IBM

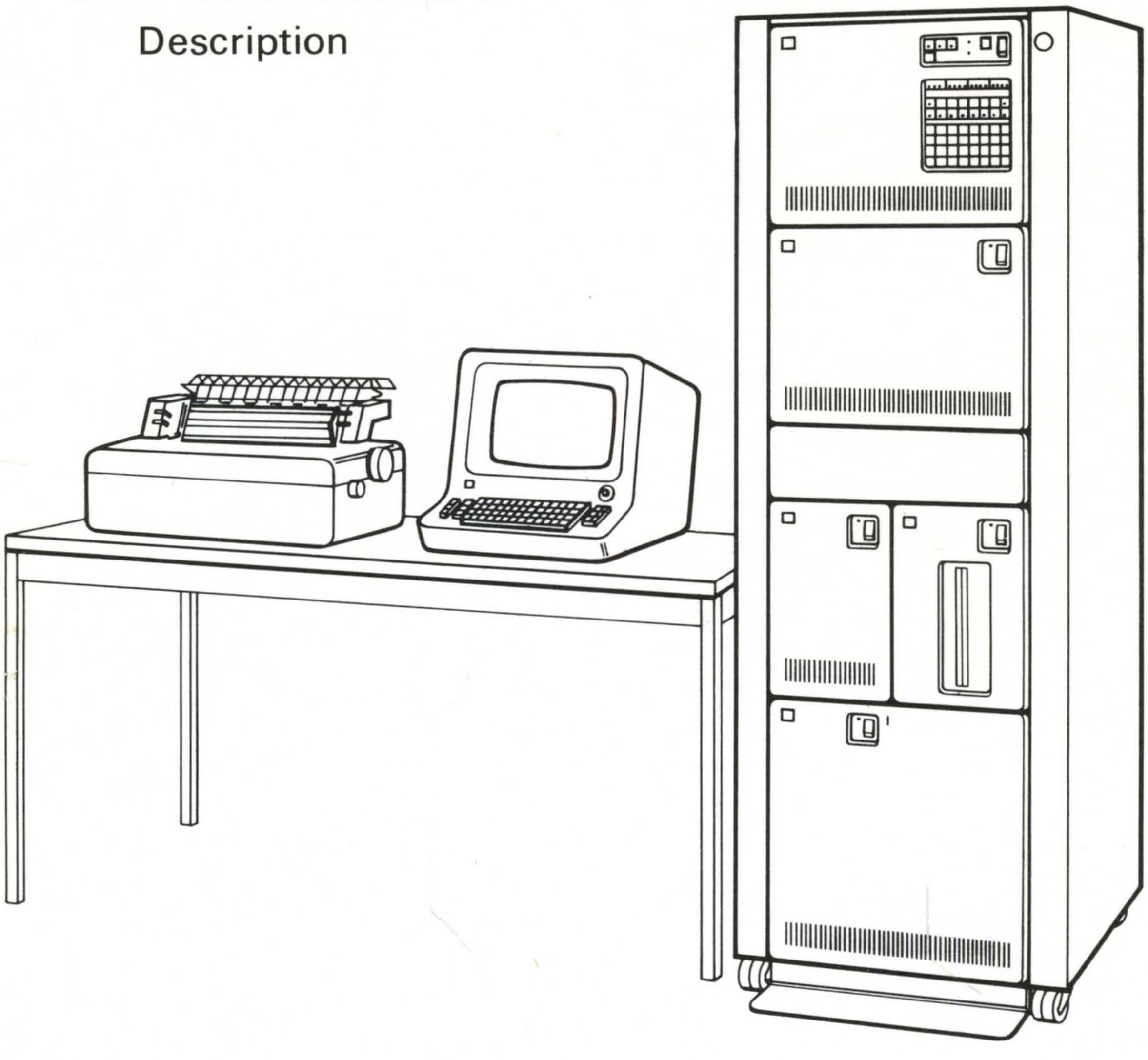
Series/1

GA34-0022-1

File No. S1-01

IBM Series/1 Model 3 4953 Processor and Processor Features Description

4953 PROCESSOR DESCRIPTION



IBM

Series/1

GA34-0022-1

File No. S1-01

IBM Series/1 Model 3 4953 Processor and Processor Features Description

4953 PROCESSOR DESCRIPTION

Second Edition (March 1977)

This is a major revision of, and obsoletes GA34-0022-0. Significant changes in this new edition include (1) rearrangement of chapters to provide a more logical flow of information and (2) removal of 4 chapters that are now included in other publications.

Contents

Preface

This publication describes the functional characteristics of the IBM 4953 Processor and the features associated with this processor. It assumes that the reader understands data processing terminology and is familiar with binary and hexadecimal numbering systems. The publication is intended primarily as a reference manual for experienced programmers who require machine code information to plan, correct, and modify programs written in the assembler language.

Summary of Publication

- *Chapter 1. Introduction* is an introduction to the system architecture. It contains a general description of the processor, storage, features, and a list of attachable I/O devices.
- *Chapter 2. Processing Unit Description* contains a description of the processor hardware including registers and indicators. The section on indicators includes examples of indicator results when dealing with signed and unsigned numbers.

  Main storage data formats and addressing are presented in this chapter.

  A section titled "Program Execution" is included and covers:
  - Basic instruction formats
  - Effective address generation
  - Processor state control
  - Initial program load (IPL)
  - Jumping and branching
  - Level switching and interrupts
  - Stack operations
- *Chapter 3. Interrupts and Level Switching* describes the priority interrupt levels and the interrupt processing for (1) I/O devices, and (2) class interrupts. Related topics are:
  - Program controlled level switching
  - Interrupt masking facilities
  - Recovery from error conditions
- *Chapter 4. Input/Output Operations* describes the I/O commands and control words that are used to operate the I/O devices. Condition codes and status information relative to the I/O operation are also explained. Specific command and status-word bit structures are contained in the I/O device description books.
- *Chapter 5. Console* describes the keys, switches, and indicators for the basic console and the optional programmer console. Typical manual operations such as storing into and displaying main storage are presented.
- *Chapter 6. Instructions* describes the basic instruction set, including indicator settings and possible exception conditions. Individual instruction word formats are included and contain bit combinations for the operation code and function fields. The instructions are arranged in alphabetical sequence based on assembler mnemonics.
- *Appendixes:*
  - Instruction execution times
  - Instruction formats
  - Assembler instruction syntax
  - Numbering systems and conversion tables
  - Character codes
  - Carry and overflow indicators
  - Reference information

Related Publications

- *IBM Series/1 System Summary*, GA34-0035.
- *IBM Series/1 Installation Manual – Physical Planning*, GA34-0029.
- *IBM Series/1 4962 Disk Storage and 4964 Diskette Unit Description*, GA34-0024.
- *IBM Series/1 4973 Line Printer Description*, GA34-0044.
- *IBM Series/1 4974 Printer Description*, GA34-0025.
- *IBM Series/1 4979 Display Station Description*, GA34-0026.
- *IBM Series/1 4982 Sensor Input/Output Unit Description*, GA34-0027.
- *IBM Series/1 Communications Features Description*, GA34-0028.
- *IBM Series/1 Attachment Features Description*, GA34-0031.
- *IBM Series/1 Battery Backup Unit Description*, GA34-0032.
- *IBM Series/1 User's Attachment Manual*, GA34-0033.

Chapter 1. Introduction

Four models of the 4953 Processor are available.

- Model A
  - 1/2 rack width unit
  - 16K bytes main storage
  - Additional storage in 16K byte increments
  - 64K bytes maximum
- Model B
  - Full rack width unit
  - 16K bytes main storage
  - Additional storage in 16K byte increments
  - 64K bytes maximum
- Model C
  - 1/2 rack width unit
  - 32K bytes main storage
  - Additional storage in 16/32K byte increments
  - 64K byte maximum
- Model D
  - Full rack width unit
  - 32K bytes main storage
  - Additional storage in 16/32K byte increments
  - 64K bytes maximum

The IBM 4953 Processor is a compact, general purpose computer and has the following general characteristics:

- Four priority interrupt levels – independent registers and status indicators for each level. Automatic and program controlled level switching.
- Main storage – read or write time is 600 nanoseconds maximum (minimum 800 nanoseconds required between two storage access cycles). Odd parity by byte is maintained throughout storage.
- TTL (transistor-transistor logic) processor technology
- Microprogram control – microcycle time: 200 nanoseconds.
- Instruction set that includes: stacking and linking facilities, multiply and divide, variable field-length byte operations, and a variety of arithmetic and branching instructions.
- Supervisor and problem states.
- Packaged in a 19-inch rack mountable unit – full width or half width.
- Basic console standard in processor unit. Programmer console optional.
- Channel capability
  - Asynchronous, multidropped channel
  - 256 I/O (input/output) devices can be addressed
  - Direct program control and cycle steal operations
  - Maximum burst data rate is 666K words per second (1.332 megabytes if transmitted in pairs). When multiple cycle stealing devices are interleaved, the aggregate data rate is also 666K words.

The processor unit contains power and space for additional features and storage. The IBM 4959 Input/Output Expansion Unit is also available for additional features.

The processor is described in the following sections of this chapter.

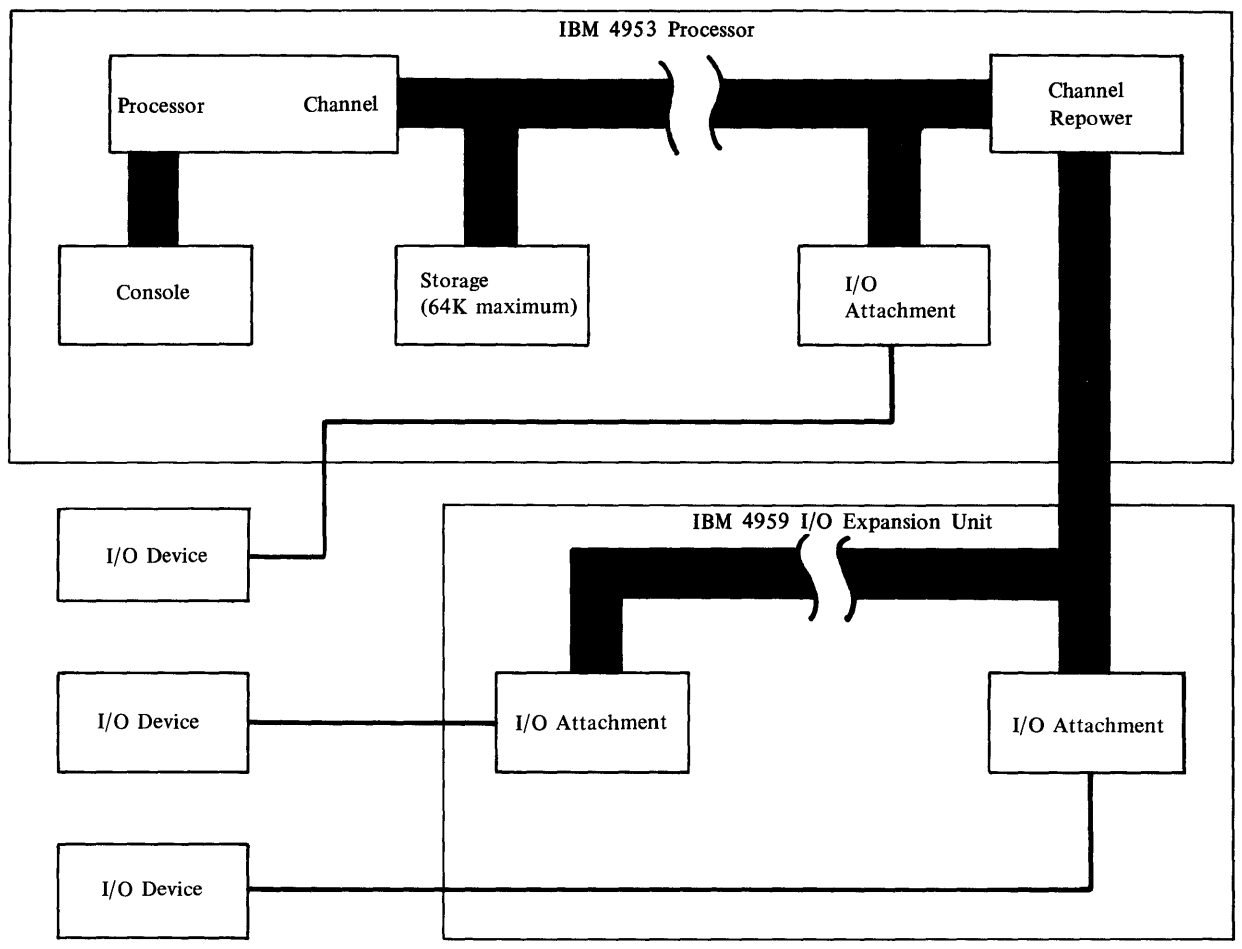


Figure 1-1. Block diagram of IBM 4953 Processor and an IBM 4959 I/O Expansion Unit

IBM 4953 Processor

Processor Options

- Storage Addition – 16,384 bytes
  - Provides additional storage in 16K byte increments for all models
  - 64K bytes maximum
- Storage Addition – 32,768 bytes
  - Provides additional storage in 32K byte increments for models C and D
  - 64K bytes maximum
- Programmer Console

Processor Description

The basic IBM 4953 Processor includes the processor, 16K bytes of storage for models A and B (32K bytes of storage for models C and D), and a basic console. These items are packaged in a unit, called the processor unit. Figure 1-1 shows a block diagram of an IBM 4953 Processor and an IBM 4959 Input/Output Expansion Unit.

The processor is microprogram controlled, utilizing a 200 nanosecond microcycle. Circuit technology is TTL.

Four priority interrupt levels are implemented in the processor. Each level has an independent set of machine registers. Level switching can occur in two ways: (1) by program control, or (2) automatically upon acceptance of an I/O interrupt request. The interrupt mechanism provides 256 unique entry points for I/O devices.

The processor instruction set contains a variety of instruction types. These include: shift, register to register, register immediate, register to (or from) storage, bit manipulation, multiple register to storage, variable byte field, and storage to storage. Supervisor and problem states are implemented, with appropriate privileged instructions for the supervisor.

The basic console is intended for dedicated systems that are used in a basically unattended environment. Only minimal controls are provided. A programmer console can be added as a feature; this console provides a variety of indicators and controls for operator-oriented systems.

Basic storage supplied is 16K bytes for models A and B; 32K bytes for models C and D. Models A and B can add additional storage in 16K byte increments up to 64K bytes maximum. Models C and D can add additional storage in 16K and/or 32K byte increments up to 64K bytes maximum. The maximum read/write access time for main storage is 600 nanoseconds. However, the minimum duration of time between successive storage cycles is 800 nanoseconds.

I/O devices are attached to the processor through the processor I/O channel. The channel directs the flow of information between the I/O devices, the processor, and main storage. The channel accommodates a maximum of 256 addressable devices.

The channel supports:

- *Direct program control operations.* Each Operate I/O instruction transfers a byte or word of data between main storage and the device. The operation may or may not terminate in an interrupt.
- *Cycle steal operations.* Each Operate I/O instruction initiates multiple data transfers between main storage and the device (65,535 bytes maximum). Cycle steal operations are overlapped with processing operations and always terminate in an interrupt.
- *Interrupt servicing.* Interrupt requests from the devices, along with cycle steal requests, are presented and polled on the interface concurrently with data transfers.

The processor is packaged in a standard 48.3 cm (19 inch) rack-mountable unit, called the processor unit. All processor units contain an integral power supply, fans, and the basic console.

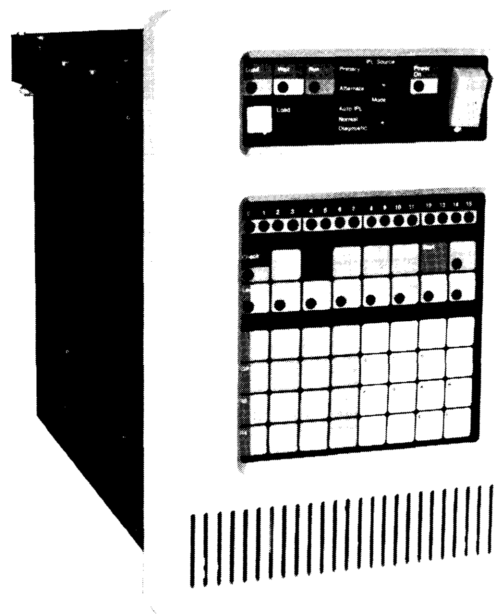
Refer to the *Series/1 Installation Manual–Physical Planning,* GA34-0029, for environmental characteristics.

Four processor models are available. Figure 1-2 shows the IBM 4953 Processor Model A, Figure 1-3 shows the IBM 4953 Processor Model B, Figure 1-4 shows the IBM 4953 Processor Model C, and Figure 1-5 shows the IBM 4953 Processor Model D.

IBM 4953 Processor Model A

This model occupies one-half the width of the standard rack and has 16K bytes of storage. It has the capacity for storage cards and/or I/O feature cards in any combination up to 4 additional cards. See Figure 1-2.

Note. Additional storage may be added in 16K byte increments up to 64K bytes maximum.



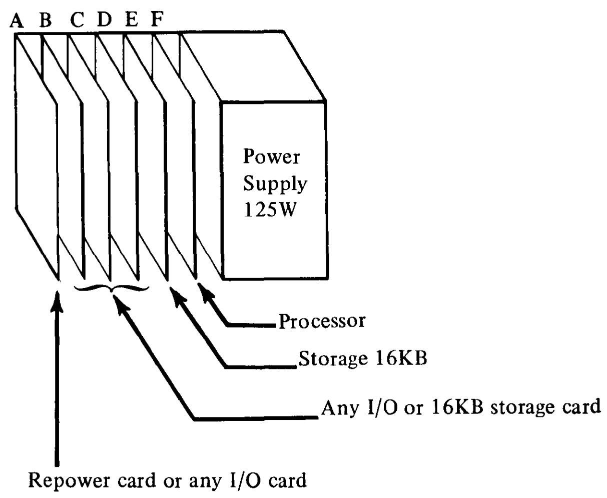
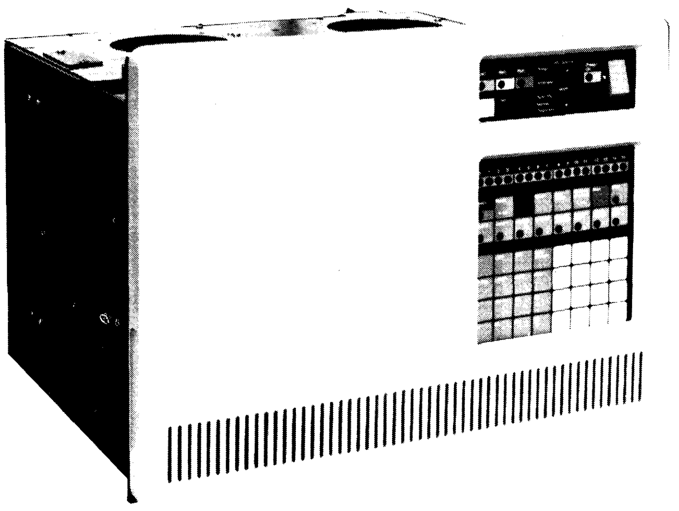


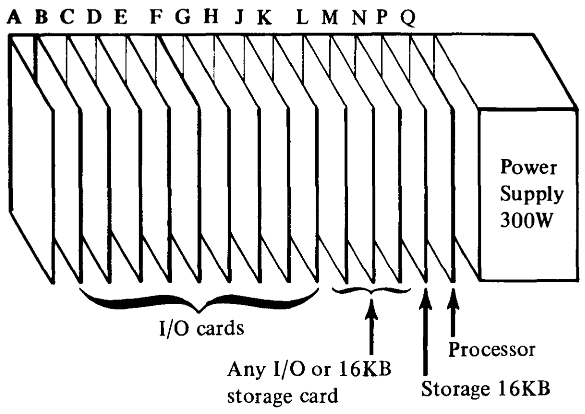
Figure 1-2. IBM 4953 Processor Model A with a Programmer Console

IBM 4953 Processor Model B

This model occupies the full width of the standard rack and has 16K bytes of storage. It has the capacity for storage cards and/or I/O feature cards in any combination up to 13 additional cards. See Figure 1-3.

Note. Additional storage may be added in 16K byte increments up to 64K bytes maximum.





If the A position is not used for the Channel Repower card, the following feature cards may be plugged in this position:

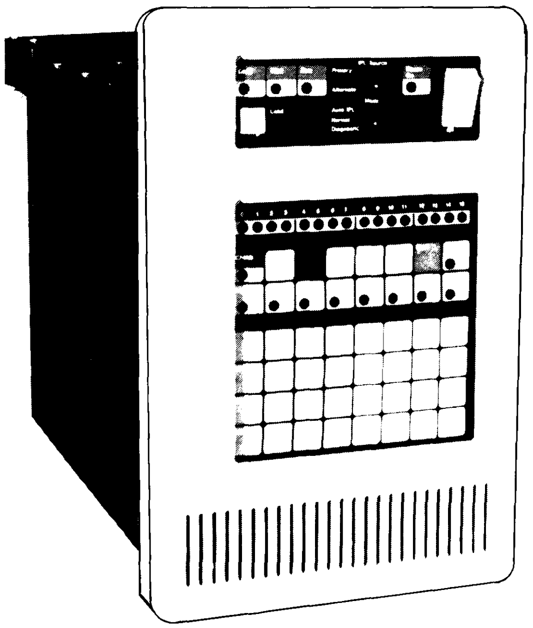
- Teletypewriter Adapter Feature using TTL voltage levels
- Teletypewriter Adapter Feature using isolated current loop where user supplies external ±12V power
- Timer Feature
- Customer Direct Program Control Adapter Feature
- 4982 Sensor Input/Output Unit Attachment Feature
- Integrated Digital Input/Output Non-Isolated Feature

Figure 1-3. IBM 4953 Processor Model B with a Programmer Console

IBM 4953 Processor Model C

This model occupies one-half the width of the standard rack and has 32K bytes of storage. It has the capacity for storage cards and/or I/O feature cards in any combination up to 4 additional cards. See Figure 1-4.

Note. Additional storage may be added in 16K/32K byte increments up to 64K bytes maximum.



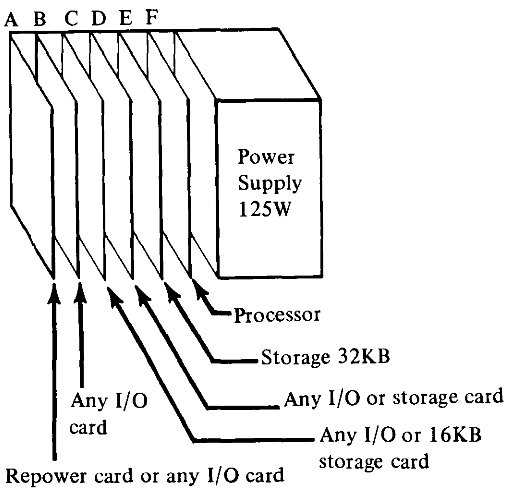
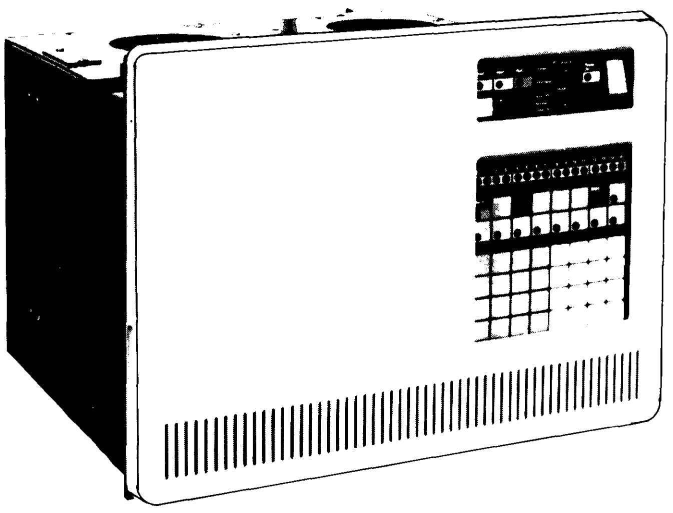


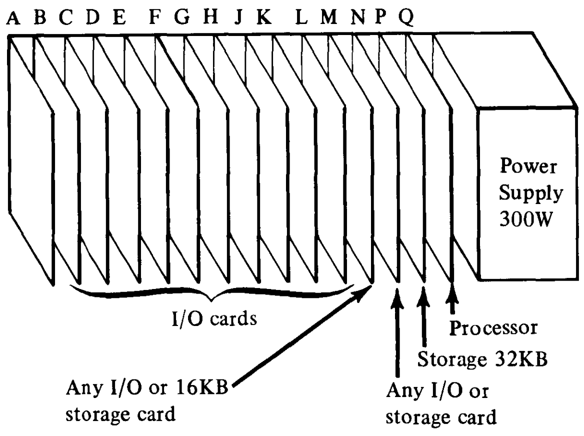
Figure 1-4. IBM 4953 Processor Model C with a Programmer Console

IBM 4953 Processor Model D

This model occupies the full width of the standard rack and has 32K bytes of storage. It has the capacity for storage cards and/or I/O feature cards in any combination up to 13 additional cards. See Figure 1-5.

Note. Additional storage may be added in 16K/32K byte increments up to 64K bytes maximum.





If the A position is not used for the Channel Repower card, the following feature cards may be plugged in this position:

- Teletypewriter Adapter Feature using TTL voltage levels
- Teletypewriter Adapter Feature using isolated current loop where user supplies external ±12V power
- Timer Feature
- Customer Direct Program Control Adapter Feature
- 4982 Sensor Input/Output Unit Attachment Feature
- Integrated Digital Input/Output Non-Isolated Feature

Figure 1-5. IBM 4953 Processor Model D with a Programmer Console

Input/Output Units and Features

- IBM 4962 Disk Storage Unit (4 models)
  - Requires 4962 Disk Storage Unit Attachment Features
- IBM 4964 Diskette Unit
  - Requires 4964 Diskette Unit Attachment Feature
- IBM 4979 Display Station
  - Requires 4979 Display Station Attachment Feature
- IBM 4973 Line Printer (2 models)
  - Requires 4973 Printer Attachment Feature
- IBM 4974 Printer
  - Requires 4974 Printer Attachment Feature
- Timers Feature (2 timers)
- Teletypewriter Adapter Feature
- Customer Direct Program Control Adapter Feature

The feature cards for attaching the I/O units can be housed in either the processor unit or the I/O expansion unit. Information about these units and features can be found in separate publications. The order numbers for these publications are listed in the preface of this manual.

Communications Features

- Asynchronous Communications Single Line Control
- Binary Synchronous Communications Single Line Control
- Binary Synchronous Communications Single Line Control/High Speed
- Synchronous Data Link Control Single Line Control
- Asynchronous Communications 8 Line Control
- Asynchronous Communications 4 Line Adapter
- Binary Synchronous Communications 8 Line Control
- Binary Synchronous Communications 4 Line Adapter
- Communications Power Feature
- Communications Indicator Panel

Refer to the publication *IBM Series/1 Communications Features Description*, GA34-0028, for a description of these features.

Sensor Input/Output Options

- Integrated Digital Input/Output Non-Isolated Feature
- IBM 4982 Sensor Input/Output Unit
  - 4982 Sensor Input/Output Unit Attachment Feature
- Features for the 4982 Sensor I/O Unit
  - Digital Input/Process Interrupt Non-Isolated
  - Digital Input/Process Interrupt Isolated
  - Digital Output Non-Isolated
  - Analog Input Control
  - Amplifier Multirange
  - Analog Input Multiplexer – Reed Relay
  - Analog Input Multiplexer – Solid State
  - Analog Output

The integrated digital input/output non-isolated feature provides digital sensor I/O and simple attachment for non-IBM equipment. This feature card can be housed in either the processor unit or the I/O expansion unit.

The 4982 sensor input/output attachment unit feature card is housed in either the processor unit or the I/O expansion unit. Refer to the publication *IBM Series/1, 4982 Sensor Input/Output Unit Description*, GA34-0027, for a description of the 4982 and associated features.

Packaging and Power Options

- IBM 4959 Input/Output Expansion Unit
- IBM 4999 Battery Backup Unit
- IBM 4997 Rack Enclosure (1-metre) – 2 models
- IBM 4997 Rack Enclosure (1.8-metre) – 2 models

The IBM 4959 Input/Output Expansion Unit is available for adding I/O feature cards beyond the capacity of the processor unit. The capacity of the I/O expansion unit is either (1) fourteen I/O cards, or (2) thirteen I/O cards plus a channel repower card. A channel repower card is required to power each additional I/O expansion unit.

The IBM 4999 Battery Backup Unit permits the processor unit (excluding external devices) to operate from a user-supplied battery when a loss or dip in line power occurs.

Other Options

Additional options such as communications cables, customer access panel, and a channel socket adapter are also available. For a list and description of system units and features, refer to the *IBM Series/1 System Summary*, GA34-0035.

Chapter 2. Processing Unit Description

Figure 2-1 shows the general data flow for the IBM 4953 Processor. The major functional units shown in the data flow are discussed in the following sections.

Main Storage

Main storage holds data and instructions for applications to be processed on the system. The data and instructions are stored in units of information called a byte. Each byte consists of eight binary data bits. Associated with each byte is a parity bit. Odd parity by byte is maintained throughout storage; even parity causes a machine check error. Formats shown in this manual exclude the parity bit(s) because they are not a part of the data flow manipulated by the instructions.

The bits within a byte are numbered consecutively, left to right, 0 through 7. When a format consists of multiple bytes, the numbering scheme is continued; for example, the bits in the second byte would be numbered 8 through 15. Leftmost bits are sometimes referred to as high-order bits and rightmost bits as low-order bits.

Bytes can be handled separately or grouped together. A *word* is a group of two consecutive bytes, beginning on an even address boundary, and is the basic building block of instructions. A *doubleword* is a group of four consecutive bytes beginning on an even address boundary.

Byte

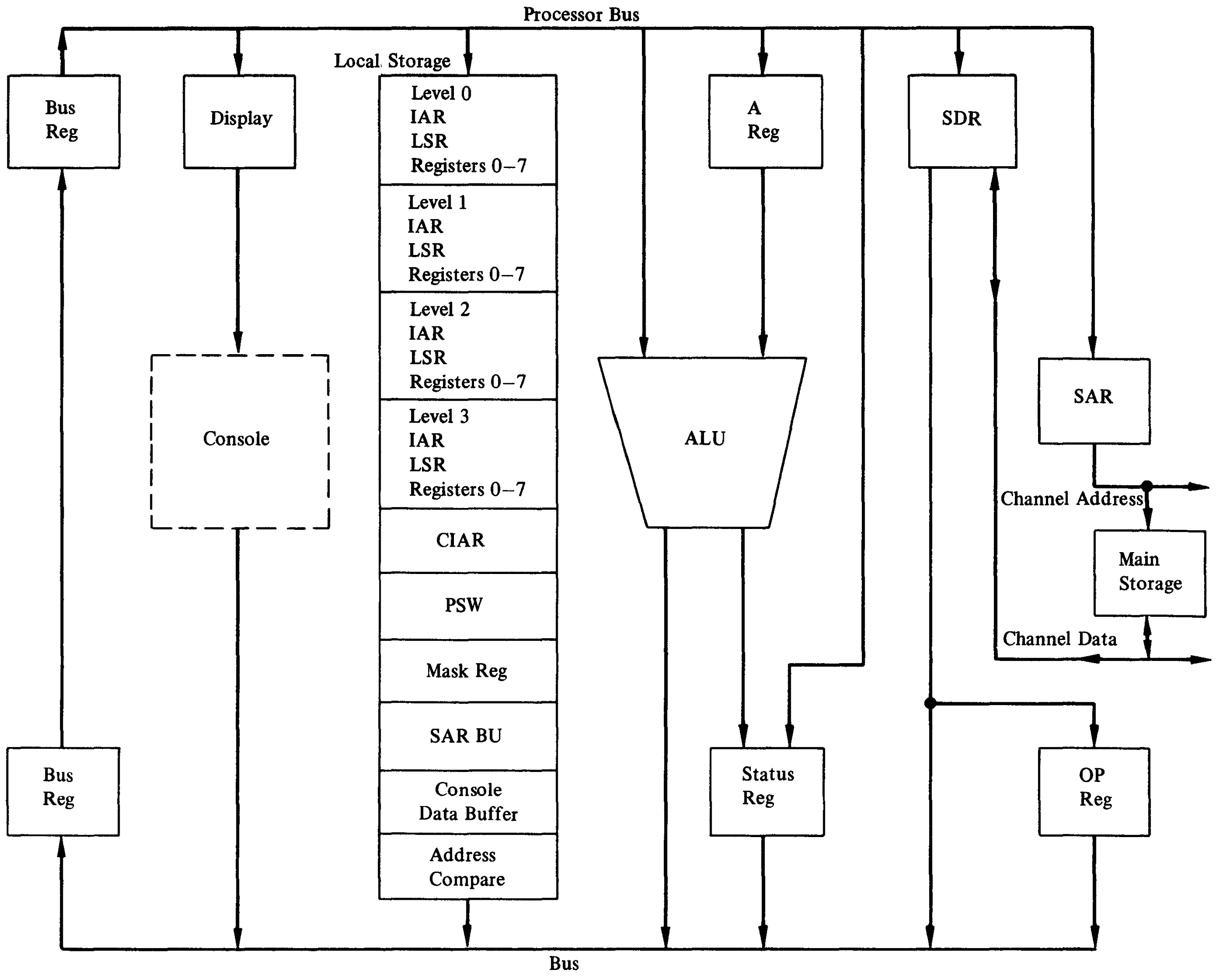
0 0 0 0 0 0 0 1
0 *7*

Word

0 0 0 0 0 0 0 0	0 0 0 0 0 0 1 0
0 *7*	*8* *15*

Doubleword

0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 1 0 0
0 *7*	*8* *15*	*16* *23*	*24* *31*



ALU - Arithmetic and logic unit	PSW - Processor status word
IAR - Instruction address register	SAR - Storage address register
LSR - Level status register	SAR BU - Storage address back-up register
Mask - Interrupt level mask register	SDR - Storage data register
OP - Operation register	

Figure 2-1. Data flow for the IBM 4953 Processor

Addressing Main Storage

Each byte location in main storage is directly addressable. Byte locations in storage are numbered consecutively, starting with location zero; each number is considered the address of the corresponding byte. Storage addresses are 16-bit unsigned binary numbers. This permits a direct addressing range of 65,536 bytes:

Address Range		
16-bit binary address	*Hexadecimal*	*Decimal*
0000 0000 0000 0000	0000	0
to	to	to
1111 1111 1111 1111	FFFF	65,535

Note. Addresses that overflow or underflow the addressing range address wrap modulo 65,536.

Instruction and Operand Address Boundaries

As previously stated, all storage addressing is defined by byte location. Instructions can refer to bits, bytes, byte strings, words, or doublewords as data operands. All word and doubleword operand addresses must be on even byte boundaries. All word and doubleword operand addresses point to the most significant (leftmost) byte in the operand. Bit addresses are specified by a byte address and a bit displacement from the most significant bit of the byte.

To provide maximum addressing range, some instructions refer to a word displacement that is added to the contents of a register. In these cases, the operand is a word and the register must contain an even byte address for valid results. Effective address generation is described in a subsequent section of this chapter.

All instructions must be on an even byte boundary. This implies that the effective address for all branch type instructions must be on an even byte boundary to be valid.

If any of the aforementioned rules are violated, a program check interrupt occurs with *specification check* set in the processor status word (PSW). The instruction is terminated.

Arithmetic and Logic Unit (ALU)

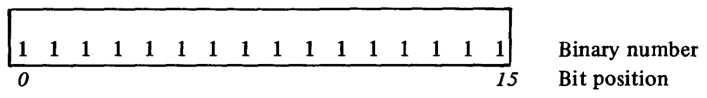
The arithmetic and logic unit (ALU) contains the hardware circuits that perform: addition; subtraction; and logical operations such as AND, OR, and exclusive OR. The ALU performs address arithmetic as well as the operations required to process the instruction operands. Operands may be regarded as signed or unsigned by the programmer. However, the ALU does not distinguish between them. Numbering representation is discussed in a subsequent section of this chapter. For many instructions, indicators are set to reflect the result of the ALU operation. The indicators are discussed in a subsequent section of this chapter.

Numbering Representation

Operands may be signed or unsigned depending on how they are used by the programmer. An unsigned number is a binary integer in which all bits contribute to the magnitude. A storage address is an example of an unsigned number. A signed number is one where the high-order bit is used to indicate the sign, and the remaining bits define the magnitude. Signed positive numbers are represented in true binary notation with the sign bit (high-order bit) set to zero. Signed negative numbers are represented in two's complement notation with the sign bit (high-order bit) set to one. The two's complement of a number is obtained by inverting each bit of the number and adding a one to the low-order bit position. Two's complement notation does not include a negative zero. The maximum positive number consists of an all-one integer field with a sign bit of zero; whereas, the maximum negative number (the negative number with the greatest absolute value) consists of an all-zero integer field with a one-bit for the sign.

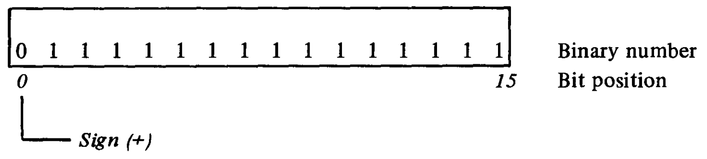
The following examples show: (1) an unsigned 16-bit number, (2) a signed 16-bit positive number, and (3) a signed 16-bit negative number.

Example of an unsigned 16-bit number:



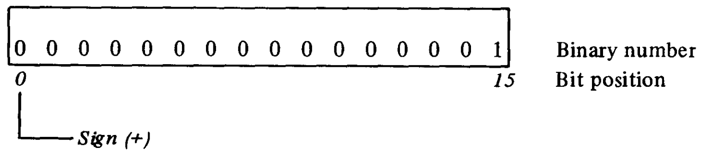
Decimal value	*65535*	*(The largest unsigned number*
Hexadecimal value	*FFFF*	*representable in 16 bits.)*

Example of a signed 16-bit positive number:



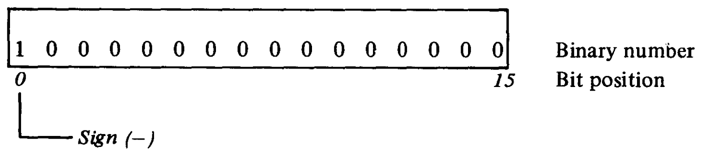
Decimal value	*+32767*	*(The largest positive signed*
Hexadecimal value	*7FFF*	*number representable in 16 bits.)*

When the number is positive, all bits to the left of the most significant bit of the number, including the sign bit, are zero:



Decimal value	*+1*
Hexadecimal value	*0001*

Example of a signed 16-bit negative number:



Decimal value	*–32768*	*(The largest negative signed*
Hexadecimal value	*8000*	*number representable in 16 bits.)*

Note. This form of representation yields a negative range of one more than the positive range.

When the number is negative, all bits to the left of the most significant bit of the number, including the sign bit, are set to one:

```
┌───────────────────────────────┐
│1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0│   Binary number
 0                            15     Bit position
 │
 └──Sign (–)

      Decimal value          –2
      Hexadecimal value      FFFE
```

When a signed-number operand must be extended with high-order bits, the expansion is achieved by prefixing a field in which each bit is set equal to the high-order bit of the operand.

Example of an 8-bit field extended to a 16-bit field:

```
                 ┌───────────────┐
                 │1 1 1 1 1 1 0 1│   Binary number
                  0             7     Bit position
                  │
                  └──Sign (–)

                      Decimal value          –3
                      Hexadecimal value      FD

┌───────────────────────────────┐
│1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1│   Binary number
 0                            15     Bit position
 │
 └──Sign (–)

      Decimal value          –3
      Hexadecimal value      FFFD
```

It must be emphasized that when performing the add and subtract operations, the machine does not regard the number as either signed or unsigned, but performs the designated operation on the values presented. Whether a given add or subtract operation is to be regarded as a signed operation or an unsigned operation is determined by the programmer's view of the values being presented as operands. The carry indicator and the overflow indicator of the LSR are changed on various operations to reflect the result of that operation. This allows the programmer to make result tests for the number representation involved. The carry and overflow indicator settings are explained in a subsequent section.

Registers

Registers in the processor are provided in two categories:

1. Per-system register (the register is provided only once and is used by all priority interrupt levels)
2. Per-level register (the register is duplicated for each priority interrupt level)

Information that must be saved when a level is preempted is retained in registers supplied for a specific level. Information that pertains only to the current process is kept in registers common to all levels. The registers in each category are listed in this section. Descriptions for each of the registers appear in subsequent sections. Only registers accessible to the program or the operator (via console operations) are discussed.

Registers supplied on a per-system basis:

- Console data buffer
- Current-instruction address register (CIAR)
- Mask register (interrupt level)
- Processor status word (PSW)
- Storage address register (SAR)

Registers supplied on a per-level basis:

- General registers (8 per level)
- Instruction address register (IAR)
- Level status register (LSR)

Note. For a specific level, the contents of the IAR, LSR, and the general registers are known as a level status block (LSB). The LSB is a 22 byte entity used by hardware and software for task control and task switching.

Per-system Registers

Console Data Buffer

The console data buffer is a 16-bit register associated with the programmer console feature. Details of how the buffer is used are explained in the programmer console section of Chapter 5. The contents of the console data buffer can be loaded into a specified general register by using the Copy Console Data Buffer (CPCON) instruction (see Chapter 6).

Current-Instruction Address Register (CIAR)

When the processor enters the stop state, the current-instruction address register (CIAR) contains the address of the last instruction that was executed. The CIAR is not addressable by software. It may be displayed from the optional programmer console. Refer to *Stop State* in this chapter for methods of entering stop state.

Mask Register

The mask register is a 4-bit register used for control of interrupts. Bit 0 controls level 0, bit 1 controls level 1, and so on.

A one bit enables interrupts on a level, while a zero bit disables interrupts. For example if bit 3 is set to a one, interrupts are enabled on level 3.

Processor Status Word (PSW)

The processor status word (PSW) is a 16-bit register used to (1) record error or exception conditions that may prevent further processing, and (2) hold certain flags that aid in error recovery. Error or exception conditions recorded in the PSW result in a class interrupt. Each bit in the PSW is described in detail in Chapter 3. The PSW can be accessed by using the Copy Processor Status and Reset (CPPSR) instruction (see Chapter 6).

Storage Address Register (SAR)

The storage address register (SAR) is a 16-bit register that contains the main-storage address for the last attempted processor storage cycle. This register is addressable by the Diagnose instruction and may be altered or displayed from the optional programmer console.

Per-level Registers

General Registers

Subsequently referred to simply as registers, the general registers are 16-bit registers available to the program for general purposes. Eight registers are provided for each level. The R and RB fields in the instructions control the selection of these registers.

Instruction Address Register (IAR)

The instruction address register (IAR) is a 16-bit register that holds the main storage address used to fetch an instruction. After an instruction has been fetched, the IAR is updated to point to the next instruction to be fetched.

Note. These registers are sometimes referred to as IAR0, IAR1, IAR2, and IAR3. The numbers represent the priority level associated with the register.

Level Status Register (LSR)

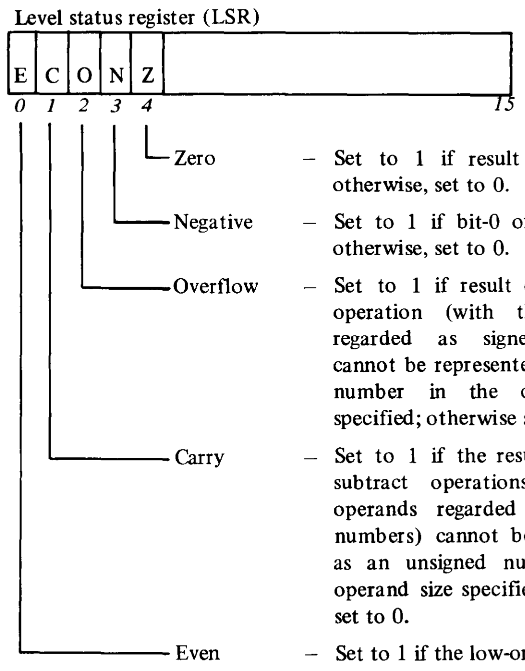
The level status register (LSR) is a 16-bit register that holds:

- Indicator bits
  - Set as a result of arithmetic, logical, or I/O operations
- A supervisor state bit
- An in-process bit
- A trace bit
- A summary mask bit.

These bits are further discussed in the following sections. Seven other bits in the LSR are not used and are always set to zero.

Indicator Bits

The indicators are located in bits 0–4 of the level status register (LSR). Figure 2-2 shows the indicators and how they are set for arithmetic operations. The indicator bits are changed or not changed depending on the instruction being executed. Some instructions do not affect the indicators, other instructions change all of the indicators, and still other instructions change only specific indicators. Refer to the individual instruction descriptions in Chapter 6 for the indicators changed by each instruction.



Zero	– Set to 1 if result is all zeros; otherwise, set to 0.
Negative	– Set to 1 if bit-0 of result is 1; otherwise, set to 0.
Overflow	– Set to 1 if result of arithmetic operation (with the operands regarded as signed numbers) cannot be represented as a signed number in the operand size specified; otherwise set to 0.
Carry	– Set to 1 if the result of add or subtract operations (with the operands regarded as unsigned numbers) cannot be represented as an unsigned number in the operand size specified; otherwise, set to 0.
Even	– Set to 1 if the low-order bit of the result is 0; otherwise, set to 0.

Figure 2-2. How indicators are set for signed and unsigned (logical) operations

The indicators are changed in a specialized manner for certain operations. These operations are described briefly. Additional information is provided in subsequent sections for those operations where more detail is required.

- *Add, subtract, or logical operations.* The even, negative, and zero indicators are *result* indicators. For add and subtract operations, the carry and overflow indicators are changed to provide information for both signed and unsigned number representations.
- *Multiply and divide operations.* Signed number operands are always assumed for these operations. The carry indicator is used to provide a divide by zero indication for the divide instruction. The overflow indicator defines an unrepresentable product for multiply operations. Refer to the individual instruction descriptions in Chapter 6.
- *Priority interrupts and input/output operations.* The even, carry and overflow indicators are used to form a three-bit condition code that is set as a binary value.
- *Compare operations.* The indicators are set in the same manner as a subtract operation.
- *Shift operations.* The carry and overflow indicators have a special meaning for shift left logical operations.
- *Complement operations.* The overflow indicator is set if an attempt is made to complement the maximum negative number. This number is not representable.
- *Set Indicators (SEIND) and Set Level Block (SELB) instructions.* All indicators are changed by the data associated with these instructions.

Even, Negative, and Zero Result Indicators

The even, negative, and zero indicators are called the result indicators. A positive result is indicated when the zero and negative indicators are both off (set to zero). These indicators are set to reflect the result of the last arithmetic, or logical operation performed. A logical operation in this sense includes data movement instructions. See the individual instruction descriptions in Chapter 6 for the indicators changed for specific instructions.

Even, Carry, and Overflow Indicators – Condition Code for Input/Output Operations

The even, carry, and overflow indicators contain the I/O condition code: (1) following the execution of an Operate I/O instruction and (2) following an I/O interrupt.

These indicators are used to form a 3-bit binary number that results in a condition code value. For additional information about condition codes, refer to:

1. Branch on Condition Code (BCC) and Branch on Not Condition Code (BNCC) instructions in Chapter 6.
2. Condition codes in Chapter 4.

Carry and Overflow Indicators – Add and Subtract Operations

A common set of add and subtract integer operations performs both signed and unsigned arithmetic. Whether a given add or subtract operation is to be regarded as a signed operation or an unsigned operation is determined by the programmer's view of the values being presented as operands. The carry and overflow indicators are set to reflect the result for both cases.

Carry Indicator Setting

The carry indicator is used to signal overflow of the result when operands are presented as *unsigned* numbers.

Overflow Indicator Setting

The overflow indicator is used to signal overflow of the result when the operands are presented as *signed* numbers.

Note. Appendix F explains the meaning of these indicators for signed and unsigned numbers. The appendix also provides examples for setting the carry indicator and for setting the overflow indicator.

Carry and Overflow Indicators – Shift Operations

The carry and overflow indicators are changed for shift left logical operations and shift left and test operations. These operations affect the indicators as follows:

1. The carry indicator is set to reflect the value of the last bit shifted out of the target register (register where bits are being shifted).
2. The overflow indicator is set to one if bit-0 of the target register was changed during the shift. Otherwise it is set to zero.

Indicators – Compare Operations

A compare operation sets the indicators in the same manner as a subtract operation. The even, negative, and zero indicators reflect the result. The carry and overflow indicators are set as described previously.

Compare instructions provide a test between two operands (without altering either operand) so that conditional branch and jump instructions may be used to control the programming logic flow. The conditions specified in branch and jump instructions are named such that, when the condition of the "subtracted from" operand relative to the other operand is true the jump or branch occurs. Otherwise, the next sequential instruction is executed. This is illustrated in the following example.

- Compare operation example

Instruction name	*Assembler mnemonic*	*Operands*
Compare word	CW	R3, R4

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0	0 1 1	1 0 0	0 0 1 0 1
0 4	*5 7*	*8 10*	*11 15*
	R3	R4	

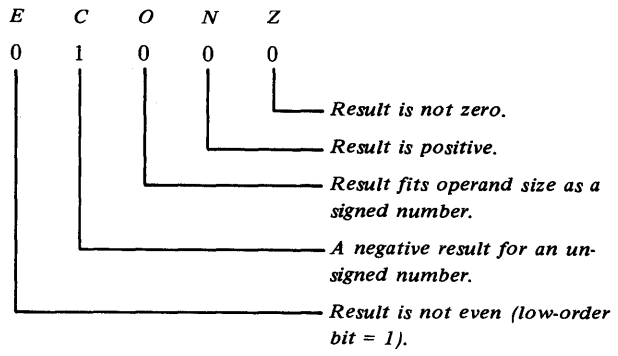
In this example, the contents of register 3 are subtracted from register 4:

		Decimal Unsigned	*Decimal Signed*
R4 contents	0000 0000 0000 0010	2	+2
R3 contents	1111 1111 1111 1011	65531	-5
Subtract result		-65529	+7

Machine operation:

Minuend	0000 0000 0000 0010	
Subtrahend	0000 0000 0000 0100	one's complement
Constant	1	for two's complement
Result	0000 0000 0000 0111	

Indicator Settings:


E C O N Z
0 1 0 0 0
Result is not zero.
Result is positive.
Result fits operand size as a signed number.
A negative result for an unsigned number.
Result is not even (low-order bit = 1).

If the programmer is comparing *unsigned* numbers, such as storage addresses, he should use the logical conditional tests (refer to Figure 2-3). In this example, assuming unsigned number representation, R4 is logically less than R3 and unequal to R3. Therefore, the following branch instructions would cause a transfer to symbolic location A (assuming register values shown in the example).

```
CW      R3,R4
BLLT    A
or
CW      R3,R4
BNE     A
```

The complementary tests (BLGT and BE) would not cause a transfer in this case.

If the programmer is comparing *signed* numbers, he should use the arithmetic conditional tests (refer to Figure 2-3). In the previous compare word example, assuming signed number representation, R4 is greater than R3 and unequal to R3. The following branch instructions would cause a transfer to symbolic location A.

```
CW      R3,R4
BGT     A
or
CW      R3,R4
BNE     A
```

The complementary tests (BLT and BE) would not cause a transfer.

Note. Jump instructions are also available for the logical and arithmetic conditional tests.

It must be emphasized again that the machine does not regard the numbers as either signed or unsigned. The compare word instruction results in a subtract operation being performed on the values presented. The programmer must then choose the correct conditional test (logical or arithmetic) for the number representation involved.

Indicators – Multiple Word Operands

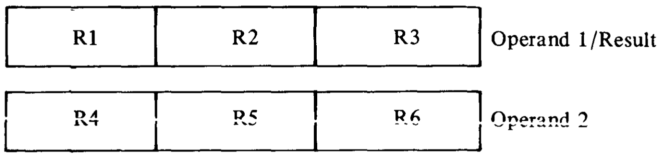
A programmer may desire to work with numbers that cannot be represented in one word or in a doubleword. It may take three or more words to represent the number.

Certain register to register instructions allow the programmer to add or subtract these multi-word operands and then have the indicators reflect the multi-word result. These instructions are:

Add Carry Register (ACY)
Add Word With Carry (AWCY)
Subtract Carry Register (SCY)
Subtract Word With Carry (SWCY)

The following two examples show how the add instructions are used. A subtract operation would be similar. See Chapter 6 for details of the individual instructions.

Example 1. (Equal length operands)



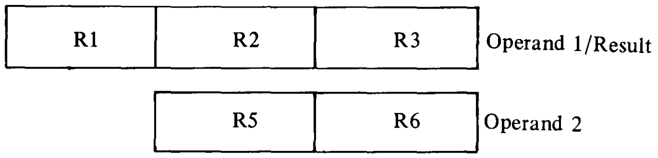
Program steps:

```
AW      R6,R3
AWCY    R5,R2
AWCY    R4,R1
```

Explanation:

Step 1: The contents of R6 are added to the contents of R3.
Step 2: The contents of R5 are added to the contents of R2 plus any carry from the previous operation.
Step 3: The contents of R4 are added to the contents of R1 plus any carry from the previous operation.

Example 2. (Unequal length operands)



Note. In this example, operand 2 must be an unsigned number or must be positive.

Program Steps:

```
AW      R6,R3
AWCY    R5,R2
ACY     R1
```

Explanation:

Step 1: The contents of R6 are added to the contents of R3.
Step 2: The contents of R5 are added to the contents of R2 plus any carry from the previous operation.
Step 3: Any carry from the previous operation is added to the contents of R1.

Note. In both examples the final indicator settings reflect the status of the 3-word result.

Even	Set on if the result low-order bit of R3 is zero.
Carry	Set on if the result cannot be represented as an unsigned 3-word number.
Overflow	Set on if the result cannot be represented as a signed 3-word number.
Negative	Set on if the result high-order bit of R1 is one.
Zero	Set on if all three result registers contain zeros.

Testing Indicators with Conditional Branch and Jump Instructions

The indicators are tested according to a selected condition when a conditional branch or a conditional jump instruction is executed. The conditions and the indicators tested for each condition are shown in Figure 2-3.

The conditional instructions are:

- Branch on Condition (BC)
- Branch on Not Condition (BNC)
- Jump on Condition (JC)
- Jump on Not Condition (JNC)

The assembler also provides extended mnemonics for the conditions shown in Figure 2-3. Refer to the individual instructions in Chapter 6.

Condition tested by conditional branch or jump instruction	*Assembler extended mnemonics*	*Indicators tested*				
		0	*1*	*2*	*3*	*4*
		E	*C*	*O*	*N*	*Z*
Zero or equal	BE, BZ, JE, JZ					1
Not zero or unequal	BNE, BNZ, JNE, JNZ					0
Positive and not zero	BP, JP				0	0
Not positive	BNP, JNP				 1	1
Negative	BN, JN				1	
Not negative	BNN, JNN				0	
Even	BEV, JEV	1				
Not even	BNEV, JNEV	0				
Arithmetically less than	BLT, JLT			0 1	1 0	
Arithmetically less than or equal	BLE, JLE			0 1 	1 0 	 1
Arithmetically greater than or equal	BGE, JGE			1 0	1 0	
Arithmetically greater than	BGT, JGT			1 0	1 0	0 0
Logically less than or equal	BLLE, JLLE		1 			 1
Logically less than (carry)	BLLT, JLLT		1			
Logically greater than	BLGT, JLGT		0			0
Logically greater than or equal (no carry)	BLGE, JLGE		0			

Legend:	*LSR bit*	*Indicator*
	0	E – Even
	1	C – Carry
	2	O – Overflow
	3	N – Negative
	4	Z – Zero

Figure 2-3. Indicators tested by conditional branch and jump instructions

Supervisor State Bit

LSR bit 8, when set to one, indicates that the processor is in the supervisor state. This state allows privileged instructions to be executed. It is set by any of the following:

1. Class interrupt
   a. Machine check condition
   b. Program check condition
   c. Power/thermal warning
   d. Supervisor Call (SVC) instruction
   e. Soft exception trap condition
   f. Trace
   g. Console interrupt
2. I/O interrupt
3. Initial program load (IPL)

When LSR bit 8 is set to zero, the processor is in problem state. For a selected priority level, the supervisor can alter the supervisor state bit by using a Set Level Block (SELB) instruction. For additional information, refer to *Processor State Control* in this chapter.

Class interrupts and I/O interrupts are described in Chapter 3. IPL is discussed in a subsequent section of this chapter.

In-process Bit

LSR bit 9, when set to one, indicates that a priority level is currently active or was preempted by a higher priority level before completing its task. Bit 9 is turned off by a Level Exit (LEX) instruction. Bit 9 can also be turned on or off by a Set Level Block (SELB) instruction. The in-process bit also affects level switching under program control. Refer to *Chapter 3. Interrupts and Level Switching.*

Trace Bit

LSR bit 10, when set to one, causes a trace class interrupt at the beginning of each instruction. The bit can be turned on or off with the Set Level Block (SELB) instruction. The trace bit aids in debugging programs. See *Class Interrupts* in Chapter 3.

Summary Mask Bit

LSR bit 11, when set to zero (disabled), inhibits all priority interrupts on all levels. When this bit is set to one (enabled), normal interrupt processing is allowed. Refer to *Summary Mask* in Chapter 3 for details relating to control of the summary mask.

Program Execution

Instruction Formats

The processor instruction formats are designed for efficient use of bit combinations to specify the operation to be performed (operation code) and the operands that participate. Some formats also include (1) an immediate data field or word, (2) an address displacement or address word, and (3) a function field that further modifies the operation code. Various combinations of these fields are used by the individual instructions. Some typical instruction formats are presented in this section. All formats are shown in the section *Instruction Formats* in Appendix B.

One Word Instructions

The basic instruction length is one word (16 bits). The operation code field (bits 0–4) is the only common field for all formats. This field, unless modified by a function field, specifies the operation to be performed. For a format without a function field, bits 5–15 specify the location of operands or data associated with an operand:

Example:

Instruction name	*Assembler mnemonic*	*Syntax*
Add Byte Immediate	ABI	byte,reg

Operation code	*R*	*Immediate*
0 0 0 0 0		
0 4	*5 7*	*8 15*

Bits 0–4	Operation code (specifies ABI instruction).
Bits 5–7	General register (0–7). This register contains data for the second operand.
Bits 8–15	Immediate data for the first operand.

In some cases the operation code is the same for a group of instructions. The format for this group includes a function field. The bit combinations in the function field then determine the specific operation to be performed.

Example:

Instruction name	*Assembler mnemonic*	*Syntax*
Add Word	AW	reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			0 1 0 0 0
0 4	*5 7*	*8 10*	*11 15*

Bits 0–4	Operation code for a group of instructions.
Bits 5–7	General register (0–7). This register contains data for the first operand.
Bits 8–10	General register (0–7). This register contains data for the second operand.
Bits 11–15	Function field. Modifies the operation code to specify the Add Word instruction.

Note. For other instruction groups, the function field may vary as to location within the format, and also the number of bits used.

Two Word Instructions

The first word of this format is identical to the one-word format. The second word (bits 16–31) contains either immediate data, an address, or a displacement. This word is used to (1) provide data for an operand, or (2) provide a main storage address or displacement for effective address generation (see *Effective Address Generation* in this chapter).

Example:

Instruction name	*Assembler mnemonic*	*Syntax*
Branch and Link	BAL	longaddr,reg

Operation code	*R1*	*R2*	*X*	*Function*
0 1 1 0 1				0 0 1 1
0 4	*5 7*	*8 10*	*11*	*12 15*

Address or displacement
16 31

Bits 0–4	Operation code
Bits 5–7	General register (0–7) for the second operand
Bits 8–10	General register (0–7) for the first operand
Bit 11	Indirect addressing bit
Bits 12–15	Function field
Bits 16–31	A main storage address used for the first operand

Note. In this example, the register designated R1 is associated with the second operand in assembler syntax.

Variable Length Instructions

Some instructions use a selectable encoded technique for generating effective addresses. This method is referred to as an address argument technique in subsequent sections. These instruction formats contain a base register (RB) field and an address mode (AM) field. If both operands are using this technique, the format contains an RB and associated AM field for each. These fields are in the first instruction word. The AM field consists of two bits and is referred to in binary notation (AM=00, 01, 10, or 11). If AM is equal to 10 or 11 an additional word is appended to the normal instruction word. For a format that contains two AM fields, two additional words may be appended. See *Effective Address Generation* in this chapter for a description of the appended words and how they are used.

For instructions with a single storage address argument, the RB field consists of two bits. An RB field of two bits with its associated AM field of two bits are referred to as a 4-bit address argument or *addr4* in assembler syntax.

Example:

Instruction name	*Assembler mnemonic*	*Syntax*
Compare byte	CB	addr4, reg

Operation code	*R*	*RB*	*AM*	*Function*
1 1 0 0 0				0 1 0 0
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Appended word, AM=10 or 11
16 31

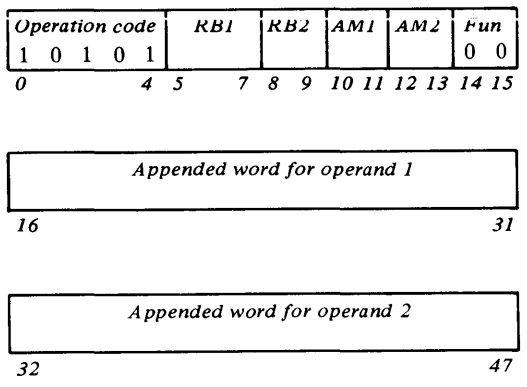
Bits 0–4	Operation Code.
Bits 5–7	General register (0–7) for the second operand.
Bits 8–9	Base register (0–3).
Bits 10–11	Address mode.
Bits 12–15	Function.
Bits 16–31	Appended word for the first operand.

Note. The register specified by the RB field is a general register that is used as a base register for effective address generation.

Some instruction formats have two storage address arguments. In this case, the first operand has a 3-bit RB field giving a 5-bit address argument (*addr5* in assembler syntax) and the second operand has a 4-bit address argument.

Example:

Instruction name	*Assembler menmonic*	*Syntax*
Add Word	AW	addr5,addr4



Bits 0–4	Operation code.
Bits 5–7	Base register (0–7) for the first operand.
Bits 8–9	Base register (0–3) for the second operand.
Bits 10–11	Address mode for the first operand.
Bits 12–13	Address mode for the second operand.
Bits 14–15	Function.
Bits 16–31	Appended word for the first operand.
Bits 32–47	Appended word for the second operand.

Notes.

1. If there is no appended word for the first operand (AM1=00 or 01), the second operand word is appended to the instruction word in bits 16–31.
2. Registers specified by the RB fields are general registers.

Names of Instruction Formats

Names have been established for several categories of instructions. Each category has the same basic instruction format, therefore, the name is related to the format. In most cases, the name indicates the location of the operands or the type of instruction.

Examples:

- Register/Register Instructions
  General registers are used by both operands.
- Storage/Storage Instructions
  Both operands reside in main storage.
- Register/Storage Instructions
  One operand uses a general register. The other operand resides in main storage.
- Register Immediate Instructions
  One operand uses a general register. The other operand uses an immediate data field. The immediate data field is the low order byte of a one-word format or the second word of a two-word (long) format.
- Shift Instructions with Immediate Count
  This is a shift instruction with the count field contained within the instruction word.
- Storage Immediate Instructions
  One operand is in main storage. The other operand uses an immediate data field. The immediate data field is the second word of a two-word format
- Parametric Instructions
  For this instruction format, a parameter field (bits 8–15) is contained within the instruction word.

Effective Address Generation

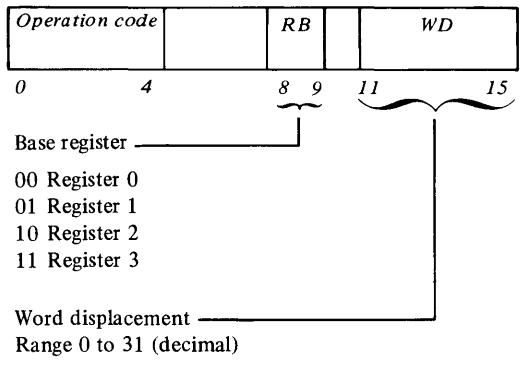
For purposes of storage efficiency, certain instructions formulate storage operand addresses in a specialized manner. These instructions have self-contained fields that are used when generating effective addresses. Standard methods for deriving effective addresses are included in this section. Other methods such as bit displacements, are explained in the individual instruction descriptions in Chapter 6.

Programming Note. For certain instructions, the effective address points to a control block rather than an operand. These instructions are:

- Copy Level Block (CPLB)
- Load Multiple and Branch (LMB)
- Pop Byte (PB)
- Pop Doubleword (PD)
- Push Byte (PSB)
- Push Doubleword (PSD)
- Push Word (PSW)
- Pop Word (PW)
- Set Level Status Block (SELB)
- Store Multiple (STM)

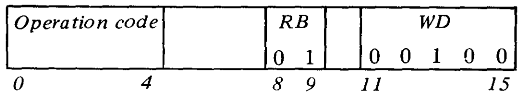
Base Register Word Displacement Short

Instruction format



The five-bit unsigned integer (WD) is doubled in magnitude to form a byte displacement then added to the contents of the specified base register to form the effective address. The contents of the base register must be even.

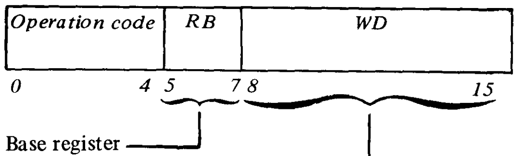
Example:



		Hex	Dec
Contents of register 1 (RB)	0000 0000 0110 0000	0060	0096
Word displacement (WD) doubled	+ 0 1000	8	8
Effective address	0000 0000 0110 1000	0068	0104

Base Register Word Displacement

Instruction format



Base register

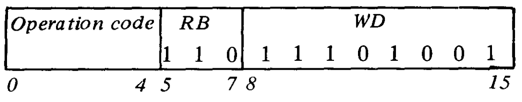
000	Register 0
001	Register 1
010	Register 2
011	Register 3
100	Register 4
101	Register 5
110	Register 6
111	Register 7

Word displacement
Range +127 to −128 (decimal)

The eight-bit *signed* integer (WD) is doubled in magnitude to form a byte displacement then added to the contents of the specified base register to form the effective address. The contents of the base register must be even.

The word displacement can be either positive or negative; bit 8 of the instruction word is the sign bit for the displacement value. If this high-order bit of the displacement field is a 0, the displacement is positive with a maximum value of +127 (decimal). If the high-order bit of the displacement field is a 1, the displacement is negative with a maximum value of -128. The negative number is represented in twos-complement form.

Example:

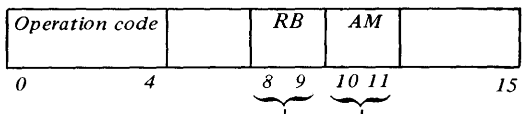


Note. This example uses a negative word displacement (-17 hex) shown in two's complement.

		Hex	Dec
Contents of register 6 (RB)	0000 0000 1000 0110	0086	0134
Word displacement (WD) doubled (sign bit is propagated left)	+ 1111 1111 1101 0010	- 2E	- 46
Effective address	0000 0000 0101 1000	0058	0088

Four-Bit Address Argument

Instruction format



Base register

00	Register 0 (AM=00 or 01)
00	No register (AM=10 or 11)
01	Register 1
10	Register 2
11	Register 3

Address mode

The Address Mode (AM) has the following significance:

AM=00. The contents of the selected base register form the effective address.

AM=01. The contents of the selected base register form the effective address. After use, the base register contents are incremented by the number of bytes in the operand. For some instructions the effective address points to a control block rather than an operand. When the effective address points to a control block, the base register contents are incremented by two.

Example:

Operation code		*RB*	*AM*	
		0 1	0 1	
0 4		*8 9*	*10 11*	*15*

		Hex	*Dec*
Effective address (contents of register 1)	0000 0000 1000 0000	0080	0128
Contents of register 1 after instruction execution			
Byte operand	0000 0000 1000 0001	0081	0129
Word operand	0000 0000 1000 0010	0082	0130
Double word operand	0000 0000 1000 0100	0084	0132

Notes.

1. For register to storage instructions, if the register specified is the same for both operands then the register will be incremented prior to using it as an operand.
2. Certain instructions (storage-to-storage) have two address arguments. Operand 1 has a 3-bit RB field with its associated AM field. Operand 2 has a 2-bit RB field with its associated AM field. If both RB fields specify the same register and both AM fields are equal to 01, the base register contents are incremented prior to fetching operand 2 and again after fetching operand 2. Assuming the same conditions but with the operand 2 AM field not equal to 01, the base register contents are incremented prior to calculating the effective address for operand 2.
3. If the effective address points to a control block rather than an operand, the base register contents are incremented by two.

AM=10. An additional word is appended to the instruction. The word has the following format.

Address or displacement
16 31

- If RB is zero, the appended word contains the effective address.
- If RB is non-zero, the contents of the selected base register and the contents of the appended word (displacement) are added to form the effective address.

Example:

Operation code		*RB*	*AM*		*Address*
		1 1	1 0		0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 4		*8 9*	*10 11*	*12 15*	*16 31*

		Hex	*Dec*
Contents of register 3	0000 1000 0000 0000	0800	2048
Contents of appended word	+ 0000 0001 0000 0000	0100	0256
Effective address	0000 1001 0000 0000	0900	2304

AM=11. An additional word is appended to the instruction.

- If RB is zero, the appended word has the format:

Indirect address
16 31

  This address points to a main storage location, on an even byte boundary, that contains the effective address.

Example:

Operation code		*RB*	*AM*		*Indirect address*
		0 0	1 1		0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0
0 4		*8 9*	*10 11*	*12 15*	*16 31*

		Hex	*Dec*
Contents of appended word	0000 0000 0101 0000	0050	0080
Effective address equals contents of storage at address 0080 (decimal)	0000 0100 0000 0000	0400	1024

- If RB is non-zero, the appended word has the format:

Displacement 1	*Displacement 2*
16 23	*24 31*

  The two displacements are unsigned eight-bit integers. Displacement 2 is added to the contents of the selected base register to generate a main storage address. The contents of this storage location are added to Displacement 1 resulting in the effective address.

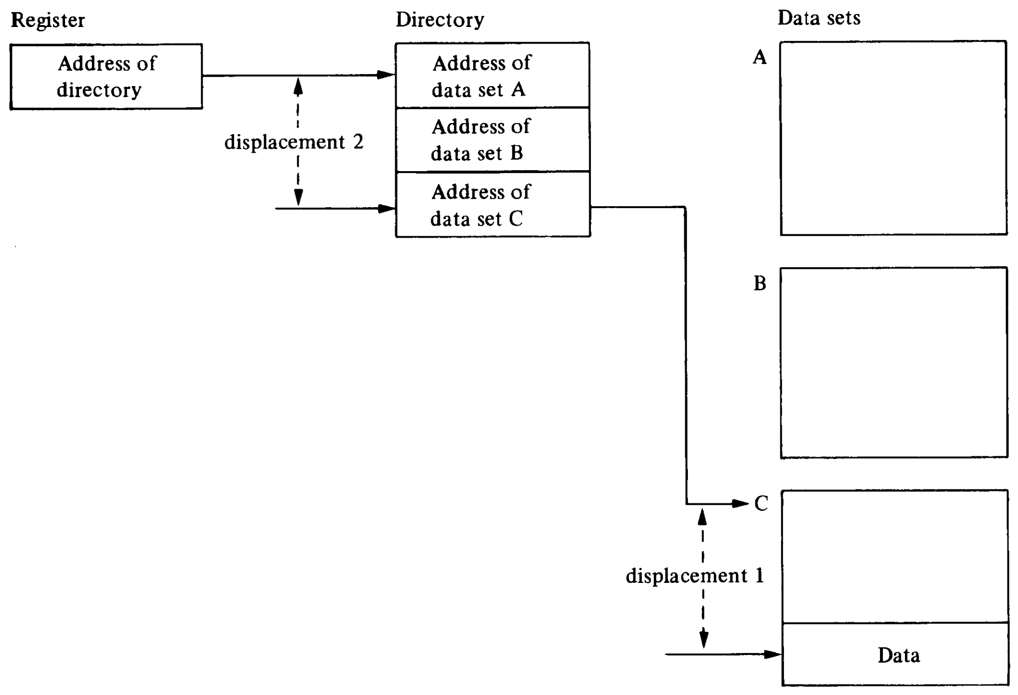
Example:

Operation code		*RB*	*AM*		*Displacement 1*	*Displacement 2*
		1 0	1 1		0 0 1 0 0 1 0 1	0 1 0 0 0 0 1 0
0	*4*	*8 9*	*10 11*	*12 15*	*16 23*	*24 31*

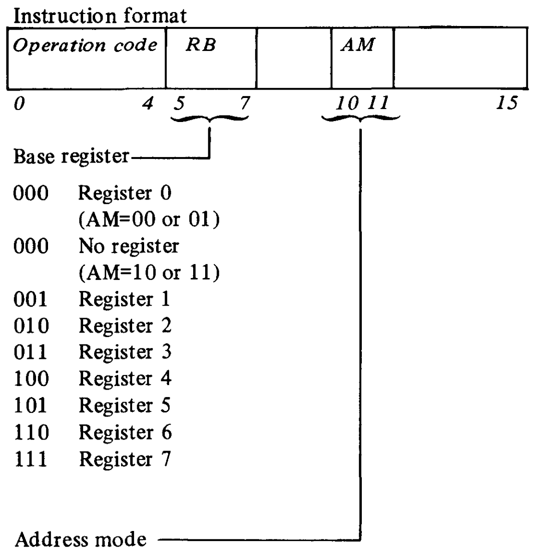
		Hex	*Dec*
Contents of register 2	0000 0101 0011 0101	0535	1333
Displacement 2	+ 0100 0010	42	66
Storage address	0000 0101 0111 0111	0577	1399
Contents of storage at address 1399 (decimal)	0000 0100 0001 0000	0410	1040
Displacement 1	+ 0010 0101	25	37
Effective address	0000 0100 0011 0101	0435	1077

Note. This example is invalid for other than a byte operand.

Programming Note. This addressing mode (AM=11, RB is non-zero) is useful for the directorized data concept. For the *addr4* or *addr5* assembler syntax, the programmer codes the form *displacement 1 (register, displacement 2)**. For addr4, the specified register is 1–3. For addr5, the specified register is 1–7. The asterisk denotes indirect addressing.

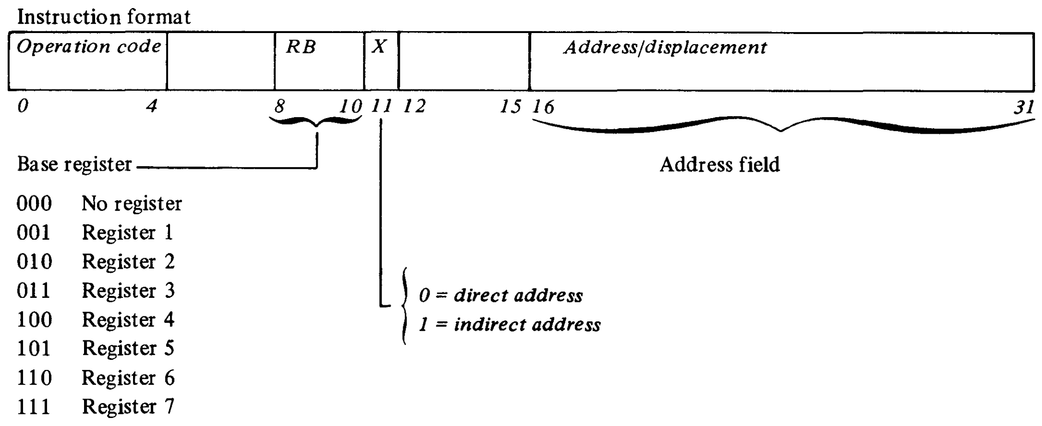


Five-Bit Address Argument



Operation of this mode is identical to the four-bit argument, but provides additional base registers.

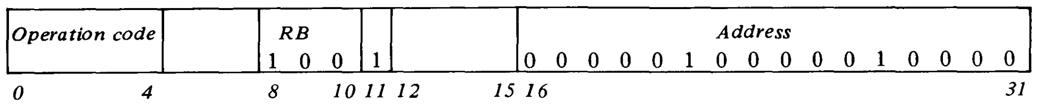
Base Register Storage Address



- If RB is zero, the address field contains the effective address.
- If RB is non-zero, the contents of the selected base register and the contents of the address field are added together to form the effective address.

Note. Bit 11, if a one, specifies that the effective addressing is indirect.

Example: Indirect address

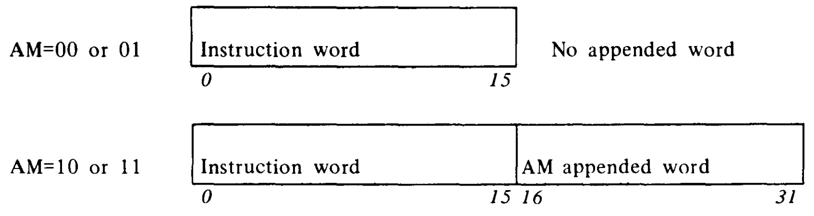


		Hex	Dec
Contents of register 4	0000 0001 0000 0000	0100	0256
Address field	+ 0000 0100 0001 0000	0410	1040
Storage address	0000 0101 0001 0000	0510	1296
Effective address			
Contents of storage at address 1296 (decimal)	0000 0110 0100 0000	0640	1600

Instruction Length Variations for Address Arguments

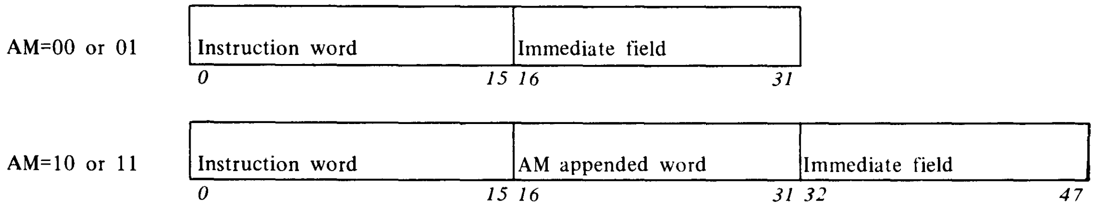
- One-word instructions that contain a single AM field become two words in length if AM is equal to 10 or 11. The AM appended word follows the instruction word.

Example:



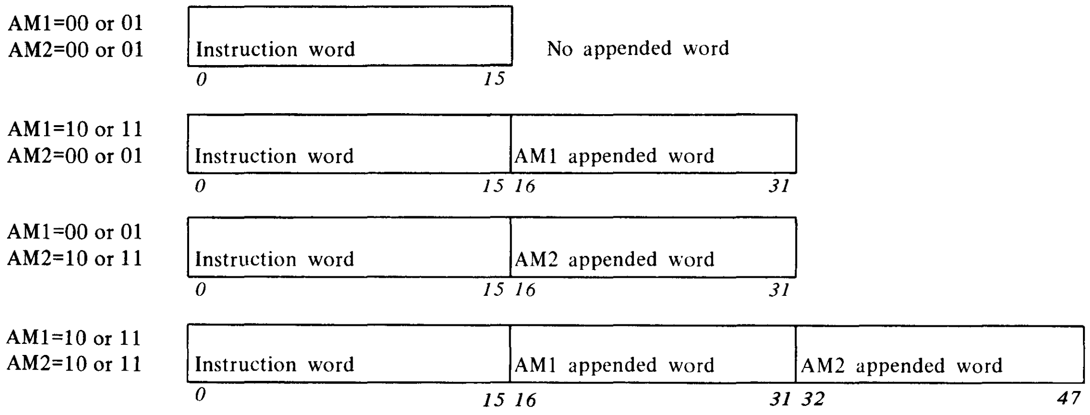
- Two-word instructions that contain a single AM field become three words in length if AM is equal to 10 or 11. The AM word is appended to the first instruction word. The *data* or *immediate* field then becomes the third word of the instruction.

Example:



- One-word instructions that contain two AM fields (AM1 and AM2) may be one, two, or three words in length depending on the values of AM1 and AM2. The AM1 word is appended first, then the AM2 word is appended.

Example:



Processor State Control

The processor is always in one of the following mutually exclusive states:

- Power off
- Stop
- Load
- Wait
- Run – when in run state, programs can be executed in either:
  - Supervisor state or
  - Problem state

Stop State

The stop state is entered when:

1. The Stop key on the programmer console is pressed.
2. The STOP instruction is executed and the mode switch on the basic console is in the Diagnostic position.
3. An address-compare occurs and the rate control on the programmer console is in the Stop on Address position.
4. An instruction has completed execution and the rate control on the programmer console is in the Instruction Step position.
5. An error occurs and the error control on the programmer console is in the Stop on Error position.
6. The Reset key on the programmer console is pressed.
7. Power-on reset occurs. For conditions 1–6, the display buffer contains the contents of the IAR.

Note. Any manual entry into Stop State is via the programmer console. The STOP instruction performs no operation if the programmer console is not installed.

While the processor is in the stop state: (1) the Stop light on the programmer console is on, (2) the functions provided on the console can be activated, and (3) no interrupt requests can be accepted by the processor.

If, when accessing main storage through the console while in stop state, a check condition arises:

1. Program check is suppressed.
2. The PSW bit(s) is set.
3. The check light is turned on.
4. The Display Register will be set to a default value of 0025.

The processor exits the stop state when:

1. The Load key on the basic console is pressed.
2. The Start key on the programmer console is pressed. When the Start key is pressed, the processor returns to the state that was exited before entering stop state. If the run state is entered, one instruction is executed before interrupts are accepted by the processor. If the stop state was entered because of a reset (power-on reset or reset key), pressing the Start key causes program execution to begin on level zero with the instruction in location zero of main storage. If the stop state was entered because of an error, with the Stop on Error switch turned on, (1) a system reset clears the error condition, or (2) pressing the Start key allows the error interrupt to be handled as if the Stop on Error switch were not on. For more information about system reset, see *State of Processor Following a Reset.*

Wait State

The processor enters wait state when: (1) Level Exit (LEX) instruction is executed and no other level is pending, or (2) a Set Level Block (SELB) instruction is executed that sets the current in-process bit off and no level is pending. While the processor is in the wait state, (1) the Wait light on the basic console is on and (2) interrupts can be accepted under control of the system mask register and the summary mask as defined by the LSR of the last active level. The processor exits the wait state when:

1. The Stop key on the programmer console is pressed.
2. The Reset key on the programmer console is pressed.
3. An I/O interrupt is accepted (the level must be enabled by the mask register).
4. A class interrupt occurs. (See Class Interrupts in Chapter 3.)

Load State

The processor enters the load state when initial program load (IPL) begins. This occurs:

1. When the Load key on the basic console is pressed.
2. After a power-on reset if the Mode switch is in the Auto IPL position.
3. A signal from a host system.

While the processor is in load state, the Load light on the basic console is on.

The processor exits the load state and enters the run state upon successful completion of the IPL. See *Initial Program Load (IPL).*

Run State

The processor enters the run state when not in the stop, wait, or load state. Run state is entered:

1. From load state upon successful completion of IPL.
2. From wait state when an interrupt is accepted.
3. From stop state when the start key is pressed. (See *Stop State.*)

The processor exits run state when entering stop, wait, or load states as previously described.

Supervisor State and Problem State

While in run state, instructions can be executed in either supervisor state or problem state. This is determined by bit 8 of the level status register (LSR):

State	*LSR bit 8*
Supervisor	1
Problem	0

Supervisor and problem states are discussed in the following sections.

Supervisor State. The processor enters supervisor state when:

1. A class interrupt occurs. This type of interrupt is caused by the following:
   a. Machine check condition
   b. Program check condition
   c. Power/Thermal warning
   d. Supervisor Call (SVC) instruction
   e. Soft exception trap condition
   f. Trace bit (LSR bit 10) set to one
   g. Console Interrupt key on the programmer console.
2. An I/O interrupt is accepted.
3. After initial program load (IPL) has completed.

Class interrupts and I/O interrupts are discussed in Chapter 3. Initial program load is discussed in a subsequent section of this chapter.

When the processor is in supervisor state, the full instruction set may be executed. The following *privileged instructions* may only be executed in supervisor state:

Copy Console Data Buffer (CPCON) Note 1
Copy Current Level (CPCL)
Copy In-Process Flags (CPIPF)
Copy Interrupt Mask Register (CPIMR)
Copy Level Status Block (CPLB)
Copy Processor Status and Reset (CPPSR)
Diagnose (DIAG)
Disable (DIS)
Enable (EN)
Level Exit (LEX)
Operate I/O (IO)
Set Console Data Lights (SECON) Note 2
Set Interrupt Mask Register (SEIMR)
Set Level Status Block (SELB)

Notes.
1. The resultant data is unpredictable if the programmer console feature is not installed.
2. Performs no operation if the programmer console feature is not installed.

Problem State. This is a state that does not allow the processor to execute the privileged instructions. The processor enters the problem state when the supervisor state bit (LSR bit 8) is turned off. This can be accomplished with a Set Level Status Block (SELB) instruction. This instruction can change the contents of the registers for a selected processor level.

While the processor is in problem state, privileged instructions cannot be executed. If a privileged instruction execution is attempted, the instruction is suppressed and a program check class interrupt occurs, with privilege violate (bit 2) set in the processor status word.

State of Processor Following a Reset

The term *reset* used in the following sections denotes the reset action that occurs during:

1. Power-on reset
2. Initial program load (IPL) reset
3. System reset initiated by pressing the Reset key on the programmer console

The following registers and conditions *are not initialized* by a reset and require program or operator action before they become valid:

- Console data buffer (Programmer Console Feature)
- General registers
- IAR on levels 1–3
- Main storage (above 16K)
- Address Compare Register

The following registers and conditions *are initialized* by a reset:

- Level Zero Indicator (turned on)
- CIAR – set to zeros
- IAR on level zero – set to zeros
- Mask register – set to ones (all levels enabled)
- LSR on level zero
  - Indicators – set to zeros
  - Supervisor state (bit 8) – set on
  - In-process (bit 9) – set on
  - Trace (bit 10) – set to zero (disabled)
  - Summary mask (bit 11) – set on (enabled)
  - All other bits – set to zeros
- PSW – set to zeros except as noted
  - Auto-IPL (bit 13) – set to zero unless the reset was caused by an Auto-IPL
  - Power/Thermal (bit 15) reflects the status of the power/thermal condition
- LSR on levels 1–3 – set to zero
- SAR – set to zeros
- Display buffer – set to all ones by Power-on Reset only (Programmer Console Feature)
- Main storage (0–16K, verified good parity Power-on Reset only)
- Check Restart (Note 1)
- Instruction Step (Note 1)
- Stop on Address (Note 1)
- Stop on Error (Note 1)

Note 1. This condition is reset by a Power-on Reset.

Initial Program Load (IPL)

An initial program load function is provided to (1) read an IPL record (set of instructions) from an external storage media, and (2) automatically execute a start-up program. An IPL record is read into storage from a local I/O device or host system. The I/O attachments for the desired IPL sources are prewired at installation time. Two local sources, primary and alternate, can be wired and either can be selected by using the IPL Source switch on the console.

IPL can be started by three methods:

1. Manually, by pressing the Load key on the console.
2. Automatically, after a power-on condition.
3. Automatically, when a signal is received from a host system. The host system can be connected through a communications adapter.

The automatic power-on IPL is selected by a mode switch on the console. When the Mode switch is in the Auto-IPL position, IPL occurs whenever power turns on (either initially or after a power failure). Power must be good to all attachments before the IPL sequence begins. Auto IPL is useful for unattended systems. A manual IPL can be initiated at any time by pressing the load key on the console (even when in run state). The mode switch has no effect on the manual IPL. For Auto-IPL and manual IPL, the local IPL source (primary or alternate) is selected. IPL from a host system can occur at any time and is initiated by the host system. The IPL record is transferred through the host-system device; for example, the communications adapter. When an auto-IPL occurs, bit 13 of the PSW is turned on to indicate the condition to the software. When a manual or host-system IPL occurs, this bit is set to zero.

During IPL main storage is loaded starting at location zero. The length of the IPL record depends on the media used by the IPL source.

Upon successful completion of an IPL, the processor enters supervisor state and begins execution on priority level zero. The summary mask is enabled and all priority interrupt levels in the mask register are enabled. The first instruction to be executed is at main storage location zero. The IPL source has a pending interrupt request on level zero. The system program must:

1. Perform housekeeping; for example, load vector table addresses in the reserved area of storage (see *Automatic Interrupt Branching* in Chapter 3).
2. Issue a Level Exit (LEX) instruction. This allows the processor to accept the interrupt from the IPL source. When the interrupt is accepted, a forced branch is taken using the device-address vector table. The vector table entry is determined by the device address of the IPL source and results in a branch to the proper program routine for handling the interrupt. The device address of the IPL source is set into bits 8–15 of register 7 on level zero. Condition code 3, device end, is reported by the IPL source. For additional information, see *I/O Interrupts* in Chapter 3.

A system reset always occurs prior to an IPL. However, if any errors occur during the IPL, the results are unpredictable.

Sequential Instruction Execution

Normally, the operation of the processor is controlled by instructions taken in sequence. An instruction is fetched from the main storage location specified in the instruction address register (IAR). The instruction address in the IAR is then increased by the number of bytes in the instruction just fetched. The IAR now contains the address of the next sequential instruction. After the current instruction is executed, the same steps are repeated using the updated address in the IAR.

A change from sequential operation can be caused by branching, jumping, interrupts, level switching, or manual intervention.

Jumping and Branching

The normal sequential execution of instructions is changed when reference is made to a subroutine; when a two-way choice is encountered; or when a segment of coding, such as a loop, is to be repeated. All of these tasks can be accomplished with branching and jumping instructions. Provision is also made for subroutine linkage, permitting not only the introduction of a new instruction address, but also the preservation of the return address and associated information.

The conditional branch and jump instructions are used to test the indicators in the LSR. These indicators are set as the result of I/O operations and most arithmetic or logical operations. Single or multiple indicators are tested as determined by the value in a three-bit field within the instruction. Refer to: (1) *Indicators* and (2) *Testing Indicators with Conditional Branch and Jump Instructions.*

Jumping

Jump instructions are used to specify a new instruction address relative to the address in the IAR. The new address must be within -256 to +254 bytes of the byte following the jump instruction.

Note. The jump instruction contains a word displacement that is converted to a byte displacement when the instruction is executed. However, when using the assembler, the programmer specifies a byte value that is converted to a word displacement by the assembler.

Branching

Branch instructions are used to specify a new full-width 16-bit address. A 16-bit value, range 0 to 65535, is contained in the second word of the instruction or in a register. The value in the second word can be used as the effective branch address or added to the contents of a base register to form an effective address. (See *Base Register Storage Address* in this chapter.)

Level Switching and Interrupts

The processor can execute programs on four different interrupt priority levels. These levels, listed in priority sequence, are numbered 0, 1, 2, and 3 with level 0 having highest priority. The processor switches from one level to another in two ways:

1. Automatically, when an interrupt request is accepted from an I/O device operating on a higher priority level than the current level.
2. Under program control, by using the Set Level Block (SELB) instruction.

Both types of level switching are discussed in detail in Chapter 3. *Class Interrupts* and *Interrupt Masking Facilities* are also discussed in Chapter 3.

Stack Operations

The processing unit provides two types of stacking facilities. Each facility is briefly described in this section. Additional information appears in subsequent sections. The two types of stacking facilities are:

1. *Data Stacking.* This facility provides an efficient and simple way to handle last-in first-out (LIFO) queues of data items and/or parameters in main storage. The data items or parameters are called stack elements. For a given queue (or stack), each element is one, two, or four bytes wide. Instructions for each element size (byte, word, or doubleword) are provided to:
   a. Push an element into a stack (register to storage).
   b. Pop an element from a stack (storage to register).
2. *Linkage stacking.* This facility provides an easy method for linking subroutines to a calling program. A word stack is used for saving and restoring the status of general registers and for allocating dynamic work areas. The Store Multiple (STM) instruction stores the contents of the registers into the stack and reserves a designated number of bytes in the stack as a work area. The Load Multiple and Branch (LMB) instruction reloads the registers, releases the stack elements, and causes a branch via register 7 back to the calling program.

Data Stacking Description

Any contiguous area of main storage can be defined as a stack. Each stack is defined by a stack control block. Figure 2-4 shows a data stack and its associated *stack control block.* Stack control blocks must be aligned on a word boundary.

The words in the stack control block are used as follows:

High Limit Address (HLA). This word contains the address of the first byte beyond the area being used for the stack. All data in the stack has a lower address than the contents of the HLA. Note that the HLA points to the first byte beyond the bottom of an empty stack.

Low Limit Address (LLA). This word designates the lowest storage location that can be used for a stack element. Note that the LLA points to the top of a stack.

Top Element Address (TEA). This word points to the stack element that is currently on top of the stack. For empty stacks, the TEA points to the same location as the high limit address (HLA).

Note. For word stacks or double word stacks, the HLA, LLA, and TEA must all contain an even address to ensure data alignment on a word boundary.

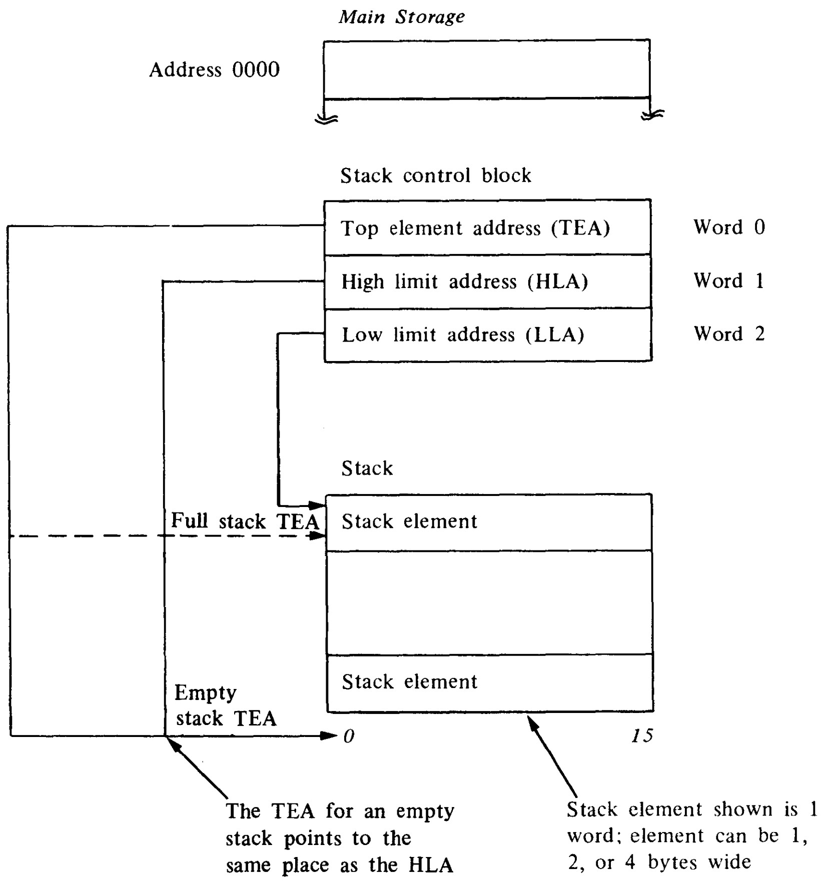
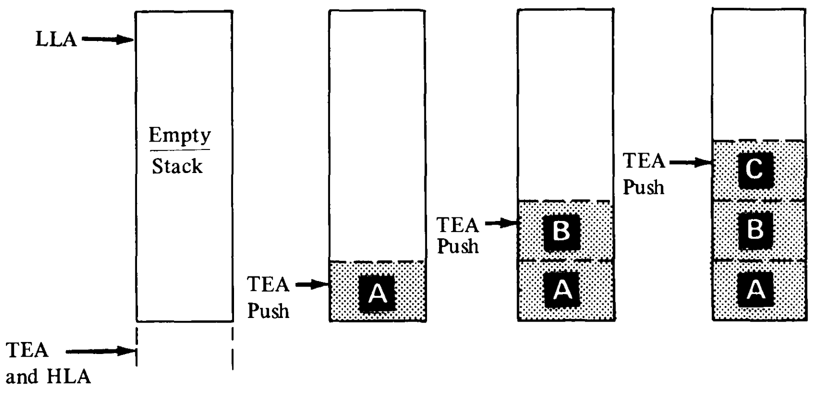


Figure 2-4. Relationship of stack control block to data stack

Push Operation. When a new element is pushed into a stack, the address value in the TEA is decremented by the length of the element (one, two, or four bytes) and compared against the LLA. If the TEA is less than the LLA, a stack overflow exists. A soft exception trap interrupt occurs with *stack exception* set in the PSW. The TEA is unchanged. If the stack does not overflow, the TEA is updated and the new element is moved to the topic location defined by the TEA.

The following diagram shows how elements are pushed into a stack. Note that each push operation always places an element at a lower address in the stack than the preceding element.

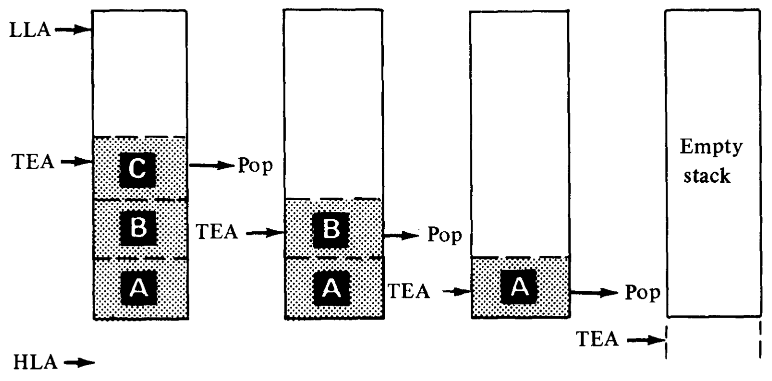


Refer to Chapter 6 for descriptions of the following instructions:

- Push Byte (PSB)
- Push Word (PSW)
- Push Doubleword (PSD)

Pop Operation. When an element is popped from a stack, the TEA is compared against the HLA. If it is equal to or greater than the HLA, an underflow condition exists. A soft exception trap interrupt occurs with *stack exception* set in the PSW. If the stack does not underflow, the stack element defined by the TEA is moved to the specified register and the TEA is incremented by the length of the element.

The following diagram shows how elements are popped from a stack.



Refer to Chapter 6 for descriptions of the following instructions:

- Pop Byte (PB)
- Pop Word (PW)
- Pop Doubleword (PD)

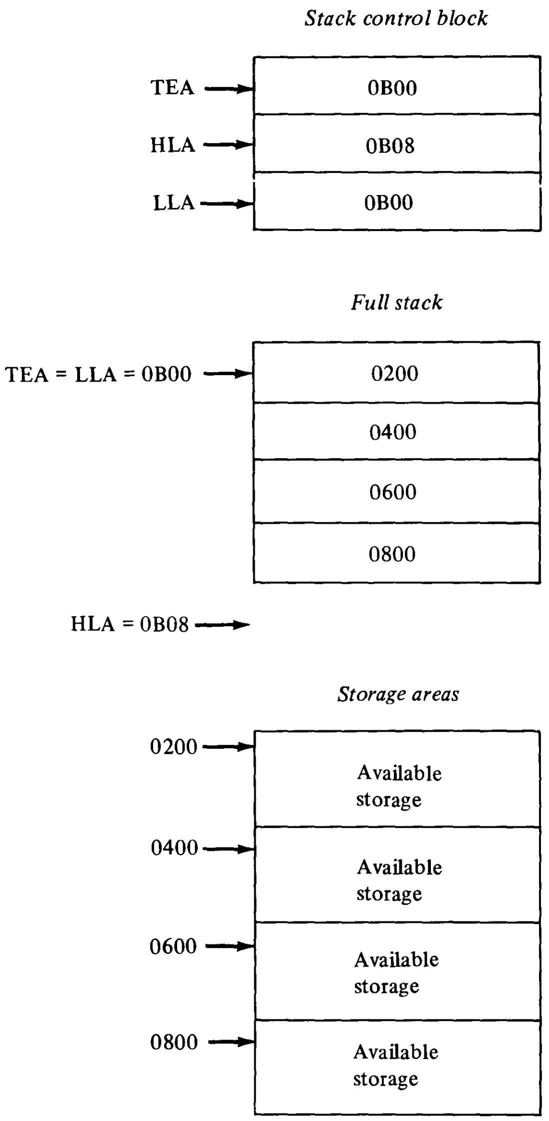
Data Stacking Example – Allocating Fixed Storage Areas

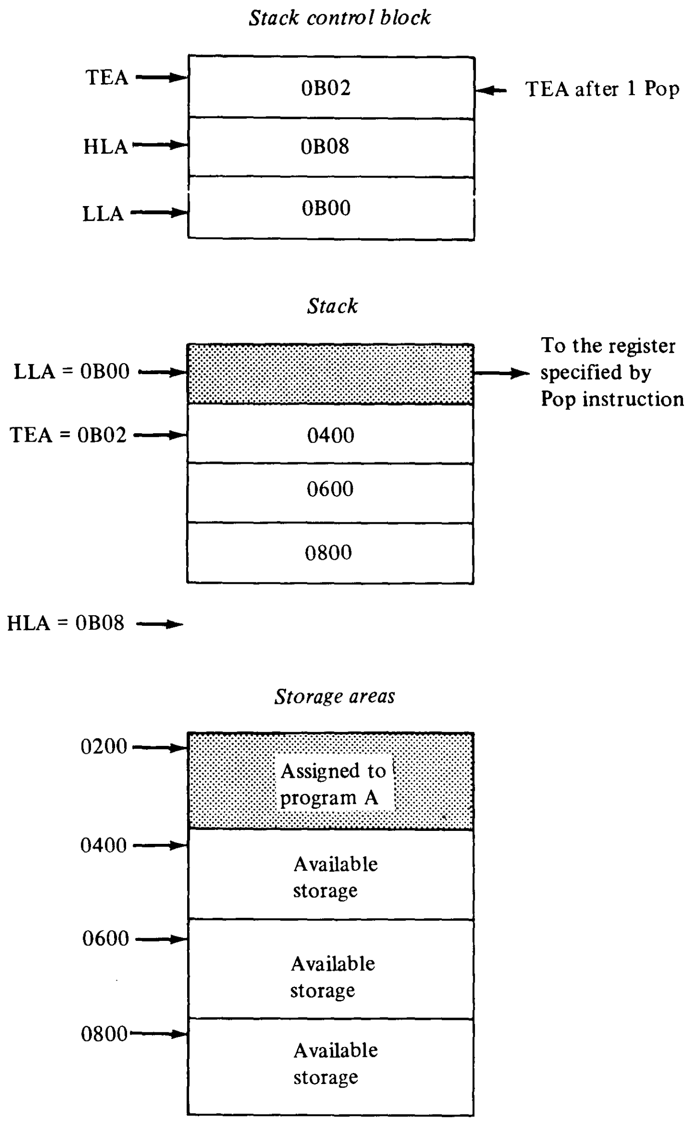
Many programs require temporary main storage work areas. It is very useful to be able to dynamically assign such work-area storage to a program only when that storage is needed. Conversely, when work-area storage is no longer needed by a program, it is desirable to free that resource so it may be used by other programs. Use of the stacking mechanism can assist in the programming of the dynamic storage management function.

The following is an example of how storage areas could be allocated using the stacking mechanism.

A stack is initialized with addresses that point to a fixed area of storage. Each element in the stack represents the starting address of a block of storage consisting of 512 bytes; e.g., addresses 0200 through 03FF. As storage is needed, the starting address for a block of storage is popped from the stack. When the block of storage is no longer needed, the starting address is pushed back into the stack.

The stack control block, stack, and storage areas appear initially as follows:



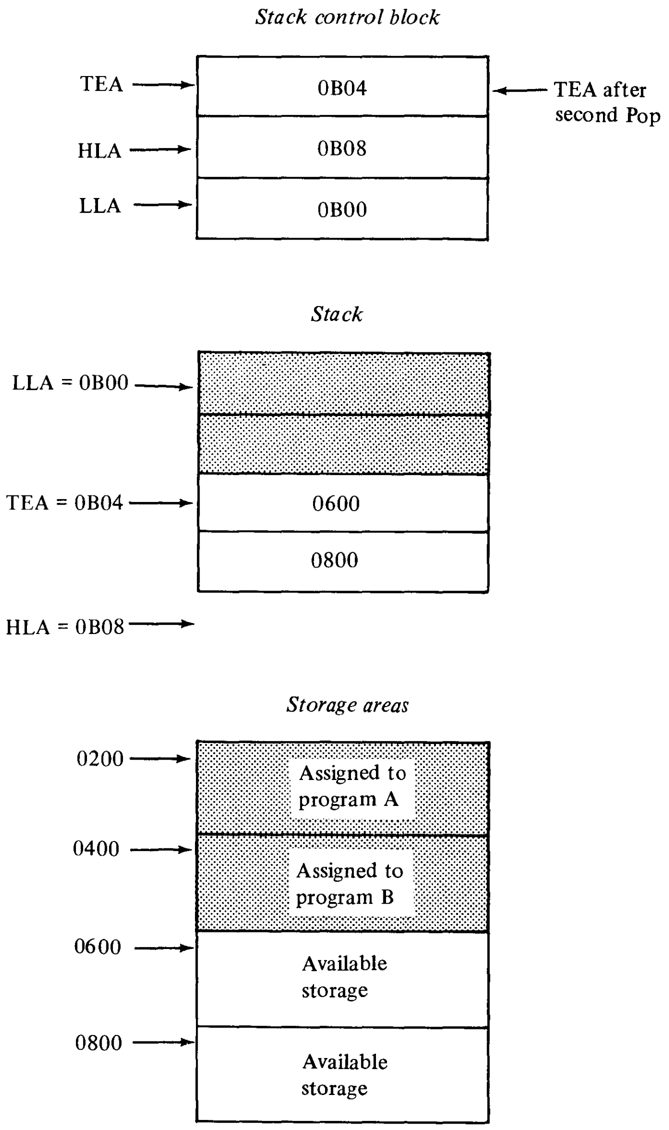


Notice that each stack element is one word long; addresses of storage areas are the stack elements; the TEA points to the lowest location of the last element because the initialized stack is *full*. Contrast this with an empty stack, in which the TEA points to the same location as the HLA.

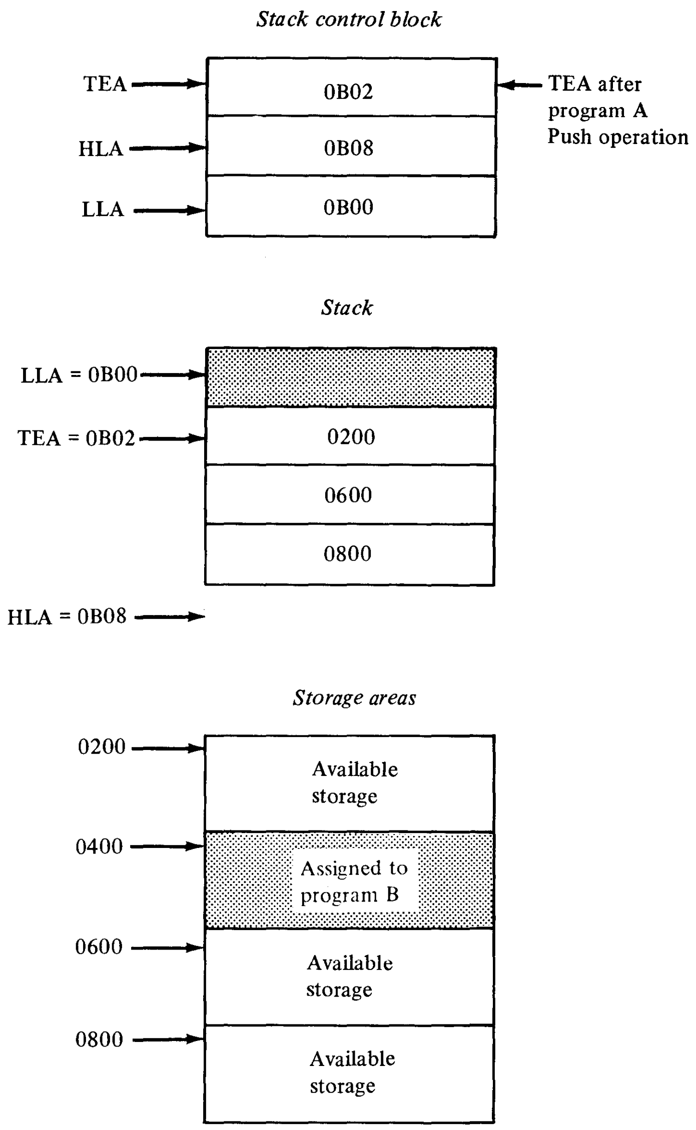
Now assume that program A requires a block of storage. Program A (or a storage management function at the request of program A) issues a pop word instruction against the stack control block. The TEA is updated as follows:

The word element popped is placed in the register specified by the pop word instruction executed by program A. This is the address of the 512-byte storage area beginning at address 0200.

At this time, assume that program B (operating on a different hardware level than program A) also requires a storage area. It too executes a pop word instruction against the stack. The next element is moved to the register specified and points to the next available storage area and the TEA is updated:



Now, before any further requests occur, program A terminates its need for a work area. Program A then issues a push word instruction against the stack and returns the address of the area it was using for use by other programs:



A similar operation will be performed by program B when it releases its storage to the stack, popping address 0400 into location 0B00. While the addresses are obviously shuffled in the stack (from the values initially established), this presents no problem since each program requires only an area of storage – it is not important where that area is located.

Linkage Stacking Description

As previously described a word-stack mechanism may be used for subroutine linkage. This mechanism saves and restores registers and allocates dynamic work areas.

The letters in the following description correspond to the letters in Figure 2-5.

The Store Multiple (STM) instruction specifies:

A Stack control block address

B Limit register (RL) number

C Number (N) of words to allocate for work areas

When the STM instruction is executed, the allocate value (N) plus the number of registers saved plus one control word is the requested block size in words. This times 2 is the size in bytes. The block size is used to decrement the TEA before an overflow check is made. If no overflow occurs the operation proceeds. The link register (R7) and register 0 through the specified limit register (RL) are saved sequentially in the stack. If register 7 is specified as the limit register, only register 7 is stored in the stack. The dynamic work space is allocated and a pointer to the work area is returned in register RL. If no work area is specified, the returned pointer contains the location of R7 in the stack. The values of RL and N are saved as an entry in the stack. The TEA is updated to point to the new top of the stack location.

When a Load Multiple and Branch (LMB) instruction is executed, the values of RL and N are retrieved from the stack and an underflow check is made. The value of RL controls the reloading of the registers; the values of RL and N are used to restore the stack pointer (TEA) to its former status. The contents of register 7 are then loaded into the instruction address register, returning program control to the calling routine.

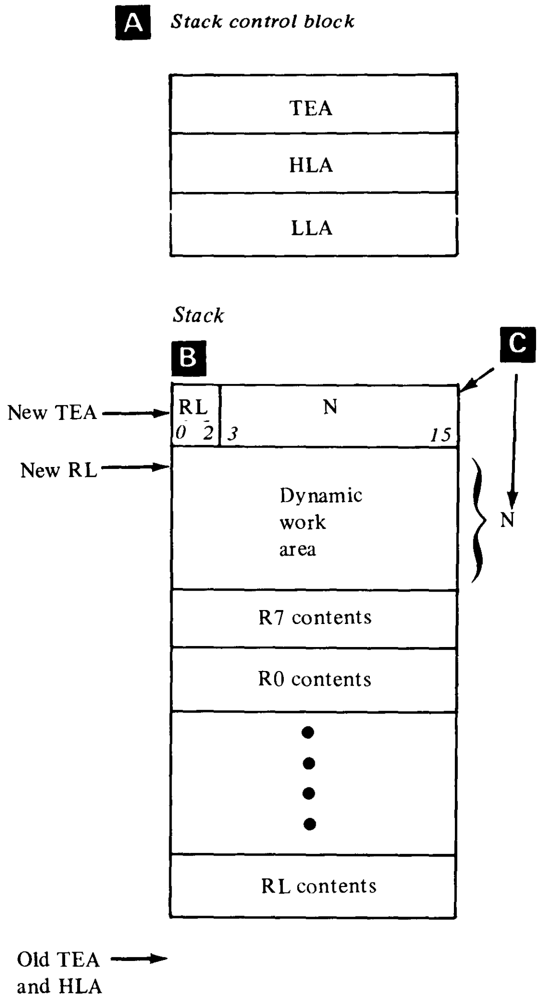


Figure 2-5. Word stack for subroutine linkage

Linkage Stacking Example – Reenterable Subroutine

A subroutine may be used by programs that operate on different interrupt levels. Rather than providing copies of the subroutine, one copy for each program that needs it, the subroutine can be made reenterable. Here, only one copy of the subroutine is provided; the single copy is used by all requesting programs. Two items must be considered in the reenterable subroutine code:

- Saving the register contents of each calling program. The subroutine is then free to use the same registers, restoring their contents to the calling-program's values just prior to returning to the calling program.
- Preserving the applicable variable data (generated by the subroutine) that is related to each call of the subroutine. That is, data associated with one call must not be disturbed when subroutine execution is restarted due to another call from a higher priority program.

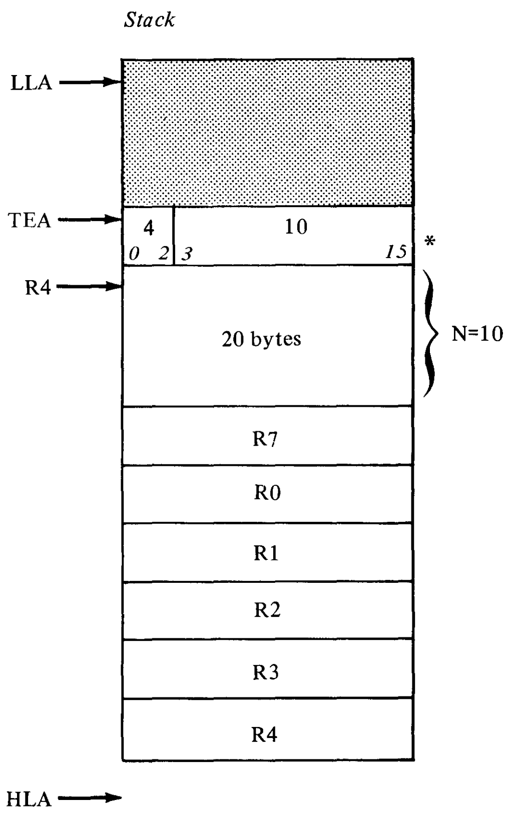
The stacking mechanism, by means of the STM and LMB instructions, handles the two items just mentioned. As an example, operation could proceed as follows:

1. Program A calls the subroutine by means of a branch and link instruction (return address is in R7).

   BAL SUBRT,7
2. The subroutine, in this example, uses registers R3 and R4 during its execution. The subroutine receives (from program A) a parameter list address in R0 and the address of the stack control block in R1. Also, the subroutine requires 20 bytes of work space. Thus, the subroutine executes, upon entry, the following store multiple instruction:

   SUBRT STM 4,(1),20

   After execution of the STM, the stack contains the following:



   * The last word contains a value that specifies the last register stored (e.g., R4 in this example) and the size of the dynamic work area (in words).

   R4 (the last register stored in the stack) is automatically loaded, during the STM operation, with the address of the work area to be used by the subroutine to hold its work data.
3. When subroutine processing for this call is completed, the subroutine executes a single load multiple instruction in order to reload the registers and return (via R7) to the calling program:

   LMB (1)

   If a second call to the subroutine has occurred prior to execution of the LMB, action similar to that just stated would occur again. However, another stack area would be used. Then, when subroutine execution is completed for the second call, and all higher priority interrupt level processing is completed, a return would be made to the interrupted subroutine for completion of processing for the first call.

Thus, multiple calls to a single subroutine are processed without interfering with the integrity of data associated with any other call to the subroutine.

Chapter 3. Interrupts and Level Switching

Introduction

Efficient operation of a central processor depends on prompt response to I/O device service requests. This is accomplished by an interrupt scheme that stops the current processor operation, branches to a device service routine, handles device service, then returns to continue the interrupted operation. One processor can control many I/O devices; therefore, an interrupt priority is established to handle the more important operations before those of lesser importance. Certain error or exception conditions (such as a machine check) also cause interrupts. These are called class interrupts and are processed in a manner similar to I/O interrupts. Both I/O and class interrupts are explained further in the following sections.

Interrupt priority is established by four priority levels of processing. These levels, listed in priority sequence, are numbered 0, 1, 2, and 3 with level 0 having highest priority. Interrupt levels are assigned to I/O devices via program control. This provides flexibility for reassigning device priority as the application changes.

Each of the four priority levels has its own set of registers. These consist of a level status register (LSR), eight general registers (R0–R7), and an instruction address register (IAR). Information pertaining to a level is automatically preserved in these hardware registers when an interrupt occurs.

Processor level switching, under program control, may be accomplished by use of the Set Level Block (SELB) instruction. Details of this method are presented in a separate section of this chapter.

I/O and class interrupts cause automatic branching to a service routine. Fixed locations in main storage are reserved for branch addresses or pointers which are referenced during interrupt processing. This storage allocation is shown in the section *Automatic Interrupt Branching* in this chapter.

Interrupt masking facilities provide additional program control over the four priority levels. System and level masking are controlled by the *Summary Mask* and the *Interrupt Level Mask Register*. Device masking is controlled by the *Device Mask*. Manipulation of the mask bits can enable or disable interrupts on all levels, a specific level, or for a specific device. See *Interrupt Masking Facilities* in this chapter.

Interrupt Scheme

As previously stated, four priority interrupt levels exist. Each I/O device is assigned to a level, dependent on the application. When an interrupt on a given level is accepted, that level remains active until (1) a Level Exit (LEX) instruction is executed, (2) a Set Level Block (SELB) instruction causes a level switch, or (3) a higher priority interrupt is accepted. In the first two cases, the active level at the time is cleared. In the latter case, the processor switches to the higher level, completes execution (including a LEX instruction), then automatically returns to the interrupted-from level. This automatic return can be delayed by other higher priority interrupts.

If an interrupt request is pending on the currently active level, it will not be accepted until after execution of a LEX instruction by the current program. If no other level of interrupt is pending when a Level Exit instruction is executed, the processor enters the wait state. In the wait state no processing is performed, but the processor can accept interrupts that are expected to occur. See Figure 3-1.

Class interrupts do not change priority levels. They are processed at the currently active level. If the processor is in the wait state when a class interrupt occurs, priority level 0 is used to process the interrupt.

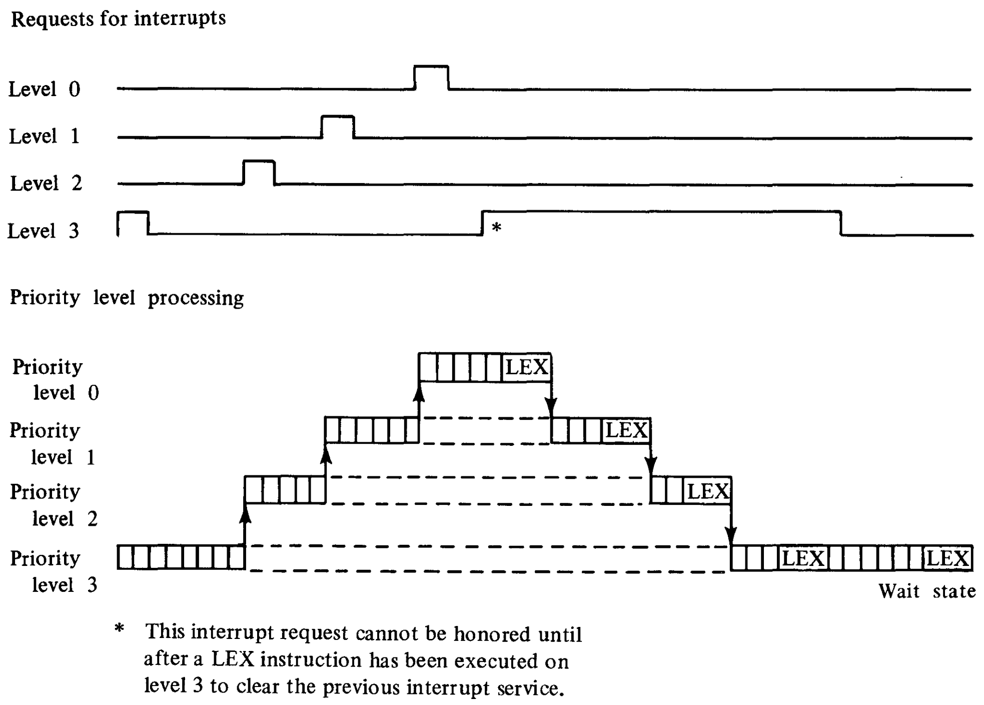

Requests for interrupts
Level 0
Level 1
Level 2
Level 3
*
Priority level processing
Priority level 0
Priority level 1
Priority level 2
Priority level 3
LEX
LEX
LEX
LEX
LEX
Wait state
* This interrupt request cannot be honored until after a LEX instruction has been executed on level 3 to clear the previous interrupt service.

Figure 3-1. Interrupt priority scheme

Automatic Interrupt Branching

Hardware processing of an interrupt includes automatic branching to a service routine. The processor uses a reserved storage area for branch information. The reserved area begins at main storage address 0000. The total size of the area depends on the number of interrupting devices attached. One word (two bytes) is reserved for each interrupting device and is related to a particular device by the device address. For example: device 00 causes a reference to location 0030, device 01 to location 0032, and so on. The device area begins at address 0030 (Hex); the reserved area is 0000 through 022F (Hex) if 256 devices (maximum number) are attached. These storage locations and contents are shown in Figure 3-2.

Main storage address (Hex)	Contents of word
022E	Device FF DDB pointer
0032	Device 01 DDB pointer
0030	Device 00 DDB pointer
002E	Reserved
002C	Reserved
002A	Reserved
0028	Reserved
0026	Reserved
0024	Reserved
0022	Soft exception trap SIA
0020	Soft exception trap LSB pointer
001E	Console interrupt SIA
001C	Console interrupt LSB pointer
001A	Trace SIA
0018	Trace LSB pointer
0016	Power failure SIA
0014	Power failure LSB pointer
0012	SVC SIA
0010	SVC LSB pointer
000E	Program check SIA
000C	Program check LSB pointer
000A	Machine check SIA
0008	Machine check LSB pointer
0006	Reserved
0004	Reserved
0002	Restart instruction word 2
0000	Restart instruction word 1

Figure 3-2. Reserved storage locations

The reserved storage locations are described as follows:

Storage Location (Hex)	*Contents*
0000–0003	Restart instruction. Following IPL a forced branch is made to location 0000.
0004–0005	Reserved.
0006–0007	Reserved.
0008–0023	Addresses used for class interrupts. The Level Status Block (LSB) pointer is the first address of an area where a level status block will be stored. The Start Instruction Address (SIA) points to the first instruction of a service routine.
0024–002F	Reserved.
0030–022F	Addresses used for I/O interrupts. The Device Data Block (DDB) pointer is the address of the first word of a device data block. This word is used to obtain the start instruction address for the service routine. See *I/O Interrupts* in this chapter.

Note. The area reserved for I/O devices varies in size depending on the number of devices. The device address determines the fixed location to be accessed. For example: Interrupts for device 01 always vector to main storage address 0032.

A device address is established by installing the appropriate connectors on the I/O feature card for the device.

I/O Interrupts

Prepare I/O Device for Interrupt

I/O device interrupt parameters are established via program control. The Operate I/O (IO) instruction initiates the device operation and in conjunction with the "Prepare" I/O command tells the device:

1. If the device can interrupt.
2. What priority level to use for interrupts. See Chapter 6 *Instructions* and Chapter 4 *Input/Output Operations* for details of the Operate I/O instruction.

Execution of the Prepare command transfers a word to the addressed device that controls its interrupt parameters. This word has the format:

Zero	Level	I
0 — 10	11 — 14	15

Bits	*Contents*
0–10	Set to zeros.
11–14	*Level.* A four-bit encoded field that assigns an interrupt priority level to the device (see note). *Example:* 0000 – level 0, 0001 – level 1, 0010 – level 2, 0011 – level 3.
15	*Device mask or I-bit.* This bit sets the interrupt mask in the device. When set to one, the device can interrupt. When set to zero, the device cannot request an interrupt.

Note. The 4953 Processor does not recognize priority levels other than zero through three; therefore, bits 11 and 12 must always be set to zero or the interrupt is lost.

An interrupting device is always able to accept and execute a Prepare command, even if it is presently busy or has an interrupt request pending from a previous command. This allows the software to change the device mask and interrupt level at any time. Any pending interrupt request is then serviced on the new interrupt level.

Present and Accept I/O Interrupt

The I/O device presents an interrupt request on its assigned priority level. This request is applied to the interrupt algorithm for acceptance determination.

For an I/O interrupt to be serviced, the following conditions must exist:

1. The summary mask must be on (enabled).
2. The mask bit (Interrupt Level Mask Register) for the interrupting level must be on (enabled).
3. For I/O interrupts the device must have its Device Mask bit on (enabled).
4. The interrupt request must be the highest priority of the outstanding requests and higher than the current level of the processor.
5. The processor must not be in the stop state.

Supervisor state is entered upon acceptance of all priority interrupts.

Following acceptance, the device sends an interrupt ID word and a condition code to the processor. The condition code is placed in the even, carry, and overflow indicators for the interrupted-to level. The ID word is placed into register 7 of the interrupted-to level. The interrupt ID word consists of an interrupt information byte (IIB) and the device address. Bits 0–7 of this word contain the interrupt information; bits 8–15 contain the device address. See Chapter 4 for condition codes and interrupt information byte (IIB) details. Hardware causes the following events to occur after the processor receives the interrupt ID word and the condition code (Figure 3-3):

- The processor hardware switches from the registers and status of the interrupted-from level to those of the interrupted-to level.
- The interrupt ID word is placed in register 7 of the interrupted-to level.
- The condition code is placed in LSR positions 0–2.
- Supervisor state is entered (LSR bit 8 is set to one).
- The processor executes an automatic branch.
  - The device address is used by hardware to fetch the DDB pointer from reserved storage.
  - The DDB pointer is placed in register 1 of the interrupted-to level.
  - The DDB pointer is used by hardware to fetch the start instruction pointer.
  - The Start Instruction Address (SIA) is loaded into the IAR of the interrupted-to level.
- Execution begins on the new level.

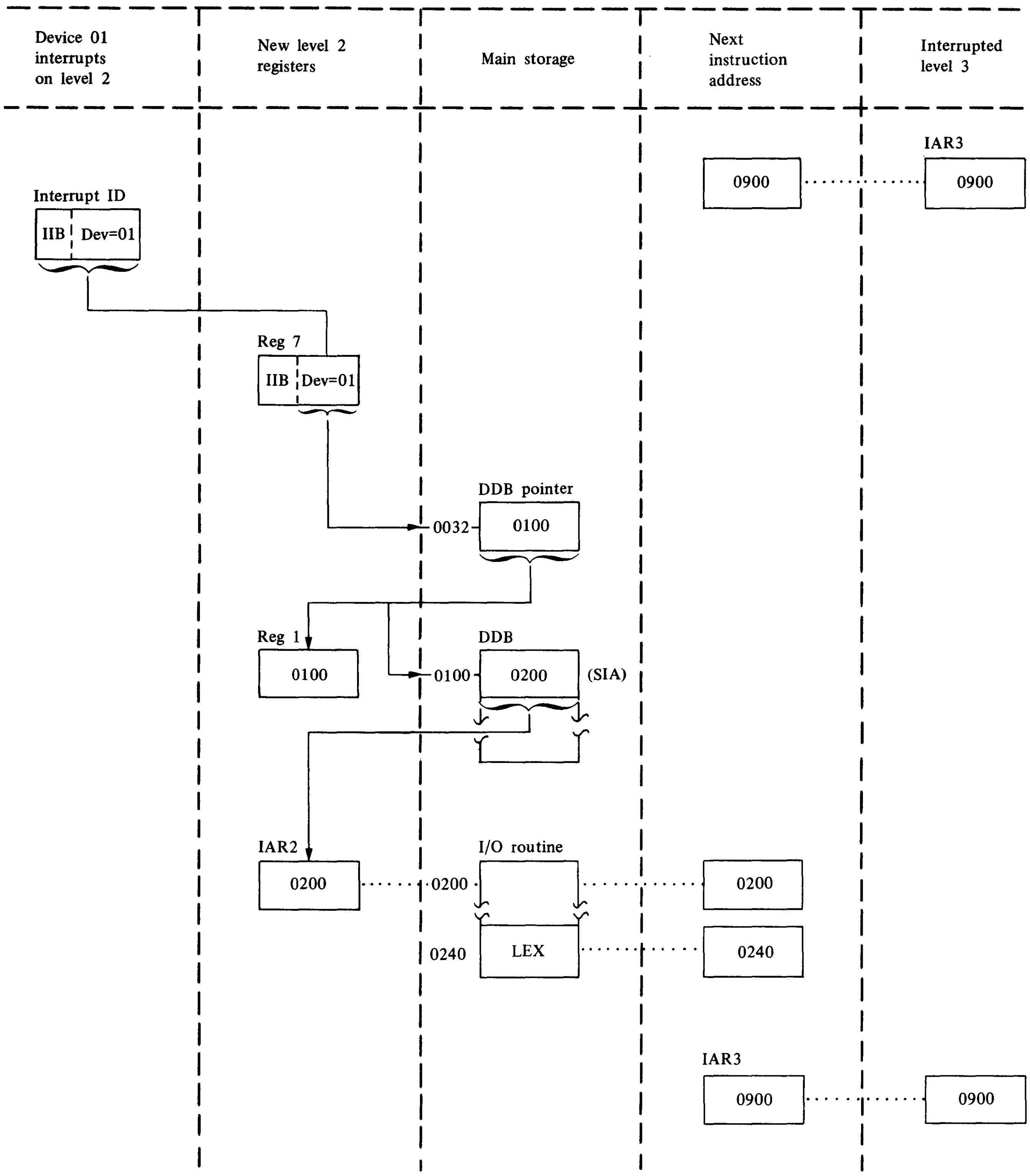


Figure 3-3. Example of I/O interrupt with automatic branching

Class Interrupts

System error or exception conditions can cause seven types of class interrupts:

1. Machine check, caused by a hardware error.
2. Program check, caused by a programming error.
3. Power/thermal warning, caused by a power or thermal irregularity.
4. Supervisor call, caused by execution of an SVC instruction.
5. Soft exception trap, caused by software.
6. Trace, caused by instruction execution (trace enabled in the current LSR).
7. Console, caused by a console interrupt when the optional programmer console is installed.

Machine check, program check, soft exception trap, and power/thermal warning are defined by bits in the processor status word. Software can refer to the processor status word for a specific condition and any related status information. See *Processor Status Word* in this chapter.

Class interrupts do not cause a change in priority level. The interrupt is serviced on the level that is active when the condition occurs. If the processor is in the wait state, the interrupt is serviced on priority level zero. Independent routines are used to handle each type of class interrupt regardless of priority level.

All class interrupts cause the processor to enter supervisor state. Refer to a subsequent section, *Present and Accept Class Interrupt*, for details of the hardware processing.

Programming Notes.

1. Two class interrupts (power/thermal warning and console) can be disabled by the summary mask.
2. If the optional programmer console is installed and Check Restart or Stop on Error are selected, machine check, power/thermal warning, and program check interrupts do not occur. See *Programmer Console Feature* in Chapter 5.

Priority of Class Interrupts

Although class interrupts are serviced on the current priority level, they are serviced according to an exception condition priority.

The following table lists the exception conditions in priority sequence with zero being the highest priority. Two exception conditions of the same priority, such as invalid storage address and specification check, may be reported to the PSW simultaneously. The table also shows the associated class interrupt vector for the exception conditions.

Priority	*Exception Condition*	*Class Interrupt Routine*
0	CPU control check I/O check	Machine check
1	Invalid function (Note 1)	Program check
2	Privilege violate	
3	Invalid function (Note 2)	
4	Not applicable on 4953 Processor	
5	Invalid storage address Specification check	
6	Storage parity	Machine check
7	Power warning Thermal warning	Power/thermal warning
8	Supervisor call	Supervisor call
9	Invalid function (Note 3)	Soft exception trap
10	Not applicable on 4953 Processor	
11	Stack exception	
12	Trace	Trace
13	Console	Console

Note 1. Caused by an illegal operation code or function combination.

Note 2. A Copy Segmentation Register (CPSR) or Set Segmentation Register (SESR) instruction is attempted. The translator feature is not available in the 4953 Processor.

Note 3. A floating-point instruction is attempted. The floating-point feature is not available on the 4953 Processor.

Present and Accept Class Interrupt

When a class interrupt occurs, it is serviced on the currently active level or on level zero (if in the wait state). Hardware processing of the interrupt causes the following:

- Register contents are saved
- Supervisor state is entered (LSR bit 8 is set to one)
- Trace is reset (LSR bit 10 is set to zero)
- Summary mask is disabled (LSR bit 11 is set to zero)
- An automatic branch is taken to a service routine.

Each type of class interrupt has an associated LSB pointer and SIA in the reserved area of main storage (see Figure 3-2). Reference is made to the reserved area to:

1. Store current level IAR, registers, and LSR into a level status block (LSB) in main storage.
2. Automatically branch to a service routine by using the start instruction address (SIA).

Note. Priority level zero is forced active when a class interrupt occurs in the wait state. The level zero LSB is stored into main storage. The in-process flag (LSR bit 9) is zero in the stored LSB.

Contents of the level status block are as follows:

Main storage address (LSB) pointer)	Instruction address register
	Zero
	Level status register
	Register 0
	Register 1
	Register 2
	Register 3
	Register 4
	Register 5
	Register 6
+14 (Hex)	Register 7
	0 15

The instruction address (contents of IAR) stored in the LSB depends on the type of class interrupt and is shown in the following chart.

Type of Class Interrupt	Contents of IAR (Stored in LSB)
Program check Soft exception trap	Address of instruction that caused the interrupt.
Supervisor call Trace Console Power/thermal warning	Address of the next instruction.
Machine check (with Sequence indicator off)	Address of instruction that caused the interrupt.
Machine check (with Sequence indicator on)	Address of instruction that was being executed at the time of the error.

Machine Check

A machine check interrupt is caused by a hardware malfunction and is considered a system-wide incident. The three types are:

1. Storage parity check (PSW bit 08)
2. CPU control check (PSW bit 10)
3. I/O check (PSW bit 11)

A level status block is stored, starting at the location in main storage designated by the machine check LSB pointer (contents of storage locations hex 0008 and 0009). The contents of the storage address register (SAR) are loaded into register seven. The machine check SIA (contents of storage locations hex 000A and 000B) is then loaded into the IAR, becoming the address of the next instruction to be fetched.

Note. When the error condition occurs, the IAR contains the true address of the first word of the instruction; it is not incremented if the error occurs in the second or third word of a long instruction.

Program Check

A program check interrupt is caused by a programming error. The types are:

1. Specification check (PSW bit 00).
2. Invalid storage address (PSW bit 01).
3. Privilege violate (PSW bit 02).
4. Invalid function (PSW bit 04).

A level status block is stored, starting at the location in main storage designated by the program check LSB pointer (contents of storage locations hex 000C and 000D). The contents of the storage address register (SAR) are loaded into register seven. The program check SIA (contents of storage locations hex 000E and 000F) is then loaded into the IAR, becoming the address of the next instruction to be fetched.

Note. A program check interrupt condition on one priority level does not affect software on other levels.

Power/Thermal Warning (PSW Bit 15)

A power/thermal warning class interrupt is initiated by:

1. A power warning signal that is generated when the power line decreases to about 85% of its rated value.
2. A thermal warning that occurs if the temperature limits inside the closure are exceeded.

In both cases, the instruction address that is stored in the LSB points to the next instruction to be executed.

A level status block is stored, starting at the location in main storage designated by the power failure LSB pointer (contents of storage locations hex 0014 and 0015). The power failure SIA (contents of storage locations hex 0016 and 0017) is then loaded into the IAR, becoming the address of the next instruction to be fetched.

A power/thermal warning interrupt can occur when the system is running or in the wait state, assuming (1) the summary mask is enabled and (2) the programmer console is not set to Check Restart or Stop on Error. These interrupts are not taken by the processor if either of the two conditions are not met.

If the optional battery backup unit is installed and a power warning occurs, PSW bit 15 remains on as long as power is supplied by the battery. If a thermal warning occurs, the processor will power down regardless of the battery backup unit. The minimum time before the processor powers down is 20 milliseconds. The IBM 4999 Battery Backup Unit is explained in a separate publication; *IBM Series/1 Battery Backup Unit Description*, GA34-0032.

Power/thermal warning interrupts are not taken by the processor until the first instruction is executed following a power-on reset, an IPL, or exit from stop state.

Note. If the processor is in the wait state when the power/thermal condition occurs:

1. The interrupt is serviced on priority level 0. The level 0 LSB is stored into main storage. Additional power/thermal interrupts are disabled at this time because the summary mask is set to zero by the class interrupt.
2. The instruction address stored in the LSB is unpredictable.

Supervisor Call

A supervisor call class interrupt is initiated by executing an SVC instruction. The SVC instruction is described in Chapter 6. A level status block is stored, starting at the main storage location designated by the supervisor call LSB pointer (contents of storage locations hex 0010 and 0011). The supervisor call SIA (contents of storage locations 0012 and 0013) is then loaded into the IAR, becoming the address of the next instruction to be fetched.

Soft Exception Trap

A soft exception trap interrupt is caused by software. The types are:

1. Invalid function (PSW bit 4)
2. Stack exception (PSW bit 6)

These exception conditions may be handled by software; therefore, they do not constitute an error condition.

A level status block is stored, starting at the location in main storage designated by the soft-exception-trap LSB pointer (contents of storage locations hex 0020 and 0021). The contents of the storage address register (SAR) are loaded into register seven. The soft-exception-trap SIA (contents of storage locations hex 0022 and 0023) is then loaded into the IAR, becoming the address of the next instruction to be fetched.

Note. The contents of register seven are unpredictable.

Trace

The trace class interrupt provides an instruction trace facility for software debugging. Instruction tracing may occur on any priority level, and is enabled by the trace bit (LSR bit 10). The tracing occurs when bit 10 of the current LSR is set to one. When trace is enabled, a trace class interrupt occurs at the beginning of each instruction. A level status block is stored, starting at the location in main storage designated by the trace LSB pointer (contents of storage locations hex 0018 and 0019). The trace SIA (contents of storage locations hex 001A and 001B) is then loaded into the IAR, becoming the address of the next instruction to be fetched.

Note. After the LSB is stored, and before the next instruction is fetched, supervisor state is set on (LSR bit 8), trace is turned off (LSR bit 10), and the summary mask is disabled (LSR bit 11).

Programming Note. When trace is enabled, a trace class interrupt occurs prior to executing each instruction. Hardware processing of the interrupt provides an automatic branch to the programmer's trace routine. To prevent retracing the same instruction, the program should exit the trace routine by using the Set Level Block (SELB) instruction with the inhibit trace (IT) bit set to one. The inhibit trace bit prevents a trace interrupt from occurring for the duration of one instruction (see SELB instruction in Chapter 6). A double trace of an instruction can also occur when the instruction is interrupted and must be reexecuted. For example: a class interrupt occurs during execution of a variable field length instruction. Under this condition, exit from the class interrupt routine should be via a SELB instruction with the inhibit trace bit set to one.

The occurrence of any class interrupt or priority interrupt causes the trace bit (LSR bit 10) to be set to zero. This action permits tracing only problem state code. If the programmer desires to trace supervisor code, he must make provisions within the service routine to enable the trace bit.

The following three conditions inhibit a trace class interrupt:

1. A Set Level Block (SELB) instruction sets the trace bit on and the in-process bit on in the LSR of a selected level lower than the current level; then, when the selected level becomes active, the first instruction executed is not preceded by a trace interrupt.
2. The programmer console is in diagnose mode and a stop instruction is encountered while tracing; then, when the Start Key is depressed, a trace interrupt does not occur prior to executing the first instruction.
3. When a level is exited by either a LEX or a SELB instruction and processing is to continue on a pending level, one instruction is executed on the pending level prior to sampling for a trace interrupt.

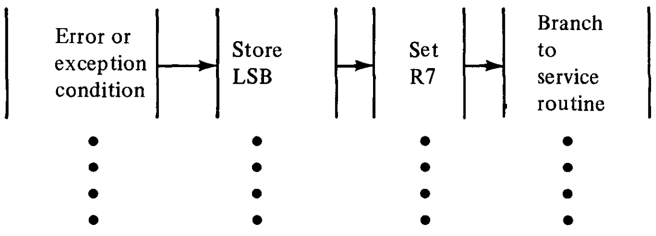
Console

A console interrupt function is provided when the optional programmer console is installed. To recognize the interrupt, the processor must have the summary mask enabled and be in the run state or wait state. A level status block is stored, starting at the main storage location designated by the console interrupt LSB pointer (contents of storage locations hex 001C and 001D). The console interrupt SIA (contents of storage locations hex 001E and 001F) is then loaded into the IAR, becoming the address of the next instruction to be fetched.

Note. If the processor is in the wait state when a console interrupt occurs, the interrupt is serviced on priority level 0.

Summary of Class Interrupts

The following chart is a summary of class interrupt processing. Each class interrupt is fully explained in separate sections of this chapter.



Class Interrupt	*LSB Pointer*	*Reg 7*	*SLA Pointer*
Machine check	0008-0009	SAR	000A-000B
Program check	000C-000D	SAR	000E-000F
Power/thermal warning	0014-0015		0016-0017
SVC	0010-0011		0012-0013
Soft exception trap	0020-0021	SAR	0022-0023
Trace	0018-0019		001A-001B
Console	001C-001D		001E-001F

Recovery from Error Conditions

Error recovery procedures, initiated by software, depend on several factors:

1. Application involved.
2. Type of error.
3. Number of recommended retries.

The error class interrupt provides an automatic branch to a service routine. This routine can interrogate the PSW for specific error and status information. The routine can then initiate corrective action or retry the failing instruction(s). If an error occurs during a priority interrupt sequence, the priority level switch is completed before the error class interrupt is processed. This facilitates automatic register retention. A reset is generated by machine check class interrupts caused by an I/O check or a CPU control check. No reset is generated by program check or power/thermal warning class interrupts. Error conditions along with error recovery information are presented in the following sections.

Program Check

A program check is caused by a programming error and initiates a program check class interrupt. Error retry depends on the application. All necessary parameters are made available for locating and, if required, correcting the invalid condition. There is no change to operands or priority level during a program check class interrupt. The stored LSB reflects conditions at the time the interrupt occurred and contains:

- The contents of all general registers.
- Status information (LSR contents).
- The address of the failing instruction (IAR contents).

The contents of the storage address register (SAR) are loaded into R7. The programmer must reference the PSW to determine the type of program check.

Storage Parity Check

A storage parity error initiates a machine check class interrupt. The error may occur when accessing a storage location that has not been validated since power on. Any retry procedure should include refreshing data in the failing location. Two unsuccessful retries are considered a permanent failure and the storage location should not be used. An IPL should be initiated.

CPU Control Check

A CPU control check occurs if hardware detects a malfunction of the processor controls. It is a machine-wide error and initiates a machine check class interrupt. A reset is generated to the channel, the I/O attachment features, and all attached I/O devices. The processor, sensor-based output points, and timer values are not reset. The generated reset should clear the error condition, but validity of any previous execution is not guaranteed. No retry is recommended. An IPL should be initiated.

I/O Check

An I/O check condition occurs if a hardware error is detected that may prevent further communication with I/O devices. A machine check class interrupt is initiated and a reset is generated to the I/O attachment features, the channel, and all I/O devices. Error recovery from an I/O check depends on the sequence indicator setting (PSW bit 12).

Sequence Indicator Set to Zero. The error occurred during an Operate I/O instruction. The address of the failing instruction (IAR contents) is available in the stored LSB. Retry should be attempted twice. After two unsuccessful retries, use of the device should be discontinued.

Sequence Indicator Set to One. The error occurred during an interrupt or cycle steal operation. The instruction address (IAR contents) stored in the LSB is not related to the error. The sequence of events leading to the I/O check is lost, along with all pending interrupt requests within the devices. Retry is not recommended.

Soft Exception Trap

A soft exception trap interrupt is the result of an exception condition that software may choose to handle dynamically. All necessary parameters are available to locate and correct the condition. The address of the instruction (IAR contents) causing the exception is preserved in the level status block in main storage. The processor is not reset. The programmer must reference the PSW to determine the soft-exception type.

Processor Status Word

The processor status word (PSW) is used to record error or exception conditions in the system that may prevent further processing. It also contains certain status flags related to error recovery. Error or exception conditions recorded in the PSW cause four of the possible seven class interrupts to occur. These are machine check, program check, soft exception trap, and power/thermal warning. See *Class Interrupts* in this chapter.

The Copy Processor Status and Reset (CPPSR) instruction can be used to examine the PSW. This instruction stores the contents of the PSW into a specified location in main storage.

The PSW is contained in a 16-bit register with the following bit representation:

Bit	*Condition*	*Class Interrupt*	*Remarks*
00	Specification check	Program check	
01	Invalid storage address	Program check	
02	Privilege violate	Program check	
03	Not used		always zero
04	Invalid function	Program check or Soft exception trap	
05	Not used		always zero
06	Stack exception	Soft exception trap	
07	Not used		always zero
08	Storage parity check	Machine check	
09	Not used		always zero
10	CPU control check	Machine check	
11	I/O check	Machine check	
12	Sequence indicator	None	Status flag
13	Auto-IPL	None	Status flag
14	Not used		always zero
15	Power/thermal warning	Power/thermal	Note 1

Note 1. The power/thermal warning class interrupt is controlled by the summary mask.

Bit 00 Specification Check. Set to one if the storage address violates the boundary requirements of the specified data type.

Bit 01 Invalid Storage Address. Set to one when an attempt is made to access a storage address outside the storage size of the system. This can occur on an instruction fetch, an operand fetch, or an operand store.

Bit 02 Privilege Violate. Set to one when a privileged instruction is attempted in the problem state (supervisor state bit in the level status register is not on).

Bit 04 Invalid Function. Set to one by one of the following conditions:

1. Attempted execution of an illegal operation code or function combination. These are:

Op code	*Function*
00111	All
01000	0001, 0010, 0011, 0101, 0110, 0111
01011	0001, 1001 (When in supervisor state)
01011	0101, 0111
01100	111
01110	11000, 11010, 11011, 11100, 11110, 11111
01111	1X1XX, 01XXX, 1X011, 10001
10110	All
11011	All
11101	1100, 1101, 1110, 1111

Note. The preceding illegal conditions cause a *program check* class interrupt to occur.

2. The processor attempts to execute reserved operation codes or function combinations. These are:

Op code	*Function*
00100	All
01011	0011, 1011 (When in supervisor state)

Note. The preceding condition causes a *soft-exception-trap* class interrupt to occur.

Bit 06 Stack Exception. Set to one when an attempt has been made to pop an operand from an empty main storage stack or push an operand into a full main storage stack. A stack exception also occurs when the stack cannot contain the number of words to be stored by a Store Muliple (STM) instruction.

Bit 08 Storage Parity. Set to one when a parity error has been detected on data being read out of storage by the processor. This error may occur when accessing a storage location that has not been validated since power on.

Bit 10 CPU Control Check. A control check will occur if no levels are active but execution is continuing. This is a machine-wide error. (See I/O check note.)

Bit 11 I/O Check. Set to one when a hardware error has occurred on the I/O interface that may prevent further communication with any I/O device. PSW bit 12 (sequence indicator) is a zero if the error occurred during an Operate I/O instruction and is set to one if the error occurred during a non-DPC transfer. The sequence indicator bit is not an error in itself but reflects the last interface sequence at any time. An I/O check cannot be caused by a software error. (See note.)

Note. The machine check class interrupt initiated by a CPU control check or I/O check causes a reset. The I/O channel and all devices in the system are reset as if a Halt I/O (channel directed command) had been executed. The processor, sensor-based output points, and timer values are not reset.

Bit 12 Sequence Indicator. This bit reflects the last I/O interface sequence to occur. See "I/O Check" described above.

Bit 13 Auto IPL. Set to one by hardware when an automatic IPL occurs.

Bit 15 Power Warning and Thermal Warning. Set to one when these conditions occur (see *Power/Thermal Warning* class interrupt in this chapter). The power/thermal class interrupt is controlled by the summary mask.

Program Controlled Level Switching

Level switching under program control may be accomplished by using the Set Level Block (SELB) instruction. This instruction is covered in detail in Chapter 6, *Instructions*, and in general it will:

- Specify the location of a level status block (LSB) at an effective address in main storage.
- Specify a selected priority level associated with the main storage LSB.
- Load the main storage LSB into the hardware LSB for the selected level.

Note. The hardware LSB consists of the following hardware registers for the selected level:

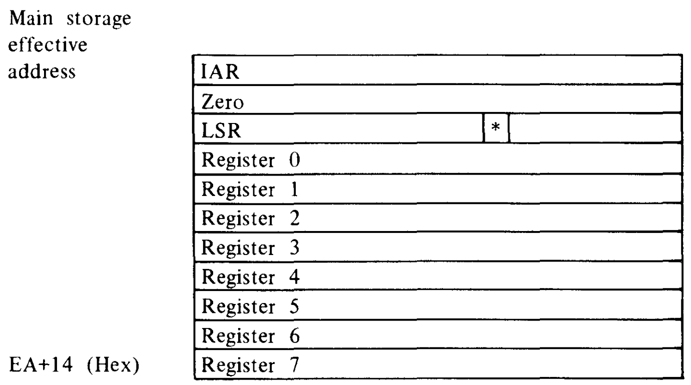
1. Instruction address register
2. Level status register
3. Eight general registers (0–7)

The system programmer should become thoroughly familiar with other effects on the processor caused by execution of the SELB instruction. These effects are determined by three factors:

1. The current execution level.
2. The selected level specified in the SELB instruction.
3. The state of the *in-process flag* (Bit 9 of the LSR) contained in the main storage LSB.

Note. Interrupt masking, provided by the summary mask and the interrupt level mask register, does not apply to program controlled level switching.

The main storage LSB and the location of the in-process flag bit are shown in the following diagram:

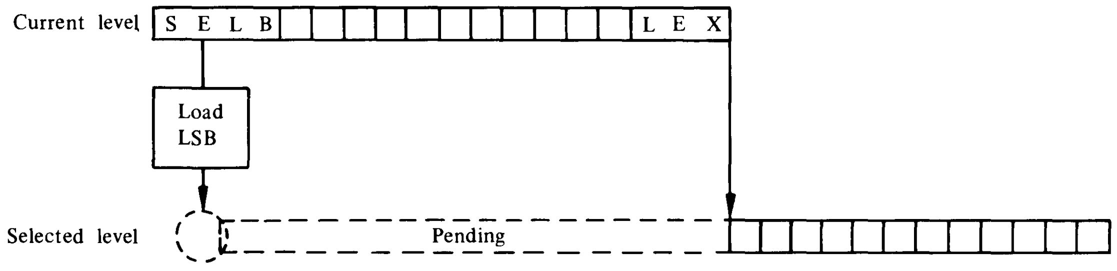


*In-process flag (bit 9)
0 = off
1 = on

Execution of the SELB instruction may result in level switching or a change in the pending status of a level as described in the following sections.

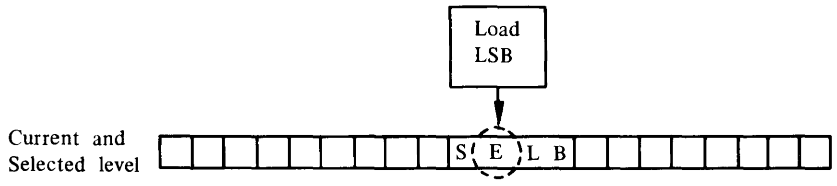
Selected Level Lower Than Current Level and In-process Flag On

These conditions cause the selected level to be pending. The main storage LSB is loaded into the hardware LSB for the selected level. Execution of a LEX instruction on the current level causes the selected level to become active providing no higher priority interrupts are being requested.



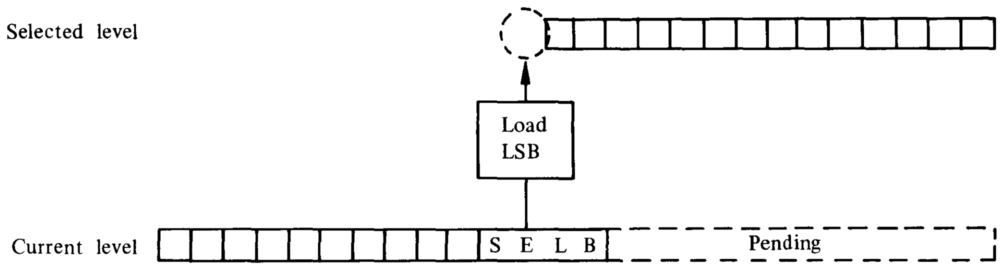
Selected Level Equal to Current Level and In-process Flag On

These conditions cause the selected level to become the current level. The main storage LSB is loaded into the hardware LSB for the selected level.



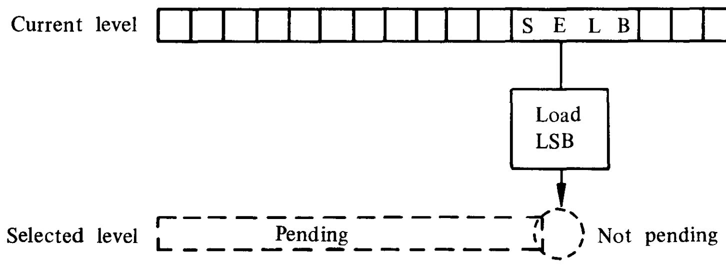
Selected Level Higher Than Current Level and In-process Flag On

These conditions cause the selected level to become the current level. The main storage LSB is loaded into the hardware LSB for the selected level. This is a level switch to the higher (selected) level and causes the lower level to be pending.



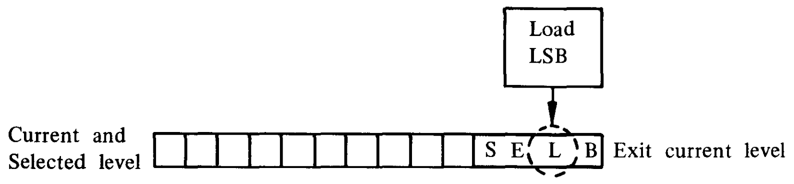
Selected Level Lower Than Current Level and In-process Flag Off

These conditions cause the selected level to be not pending. The main storage LSB is loaded into the hardware LSB for the selected level.



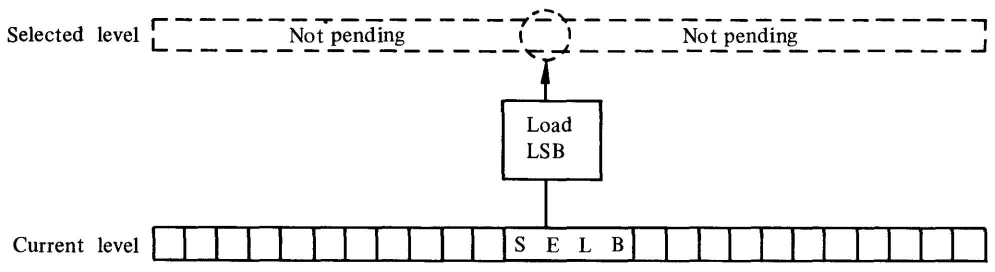
Selected Level Equal to Current Level and In-process Flag Off

These conditions cause an exit from the current level. This exit is identical to executing a LEX instruction with the exception that the main storage LSB is loaded into the hardware LSB for the selected level. See LEX instruction in Chapter 6.



Selected Level Higher Than Current Level and In-process Flag Off

The main storage LSB is loaded into the hardware LSB for the higher (selected) level.



Interrupt Masking Facilities

Three levels of priority interrupt masking are provided to the programmer for control of the interrupt processing. These consist of:

1. Summary Mask (LSR bit 11)
2. Interrupt Level Mask Register
3. Device Mask (I-bit)

Each masking facility has specific control as explained in the following sections.

Summary Mask

The summary mask provides a masking facility for priority interrupts and certain class interrupts. The state of the summary mask (enabled or disabled) is controlled by bit 11 in the level status register (LSR) of the active priority level. When bit 11 is set to zero, the summary mask is disabled and prevents (1) all priority interrupts regardless of priority level, and (2) power/thermal and console class interrupts. All other class interrupts are not masked. When bit 11 is set to one, the mask is enabled and the interrupts are allowed.

The summary mask is disabled and enabled as follows:

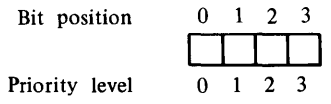
- Disabled (Set to Zero)
  1. When a Supervisor Call (SVC) instruction is executed, the summary mask for the active level is disabled.
  2. Execution of a Disable (DIS) instruction, with bit 15 of the instruction equal to one, causes the summary mask for the active level to be disabled.
  3. All class interrupts disable the active level summary mask.
  4. The summary mask for a selected level is disabled by executing a Set Level Block (SELB) instruction with bit 11 of the LSR to be loaded equal to zero.
  5. The summary mask bits for priority levels 1–3 are set to zero by a system reset, power-on reset, or IPL.
- Enabled (Set to One)
  1. Execution of an Enable (EN) instruction, with bit 15 of the instruction equal to one, causes the active level summary mask to be enabled.
  2. The summary mask for a selected level is enabled by executing a Set Level Block (SELB) instruction with bit 11 of the LSR to be loaded equal to one.
  3. The level zero summary mask is enabled by a system reset, power-on reset, or IPL.
  4. The summary mask for the interrupted-to level is enabled by a priority interrupt.

Note. If the processor is in the wait state, the summary mask is enabled or disabled as defined by bit 11 in the LSR of the last active priority level.

Interrupt Level Mask Register

The interrupt level mask register is a 4-bit register used for control of interrupts on specific priority levels. Each level is controlled by a separate bit of the mask register as shown below:

Interrupt Level Mask Register



With a bit position set to one, the corresponding priority level is enabled and permits interrupts. With a bit position set to zero, the corresponding priority level is disabled. The Set Interrupt Mask Register (SEIMR) instruction is used to control bit settings in the interrupt level mask register. The Copy Interrupt Mask Register (CPIMR) instruction may be used to interrogate the register.

Note. All levels are enabled (set to one) by a system reset, power-on reset, or IPL.

Device Mask (I-bit)

Each interrupting device contains a one-bit mask called the device mask or interrupt bit (I-bit). Interrupts by the device are permitted when its device mask is enabled (set to one). With the device mask bit disabled (set to zero), that device cannot cause an interrupt. The device mask is controlled by a *Prepare* command in conjunction with an Operate I/O instruction. See Chapter 6, *Instructions*, and Chapter 4, *Input/Output Operations.*

Chapter 4. Input/Output Operations

Input/output (I/O) operations involve the use of input/output devices. These devices are attached to the processor and main storage via the I/O channel with the channel directing the flow of information. The I/O channel can accommodate a maximum of 256 addressable devices. The general data flow is shown in Figure 4-1.

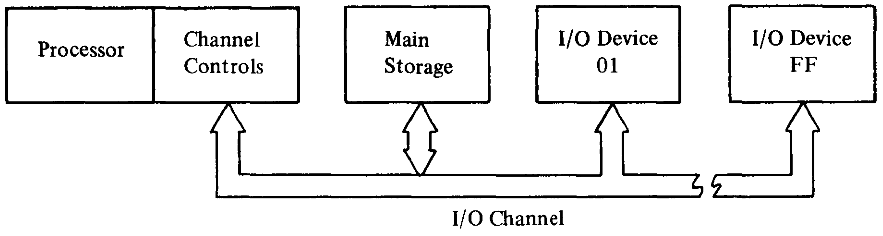


Figure 4-1. Block diagram of Series/1 Model 3 system

The channel supports three basic types of operations:

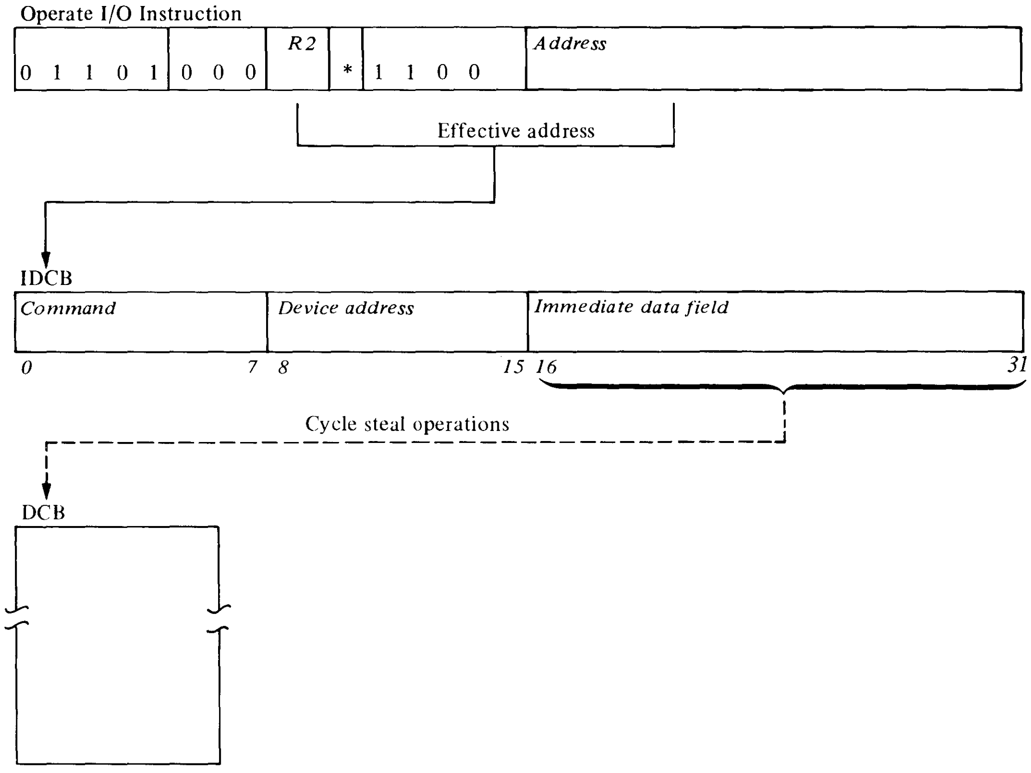
- *Direct Program Control (DPC) Operations* – An immediate data transfer is made between main storage and the device for each Operate I/O instruction. The data may consist of one byte or one word. The operation may or may not terminate with an interrupt.
- *Cycle Steal Operations* – An Operate I/O instruction can initiate cycle-stealing data transfers of up to 65,535 bytes between main storage and the device. Cycle steal operations are overlapped with processing operations. Word or byte transfers, DCB chaining, burst mode, and program controlled interrupt can be supported. All cycle stealing operations terminate with an interrupt.
- *Interrupt Servicing* – Four preemptive priority interrupt levels are available to facilitate device service. The device interrupt level is assignable by the program. In addition, the device interrupt capability may be masked under program control. Interrupt requests, along with cycle steal requests, are presented and polled concurrently with DPC and cycle-steal data transfers.

The channel provides comprehensive error checking including time-outs, sequence checking, and parity checking. Error, exception, and status reporting are facilitated by: (1) recording condition codes in the processor during execution of Operate I/O instructions, and (2) recording condition codes and an Interrupt Information Byte (IIB) in the processor during interrupt acceptance. Additional status words may be used by the device as necessary to describe its status (see *I/O Condition Codes and Status Information* in this chapter).

Operate I/O Instruction

The Operate I/O instruction initiates all I/O operations from the processor. It is a privileged instruction and is independent of specific I/O parameters. The generated effective address points to an immediate device control block (IDCB) in main storage. The IDCB consists of two words that contain an I/O command, a device address, and an immediate data field. For DPC operations, the immediate data field is used as a device data word. For cycle steal operations, the immediate data field points to a device control block (DCB) that provides additional information needed for the operation. For more details of the *Operate I/O Instruction* refer to Chapter 6.

Operate I/O Instruction

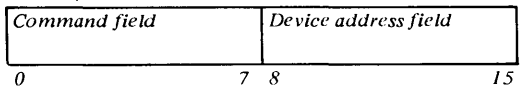


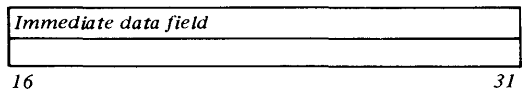
**Indirect addressing bit*

Immediate Device Control Block (IDCB)

The location in storage specified by the Operate I/O instruction contains the first word of the IDCB. The IDCB contains an I/O command that describes the specific nature of the I/O operation. This command is used by the channel for execution of the operation. The IDCB must always be on a word address boundary and has the following format:

IDCB (immediate device control block)





Command field (bits 0–7)

Bit 0	*Channel directed.* If this bit is equal to one, the I/O command is directed to the channel rather than to a specific device. The *Halt I/O* command is the only valid channel directed command. Any other command with bit 0 set to one causes a *command reject* exception condition.
Bit 1	*Read/Write.* If this bit is equal to one, the data contained in the immediate field is transferred to the addressed I/O device. If this bit is equal to zero, the immediate field contains the data received from the I/O device at the conclusion of the IO instruction.
Bits 2–3	*Function.* This field specifies the general type of I/O operation to be performed (see Figure 4-2).
Bits 4–7	*Modifier.* This field contains four bits for further specification of a function, if required (see Figure 4-2).

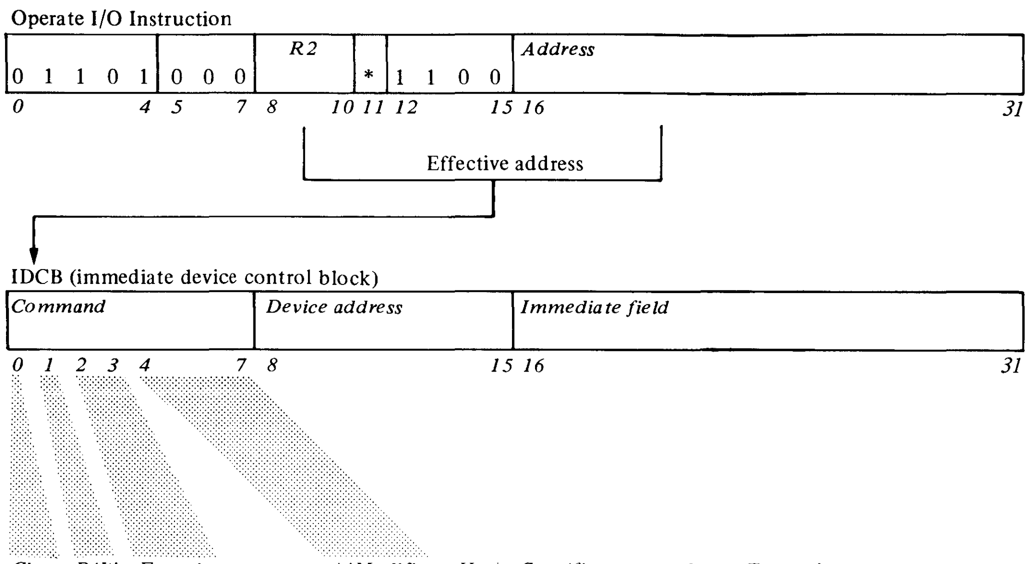
Device address field (bits 8–15)

This byte contains the I/O device address. The address range is 00 through FF (hex).

Immediate data field (bits 16–31)

This field contains a device data word for DPC operations. It contains the address of a device control block for cycle steal operations.

Figure 4-2 shows the relationship of the IDCB and the Operate I/O instruction. It also contains a chart of the various I/O commands. The Start command and the Start Cycle Steal Status command are used to initiate cycle steal operations. The remaining commands are used for DPC operations only.



Chan	R/W	Function	**Modifier	Hex	Specific command	Type of operation
0	0	00 Read	XXXX	0X	Read	DPC
0	0	01 Read	XXXX	1X	Read	DPC
0	0	10 Read status	0000	20	Read ID	DPC
0	0	10 Read status	XXXX	2X	Read status	DPC
0	0	11		3X	Unused***	Unused
0	1	00 Write	XXXX	4X	Write	DPC
0	1	01 Write	XXXX	5X	Write	DPC
0	1	10 Control	0000	60	Prepare	DPC
0	1	10 Control	XXXX	6X	Control	DPC
0	1	10 Control	1111	6F	Device reset	DPC
0	1	11 Start	XXXX	7X	Start	Cycle steal
0	1	11 Start	1111	7F	Start cycle steal status	Cycle steal
1	1	11 Channel	0000	F0	Halt I/O	Channel

*Indirect addressing bit.

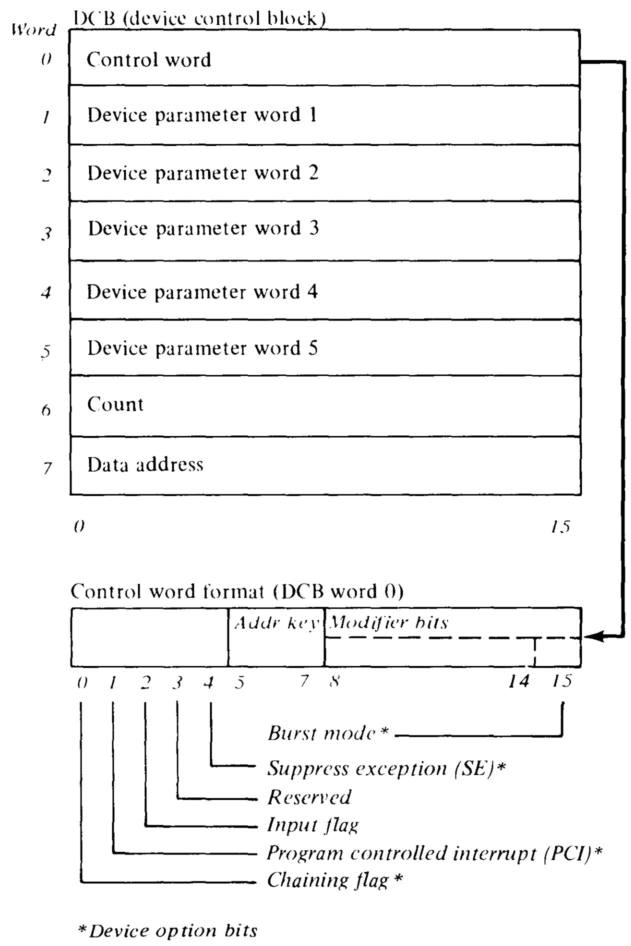
**Modifier XXXX is device dependent. Other modifiers are system defined.

***To avoid future code obsolescence, this command format must not be used.

Figure 4-2. IDCB and I/O commands

Device Control Block (DCB)

This section describes the device control block that is used for a cycle steal operation. The actual cycle steal operation is explained in a later section of this chapter. The DCB is an eight-word control block residing in the supervisor area of main storage. It contains the specific parameters of a cycle steal operation. The device fetches the DCB using the cycle steal mechanism. The format of the DCB is shown in Figure 4-3.



Word
DCB (device control block)
0 Control word
1 Device parameter word 1
2 Device parameter word 2
3 Device parameter word 3
4 Device parameter word 4
5 Device parameter word 5
6 Count
7 Data address
0 15
Control word format (DCB word 0)
Addr key
Modifier bits
0 1 2 3 4 5 7 8 14 15
Burst mode*
Suppress exception (SE)*
Reserved
Input flag
Program controlled interrupt (PCI)*
Chaining flag*
*Device option bits

Figure 4-3. Device control block

The DCB words have the following meanings:

Control word

Bit 0*	*Chaining flag.* If this bit is equal to one, a DCB chaining operation is indicated.
Bit 1*	*Programmed controlled interrupt (PCI).* If this bit is equal to one, the device presents a programmed controlled interrupt (PCI) at the completion of the DCB fetch.
Bit 2	*Input flag.* The setting of this bit tells the device the direction of data transfer. 0 = Output (main storage to device) 1 = Input (device to main storage) For bidirectional data transfers under one DCB operation, this bit must be set to one. For control operations involving no data transfer, this bit must be set to zero.
Bit 3	*Reserved.* This bit must be set to zero to avoid future code obsolescence.
Bit 4*	*Suppress exception (SE).* If this bit is equal to one, the device is allowed to suppress the reporting of certain exception conditions. The device can then take alternative action depending on the condition.
Bits 5–7	*Cycle steal address key.* Not used on the 4953 Processor.
Bits 8–15	*Modifier.* These are device dependent bits with one exception. When a device uses burst mode, it is specified in bit 15. These bits may be used for functions that are unique to a particular device.

*Chaining, PCI, and SE are device options that are available on a device feature basis. Any bit not used by the device should be set to zero although it is not checked by the device.

Refer to the *Cycle-Steal Device Options* section of this chapter.

Device Parameter Words 1–2

These parameter words are device-dependent control words and are implemented as required. Refer to the individual device publications for definition.

Device Parameter Word 3

When PCI is specified, the high-order byte (bits 0–7) of this word is used for a DCB identifier. The device places the identifier in the interrupt information byte when the PCI is processed. The low-order byte (bits 8–15) is always device dependent. The high-order byte is device dependent when PCI is not specified.

Device Parameter Word 4

If suppress exception (SE) is used by a device, this word specifies a 16-bit main storage address called the *status address*. This address points to a *residual status block* that is stored by the device following completion of the DCB operation.

If suppress exception is not used by a device, a residual status block is not stored. In this case, parameter word 4 is device dependent. Refer to *Cycle-Steal Device Option* in this chapter.

Device Parameter Word 5

If the DCB chaining bit (bit 0 of the control word) is equal to one, this word specifies a 16-bit main storage address of the next DCB in the chain. If chaining is not indicated, this parameter word is device dependent.

Count

The count word contains a 16-bit unsigned integer representing the number of data bytes to be transferred for the current DCB. Count is specified in bytes with a range of 0 through 65,535. The count specification must be even for word-only devices.

Data Address

This word contains the starting main storage address for the data transfer.

Programming Considerations When Using the DCB

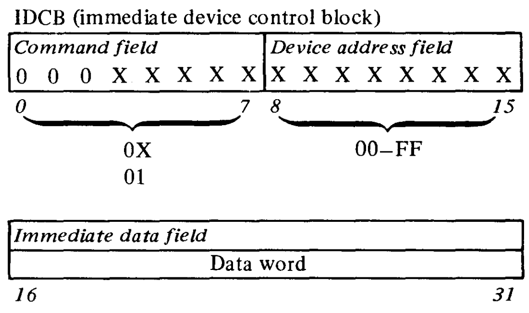
1. Only those words required for the cycle stealing operation are fetched by the device and they may be fetched in any order. Contents of the words must be specified correctly; if not, the device records a *DCB specification check* in the interrupt status byte and terminates the cycle steal operation with an exception interrupt.
2. The DCB address (in the DCB), the chain address, and the status address must be even (word boundary). If the DCB address is odd, the device records a *command reject* condition code and terminates the cycle steal operation. An odd chain address or status address results in a *DCB specification* check.

Note. Condition code and status recording are explained in detail in a separate section of this chapter.

I/O Commands

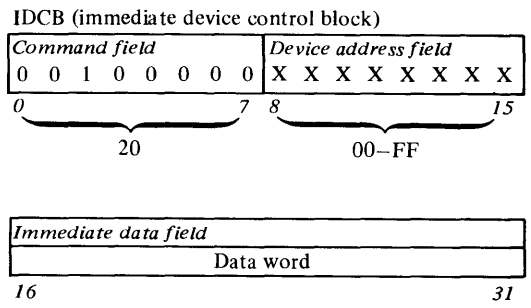
This section describes each I/O command and shows the related IDCB. The command field (bits 0–7) of the IDCB contains the binary value of the command. An X in this field means the value is device dependent.

Read

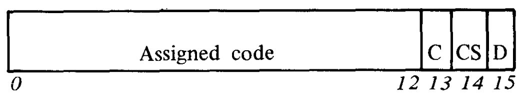


This command transfers a word or byte from the addressed device to the data word of the IDCB. If a single byte is transferred, it is placed in bits 24–31 of the data word with bits 16–23 set to zeros. Correct parity is always maintained and checked for both bytes on the I/O channel. The individual devices may use either the 0X or 1X type of read command. The two commands operate the same in the channel.

Read ID



This command transfers an identification (ID) word from the device to the data word of the IDCB. The device identification word contains physical information about the device and may be used to determine the devices that are attached to the system. This word is not related to the interrupt ID word associated with interrupt processing. The device ID word format is:



Bits 0–12	Unique identification code for the device
Bit 13	Zero – not a controller device or the device does not report delayed command reject One – controller device or any device that reports delayed command reject
Bit 14	Zero – not a cycle steal device One – cycle steal device
Bit 15	Zero – IBM device One – OEM device

Note. A controller may control more than one I/O device and is not directly addressable, but is not transparent to software. That is, the controller may cause busy or exception conditions as opposed to those caused by an attached I/O device.

Read Status

IDCB (immediate device control block)

Command field	*Device address field*
0 0 1 0 X X X X	X X X X X X X X
0 7	*8 15*
2X	00–FF

Immediate data field
Data word
16 31

This command transfers a device status word from the device to the data word of the IDCB. Contents of the status word are device dependent.

Write

IDCB (immediate device control block)

Command field	*Device address field*
0 1 0 X X X X X	X X X X X X X X
0 7	*8 15*
4X 5X	00–FF

Immediate data field
Data word
16 31

This command transfers a word or byte to the addressed device from the data word of the IDCB. The individual device may use either format of the command. If a single byte is to be transferred, it must be placed in bits 24–31 of the data word and bits 16–23 must be set to zero. A byte oriented device may ignore bits 16–23 (including the parity bit on the I/O channel) but these bits should be zeros to avoid future code obsolescence.

Note. Both bytes of the IDCB data word are fetched by the channel and placed on the I/O data bus (in good parity) even if not required by the device.

Prepare

IDCB (immediate device control block)

Command field	*Device address field*
0 1 1 0 0 0 0 0	X X X X X X X X
0 7	*8 15*
60	00–FF

Immediate data field		
Zeros	Level	I
16 26	*27 30*	*31*

This command transfers a word (to the addressed device) that controls the device interrupt parameters. The word is transferred from the immediate data field of the IDCB in the format shown. A priority interrupt level is assigned to the device by the *level* field. The I-bit (device mask) controls the device interrupt capability. If the I-bit equals 1, the device is allowed to interrupt. If the I-bit equals 0, the device cannot interrupt. See *Prepare I/O Device for Interrupt* in Chapter 3.

Note. The IBM 4953 Processor does not recognize a priority level other than 0–3. Lost interrupts result if a device is prepared for a level other than 0–3.

Control

IDCB (immediate device control block)

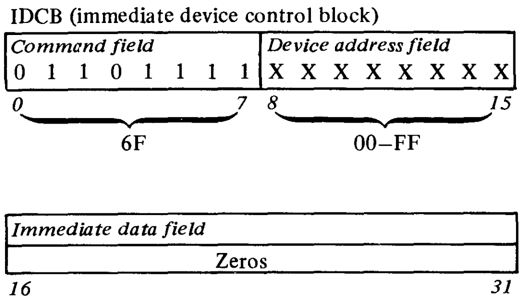
Command field	*Device address field*
0 1 1 0 X X X X	X X X X X X X X
0 7	*8 15*
6X	00–FF

Immediate data field
Data word
16 31

This command initiates a control action in the addressed device. A word, or byte, transfer from the data word of the IDCB to the addressed device may or may not occur, depending on device requirements. If a single byte is to be transferred it must be placed in bits 24–31 of the data word and bits 16–23 must be set to zero.

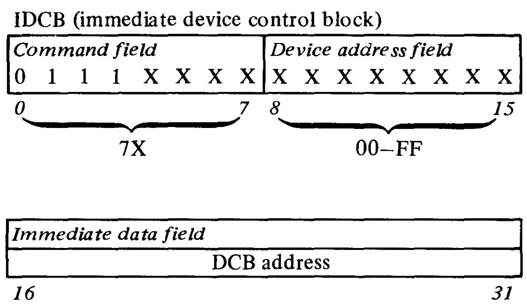
Note. Both bytes of the IDCB data word are fetched by the channel and placed on the I/O data bus (in good parity) even if not required by the device.

Device Reset



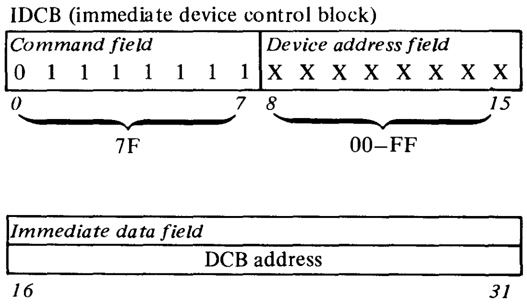
This command resets the addressed device. A pending interrupt from this device (or a busy condition) is cleared. The device mask (I-bit) is not changed. A device must always accept and execute this command. There is no change to the assigned priority level for the device. The residual address (device status) and output sensor points are not affected. Parity checking of the IDCB data word is not performed.

Start



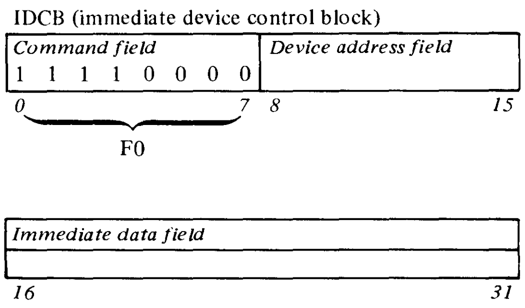
This command initiates a cycle steal operation for the addressed device. The second word of the IDCB is transferred to the device. It contains a 16-bit logical storage address of a device control block (DCB) to be used by the device. See *Cycle Steal* in this chapter.

Start Cycle Steal Status



This command initiates a cycle steal operation for the addressed device. Its purpose is to collect status information from the addressed device. The second word of the IDCB is transferred to the device and contains a 16-bit logical address of a device control block (DCB). See *Start Cycle Steal Status Operation* in this chapter.

Halt I/O



This is a *channel* directed command that causes a halt of all I/O activity on the I/O channel and resets all devices. No data is associated with this command. All pending device interrupts are cleared. Device priority-interrupt-level assignments and device masks (I-bits) are unchanged. The residual address (device status) and output sensor points are not affected.

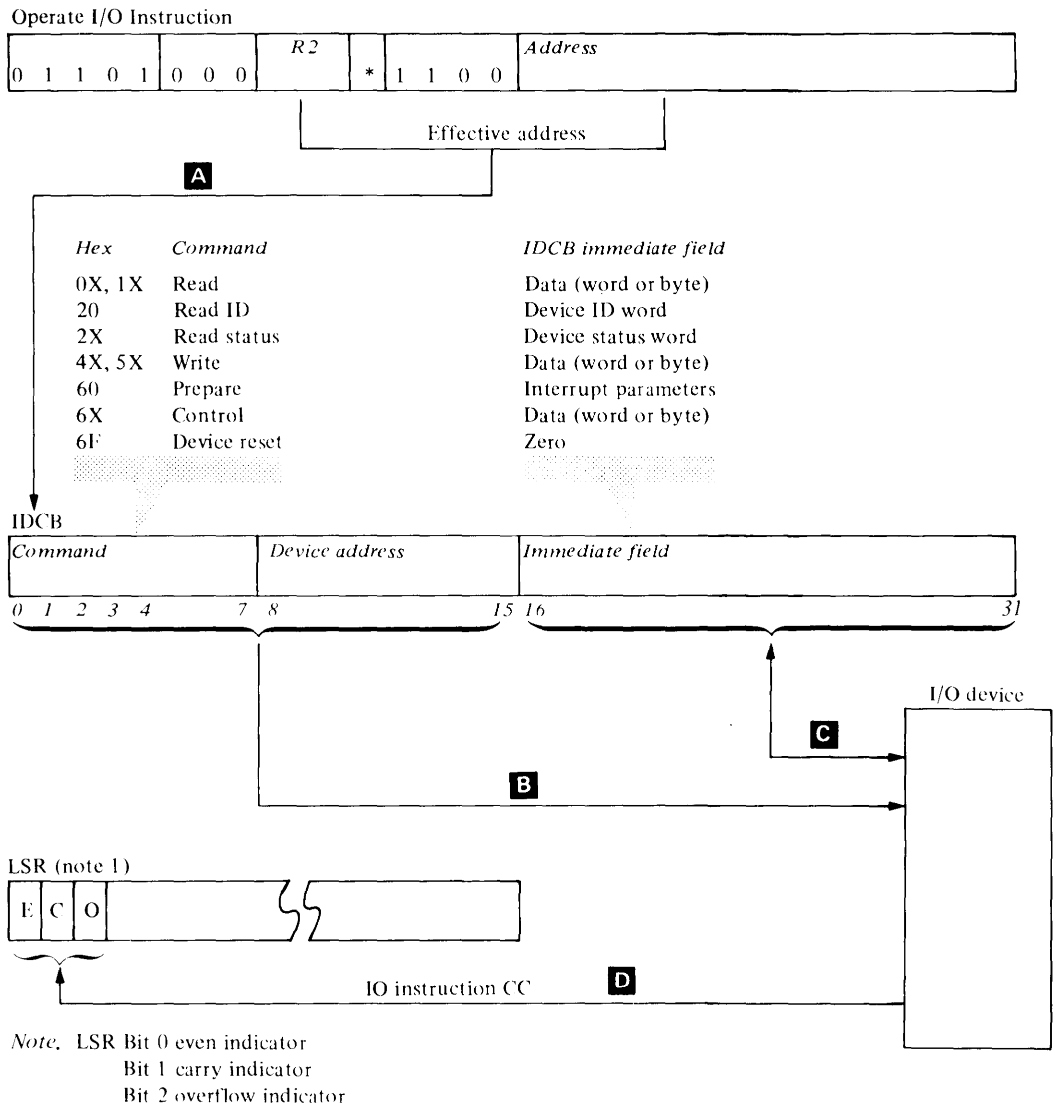
Notes.
1. The channel is always able to accept and execute this command.
2. Halt I/O is the only valid channel directed command.

DPC Operation

A DPC operation causes an immediate transfer of data or control information to or from an I/O device. An Operate I/O instruction must be executed for each data transfer and causes the following events to occur (refer to Figure 4-4).

1. The Operate I/O instruction points to an IDCB in main storage. **A**
2. The I/O channel uses the IDCB to select the addressed device and to determine the operation to perform. **B**
3. The I/O channel sends data to the device from main storage, or from the device to main storage. **C**
4. The device sends an IO instruction condition code to the level status register (LSR) in the processor. **D**

Notes.
1. The DPC operation may end with a priority interrupt if the device has this capability. Refer to *I/O Interrupts* in Chapter 3.
2. There are two types of condition codes: the first is an I/O instruction condition code and is available immediately after completion of an Operate I/O instruction; the second is an interrupt condition code and is presented upon acceptance of a priority interrupt. The code significance is different for the two cases. Refer to *I/O Condition Codes and Status Information* in this chapter.



Hex	*Command*	*IDCB immediate field*
0X, 1X	Read	Data (word or byte)
20	Read ID	Device ID word
2X	Read status	Device status word
4X, 5X	Write	Data (word or byte)
60	Prepare	Interrupt parameters
6X	Control	Data (word or byte)
6F	Device reset	Zero

Note. LSR Bit 0 even indicator
Bit 1 carry indicator
Bit 2 overflow indicator

Figure 4-4. Direct program control I/O operation

Cycle Steal

The cycle steal mechanism allows data service to or from an I/O device while the processor is processing instructions. This overlapped operation allows multiple data transfers to be started by one Operate I/O instruction. The processor executes the Operate I/O instruction, then continues processing instructions while the I/O device steals main storage data cycles when needed. The channel resolves contention among multiple devices requesting cycle steal transfers. The operation always ends with a priority interrupt from the device.

The cycle steal operation includes certain capabilities that are provided on a device feature basis:

1. Burst mode
2. DCB chaining
3. Programmed controlled interrupt (PCI)
4. Suppress exception (SE)
5. Storage addresses and data transfers by byte or word

See the *Cycle-Steal Device Options* section of this chapter for details of these facilities.

All cycle steal operations terminate with a priority interrupt, providing, the device has executed a successful *Prepare* command, with the device mask (I-bit) enabled. If the device mask is disabled, the interrupt presentation is blocked and the device remains busy until (1) the condition is cleared by a reset, or (2) the proper Prepare command is executed.

All cycle steal operations are started by an Operate I/O instruction that points to an IDCB. The immediate data field of the IDCB contains the address of a device control block (DCB). The DCB is fetched by the device using the cycle-steal mechanism. Within the DCB are specific parameters of the cycle steal operation. See *Device Control Block* in this chapter.

There are two types of cycle steal commands:

- Start
- Start Cycle Steal Status.

Start Operation

A cycle steal operation begins after successful execution of the *Start* command. The IDCB, pointed to by an Operate I/O instruction, has the format:

IDCB (immediate device control block)

Command field	*Device address field*
0 1 1 1 X X X X	X X X X X X X X
0 *7*	*8* *15*
7X	00–FF

Immediate data field
DCB address
16 *31*

The command modifier (X) is device dependent. The DCB address always specifies a word boundary and is the starting storage address of the DCB. This address is used by the device to fetch the DCB, using the cycle steal mechanism.

A cycle steal operation is presented in the following chart. Use Figure 4-5 in conjunction with this chart. Condition codes used in the chart are fully explained in the section *I/O Condition Codes and Status Information* in this chapter.

Note. An I/O device must be properly prepared (using a *Prepare* command), before it is allowed to interrupt.

Cycle steal major steps	*Remarks*
Start cycle steal	1. Execute IO instruction. 2. IDCB contains *Start* command and points to a DCB. The DCB address is sent to the device. **A** 3. Device presents condition code 7 (bits 0–2 in the LSR). **B**
Device fetches DCB	1. Device uses cycle steal mechanism to fetch DCB. **C**
Data transfer	1. Data is transferred to or from the device in word or byte format. **D** 2. Transfer continues until count in DCB is exhausted.
Termination (no error condition)	1. Device presents interrupt request. 2. Channel polls I/O attachment feature and accepts request. 3. Device sends interrupt ID word and interrupt condition code 3 (device end).
Termination (Exception condition)	1. Device presents interrupt request. 2. Channel polls I/O attachment feature and accepts request. 3. Device sends interrupt ID word and interrupt condition code 2 (exception).

Note. Other events that might occur during the cycle steal operation are:

Chaining	1. Device completes the current DCB operation but does not present an interrupt request. 2. Device fetches next DCB in the chain. **E**
Program Controlled interrupt	1. Device fetches DCB (PCI bit = 1). 2. Device initiates an interrupt and sends an interrupt ID word and interrupt condition code 1 (PCI).
Suppress exception	1. Device completes current operation. 2. Device stores status at the main storage location defined by DCB parameter word 4.

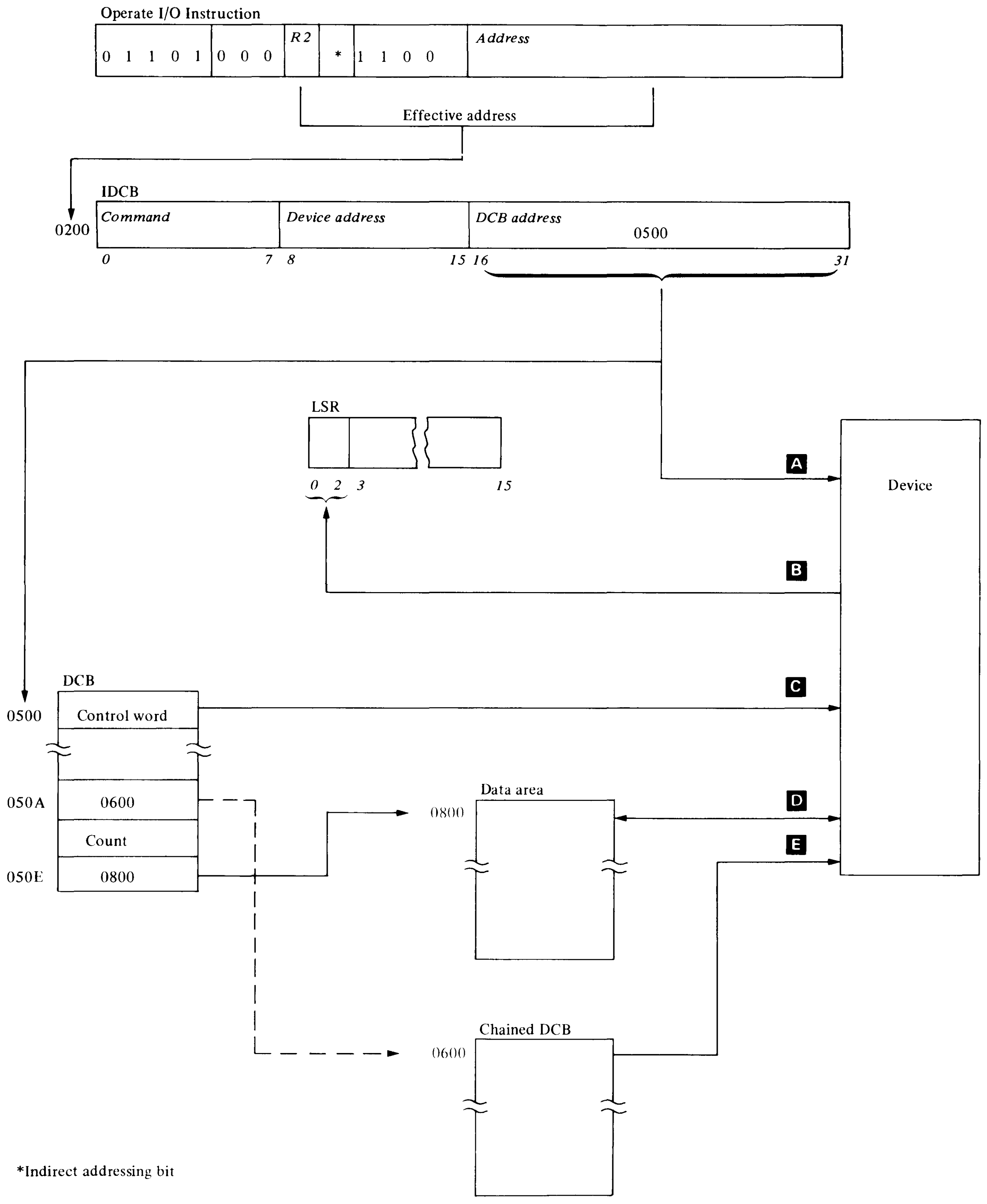


Figure 4-5. Example of cycle steal control information

Start Cycle Steal Status Operation

The purpose of this operation is to obtain data from the device if the previous cycle steal operation terminates due to an error or exception condition. The operation is initiated by a *Start Cycle Steal Status* command. The IDCB format is:

IDCB (immediate device control block)

Command field	*Device address field*
0 1 1 1 1 1 1 1	X X X X X X X X
0 … *7*	*8* … *15*
7F	00–FF

Immediate data field
DCB address
16 … *31*

This command uses a special DCB format with some words and fields to set to zeros (see Figure 4-6).

DCB (device control block)

Word	
0	*Control word* 0 0 1 0 0 | 0 0 0 | 0 0 0 0 0 0 0 0
1	Not used (zeros)
2	Not used (zeros)
3	Not used (zeros)
4	Not used (zeros)
5	Not used (zeros)
6	Byte count
7	Data address
	0 … *15*

Figure 4-6. DCB for start cycle steal status operation

Programming Note.
Concerning the DCB for the start cycle steal status operation:

1. Bits designated as zero are not checked by hardware (see Figure 4-6).
2. The count is specified in bytes.
3. The maximum count is device dependent.
4. The validity of a count value less than the maximum value is device dependent.
5. If the maximum count is exceeded, or a count value is specified that indicates the partial storing of a word length parameter, the device records a *DCB specification check* in the ISB and terminates the operation.
6. An odd data address also results in a *DCB specification check.*

Data is transferred to main storage starting at the data address specified in the DCB. This data consists of residual parameters and device dependent status information and has the following format:

Word 0	Residual address
Word 1	Device cycle steal status word 1
Word 2	Device dependent status word
•	*0* … *15*
•	
•	
•	

Residual Address. This word contains the main storage address of the last attempted cycle steal transfer associated with a *Start* command. It may be a data address, a DCB address, or a residual-status-block address. It is updated to the current cycle-steal storage address upon execution of cycle steal transfers. For word transfers, the residual address points to the high-order byte of the word. If an error occurs during a start cycle steal status operation, this address (as contained within the device) is not altered. Device reset, Halt I/O, machine check, and system reset have no effect on the residual address in the device. It is cleared by a power-on reset. Following a power-on reset the residual address is:

- 0000 (Hex) for a byte-oriented device.
- 0001 (Hex) for a word-oriented device.

Device Cycle-Steal-Status Word 1. This word contains the residual byte count of the previous cycle steal operation associated with a start command. The byte count is initialized by the count field of a DCB associated with a *Start* command, and is updated as each byte of data is successfully transferred via a cycle steal operation. It is not updated by cycle-steal transfers into the residual status block. The residual byte count is not altered if an error occurs during a start cycle steal status operation. It is reset by (1) power-on reset, (2) system reset, (3) device reset, (4) Halt I/O, and (5) machine check condition.

Note. The contents of the device cycle-steal-status word 1 are device dependent if the device does not: (1) implement suppress exception (SE), or (2) store a residual byte count as part of its cycle-steal status.

Device Dependent Status Words. The number and contents of these words are specified by the individual device. Three conditions can cause bits to be set in the device dependent status words (refer to individual device publications).

1. Execution of an I/O command that causes an exception interrupt.
2. Asynchronous conditions in the device that indicate an error, exception, or a state condition.
3. As defined by the individual device.

The bits are reset as follows:

1. For the first condition listed above, the bits are reset by the acceptance of the next I/O command (except Start Cycle Steal Status) following the exception interrupt. These bits are also reset by a power-on reset, system reset, or execution of a Halt I/O command.
2. For the second condition, the bits are reset on a device dependent basis.
3. For the third condition, the bits are reset as defined by the individual device.

Cycle-steal Device Options

The I/O channel supports operations such as burst mode and chaining when required by individual devices. Bits in the DCB control word are used to activate these operations. Refer to the individual device publications for the device options used. The following sections explain the operations.

Burst Mode

Burst mode, when used by a device, is specified in bit 15 of the DCB control word. If bit 15 is equal to one, the transfer of data takes place in burst mode. This mode dedicates the I/O channel to the device until the last data transfer for the DCB is completed. Cycle steal interleave, by other devices, is prevented. Burst mode also prevents any priority interrupt request from being accepted by the processor.

The maximum burst rate for the 4953 channel is 1.332 megabytes per second.

Chaining

The purpose of chaining is to allow the programmer to sequence an I/O device through a set of operations by using a chain of DCBs. Bit 0 of the DCB control word (when set to one) indicates a chaining operation. This means that the chained DCB, fetched by the device, is interpreted as a new operation (or function) to be performed. The DCB may be equal to, but not a continuation of, the operation specified by the previous DCB.

When the current DCB indicates a chaining operation, device parameter word 5 of the DCB must contain a main storage address that points to the next DCB in the chain. The device completes the current operation but does not present an interrupt request (excluding PCI) to the processor. Instead, the device fetches the next DCB in the chain and continues operation.

Note. The chaining operation has no effect on programmed controlled interrupt (PCI). These interrupts, when specified in the DCB, still occur at the completion of the DCB fetch operation.

Programmed Controlled Interrupt (PCI)

Bit 1 of the DCB control word (when set to one) tells the device to present a PCI to the processor at the completion of the DCB fetch prior to data transfer.

When the PCI is serviced, a DCB identifier byte is returned to the processor in the interrupt information byte (IIB). Refer to DCB device parameter word 3 in this chapter. Two conditions should be noted by the programmer:

1. Chaining and data transfers associated with the DCB may commence even if the PCI is pending.
2. If the PCI is pending when the device encounters the next interrupt causing condition, the PCI condition is discarded by the device and replaced with the new interrupt condition.

Suppress Exception (SE)

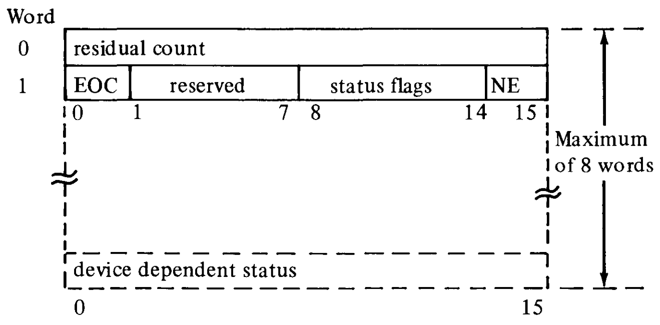
When a device uses this option it is allowed to suppress the reporting of certain exception conditions that would normally cause an exception interrupt. The device is then allowed to take alternative action depending on the condition. The suppressed exception conditions are reported to the programmer as status information upon completion of the operation. Refer to a subsequent section, *Suppression of Exceptions*, for details of the various actions a device might take.

The suppress exception option also provides for automatic logging of status information (including suppressed exceptions) into main storage. When the SE bit for a DCB is set to one, the device always stores a residual status block into main storage after successful completion of the data transfer for the DCB. Device parameter word 4 of the DCB must be used to specify the starting main storage address for the residual status block. Note that a residual status block is stored even if there are no exception conditions to be suppressed.

The following section shows the residual status block that is stored.

Residual Status Block

The residual status block is stored into main storage at the location pointed to by the status address (DCB word 4). The size of a residual status block is fixed for each device with a limit of 8 words total. The format is:



Word 0 Contains the residual byte count associated with the DCB.

Word 1 EOC is the End of Chain bit and is set to one for all conditions that would terminate a chaining operation. NE is the No Exception bit and is set to one when the operation is completed and no exceptions are reported. The Status Flags are device dependent flags that indicate suppressed exception conditions.

Any additional words are device dependent as to number and content. Refer to the individual device publications for the additional status information and, also, the bit significance of the status flags.

Suppression of Exceptions

An exception condition can be suppressed by a device only when it occurs during a data transfer operation. It cannot be suppressed if it occurs during (1) a DCB fetch, (2) storing of a residual status block, or (3) a cycle steal status operation. A second requirement of a suppressible exception is that the device be capable of continuing operation in a normal and predictable manner after occurrence of the exception. If these conditions are not met, the exception condition causes an exception interrupt. When a suppressible exception is encountered, the device initiates one of a possible three types of action depending on the device and the exception condition. Note that the number of action types used by a device and the suppressible exceptions for each type are a device specification. Refer to the individual device publication. The three action types are:

1. Suppress Exception and Continue. The exception condition occurs but data transfer is allowed to proceed. At the completion of the data transfer (defined by the DCB) a residual status block is stored with word one set as follows:
   - A status flag for this exception is set to one.
   - If the DCB specifies chaining, then the EOC bit is set to zero. Otherwise, it is set to one.
   - The NE bit is set to zero.

   The device may then continue with the next DCB if chaining is specified.
2. Suppress Exception and Terminate Data Transfer. Upon detecting the exception condition, the device terminates the data transfer for this DCB. It then stores a residual status block containing:
   - A status flag for the exception condition.
   - EOC bit set to zero, if chaining. Otherwise, set to one.
   - NE bit set to zero.

   The device may then continue with the next DCB if chaining is specified.

   Programming Note. For some devices, the most common exception condition of this type is incorrect length record (ILR). For example, the data transfer is completed prior to the count reaching zero.

   In certain communications devices a short ILR is considered normal operation. When a short ILR occurs in this type device, the residual byte count is sufficient to indicate the condition; therefore, the NE bit may be set to indicate no exception.
3. Suppress Exception and Terminate Chain. Upon detecting this exception condition, the device terminates the data transfer for this DCB. It ignores any commands specifying further chaining.

   The device stores a residual status block containing:
   - A status flag for the exception condition
   - EOC bit set to one
   - NE bit set to zero.

   The device then presents a device end interrupt. Refer to *Interrupt Condition Codes* in a subsequent section of this chapter.

Programming Note. In certain communication devices a change-of-direction character is considered normal operation. When a change-of-direction character occurs in this type device, the EOC bit is sufficient to indicate the condition; therefore, the NE bit may be set to indicate no exception.

Priority of Suppress Exception Actions. Multiple exceptions that are suppressible can occur during an operation. They are noted in the residual status block by setting multiple status flags. The type of action taken by a device depends on the exception/action combination with highest priority. The priority sequence is type 3, type 2, and type 1 with type 3 having the highest priority.

Cycle-steal Termination Conditions

The following chart shows the action that occurs at the end of a DCB operation depending on the function specified and the exception conditions encountered:

CHN	*SE*	*Suppressible exception*	*Non-Suppressible exception*	*No exception*
0	0	I (XCT)	I (XCT)	I (DE)
0	1	I (PDE)	I (XCT)	I (DE)
1	0	I (XCT)	I (XCT)	CC
1	1	*I (PDE)/CC	I (XCT)	I (DE)

CC – DCB chaining

CHN – Chaining flag (bit 0 of the DCB control word)

I (DE) – Device end interrupt

I (PDE) – Permissive device end interrupt (see device end interrupt)

I (XCT) – Exception interrupt

SE – Suppress exception (bit 4 of the DCB control word)

*Dependent on the specific exception condition in the individual device.

I/O Condition Codes and Status Information

Each time an Operate I/O instruction is issued, the device, controller, or channel immediately reports to the processor one of seven condition codes pertaining to execution of the I/O command. These codes are called *IO instruction* condition codes. Three bits are used to encode a condition code value (range 0 through 7). The bits are recorded in the even, carry, and overflow positions of the LSR and may be interrogated by specific instructions such as *Branch on Condition Code* and *Branch on Not Condition Code.* (See BCC and BNCC in Chapter 6.)

For interrupting devices, condition codes are also reported during a priority interrupt. These codes are called *Interrupt* condition codes and pertain to operations that continue beyond execution of the Operate I/O instruction (such as cycle stealing of data). The interrupt condition codes are recorded in the LSR and interrogated in the same manner as the I/O instruction codes. Along with the interrupt condition code, the device also transfers an interrupt ID word to the processor. Bits 0 through 7 of the interrupt ID word contain status information related to the interrupt processing and are called the interrupt information byte (see *Interrupt ID Word* in this chapter).

Figure 4-7 presents an overall view of condition code reporting along with status information. Details of the condition codes and status information are discussed in the following sections. Note that there are two unique sets of condition codes (IO instruction and interrupt) and that most status information is device dependent.

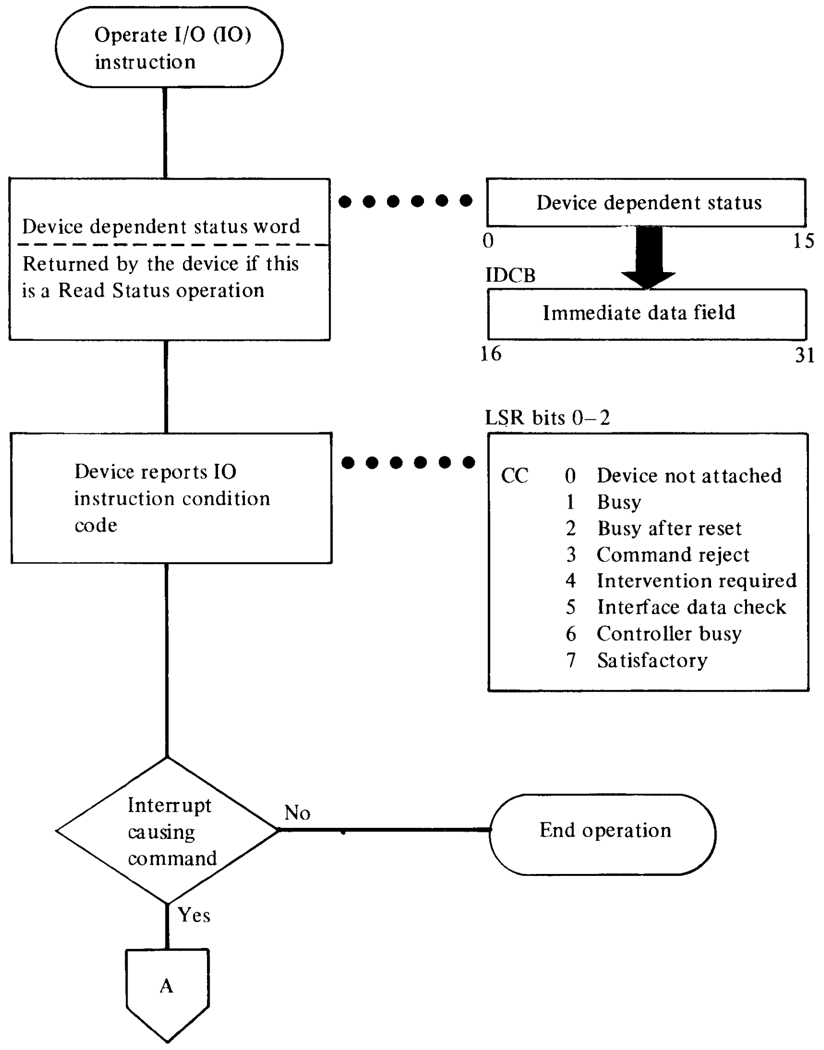


Figure 4-7. Condition codes, status words, and status bytes received from a device (Part 1)

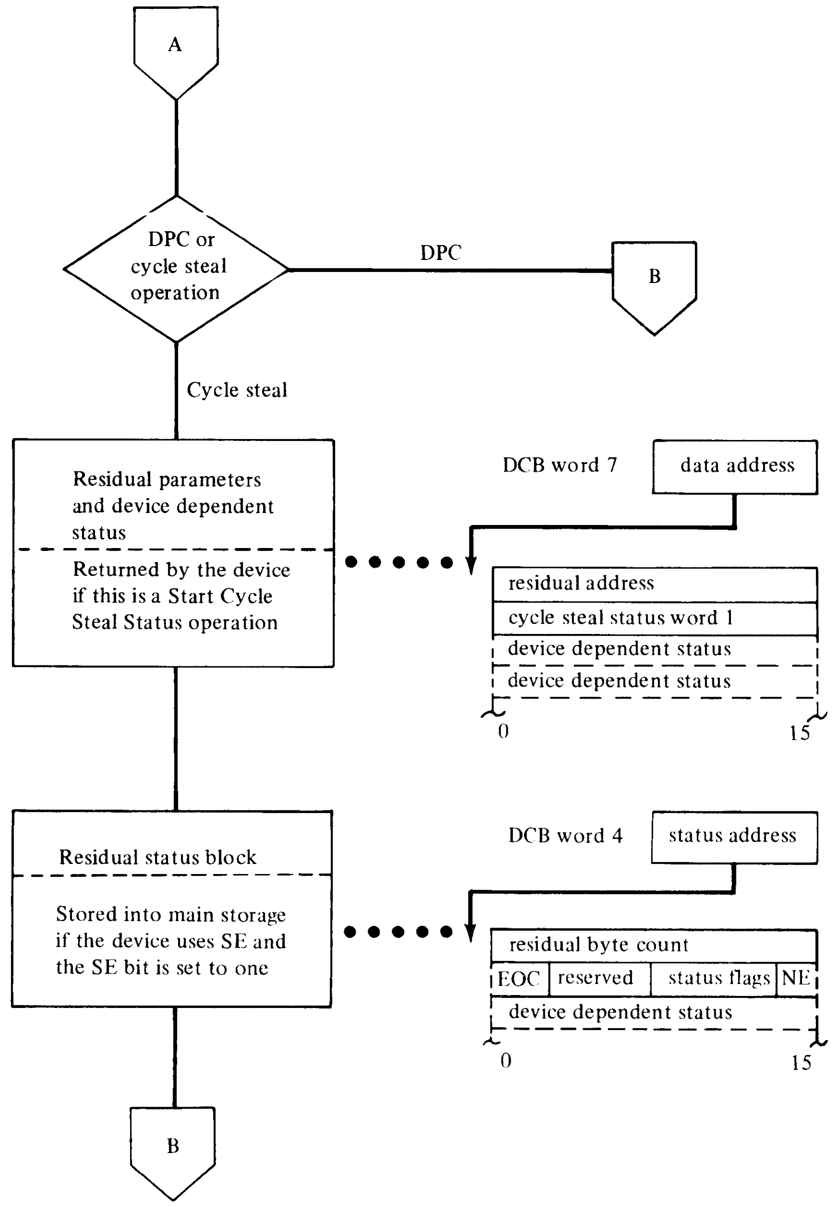
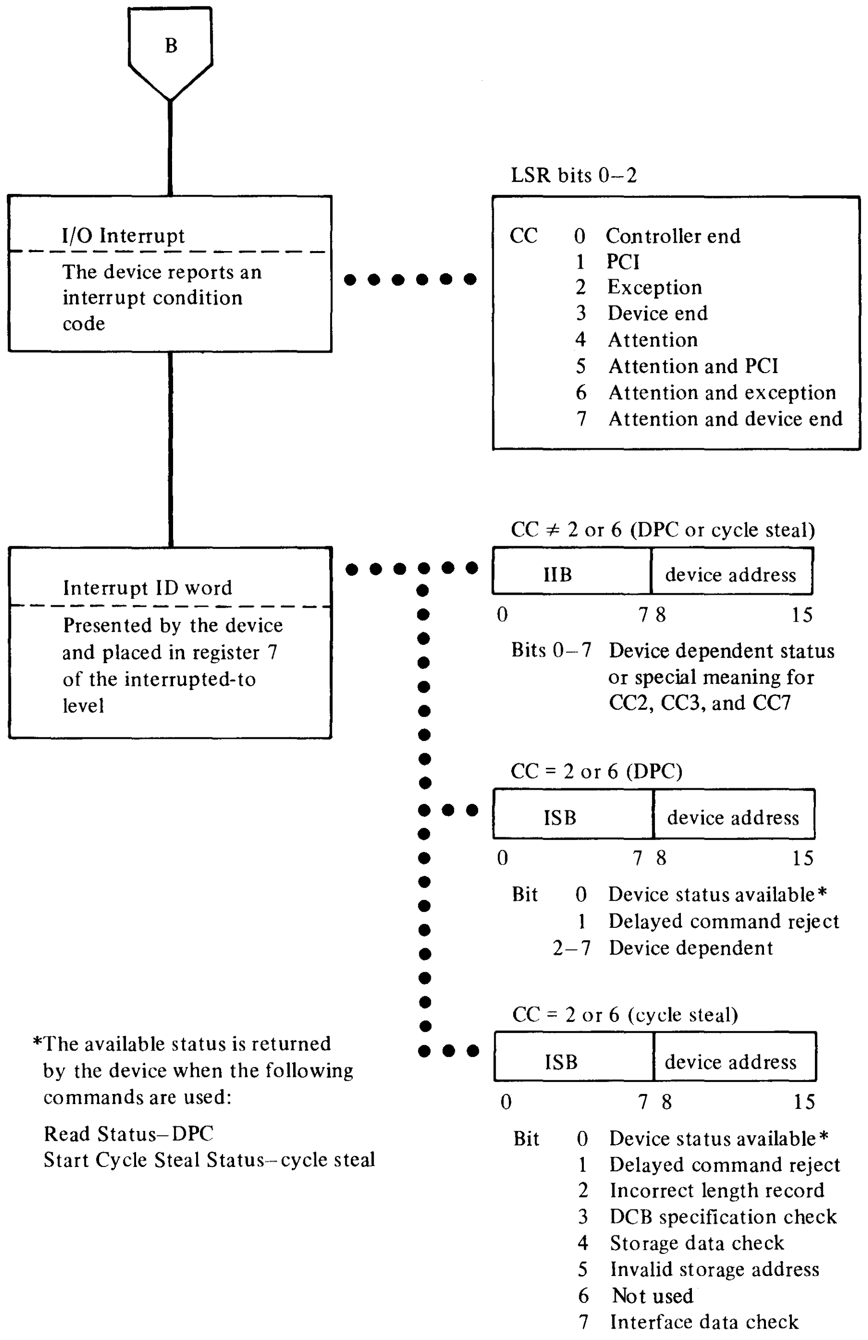

A
DPC or cycle steal operation
DPC
B
Cycle steal
Residual parameters and device dependent status
Returned by the device if this is a Start Cycle Steal Status operation
DCB word 7
data address
residual address
cycle steal status word 1
device dependent status
device dependent status
0
15
Residual status block
Stored into main storage if the device uses SE and the SE bit is set to one
DCB word 4
status address
residual byte count
EOC
reserved
status flags
NE
device dependent status
0
15
B

Figure 4-7. Condition codes, status words, and status bytes received from a device (Part 2)



I/O Interrupt

The device reports an interrupt condition code

LSR bits 0–2

CC	0	Controller end
	1	PCI
	2	Exception
	3	Device end
	4	Attention
	5	Attention and PCI
	6	Attention and exception
	7	Attention and device end

Interrupt ID word

Presented by the device and placed in register 7 of the interrupted-to level

CC ≠ 2 or 6 (DPC or cycle steal)

IIB (0–7)	device address (8–15)

Bits 0–7 Device dependent status or special meaning for CC2, CC3, and CC7

CC = 2 or 6 (DPC)

ISB (0–7)	device address (8–15)

Bit	0	Device status available*
	1	Delayed command reject
	2–7	Device dependent

CC = 2 or 6 (cycle steal)

ISB (0–7)	device address (8–15)

Bit	0	Device status available*
	1	Delayed command reject
	2	Incorrect length record
	3	DCB specification check
	4	Storage data check
	5	Invalid storage address
	6	Not used
	7	Interface data check

*The available status is returned by the device when the following commands are used:

Read Status–DPC
Start Cycle Steal Status–cycle steal

Figure 4-7. Condition codes, status words, and status bytes received from a device (Part 3)

IO Instruction Condition Codes

These codes are reported during execution of an Operate I/O instruction.

Condition code (CC) value	*LSR position Even*	*Carry*	*Over-flow*	*Reported by*	*Meaning*
0	0	0	0	channel	Device not attached
1	0	0	1	device	Busy
2	0	1	0	device	Busy after reset
3	0	1	1	chan/dev	Command reject
4	1	0	0	device	Intervention required
5	1	0	1	chan/dev	Interface data check
6	1	1	0	controller	Controller busy
7	1	1	1	chan/dev	Satisfactory

CC=0 *Device not attached.* Reported by the channel when the addressed device is not attached to the system.

CC=1 *Busy.* Reported by the device when it is unable to execute a command because it is in the busy state. The device enters the busy state upon acceptance of a command that requires an interrupt for termination. It exits the busy state when the processor accepts the interrupt. Certain devices also enter the busy state when an external event occurs that results in an interrupt. When this condition code is reported, a subsequent priority interrupt from the addressed device always occurs.

CC=2 *Busy after reset.* Reported by the device when it is unable to execute a command because of a reset and the device has not had sufficient time to return to the quiescent state. No interrupt occurs to indicate termination of this condition.

CC=3 *Command Reject.* Reported by the device or the channel when:

1. A command is issued (in the IDCB) that is outside the device command set.
2. The device is in an improper state to execute the command.
3. The IDCB contains an incorrect parameter. For example: an odd byte DCB address, or an incorrect function/modifier combination.

When a cycle-steal device reports command reject, it does not fetch the DCB.

CC=4 *Intervention required.* Reported by the device when it is unable to execute a command due to a condition requiring manual intervention to correct.

CC=5 *Interface data check.* Reported by the device or the channel when a parity error is detected on the I/O data bus during a data transfer.

CC=6 *Controller busy.* This condition is reported by a device controller, not the addressed device, when the controller is busy. It is reported only by controllers that have two or more devices attached (each device having a unique address). When this condition code is reported, a subsequent controller-end interrupt always occurs.

CC=7 *Satisfactory.* Reported by the device on the channel when it accepts the command.

These condition codes are mutually exclusive and have a priority sequence. That is, beginning with CC=7, each successive condition code through CC=0 takes precedence over the previous code. For example, if a device cannot accept a command because it is busy, it reports CC=1, irrespective of error conditions encountered.

Note. The only exception is CC=6 (controller busy). This condition code may have a variable priority depending on the particular controller.

Interrupt Condition Codes

These condition codes are reported by the device or controller during priority interrupt acceptance.

Condition code(CC) value	*LSR position Even*	*Carry*	*Over-flow*	*Reported by*	*Meaning*
0	0	0	0	controller	Controller end
1	0	0	1	device	Program controlled interrupt (PCI)
2	0	1	0	device	Exception
3	0	1	1	device	Device end
4	1	0	0	device	Attention
5	1	0	1	device	Attention and PCI
6	1	1	0	device	Attention and exception
7	1	1	1	device	Attention and device end

CC=0 *Controller end.* Reported by a controller when *controller busy* (IO instruction condition code) has been previously reported one or more times. It signifies that the controller is now free to accept I/O commands for devices under its control. The device address reported with controller end is always the lowest address (numerical value) of the group of devices serviced by the controller. The interrupt information byte, in the interrupt ID word, is set to zero.

CC=1 *Program controlled interrupt.* Reported when the interrupt indicates that a DCB with the PCI bit set to one has been transferred by cycle steal to the device and no error or exception condition has occurred. The device places a DCB identifier into the interrupt information byte.

CC=2 *Exception.* Reported when an error or exception condition is associated with the interrupt. The condition is described in the interrupt status byte (ISB) or in device dependent status words.

CC=3 *Device end.* Reported when no error, exception, or attention condition has occurred during the I/O operation, and the interrupt is not the result of a PCI. For example: an operation has terminated normally.

Note. If the device has come to a normal end while using suppress exception (SE bit set to one) and an exception was suppressed since the last *Start* command, then bit zero of the interrupt status byte is set to one. The condition is called *permissive device end* (PDE) and indicates that errors or exceptions have been suppressed. Related status information is contained in the residual status block.

CC=4 *Attention.* Reported when the interrupt was caused by an external event rather than execution of an Operate I/O instruction. Additional status information is not provided unless the event requires further definition; for example, code bits for a keyboard function.

CC=5 *Attention and PCI.* Reported when attention and PCI are both present. In this case the interrupt information byte contains the DCB identifier, and the attention must be singular in meaning.

CC=6 *Attention and exception.* Reported when attention and exception are both present.

CC=7 *Attention and device end.* Reported when attention and device end are both present. For this condition code, device end could also mean permissive device end. Refer to interrupt condition code 3.

The interrupt condition codes are mutually exclusive with each other but have no priority sequence.

I/O Status Information

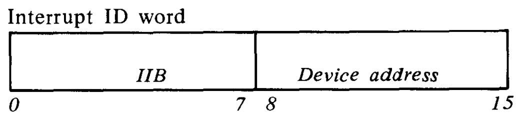
Some form of status information is transferred from the device to the processor as a result of:

- A read status operation (see *Read Status* command in this chapter).
- A start cycle steal status operation (see *Start Cycle Steal Status Operation* in this chapter).
- Storing a residual status block (see *Cycle-Steal Device Options* in this chapter).
- A priority interrupt.

The interrupt status information is detailed in the following two sections (*Interrupt ID Word* and *Interrupt Status Byte*).

Interrupt ID Word

Acceptance of an I/O interrupt causes the device to present an interrupt ID word to the processor. Presentation of the interrupt ID word is explained in Chapter 3 (see I/O Interrupts). This word has the following format:



Bits 0–7 *Interrupt information byte (IIB).* For interrupt condition codes 2 and 6, the IIB has a special format and is called an interrupt status byte (ISB). Refer to *interrupt status byte* in this section. For most other interrupt condition codes, implementation of the IIB is device dependent. Exceptions are:

1. CC=0. The IIB is set to zero.
2. CC=3 or 7. Bit zero may be set to one if suppress exception is in effect.

Bits 8–15 *Device address.* This byte contains the address of the interrupting device.

Interrupt Status Byte (ISB)

The ISB is a special format of the interrupt information byte (IIB) and contains detailed information on the nature of the interrupt. The ISB is reported only for error or exception conditions (interrupt condition codes 2 or 6). The ISB bits are normally set as a result of:

1. Status errors that occur during a DPC operation that cannot be indicated via a condition code.
2. Status errors that occur during a cycle steal operation.

The ISB is never reported as zero unless the condition code presentation of 2 or 6 is singular in meaning for devices that do not cycle steal. After the processor has accepted the interrupt request, the device resets the ISB.

Bits 0–7 of the two special formats are explained in the following sections.

ISB (devices that do not cycle steal):

Bit 0 *Device dependent status available.* This bit set to one signifies that additional status information is available from the device. The information content and method of reading is described in the individual device publications.

Bit 1 *Delayed Command reject.* This bit is set to one if the device cannot execute the command (specified in the IDCB) due to an incorrect parameter in the IDCB, or it cannot execute the command due to its present state. For example: (1) the IDCB specifies an incorrect function/modifier combination, or (2) the device is temporarily not ready. The operation in progress is terminated. Command reject is set in the ISB only if the device cannot report IO instruction condition codes for the condition.

Bits 2–7 *Device dependent.* These bits, if used, are described in the individual device publications.

ISB (cycle stealing device):

Bit 0 *Device dependent status available.* This bit, when set to one, signifies that: (1) additional status information is available from the device, or (2) the device is in an improper state to execute a function specified by a DCB.

The operation is terminated. The content and method of reading the additional status information is described in the individual device publications.

Note. When bit 0 of the ISB is equal to one and bits 2–7 are zeros, the contents of the residual-address word (cycle steal status) are defined by the device.

Bit 1 *Delayed command reject.* This bit is set to one if the device cannot execute the command due to one of the following conditions:

1. The IDCB contains an incorrect parameter. Examples are (a) an odd-byte DCB address, or (b) an incorrect function/modifier combination.
2. The present state of the device, such as a *not ready* condition, prevents execution of an I/O command specified in the IDCB.

Delayed command reject is set in the ISB only if the device cannot report IO instruction condition codes for the condition. The operation is terminated. The DCB is not fetched.

Bit 2 *Incorrect length record.* This bit is set to one when the device encounters a mismatch between byte count and actual record length after beginning execution of the DCB. For example: the byte count is reduced to zero (with chaining flag off) and no end of record encountered. Incorrect length record is not reported when the SE bit in the control word is set to one. Reporting of incorrect length record is a device dependent feature and may be implemented regardless of the suppress exception feature. The operation is terminated.

Bit 3 *DCB specification check.* This bit is set to one when the device cannot execute a command due to an incorrect parameter specification in the DCB. Examples are (1) an odd-byte DCB chaining or status address, (2) the byte count is odd for a word-only device, (3) an odd-byte data address for a word-only device, (4) an invalid command or invalid bit settings in the control word, or (5) an incorrect count.

The operation is terminated.

Bit 4 *Storage data check.* This error condition applies to cycle steal output operations only. If the bit is set to one, it indicates that the main storage location accessed during the current output cycle contained bad parity. Parity in main storage is not corrected. The device terminates the operation. The bad parity data is not transferred to the I/O data bus. No machine check condition occurs. See Figure 4-8 for other bits that may be present.

Bit 5 *Invalid storage address.* When set to one, this bit indicates that, during a cycle steal operation, the device has presented a main storage address that is outside the storage size of the system.

Invalid storage address can occur on a data transfer or on a DCB fetch operation. In either case, the cycle steal operation is terminated. See Figure 4-8 for other bits that may be present.

Bit 6 Not used.

Bit 7 *Interface data check.* This bit set to one indicates that a parity error has been detected on the I/O data bus during a cycle steal data transfer. The condition may be detected by the channel or the I/O device. In either case, the operation is terminated. See Figure 4-8 for other bits that may be present.

Conditions			*Bit results*			
I/O operation	*Invalid storage address*	*Incorrect data parity*	*4*	*5*	*7*	
Write	No	No	0	0	0	
Write	Yes	No	0	1	0	
Read	No	No	0	0	0	
Read	Yes	No	0	1	0	*
Write	No	Yes	0	0	1	
Write	Yes	Yes	0	1	1	
Read	No	Yes	1	0	1	
Read	Yes	Yes	1	1	1	

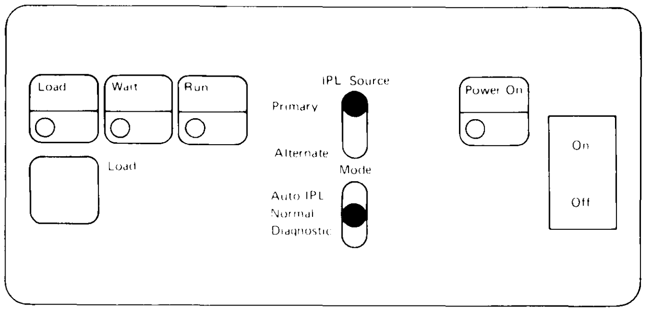
*This condition not possible.

Figure 4-8. Bit result chart

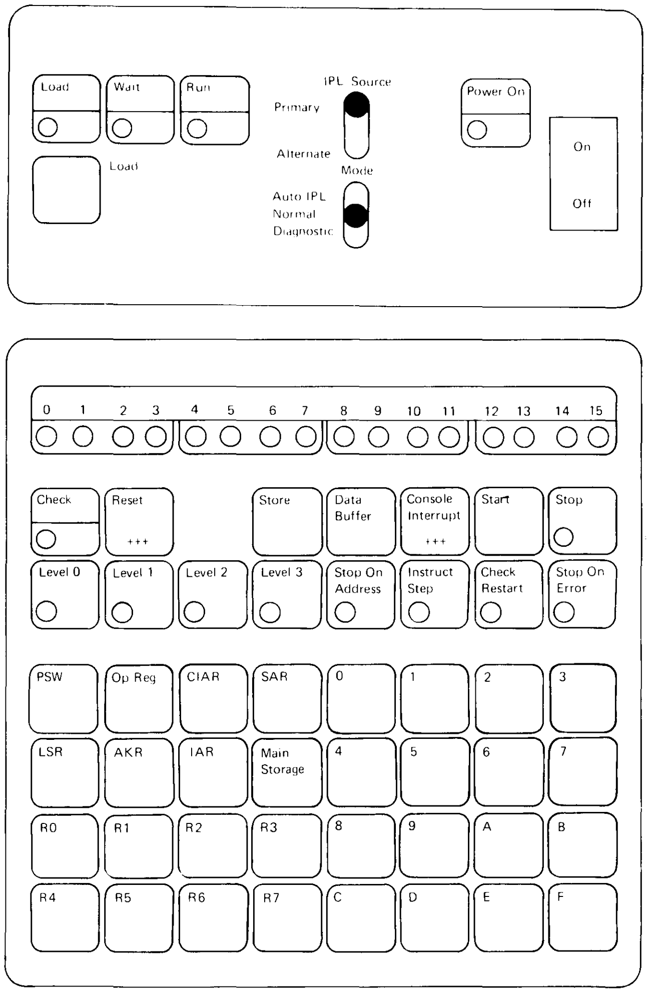
Chapter 5. Console

There are two configurations of consoles available for the IBM 4953 Processor. The Basic Console is standard, and remains with the processor. The Programmer Console is an optional feature that is added to the processor when the option is selected.

Configuration 1
Basic Console



Configuration 2
Basic Console and
Programmer Console



Configuration 1 is primarily intended for those systems that are totally dedicated to a particular application, where operator intervention is not needed during the execution of the application.

Configuration 2 is aimed at operator oriented systems where various programs are entered and executed during the day. This type of environment requires a more versatile console arrangement for program and machine problem determination, and for manual alteration of data and programs in storage.

Basic Console

Each IBM 4953 Processor comes equipped with the standard Basic Console. The Basic Console provides the following capabilities:

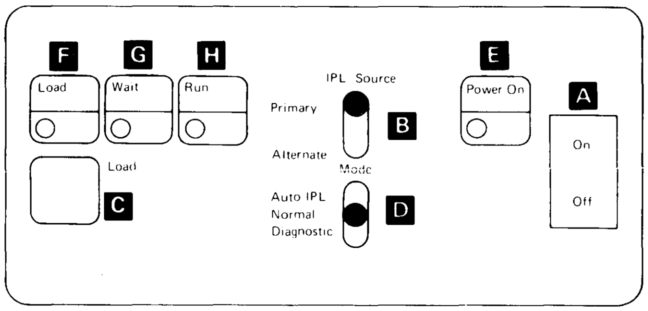
- Power On/Off switch for the processor card file
- Load key for IPL (initial program load)
- Load, Wait, Run, and Power On indicators
- Mode switch to select: Diagnostic mode, Auto IPL, or Normal mode
- IPL source switch to select a primary or alternate IPL device.

Keys and Switches

A	Power On/Off	When this switch is placed in the On position, power is applied to the processor card file. After all power levels are up, the Power On indicator is turned on. When this switch is placed in the Off position, power is removed from the processor card file and the Power On indicator is turned off.
B	IPL Source	This switch selects the I/O device to be used for program loading. In the Primary position, the device that was pre-wired as the primary IPL device is selected. In the Alternate position, the device that was pre-wired as the alternate IPL device is selected.
C	Load	Pressing this key causes a *system reset*, then the initial program load (IPL) sequence is started. The Load indicator is turned on and remains on until the IPL sequence is completed. When the IPL is completed, instruction execution begins at location zero on level zero.
D	Mode	This switch has the following positions: ● Auto IPL – In this position, an IPL is initiated after a successful power-on sequence. Bit 13 of the PSW is set to indicate to the software that an automatic IPL was performed. In this mode STOP instructions are treated as no-ops. ● Normal – This position is for attended operation. In this mode STOP instructions are treated as no-ops. ● Diagnostic – This position has no function without the Programmer Console. This position places the processor in a diagnostic mode if the Programmer Console is attached. When the processor is in diagnostic mode, STOP instructions cause the processor to enter stop state.

Indicators

E	Power On	On when the proper power levels are available to the system.
F	Load	On when the machine is performing an initial program load (IPL).
G	Wait	On when an instruction that exits the active level has been executed and no other levels or interrupts are pending.
H	Run	On when the machine is executing instructions.



Programmer Console

The Programmer Console is an optional feature that can be ordered with the IBM 4953 Processor or may be field installed at a later date. The Programmer Console provides the following capabilities:

- Start and stop the processor.
- Display or alter any storage location.
- System reset.
- Select any of the four interrupt levels for display or alter purposes.
- Display or alter the storage address register (SAR), instruction address register (IAR), console data buffer, or any general purpose register.
- Display but not alter the level status register (LSR), current instruction address register (CIAR), op register, or processor status word (PSW).
- Stop-on-address.
- Stop-on-error.
- Instruction step.
- Check restart.
- Request a console interrupt.
- Check indicator, on when a machine check or program check class interrupt has occurred.

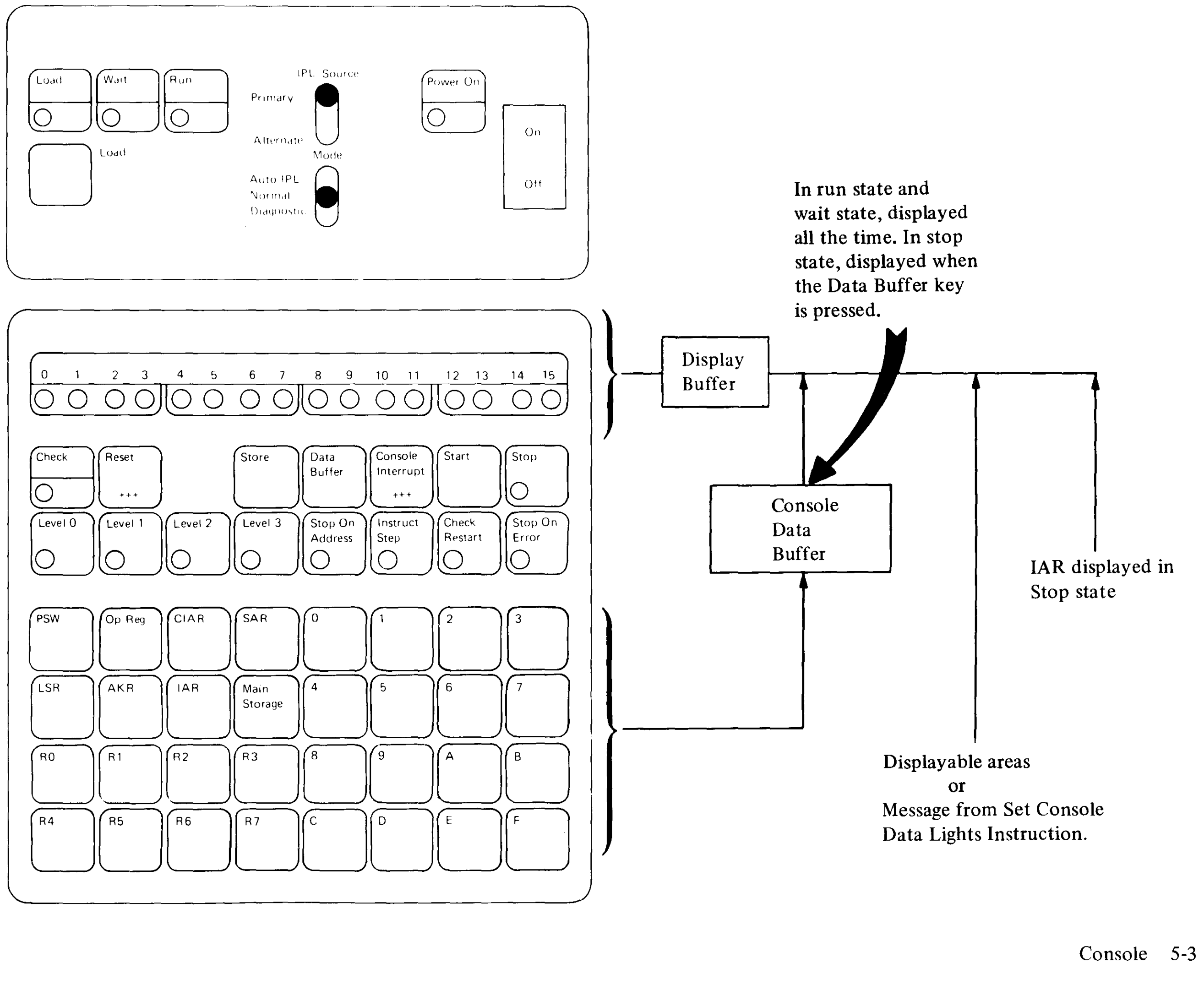
The Programmer Console is touch sensitive with a tone generator providing an audio response tone whenever a key depression has been accepted and serviced by the processor.

Console Display

When the processor is in run state or wait state, the console data buffer is displayed in the data display indicators. The only exception to this is when in run state a Set Console Data Lights instruction writes a message to the data display. This message remains displayed until the processor enters stop state or the Data Buffer Key is pressed. When the Data Buffer Key is pressed, the console data buffer is again displayed in the data display indicators.

When the processor enters stop state, the IAR is displayed in the data display indicators. Any system resource that has a corresponding select key on the console can be displayed while in stop state. Once data has been entered into the console data buffer, it remains there until other data is entered. The console data buffer can be displayed at any time, during either run state, wait state, or stop state, by pressing the Data Buffer key.

After a power-on reset, the data display indicators are all set on, and the level indicators are set off.



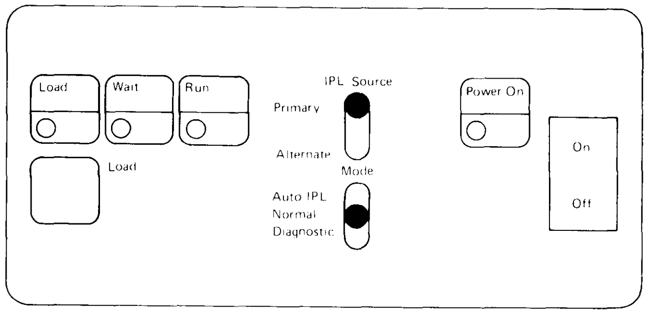
Indicators

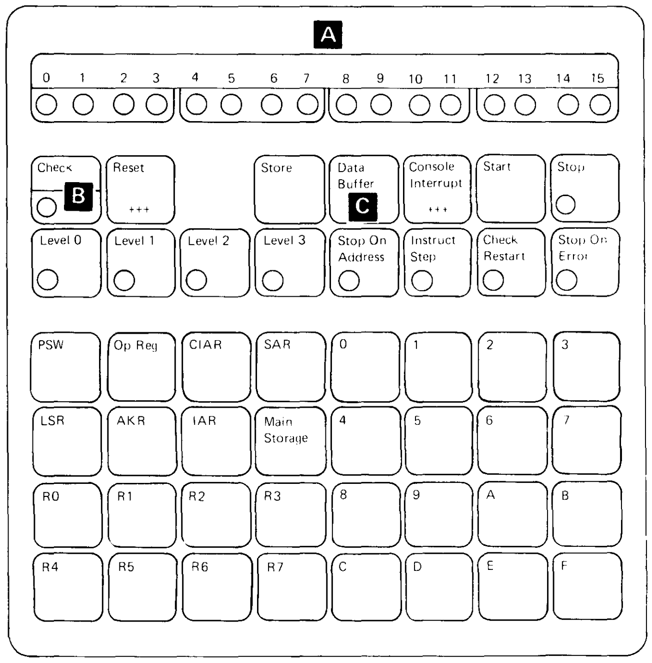
A Data Display

- When the processor is in run state, the console data buffer is displayed in the data display indicators.
- When the processor enters stop state, the IAR is displayed unless another system resource is selected.
- To display the contents of the console data buffer after a system resource has been displayed, press the Data Buffer key. **C**

B Check

On when a machine check, program check, or power/thermal warning class interrupt has occurred while in process mode or in stop-on-error mode. The check indicator remains on until either the check condition is cleared, or any console key is pressed while in the stop state. The check condition is cleared by the Reset key, Load key, or the execution of a Copy Processor Status and Reset instruction (which resets the check bits). If a main storage display of a location causes a parity error, an invalid storage address, or a specification check, the check indicator is turned on, or appears to stay on.



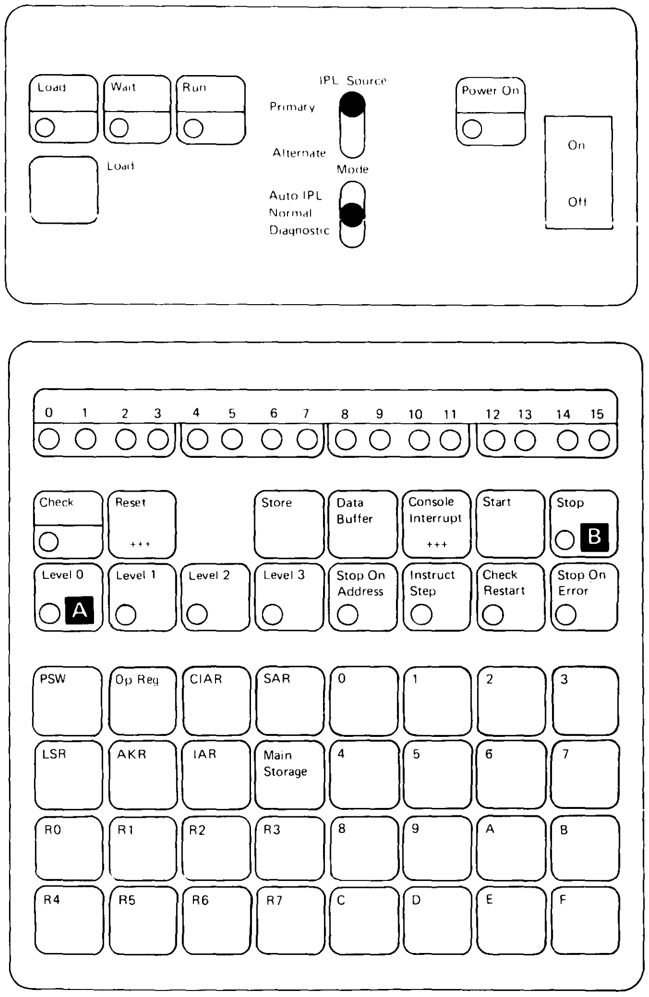


Combination Keys/Indicators

There are nine combination key/indicators:

- Level 0, 1, 2, and 3
- Stop
- Stop On Address
- Instruct Step
- Check Restart
- Stop On Error

A	Level 0–3	The current active level is always displayed by one of the level indicators. When in the stop state, pressing any of the level keys causes that level to be selected and the associated indicator is turned on.
B	Stop	This indicator is on when the processor is in the stop state. Stop state is entered in the following ways: • By pressing the Stop key. – In run state the current instruction is completed. – In wait state, stop state is entered directly. • By execution of the Stop instruction (diagnostic mode only). • When an address compare occurs in stop-on-address mode. • When an error occurs in stop-on-error mode. • By pressing the Reset key. • When a power-on reset occurs. • By selecting the Instruction step mode while in run state.



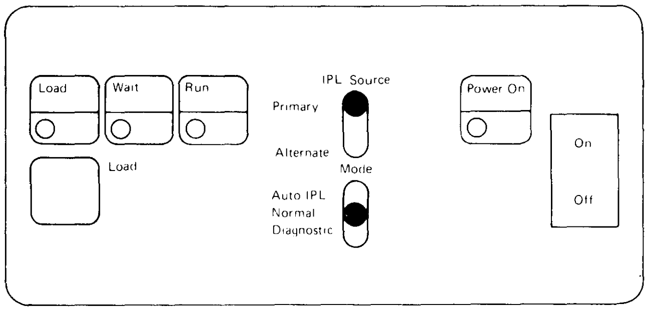
The Stop On Address key and the Instruct Step key are mutually exclusive. When one is pressed, the other is reset if it was on.

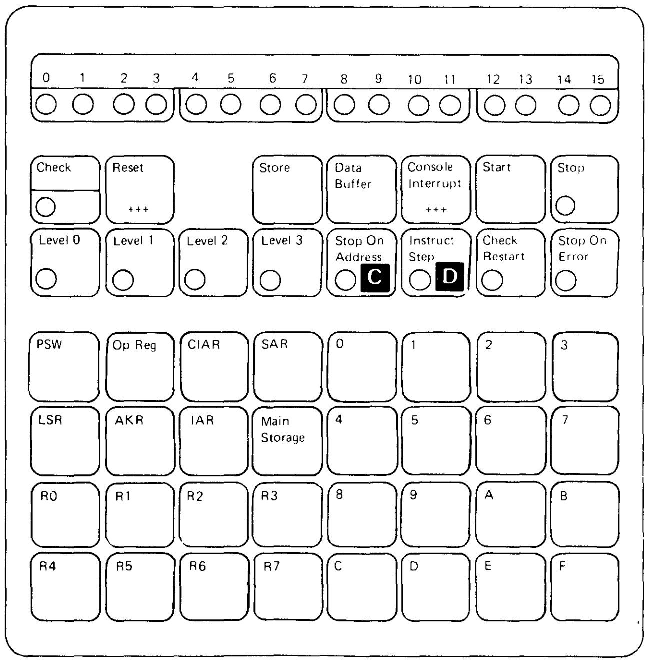
C	Stop on Address	This key places the processor in stop on address mode. Pressing the Stop On Address key a second time resets stop on address mode and turns off the indicator.
D	Instruct Step	Pressing the Instruct Step key places the processor in instruction step mode and turns the Instruct Step indicator on. The Stop On Address indicator is turned off if it was on.

If the processor is in run state, pressing this key causes the processor to enter stop state. Pressing the Instruct Step key a second time resets instruction step mode, the processor remains in stop state.

To operate in instruction step mode:

- Key the desired starting address and store into the IAR.
- Press the Instruct Step key.
- Press the Start key. The instruction located at the selected address is executed, the processor returns to stop state. The IAR is updated to the next instruction address, this address is displayed in the data display indicators.
- Each subsequent depression of the Start key causes one instruction to be executed and the IAR is updated to the next instruction address.





Stop On Address Mode

Processor must be in stop state to set the compare address.

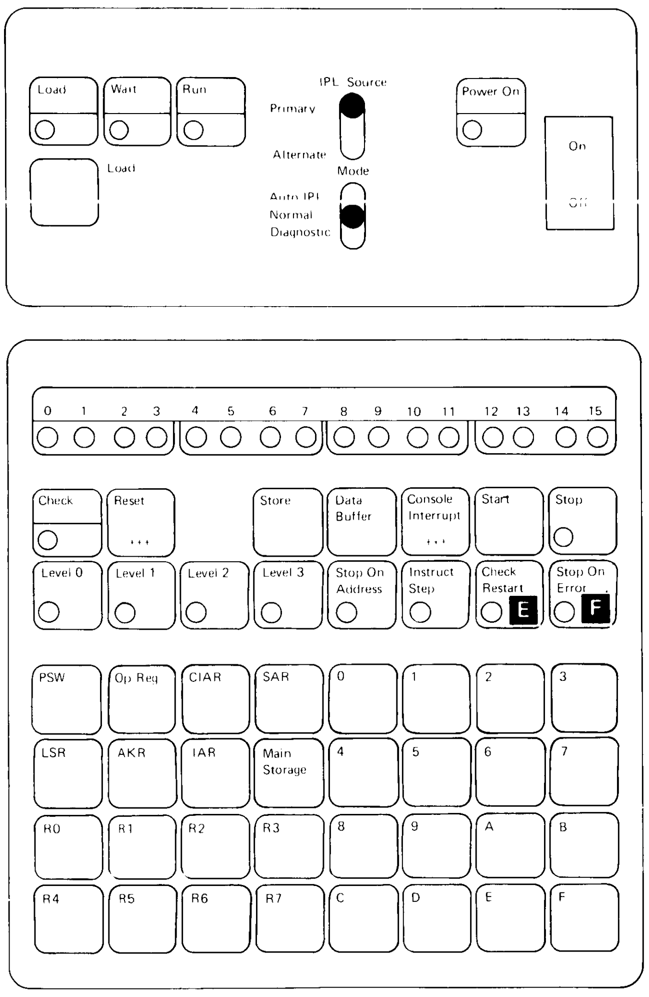
1. Press Stop On Address Key.
2. Key in selected address.
3. Press Store Key. The selected address is placed in the stop on address buffer.
4. Press Start Key. Execution begins at current IAR address on the current level.

When the selected address is loaded into the IAR, the processor enters stop state. To exit stop state press the Start key; execution begins at the next sequential address.

Note. When running in Stop on Address Mode, instruction execution time is increased by 7.8 microseconds per instruction.

The Check Restart key and the Stop On Error key are mutually exclusive. When one is pressed the other is reset if it was on.

E	Check Restart	Pressing this key places the processor in check restart mode. While in this mode, a program check, or machine check, or a power/thermal warning class interrupt causes the processor to be reset and execution to restart at address zero on level zero. *Note.* The power/thermal warning class interrupt is controlled by the summary mask.
F	Stop On Error	Pressing the Stop On Error key places the processor in stop on error mode. Any program check, machine check, or power/thermal warning causes the processor to enter stop state. To determine the cause of the error, display the PSW (see table). To restart the processor, press the Reset key then the Start key. Pressing only the Start key allows the processor to proceed with the class interrupt as if stop mode has not occurred. Note that the check indicator may have been turned off while in stop state. After the class interrupt routine is completed, control may be returned to the instruction that caused the error and an attempt to re-execute the instruction may be made. Note that some instructions are not re-executable because operand registers or storage locations were changed before the instruction was terminated because of the initial error. In these cases, the operator must be familiar with the program because manual restoration of affected locations must be made before restart is attempted. *Note.* The power/thermal warning class interrupt is controlled by the summary mask.



Processor Status Word (PSW) Table

Bit	*Meaning*	*Category*
0	Specification check	Program check
1	Invalid storage address	Program check
2	Privilege violate	Program check
3	Not used	
4	Invalid function (may be program check)	Soft exception
5	Not used	
6	Stack exception	Soft exception
7	Not used	
8	Storage parity check	Machine check
9	Not used	
10	CPU control check	Machine check
11	I/O check	Machine check
12	Sequence indicator	Status flag
13	Auto-IPL	Status flag
14	Not used	
15	Power/thermal warning	Power/Thermal

Bits not used are always zero.

Keys and Switches

A Reset — This key initiates a system reset that performs the following functions:

- Interrupt mask set to all levels enabled.
- LSR on level zero – indicators set to zero, summary mask enabled, supervisor state and in-process flag turned on, trace disabled.
- LSRs for levels 1–3 set to zeros.
- PSW set to zero.
- SAR set to zeros.
- CIAR set to zeros.
- IAR on level zero – set to zeros.

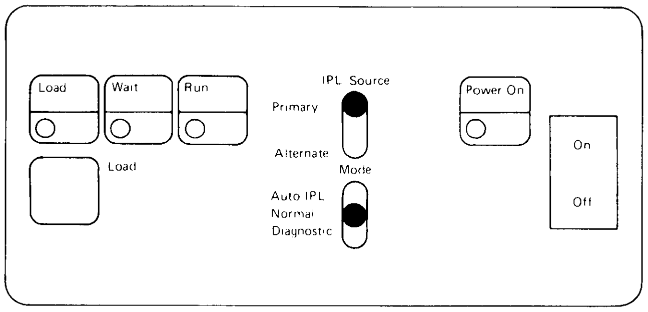
After the system reset is completed, the processor is placed in the stop state with stop indicator on.

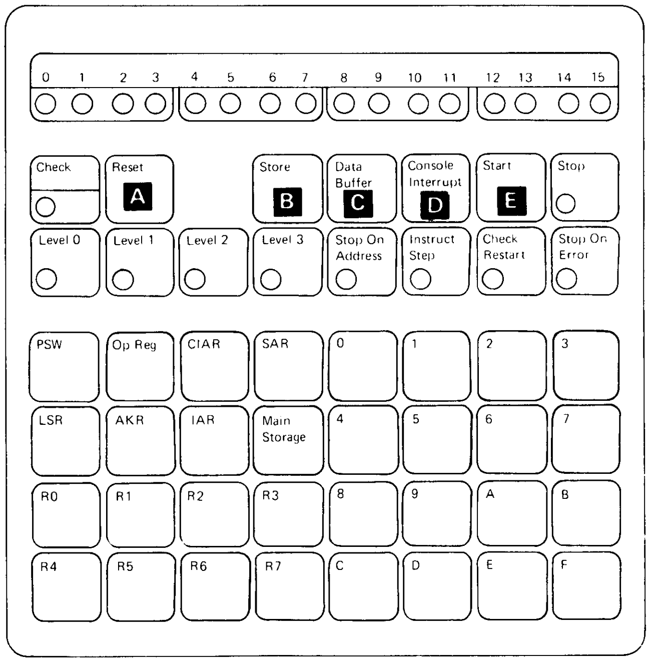
The following resources are not effected by system reset:

- General registers (all levels)
- IARs (levels 1–3)
- Main storage
- Console data buffer
- Stop on Address buffer.

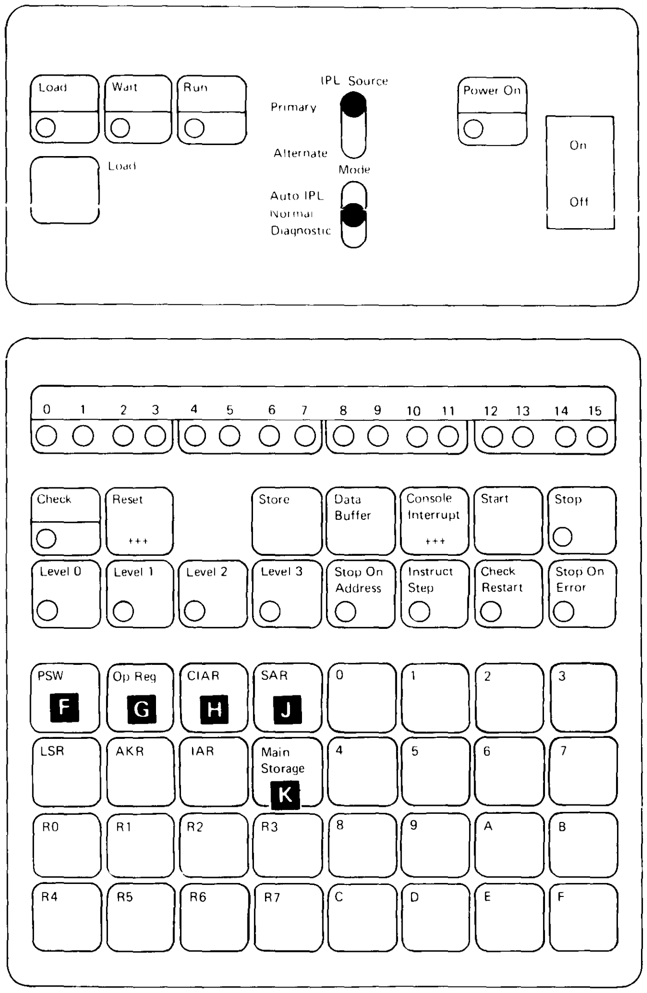
B	Store	This key is effective only when the processor is in stop state. Pressing this key causes the last data entry to be stored in the last selected resource.
C	Data Buffer	Pressing this key causes the contents of the console data buffer to be displayed in the data display indicators.
D	Console Interrupt	The effect of this key depends on the state of the processor. If the processor is in the stop or load states, this key has no effect. If the processor is in the run or wait state and the summary mask is enabled, a console class interrupt occurs. *Note.* If the summary mask is enabled by the program while the key is being activated, a console class interrupt occurs.
E	Start	Effective in stop state only. Stop state is exited and the processor resumes execution at the address in the IAR on the current level. If stop state was entered from system reset, execution begins at address zero, level zero. If stop state was entered from wait state, the processor returns to wait state.

Note. The Reset and Console Interrupt keys have an indication (+++) on the face of the keys. This signifies that additional pressure must be used to activate these keys. This is to minimize the possibility of the operator inadvertently activating these functions.





F	PSW	Pressing this key selects the processor status word. The contents of the PSW are displayed in the data display indicators. Data cannot be stored into the PSW from the console.
G	Op Reg	Pressing this key selects the Op register and displays the contents in the data display indicators. Data cannot be stored into the Op register from the console.
H	CIAR	Pressing this key after entering stop state causes the address of the instruction just executed to be displayed. Data cannot be stored into the CIAR from the console.
J	SAR	Pressing this key while in stop state displays the contents of the storage address register. An address can be stored into the SAR to address main storage for display or store operations. Bit 15 of the SAR cannot be set from the console.
K	Main Storage	Pressing this key selects main storage as the facility to be accessed by the console. When this key is pressed, the contents of the main storage location addressed by the SAR is displayed in the data display indicators. Procedures for displaying and storing main storage are provided in subsequent sections of this chapter.



Level Dependent Keys

The following keys select registers that are duplicated in hardware for each of the four interrupt levels:

- LSR
- IAR
- General purpose registers 0–7

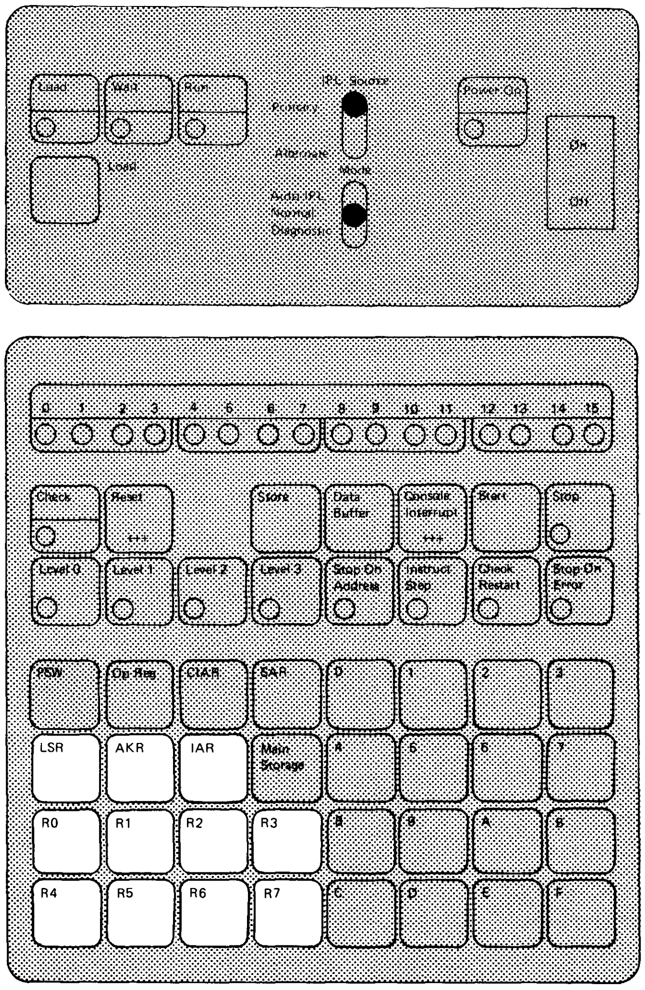
Pressing any of these keys, once a level has been selected, causes the contents of that register to be displayed in the data display indicators.

The level status register (LSR) is displayable only; data cannot be stored into this register.

The AKR key is not functional on the 4953 Processor and does not respond with an audio tone when pressed.

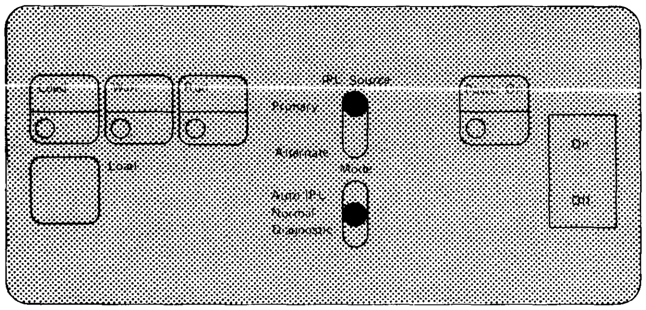
Bit 15 of the IARs cannot be changed from the console.

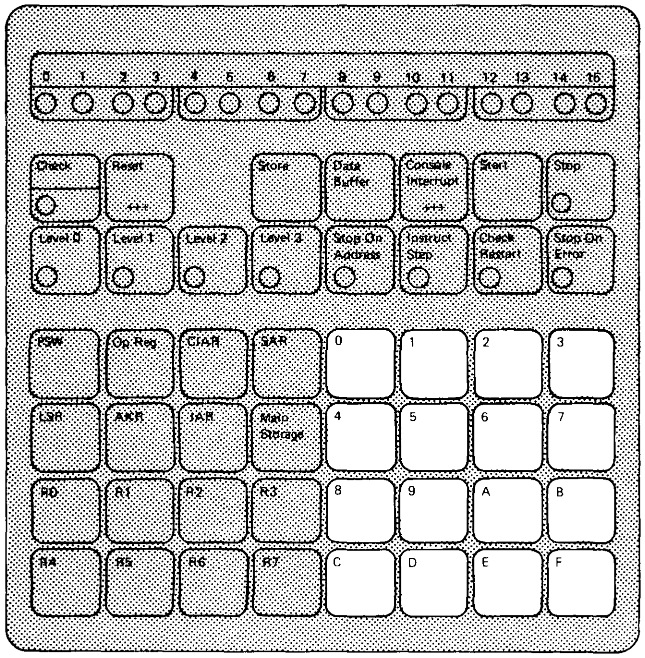
Pressing the Store key after selecting an LSR or AKR results in no action taken and no audio tone response.



Data Entry Keys

The sixteen data entry keys are used to enter data into the selected resource.





Example:

Data to be entered: F3A8

Action	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Press data entry key F	○	○	○	○	○	○	○	○	○	○	○	○	●	●	●	●
Press data entry key 3	○	○	○	○	○	○	○	○	●	●	●	●	○	○	●	●
Press data entry key A	○	○	○	○	●	●	●	●	○	○	●	●	●	○	●	○
Press data entry key 8	●	●	●	●	○	○	●	●	●	○	●	○	●	○	○	○

Data display indicators

Legend:

● - Indicator on

○ - Indicator off

Displaying Main Storage Locations

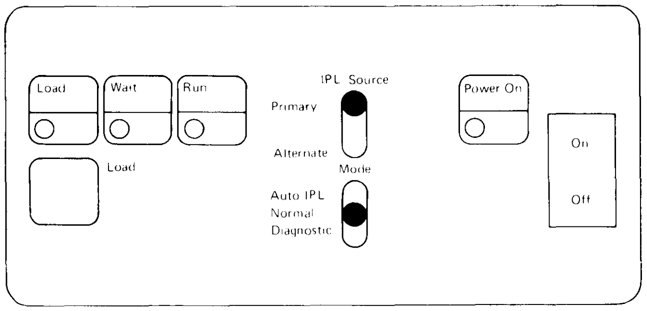
- Processor must be in stop state.

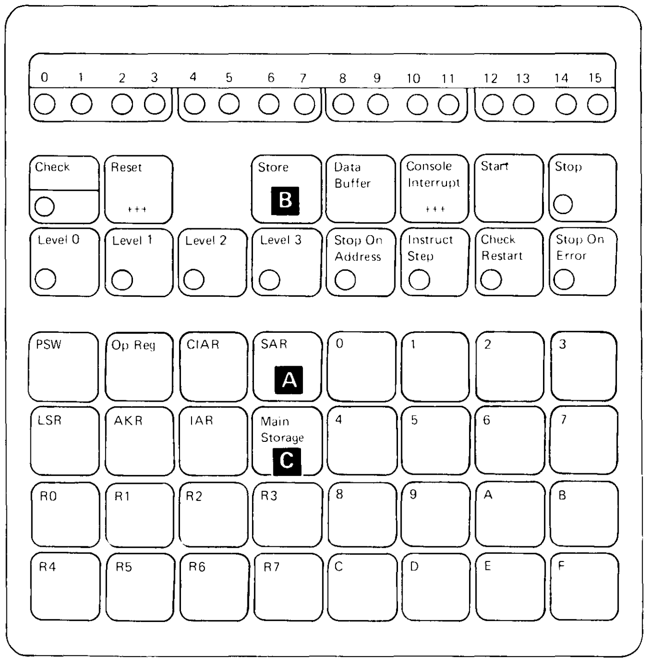
1. Press the SAR key. A
   The contents of the SAR are displayed in the data display indicators.
2. Key in the selected address (four hex characters). This address is displayed in the data display indicators.
3. Press the Store key. B
   The address that is displayed is stored into the SAR.
4. Press the Main Storage key. C
   The contents of the addressed storage location are displayed in the data display indicators. To display sequential main storage locations, continue pressing the Main Storage key. The storage address is incremented by +2 each time the Main Storage key is pressed, and the contents of the addressed location are displayed.

Storing Into Main Storage

- Processor must be in stop state.

1. Press the SAR key. A
   The current contents of the SAR are displayed in the data display indicators.
2. Key in the selected address (four hex characters). The address is displayed in the data display indicators.
3. Press the Store key. B
   The address displayed in the data display indicators is stored into the SAR.
4. Press the Main Storage key. C
   The contents of the addressed storage location are displayed in the data display indicators.
5. Key in the data that is to be stored into main storage. This data is displayed in the data display indicators.
6. Press the Store key. B
   The data that is displayed is stored at the selected storage location. Each subsequent pressing of the Store key causes the SAR to be incremented by +2, and the data stored at that location is displayed.





Displaying Registers

- Processor must be in stop state.

1. Select the proper level by pressing the appropriate Level key. **A**

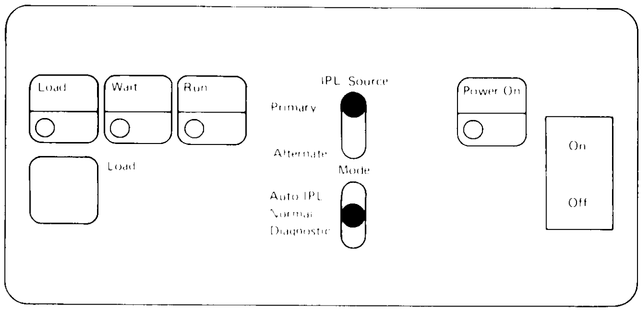
   The contents of any register associated with the selected level can now be displayed by pressing the register key.
2. Press the desired register key. The contents of that register are displayed in the data display indicators. **B**

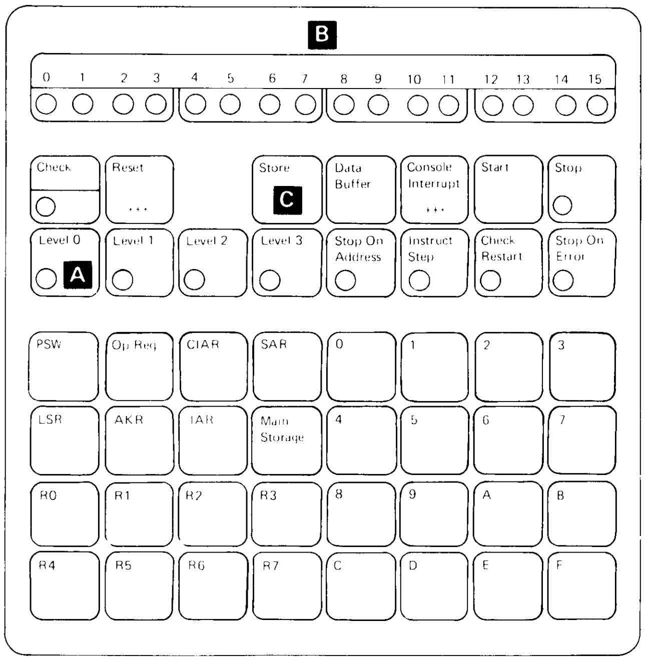
Storing Into Registers

- Processor must be in stop state.

1. Select the proper level by pressing the appropriate Level key. **A**
2. Press the key for the register where data is to be stored. The contents of that register are displayed in the data display indicators. **B**
3. Key in the data that is to be stored. This data is displayed in the data display indicators.
4. Press the Store key. **C**

   The data that is displayed is stored into the selected register.





Chapter 6. Instructions

The instructions for the IBM 4953 Processor are described in this chapter. A complete listing of instruction formats is contained in Appendix B. Instruction timings are contained in Appendix A. Indicator settings are listed for each instruction. For additional indicator information, refer to *Indicators* in Chapter 2.

Exception Conditions

Exception conditions that might occur during instruction execution are shown in abbreviated form with each instruction description. Refer to the following sections for a detailed description of these conditions.

Program Check Conditions

Invalid Function

(1) An illegal operation code or function combination is encountered during instruction execution, or (2) while in supervisor state, the processor attempts to execute one of the following instructions: operation code 01011 with a function of 0001 or 1001.

A program check class interrupt occurs with *invalid function* (bit 4) set in the PSW. See *Processor Status Word* in Chapter 3 for a list of invalid functions.

Invalid Storage Address

Instruction Word or Operand. One or more words of the instruction or the effective address is outside the installed storage size of the system. The instruction is suppressed unless otherwise noted in the individual instruction description.

A program check class interrupt occurs with *invalid storage address* (bit 1) set in the PSW.

Privilege Violate

Privileged Instruction. A privileged instruction is encountered while in problem state. The instruction is suppressed.

A program check class interrupt occurs with *privilege violate* (bit 2) set in the PSW. See *Processor Status Word* in Chapter 3 for a list of privileged instructions.

Specification Check

Operand Address. The generated effective address has violated an even-byte boundary requirement.

Indirect Address. When using addressing mode (AM=11), the indirect address is not on an even-byte boundary.

The instruction is suppressed unless otherwise noted in the individual instruction description. A program check class interrupt occurs with *specification check* (bit 0) set in the PSW.

Note. A specification check can also occur during a Supervisor Call (SVC) instruction if the SVC LSB pointer or the SVC SIA pointer violates an even-byte boundary requirement.

Soft Exception Trap Conditions

Invalid Function

(1) Operation code 00100 is attempted
(2) operation code 10110 is attempted or
(3) in supervisor state, operation code 01011 with a function of 0011 or 1011 is attempted. The instruction is suppressed. A soft-exception-trap class interrupt occurs with *invalid function* (bit 4) set in the PSW. See *Processor Status Word* in Chapter 3 for a list of invalid functions.

Stack Exception

(1) The stack is full and a Push instruction or a Store Multiple (STM) instruction is attempted, (2) the stack is empty and a Pop instruction or a Load Multiple and Branch (LMB) instruction is attempted, or (3) the stack cannot contain the number of words to be stored by a Store Multiple instruction.

The instruction is suppressed. A soft-exception-trap class interrupt occurs with *stack exception* (bit 6) set in the PSW.

Note. When the AM field is equal to 01, the register specified by the RB field is incremented before the class interrupt occurs.

Instruction Termination or Suppression

Exception conditions that occur during instruction processing might cause the instruction to be terminated or suppressed. When an instruction is terminated, partial execution has taken place and may have caused a change to registers, indicators, or main storage. When an instruction is suppressed, there has been no execution, therefore, no changes. Refer to *Exception Conditions* in the previous section.

Compatibility

The IBM 4955 Processor has more features and instructions than the IBM 4953 Processor. Consideration should be given to the differences between the processors when writing programs to permit possible future replacement with a larger system.

The following section describes the action taken by the IBM 4953 Processor when instructions that apply to the IBM 4955 Processor are encountered.

Soft Exception Trap

The following instructions cause a soft exception trap class interrupt with *invalid function*, bit 04 in the PSW, set to 1.

Op Code	*Function*
00100	all
01011	0011, 1011 (supervisor state only)

No Operation

In supervisor state the following instructions are recognized and executed as a No Operation instruction.

Op Code	*Function*
01011	0010, 0100, 1010, 1100
01100	110
01111	10010, 11010

Program Check

1. In supervisor state the following instructions cause a program check class interrupt with *invalid function*, bit 04 in the PSW, set to 1.

Op Code	*Function*
01011	0001, 1001
10110	all

2. In problem state the following instructions cause a program check class interrupt with *privilege violate*, bit 02 in the PSW, set to 1.

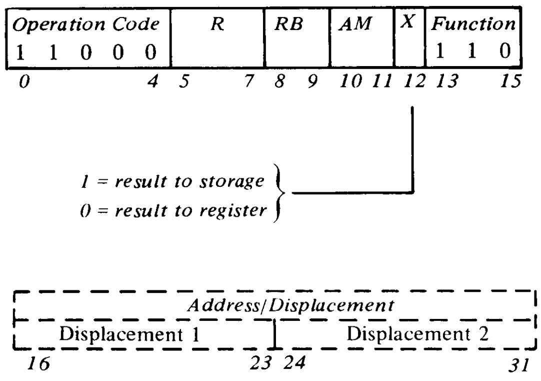
Op Code	*Function*
01011	0001, 0010, 0011, 0100, 1001, 1010, 1011, 1100
01100	110
01111	10010, 11010

Instruction Descriptions

The following descriptions are in alphabetical sequence based on assembler mnemonics. However, extended mnemonics are listed under the appropriate machine instruction. For example: branching and jumping instructions.

Add Byte (AB)

AB reg,addr4
addr4,reg



An add operation is performed between the least significant byte of the register specified by the R field and the location specified by the effective address in main storage. (See *Effective Address Generation* in Chapter 2.) Bit 12 of the instruction specifies the destination of the result. The source operand and high-order byte of the register are unchanged.

Indicators

Carry. Turned on if a carry is detected out of the high-order bit position of the byte. If no carry is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the sum cannot be represented in one byte; i.e., if the sum is less than -2^7 or greater than $+2^7-1$.

If an overflow occurs, the result contains the correct low-order eight bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

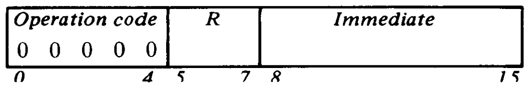
Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address).

Add Byte Immediate (ABI)

ABI byte,reg



The immediate field is expanded to 16 bits by sign propagation to the eight high-order bits. The field is then added to the contents of the register specified by the R field. The result is placed in the register specified by the R field.

Indicators

Carry. Turned on if a carry is detected out of the high-order bit position of the word. If no carry is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the sum cannot be represented in one word; i.e., if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

No program checks occur.

Add Carry Register (ACY)

ACY reg

Operation code		*R2*	*Function*
0 1 1 1 0	0 0 0		0 1 1 0 0
0 4	*5 7*	*8 10*	*11 15*

The value of the carry indicator on entry is added to the contents of the register specified by the R2 field, and the result is placed in the register specified by the R2 field. Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

Programming Note. This instruction can be used when adding multiple word operands. See *Indicators – Multiple Word Operands* in Chapter 2.

Indicators

Carry. Turned on if a carry is detected out of the high-order bit position of the word. If no carry is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the sum cannot be represented in one word; i.e., if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even. Unchanged.

Negative. Changed to reflect the result.

Zero. If on at entry, changed to reflect the result. If off at entry, it remains off.

Program Check Conditions

No program checks occur.

Add Doubleword (AD)

Register/Storage Format

AD reg,addr4
addr4,reg

Operation Code	R	RB	AM	X	Function
1 1 0 1 0					1 1 0
0 4	5 7	8 9	10 11	12	13 15

1 = result to storage
0 = result to register

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

An add operation is performed between the register pair specified by the R field (R and R+1) and the doubleword in main storage specified by the effective address. (See *Effective Address Generation* in Chapter 2.) Bit 12 of the instruction specifies the destination of the result. The source operand is unchanged.

If the R field equals 7, register 7 and register 0 are used.

Indicators

Carry. Turned on if a carry is detected out of the high-order bit position of the doubleword. If no carry is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the sum cannot be represented in the doubleword; i.e., if the sum is less than -2^{31} or greater than $+2^{31}-1$.

If an overflow occurs, the result contains the correct low-order 32 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Storage/Storage Format

AD addr5,addr4

Operation code	RB1	RB2	AM1	AM2	Fun
1 0 1 0 1					1 0
0 4	5 7	8 9	10 11	12 13	14 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

Address/Displacement	
Displacement 1	Displacement 2
32 39	40 47

The address arguments generate the effective addresses of two operands in main storage. (See *Effective Address Generation* in Chapter 2.) Doubleword operand 1 is added to doubleword operand 2. The result replaces operand 2. Operand 1 is unchanged.

Indicators

Carry. Turned on if a carry is detected out of the high-order bit position of the doubleword. If no carry is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the sum cannot be represented in the doubleword; i.e., if the sum is less than -2^{31} or greater than $+2^{31}-1$.

If an overflow occurs, the result contains the correct low-order 32 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. The instruction is terminated. If RB1 and RB2 specify the same register and AM1=01, the register is incremented before the program check interrupt occurs.

Specification Check. Even byte boundary violation (indirect address or operand address).

Add Word (AW)

Register/Register Format

AW reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			0 1 0 0 0
0 4	*5 7*	*8 10*	*11 15*

The contents of the register specified by the R1 field are added to the contents of the register specified by the R2 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field remain unchanged if R1 and R2 do not specify the same register.

Indicators

Carry. Turned on if a carry is detected out of the high-order bit position of the word. If no carry is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the sum cannot be represented in one word; i.e., if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

No program checks occur.

Register/Storage Format

AW reg,addr4
addr4,reg

Operation Code	*R*	*RB*	*AM*	*X*	*Function*
1 1 0 0 1					1 1 0
0 4	*5 7*	*8 9*	*10 11*	*12*	*13 15*

1 = result to storage
0 = result to register

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

An add operation is performed between the register, specified by the R field and the location specified by the effective address in main storage. (See *Effective Address Generation* in Chapter 2.) Bit 12 of the instruction specifies the destination of the result. The source operand is unchanged.

Indicators

Carry. Turned on if a carry is detected out of the high-order bit position of the word. If no carry is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the sum cannot be represented in one word; i.e., if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

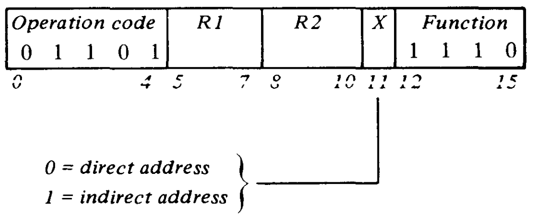
Program Check Conditions

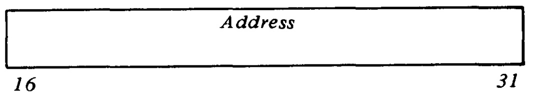
Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Storage to Register Long Format

AW longaddr,reg





The contents of the main storage location specified by an effective address are added to the contents of the register specified by the R1 field. The result is placed in the register specified by the R1 field.

The effective main storage address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals zero, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0 (direct address). The result from step 1 is the effective address.

   Bit 11=1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Indicators

Carry. Turned on if a carry is detected out of the high-order bit position of the word. If no carry is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the sum cannot be represented in one word; i.e., if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

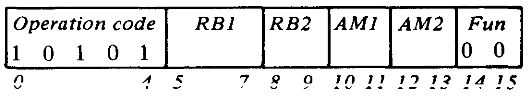
Program Check Conditions

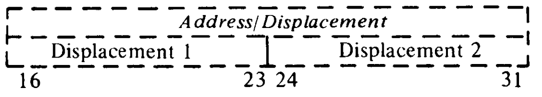
Invalid Storage Address. Instruction word or operand.

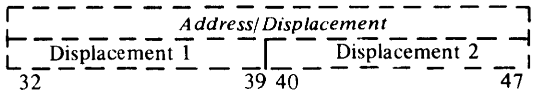
Specification Check. Even byte boundary violation (indirect address or operand address).

Storage/Storage Format

AW addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. (See *Effective Address Generation* in Chapter 2.) Word operand 1 is added to word operand 2. The result replaces operand 2. Operand 1 is unchanged.

Indicators

Carry. Turned on if a carry is detected out of the high-order bit position of the word. If no carry is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the sum cannot be represented in one word; i.e., if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

Specification Check. Even byte boundary violation (indirect address or operand address).

Add Word With Carry (AWCY)

AWCY reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			0 1 0 0 1
0 4	*5 7*	*8 10*	*11 15*

This instruction adds three terms together:

(R1)	the contents of the register specified by the R1 field.
(R2)	the contents of the register specified by the R2 field.
C	the value of the carry indicator at entry.

The contents of the register specified by the R1 field are unchanged if R1 and R2 do not specify the same register. The final result replaces the contents of the register specified by the R2 field.

Programming Note. This instruction can be used when adding multiple word operands. See *Indicators – Multiple Word Operands* in Chapter 2.

Indicators

Carry. Turned on if a carry is detected out of the high-order bit position of the word. If no carry is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the sum cannot be represented in one word; i.e., if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even. Unchanged.

Zero. If on at entry, set to reflect the result. If off at entry, remains off.

Negative. Changed to reflect the result.

Program Check Conditions

No program checks occur.

Add Word Immediate (AWI)

Register Immediate Long Format

AWI word,reg[,reg]

Operation code	*R1*	*R2*	*Function*
0 1 1 1 1			0 0 0 0 1
0 4	5 7	8 10	11 15

Immediate
16 31

The immediate field is added to the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are unchanged if R1 and R2 do not specify the same register.

Indicators

Carry. Turned on if a carry is detected out of the high-order bit position of the word. If no carry is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the sum cannot be represented in one word; i.e., if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word.

Storage Immediate Format

AWI word,addr4

Format without appended word for effective addressing (AM = 00 or 01)

Operation code		*RB*	*AM*	*Function*
0 1 0 0 0	0 0 0			1 0 0 1
0 4	5 7	8 9	10 11	12 15

Immediate
16 31

Format with appended word for effective addressing (AM = 10 or 11)

Operation code		*RB*	*AM*	*Function*
0 1 0 0 0	0 0 0			1 0 0 1
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

Immediate
32 47

The immediate field is added to the contents of the location specified by the effective address. (See *Effective Address Generation* in Chapter 2.) The result replaces the contents of the storage location specified by the effective address.

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

The immediate operand is unchanged.

Indicators

Carry. Turned on if a carry is detected out of the high-order bit position of the word. If no carry is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the sum cannot be represented in one word; i.e., if the sum is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the sum; the carry indicator contains the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

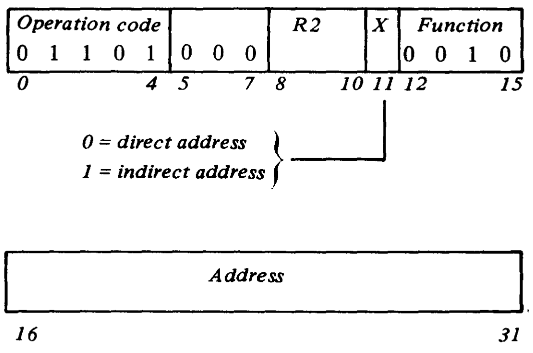
Specification Check. Even byte boundary violation (indirect address or operand address).

Branch Unconditional (B)

B longaddr

Extended Assembler Mnemonic

BX vcon Branch External



An effective branch address is generated and loaded into the instruction address register, becoming the next instruction to be fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals zero, no register contributes to the address generation. The contents of R2 are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

   Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Instruction word or effective branch address.

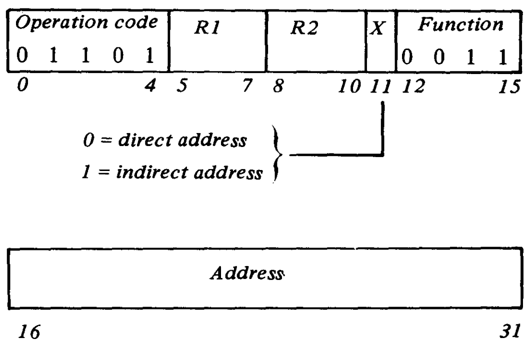
Specification Check. Even byte boundary violation (indirect address or branch address).

Branch and Link (BAL)

BAL longaddr,reg

Extended Assembler Mnemonic

BALX vcon,reg Branch and Link External



The updated value of the instruction address register (the address of the next sequential instruction) is stored into the register specified by the R1 field. An effective branch address is then generated and loaded into the instruction address register, becoming the next instruction to be fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals zero, no register contributes to the address generation. The contents of R2 are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

Programming Note. If R1 and R2 specify the same register the initial contents are used in effective address computation and subsequently overwritten by the return data.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Instruction word or effective branch address. No branch is taken, but the contents of the register specified by the R1 field are still changed.

Specification Check. Even byte boundary violation (indirect address or branch address). No branch is taken but the contents of the R1 register are changed.

Branch and Link Short (BALS)

BALS (reg,jdisp)*
(reg)*
addr*

Operation code	R	Word displacement
1 1 1 1 1		
0 4	5 7	8 15

The updated contents of the instruction address register (the location of the next sequential instruction) are stored in register 7.

Bit 8 of the word displacement field is propagated left by 7 bit positions and a zero is appended at the low order end, resulting in a 16-bit word. (Word displacement is converted to a byte displacement.) This value is added to the contents of the register specified by R to form an effective address. The contents of the storage location specified by the effective address are stored into the instruction address register, and become the address of the next instruction to be fetched.

Programming Note. If the implied register (R7) is used as a base register, the initial contents of R7 are used in effective address computation and subsequently overwritten by the return data.

Indicators

No indicators are changed.

Program Check Conditions

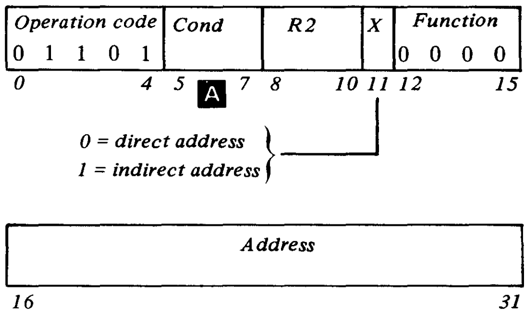
Invalid Storage Address. Effective address. Branching does not occur but storing of the updated instruction address into R7 does occur.

Specification Check. Even byte boundary violation (effective address). Branching does not occur but storing of the updated instruction address into R7 does occur.

Branch On Condition (BC)

Mnemonic	*Operand syntax*	*Instruction name*	*Condition field bits (see* A *)*
BC	cond,longaddr	Branch on Condition	Any value listed below

Extended Mnemonic	*Operand syntax*	*Instruction name*	*Condition field bits (see* A *)*
BE	longaddr	Branch on Equal	000
BOFF	longaddr	Branch if Off	000
BZ	longaddr	Branch on Zero	000
BP	longaddr	Branch on Positive	001
BMIX	longaddr	Branch if Mixed	001
BN	longaddr	Branch if Negative	010
BON	longaddr	Branch if On	010
BEV	longaddr	Branch on Even	011
BLT	longaddr	Branch on Arithmetically Less Than	100
BLE	longaddr	Branch on Arithmetically Less Than or Equal	101
BLLE	longaddr	Branch on Logically Less Than or Equal	110
BCY	longaddr	Branch on Carry	111
BLLT	longaddr	Branch on Logically Less Than	111



This instruction tests the condition of the various indicators (LSR bits 0–4). If the condition tested is met, the effective branch address is loaded into the instruction address register and becomes the next address to be fetched.

If the condition tested is not met, the next sequential instruction is fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals zero, no register contributes to the address generation. The contents of R2 are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

Indicators

All indicators are unchanged.

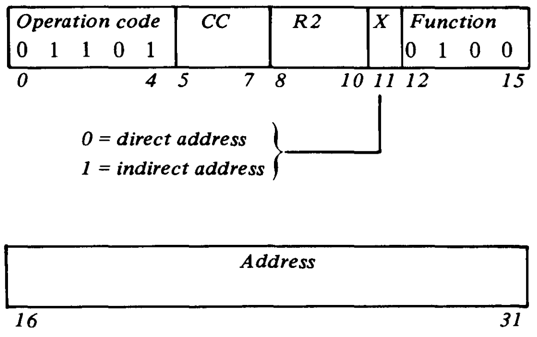
Program Check Conditions

Invalid Storage Address. Instruction word or effective address.

Specification Check. Even byte boundary violation (indirect address or branch address).

Branch On Condition Code (BCC)

BCC	cond,longaddr	
Extended mnemonic		
BNER	longaddr	Branch on Not Error (CC field = 111)



The value of the CC field is compared to the even, carry, and overflow indicators. These indicators hold the I/O condition code: (1) following an I/O instruction or (2) following an I/O interrupt.

CC bit	*Indicator*
5	Even
6	Carry
7	Overflow

If the conditions match, an effective branch address is generated and loaded into the instruction address register, becoming the next instruction to be fetched.

If the conditions do not match the next sequential instruction is fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals zero, no register contributes to the address generation. The contents of R2 are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Instruction word or effective address.

Specification Check. Even byte boundary violation (indirect address or branch address).

I/O Condition Codes

The I/O condition codes are summarized in the following tables. Refer to Chapter 4 for a detailed description of each condition code value. Also refer to the specific I/O device descriptions because some devices do not report all condition codes.

Condition Codes Reported After I/O Instruction.

Condi-tion Code	*Indicators Even*	*Carry*	*Overflow*	*Meaning*
0	0	0	0	Device not attached
1	0	0	1	Busy
2	0	1	0	Busy after reset
3	0	1	1	Command reject
4	1	0	0	Intervention required
5	1	0	1	Interface data check
6	1	1	0	Controller busy
7	1	1	1	Satisfactory

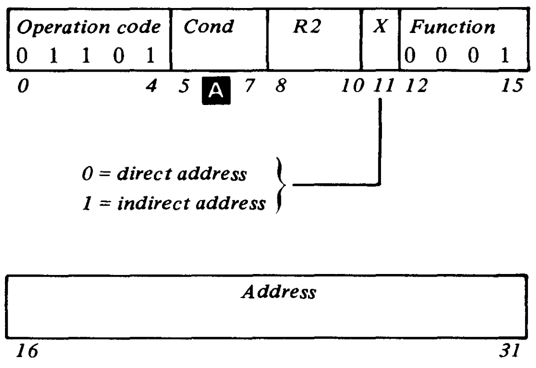
Condition Codes Reported During an I/O Interrupt.

Condi-tion Code	*Indicators Even*	*Carry*	*Overflow*	*Meaning*
0	0	0	0	Controller end
1	0	0	1	PCI (program control interrupt)
2	0	1	0	Exception
3	0	1	1	Device end
4	1	0	0	Attention
5	1	0	1	Attention and PCI
6	1	1	0	Attention and exception
7	1	1	1	Attention and device end

Branch On Not Condition (BNC)

Mnemonic	*Operand syntax*	*Instruction name*	*Condition field bits (see A)*
BNC	cond,longaddr	Branch on Not Condition	Any value listed below

Extended Mnemonic	*Operand syntax*	*Instruction name*	*Condition field bits (see A)*
BNE	longaddr	Branch on Not Equal	000
BNZ	longaddr	Branch on Not Zero	000
BNOFF	longaddr	Branch if Not OFF	000
BNP	longaddr	Branch on Not Positive	001
BNMIX	longaddr	Branch on Not Mixed	001
BNN	longaddr	Branch on Not Negative	010
BNON	longaddr	Branch if Not On	010
BNEV	longaddr	Branch on Not Even	011
BGE	longaddr	Branch on Arithmetically Greater Than or Equal	100
BGT	longaddr	Branch on Arithmetically Greater Than	101
BLGT	longaddr	Branch on Logically Greater Than	110
BLGE	longaddr	Branch on Logically Greater Than or Equal	111
BNCY	longaddr	Branch on No Carry	111



This instruction tests the various indicators (LSR bits 0–4). If the condition tested is met, the effective branch address is loaded into the instruction address register and becomes the next address to be fetched.

If the condition tested is not met, the next sequential instruction is fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals zero, no register contributes to the address generation. The contents of R2 are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

Indicators

All indicators are unchanged.

Program Check Conditions

Invalid Storage Address. Instruction word or effective address.

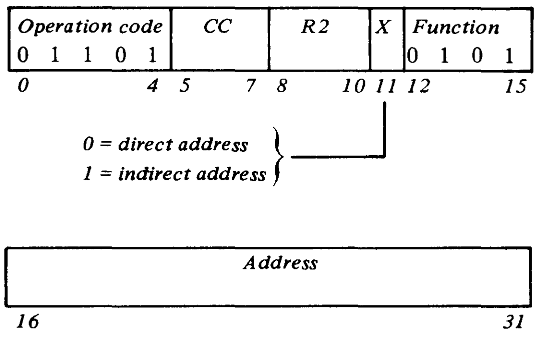
Specification Check. Even byte boundary violation (indirect address or branch address).

Branch On Not Condition Code (BNCC)

BNCC cond,longaddr

Extended mnemonic

BER	longaddr	Branch on Error (CC Field=111)



The value of the CC field is compared to the even, carry, and overflow indicators. These indicators hold the I/O conditions code: (1) following an I/O instruction or (2) following an I/O interrupt.

CC bit	*Indicator*
5	Even
6	Carry
7	Overflow

If the conditions do not match, an effective branch address is generated and loaded into the instruction address register, becoming the next instruction to be fetched.

If the conditions match, the next sequential instruction is fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals zero, no register contributes to the address generation. The contents of R2 are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Instruction word or effective address.

Specification Check. Even byte boundary violation (indirect address or branch address).

I/O Condition Codes

The I/O condition codes are summarized in the following tables. Refer to Chapter 4 for a detailed description of each condition code value. Also refer to the specific I/O device descriptions because some devices do not report all condition codes.

Condition Codes Reported After I/O Instruction.

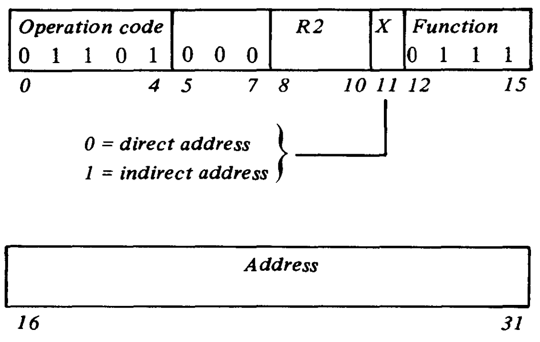
Condition Code	*Indicators Even*	*Carry*	*Overflow*	*Meaning*
0	0	0	0	Device not attached
1	0	0	1	Busy
2	0	1	0	Busy after reset
3	0	1	1	Command reject
4	1	0	0	Intervention required
5	1	0	1	Interface data check
6	1	1	0	Controller busy
7	1	1	1	Satisfactory

Condition Codes Reported During an I/O Interrupt.

Condition Code	*Indicators Even*	*Carry*	*Overflow*	*Meaning*
0	0	0	0	Controller end
1	0	0	1	PCI (program controlled interrupt)
2	0	1	0	Exception
3	0	1	1	Device end
4	1	0	0	Attention
5	1	0	1	Attention and PCI
6	1	1	0	Attention and Exception
7	1	1	1	Attention and device end

Branch On Not Overflow (BNOV)

BNOV longaddr



The overflow indicator is tested. If the indicator is off, the effective branch address is loaded into the instruction address register and becomes the next address to be fetched.

If the overflow indicator is on, the next sequential instruction is fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals zero, no register contributes to the address generation. The contents of R2 are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

   Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

Indicators

All indicators are unchanged.

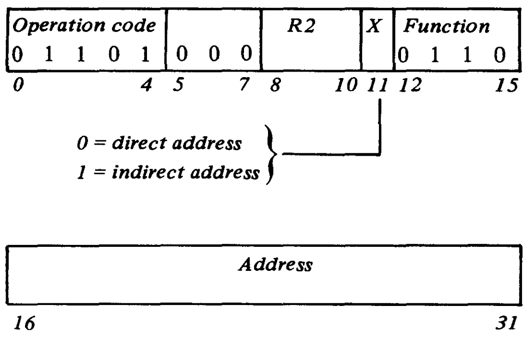
Program Check Conditions

Invalid Storage Address. Instruction word or effective address.

Specification Check. Even byte boundary violation (indirect address or branch address).

Branch On Overflow (BOV)

BOV longaddr



The overflow indicator is tested. If the indicator is on, the effective branch address is loaded into the instruction address register and becomes the next address to be fetched.

If the overflow indicator is off, the next sequential instruction is fetched.

The effective branch address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals zero, no register contributes to the address generation. The contents of R2 are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0. The result from step 1 is a direct address and is loaded into the instruction address register.

   Bit 11=1. The result from step 1 is an indirect address. The contents of the main storage location specified by the result are loaded into the instruction address register.

   Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

Indicators

All indicators are unchanged.

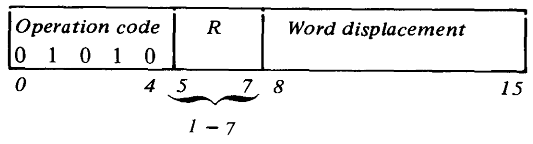
Program Check Conditions

Invalid Storage Address. Instruction word or effective address.

Specification Check. Even byte boundary violation (indirect address or branch address).

Branch Indexed Short (BXS)

BXS (reg$^{1-7}$,jdisp)
(reg$^{1-7}$)
addr



Bit 8 of the word displacement field is propagated left seven bit positions and a zero is appended at the low order end, resulting in a 16-bit word. (Word displacement is converted to a byte displacement.) This value is added to the contents of the register specified by the R field, and the result is stored into the instruction address register, becoming the address of the next instruction to be fetched.

Note. The hardware format of this instruction is identical to the format used for the Jump Unconditional (J) and No Operation (NOP) instructions.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Effective address.

Specification Check. Even byte boundary violation (branch address).

Compare Byte (CB)

Register/Storage Format

CB addr4,reg

Operation code	*R*	*RB*	*AM*	*Function*
1 1 0 0 0				0 1 0 0
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

The contents of the location specified by the effective address in main storage are subtracted from the least significant byte of the register specified by the R field. (*Effective Address Generation* is explained in Chapter 2.)

Neither operand is changed.

Bit 12 of the instruction is not used and must be set to zero to avoid future code obsolescence.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the byte. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one byte; i.e., if the difference is less than -2^7 or greater than $+2^7-1$.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address).

Storage/Storage Format

CB addr5,addr4

Operation code	*RB1*	*RB2*	*AM1*	*AM2*	*Fun*
1 0 0 0 0					1 1
0 4	*5 7*	*8 9*	*10 11*	*12 13*	*14 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

Address/Displacement	
Displacement 1	Displacement 2
32 39	*40 47*

The address arguments generate the effective addresses of the two operands in main storage. (*Effective Address Generation* is explained in Chapter 2.) Byte operand 1 is subtracted from byte operand 2. Neither operand is changed.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the byte. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one byte; i.e., if the difference is less than -2^7 or greater than $+2^7-1$.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

Specification Check. Even byte boundary violation (indirect address).

Compare Byte Immediate (CBI)

CBI byte,reg

Operation code	*R*	*Immediate*
1 1 1 1 0		
0 4	*5 7*	*8 15*

The immediate field is extended to 16 bits by sign propagation to the eight high-order bit positions. The result is subtracted from the contents of the register specified by the R field. Neither operand is changed.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the word. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one word; i.e., if the difference is less than -2^{15} or greater than $+2^{15}-1$.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

No program checks occur.

Compare Doubleword (CD)

Register/Storage Format

CD addr4,reg

Operation code	R	RB	AM	Function
1 1 0 1 0				0 1 0 0
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

The contents of the doubleword in main storage specified by the effective address are subtracted from the contents of the register pair specified by the R field and R+1. (*Effective Address Generation* is explained in Chapter 2.)

Neither operand is changed.

Bit 12 of the instruction is not used and must be set to zero to avoid future code obsolescence. If the R field equals 7, register 7 and register 0 are used.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the doubleword. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in the doubleword; i.e., if the difference is less than -2^{31} or greater than $+2^{31}-1$.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Storage/Storage Format

CD addr5,addr4

Operation code	RB1	RB2	AM1	AM2	Fun
1 0 0 1 0					1 1
0 4	*5 7*	*8 9*	*10 11*	*12 13*	*14 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

Address/Displacement	
Displacement 1	Displacement 2
32 39	*40 47*

The address arguments generate the effective addresses of two operands in main storage. (*Effective Address Generation* is explained in Chapter 2.) Doubleword operand 1 is subtracted from doubleword operand 2. Neither operand is changed.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the operand. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one doubleword; i.e., if the difference is less than -2^{31} or greater than $+2^{31}-1$.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

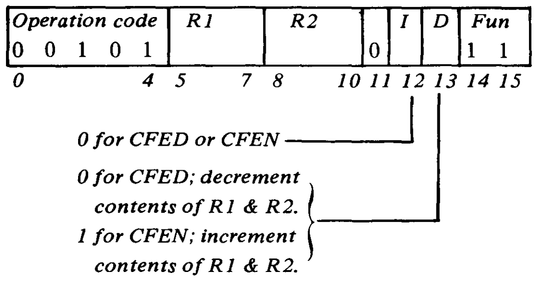
Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

Specification Check. Even byte boundary violation (indirect address or operand address).

Compare Byte Field Equal and Decrement (CFED)

Compare Byte Field Equal and Increment (CFEN)

CFED (reg),(reg)
CFEN (reg),(reg)


Operation code
0 0 1 0 1
R1
R2
0
I
D
Fun
1 1
0 4 5 7 8 10 11 12 13 14 15
0 for CFED or CFEN
0 for CFED; decrement contents of R1 & R2.
1 for CFEN; increment contents of R1 & R2.

This instruction compares two fields in main storage on a byte for byte basis. Register 7 contains the number of bytes to be compared. This number is decremented after each byte is compared. The register specified by R1 contains the address of operand 1. The register specified by R2 contains the address of operand 2. Operand 1 is subtracted from operand 2, but neither operand is changed. After each byte is compared, the addresses in R1 and R2 are incremented or decremented (determined by bit 13 of the instruction). The operation terminates when either:

1. An equal condition is detected, or
2. The number of bytes specified in register 7 has been compared.

When an equality occurs, the addresses in the registers point to the next operands to be compared, but the count in R7 is not updated.

Bit 11 of the instruction is not used and must be set to zero to avoid future code obsolescence.

See Scan Byte Field Equal and Decrement (SFED) and Scan Byte Field Equal and Increment (SFEN) for other versions of this machine instruction.

Notes.
1. If the specified count in R7 is zero, the instruction performs no operation (No-op).
2. Variable field length instructions can be interrupted. When this occurs and the interrupted level resumes operation, the processor treats the uncompleted instruction as a new instruction with the remaining byte count specified in register 7.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the byte. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one byte; i.e., if the difference is less than -2^7 or greater than $+2^7-1$.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

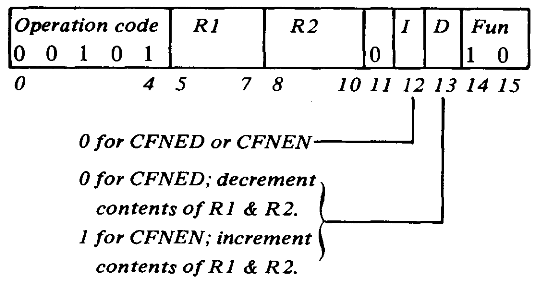
Invalid Storage Address. Operand. The instruction is terminated.

CFNED
CFNEN

Compare Byte Field Not Equal and Decrement (CFNED)

Compare Byte Field Not Equal and Increment (CFNEN)

CFNED (reg),(reg)
CFNEN (reg),(reg)



Operation code
0 0 1 0 1
R1
R2
0
I
D
Fun
1 0
0 4 5 7 8 10 11 12 13 14 15
0 for CFNED or CFNEN
0 for CFNED; decrement contents of R1 & R2.
1 for CFNEN; increment contents of R1 & R2.

This instruction compares two fields in main storage on a byte for byte basis. Register 7 contains the number of bytes to be compared. This number is decremented after each byte is compared. The register specified by R1 contains the address of operand 1. The register specified by R2 contains the address of operand 2. Operand 1 is subtracted from operand 2, but neither operand is changed. After each byte is compared, the addresses in R1 and R2 are incremented or decremented (determined by bit 13 of the instruction). The operation terminates when either:

1. An unequal condition is detected, or
2. The number of bytes specified in register 7 has been compared.

When an inequality occurs, the addresses in the registers point to the next operands to be compared, but the count in R7 is not updated.

Bit 11 is not used and must be set to zero to avoid future code obsolescence.

See Scan Byte Field Not Equal and Decrement (SFNED) and Scan Byte Field Not Equal and Increment (SFNEN) for other versions of this machine instruction.

Notes.
1. If the specified count in R7 is zero, the instruction performs no operation (no-op).
2. Variable field length instructions can be interrupted. When this occurs and the interrupted level resumes operation, the processor treats the uncompleted instruction as a new instruction with the remaining byte count specified in register 7.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the byte. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one byte; i.e., if the difference is less than -2^7 or greater than $+2^7-1$.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Operand. The instruction is terminated.

Complement Register (CMR)

CMR reg[,reg]

Operation code	R1	R2	Function
0 1 1 1 0			0 0 1 1 0
0 4	5 7	8 10	11 15

The contents of the register specified by the R1 field are converted to the two's complement. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are unchanged if R1 and R2 do not specify the same register.

Indicators

Carry. Reset. Then turned on if the number to be complemented is zero.

Overflow. Reset. Then turned on if the number to be complemented is the maximum negative number representable.

Even, Negative, and Zero. Unchanged.

Program Check Conditions

No program checks occur.

Copy Current Level (CPCL)

CPCL reg

Operation code		R2	Function
0 1 1 1 1	0 0 0		1 1 0 0 1
0 4	5 7	8 10	11 15

The register specified by the R2 field is loaded as follows:

- Bits 0–13 are set to zero.
- Bits 14–15 are set to the binary-encoded current level. For example if the current level is three, bits 14–15 are set to 11.

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

Indicators

All indicators are unchanged.

Program Check Conditions

Privilege Violate. Privileged instruction.

Copy Console Data Buffer (CPCON)

CPCON reg

Operation code		R2	Function
0 1 1 1 1	0 0 0		1 1 0 0 0
0 4	5 7	8 10	11 15

The contents of the console data buffer are loaded into the register specified by the R2 field. The contents of the buffer are unchanged.

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence. If the programmer console is not installed, the data loaded into the specified register is undefined.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Privilege Violate. Privileged instruction.

Copy Interrupt Mask Register (CPIMR)

CPIMR addr4

Operation code		RB	AM	Function
0 1 0 1 1	0 0 0			1 0 0 0
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

The contents of the interrupt mask register are stored at the word location in main storage specified by the effective address. (See *Effective Address Generation* in Chapter 2.) The interrupt mask register is unchanged.

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

The mask is represented in a bit significant manner as follows:

Mask bit	*Interrupt level*
0	0
1	1
2	2
3	3

Bits 4–15 are set to zero.

A mask bit set to "1" indicates that the level is enabled.
A mask bit set to "0" indicates that the level is disabled.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Privilege Violate. Privileged instruction.

Specification Check. Even byte boundary violation (indirect address or operand address).

Copy In-process Flags (CPIPF)

CPIPF addr4

Operation code		RB	AM	Function
0 1 0 1 1	0 0 0			1 1 0 1
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

The in-process flags for each level are stored at the *word* location in main storage specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.)

The in-process flags are not changed. The flags are stored in a bit significant manner with bit zero representing level zero, and so on. Bits 4–15 are set to zero.

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

This instruction permits the supervisor on the current level to inspect the in-process flags of the other levels. The in-process flag, bit 9 of the level status register, is on when a level is active or pending (previously interrupted by a higher level).

Indicators

All indicators are unchanged.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Privilege Violate. Privileged instruction.

Specification Check. Even byte boundary violation (indirect address or operand address).

Copy Level Block (CPLB)

CPLB reg,addr4

Operation code	R	RB	AM	Function
0 1 0 1 1				1 1 1 0
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

This instruction stores a level status block (LSB) into 11 words of main storage beginning with the location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.) The contents of the LSB are not changed.

The register specified by the R field contains the binary encoded level of the LSB to be stored. The binary encoded level is placed in bits 14–15 of the register. Bits 0–13 are not used and must be zero.

Using this one instruction, the supervisor can copy the information contained in the hardware registers assigned to a program operating on any level. Most instructions are restricted to the registers associated with the current level. After executing a CPLB instruction, the supervisor can:

1. Use the information just stored; for example, the contents of the general registers or the LSR.
2. Assign the level to another task by executing a Set Level Block (SELB) instruction that points to a different level status block.

In the second case, the supervisor can restart the preempted program at a later time by executing another SELB instruction that points to the previously stored level status block.

Programming Note. If the AM field equals 01, the contents of the register specified by the RB field are incremented by 2.

Indicators

All indicators are unchanged.

Program Check Conditions

Invalid Storage Address. Instruction word or the 11 word main storage area. The instruction is terminated. If the main storage area being accessed is partially outside the installed storage size, a partial data transfer occurs.

Privilege Violate. Privileged instruction.

Specification Check. Even byte boundary violation (indirect address or operand address).

Level Status Block Format

EA	IAR
	Zeros
	LSR
	Register 0
	Register 1
	Register 2
	Register 3
	Register 4
	Register 5
	Register 6
EA+20 (+14 hex)	Register 7

EA=effective address

Format of Register Specified by R in CPLB Instruction

Reserved	Level
0 0 0 0 0 0 0 0 0 0 0 0 0 0	
0 13	14 15

Level 0	*0 0*
Level 1	*0 1*
Level 2	*1 0*
Level 3	*1 1*

Copy Level Status Register (CPLSR)

CPLSR reg

Operation code		*R2*	*Function*
0 1 1 1 0	0 0 0		0 1 1 1 0
0 4	*5 7*	*8 10*	*11 15*

The level status register is loaded into the register specified by the R2 field. The level status register is unchanged. Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

Indicators

All indicators are unchanged.

Program Check Conditions

No program checks occur.

Copy Processor Status and Reset (CPPSR)

CPPSR addr4

Operation code	*R*	*RB*	*AM*	*Function*
0 1 0 1 1	0 0 0			1 1 1 1
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

The contents of the processor status word (PSW) are stored at the word location in main storage specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.)

This instruction resets bits 0–12 of the PSW. Bits 13–15 are unchanged.

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

Indicators

All indicators are unchanged.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Privilege Violate. Privileged instruction.

Specification Check. Even byte boundary violation (indirect address or operand address).

Program Status Word (PSW) Format

PSW bit	*Meaning*
0	Specification check
1	Invalid storage address
2	Privilege violate
3	Not used
4	Invalid function
5	Not used
6	Stack exception
7	Not used
8	Storage parity check
9	Not used
10	CPU control check
11	I/O check
12	Sequence indicator
13	Auto-IPL
14	Not used
15	Power/thermal warning

Compare Word (CW)

Register/Register Format

CW reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			0 0 1 0 1
0 4	*5 7*	*8 10*	*11 15*

The contents of the register specified by the R1 field are subtracted from the contents of the register specified by the R2 field. The contents of both registers are unchanged.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the word. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one word; i.e., if the difference is less than -2^{15} or greater than $+2^{15}-1$.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

No program checks occur.

Register/Storage Format

CW addr4,reg

Operation code	*R*	*RB*	*AM*	*Function*
1 1 0 0 1				0 1 0 0
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

The contents of the word in main storage specified by the effective address are subtracted from the contents of the register specified by the R field. (*Effective Address Generation* is explained in Chapter 2.)

Both operands are unchanged.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the word. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one word; i.e., if the difference is less than -2^{15} or greater than $+2^{15}-1$.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Storage/Storage Format

CW addr5,addr4

Operation code	*RB1*	*RB2*	*AM1*	*AM2*	*Fun*
1 0 0 0 1					1 1
0 4	*5 7*	*8 9*	*10 11*	*12 13*	*14 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

Address/Displacement	
Displacement 1	Displacement 2
32 39	*40 47*

The address arguments generate the effective addresses of two operands in main storage. (See *Effective Address Generation* in Chapter 2.) Word operand 1 is subtracted from word operand 2. Neither operand is changed.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the word. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one word; i.e., if the difference is less than -2^{15} or greater than $+2^{15}-1$.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

Specification Check. Even byte boundary violation (indirect address or operand address).

Compare Word Immediate (CWI)

Register Immediate Long Format

CWI word, reg

Operation code	*R1*		*Function*
0 1 1 1 1		0 0 0	0 0 1 1 0
0 4	5 7	8 10	11 15

Immediate
16 31

The immediate field is subtracted from the contents of the register specified by the R1 field. The contents of the register specified by the R1 field are unchanged.

Bits 8–10 are not used and must be set to zero to avoid future code obsolescence.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the word. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one word; i.e., if the difference is less than -2^{15} or greater than $+2^{15}-1$.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word.

Storage Immediate Format

CWI word,addr4

Format without appended word for effective addressing (AM = 00 or 01)

Operation code		*RB*	*AM*	*Function*
0 1 0 0 0	0 0 0			1 1 1 1
0 4	5 7	8 9	10 11	12 15

Immediate
16 31

Format with appended word for effective addressing (AM = 10 or 11)

Operation code		*RB*	*AM*	*Function*
0 1 0 0 0	0 0 0			1 1 1 1
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

Immediate
32 47

The immediate word is subtracted from the contents of the location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.)

Bits 5–7 are not used and must be set to zero to avoid future code obsolescence. Both operands are unchanged.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the word. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one word; i.e., if the difference is less than -2^{15} or greater than $+2^{15}-1$.

Even, Negative, and Zero. Changed to reflect the result.

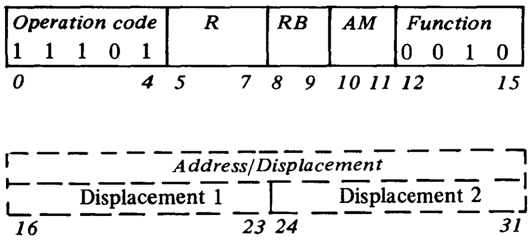
Program Check Conditions

Invalid Storage Address. Instruction word or operand.

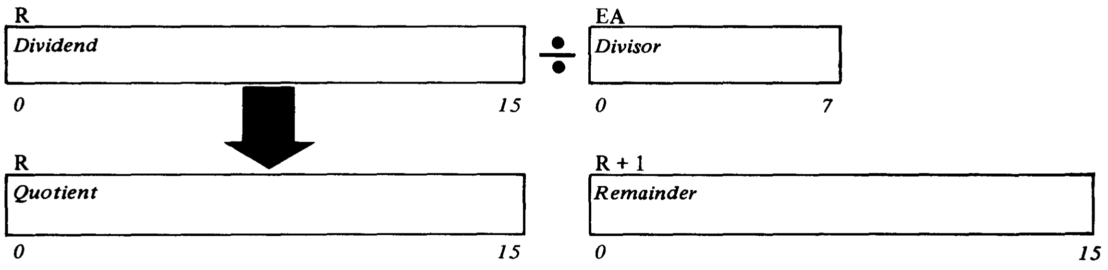
Specification Check. Even byte boundary violation (indirect address or operand address).

Divide Byte (DB)

DB addr4,reg



A divide operation is performed between the word dividend contained in the register specified by the R field and the byte divisor at the location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.) The 1-word quotient replaces the contents of the specified register while the 1-word remainder is placed in the register specified by R+1. If the R field specifies register 7, the remainder is placed in register 0.



Indicators

Overflow. Cleared, then turned on if division by zero is attempted, or if the quotient cannot be represented in one word. If overflow occurs, the remaining indicators and the contents of the specified register are undefined.

Carry. Cleared, then turned on (together with the overflow indicator) if the overflow was caused by an attempt to divide by zero.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. If the AM field equals 01, the contents of the register specified by the RB field are incremented.

Specification Check. Even byte boundary violation (indirect address).

Divide Doubleword (DD)

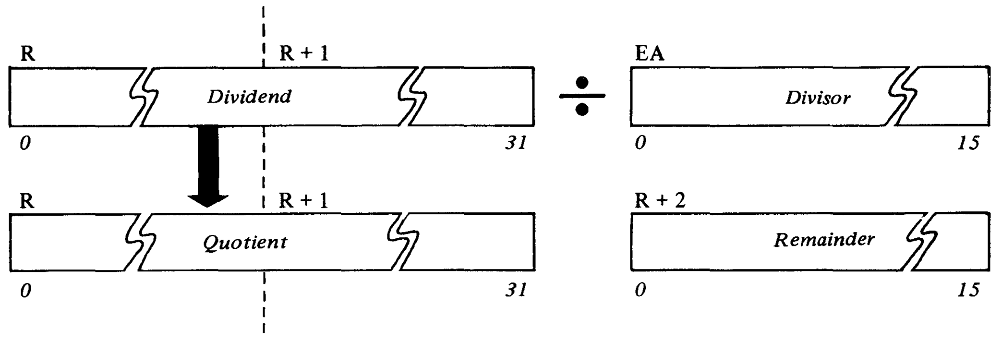
DD addr4,reg

Operation code	R	RB	AM	Function
1 1 1 0 1				1 0 1 0
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

A divide operation is performed between the doubleword dividend contained in the registers, specified by the R field and R+1, and the word divisor at the location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.) The doubleword quotient replaces the contents of the specified registers (least significant word is in R+1). The one-word remainder is placed in the register specified by R+2.

The R field wraps from 7 to 0; e.g., if R specifies register 6, registers 6, 7, and 0 are used.



Programming Note. If the AM field equals 01, the contents of the register specified by the RB field are incremented by 2.

Indicators

Overflow. Cleared, then turned on if division by zero is attempted, or if the quotient cannot be represented in a doubleword. If overflow occurs, the remaining indicators and the contents of the specified registers are undefined.

Carry. Cleared, then turned on (together with the overflow indicator) if the overflow was caused by an attempt to divide by zero.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

DIAG

Diagnose (DIAG)

DIAG ubyte

Operation code	*Function*	*Parameter*
0 1 1 0 0	1 0 1	
0 4	*5 7*	*8 15*

The Diagnose instruction is used for controlling or testing various hardware functions in a machine dependent manner.

The parameter field has the following bit significance.

Bits	*Functions*
8&9 = 00	Storage select – word
= 01	Storage select – byte
= 10	Local storage register select
= 11	I/O interface select
10 = unused	Must be set to zero
11 = unused	Must be set to zero
12 = 0	Storage to register
= 1	Register to storage
13 = 1	Set system ID
14 = 0	Disable
= 1	Enable
15 = unused	Must be set to zero

Functions

Storage Select – Word.

Storage Select – Byte. These functions, with bit 14 (enable/disable function), allow the inhibiting of storage parity generation or parity checking for the data cycle executed within the instruction. Bit 12 (storage to register/register to storage function) specifies the direction of the data transfer. The storage address for this storage cycle is contained in register 7 of the current active level. The data is contained in register 0 of the current active level.

Local Storage Register Select. This function transfers data between main storage and any local storage location by directly addressing local storage. Two additional words are appended to the *Diagnose* instruction when this function is specified.

Additional words when accessing local storage

	Stack address
0 0 0 0 0 0 0 0 0 0	
16 25	*26 31*

Immediate data
32 47

The bits in these words are defined as follows:

Bits	*Significance*
16–25	Must be set to zero
26–31	Local storage address
32–47	Immediate data field

Bit 12 of the Parameter field specifies the direction of the data transfer.

Bit 12 = 0	Immediate data field to local storage
= 1	Local storage to immediate data field

The following chart shows the addresses for the registers in local storage.

Bits 26	*27*	*28*	*29*	*30*	*31*	*Register Selected*
0	0	0	0	0	0	Work Reg 2
0	0	0	0	0	1	Work Reg 3
0	0	0	0	1	0	PSW
0	0	0	0	1	1	Mask
0	0	0	1	0	0	SAR
0	0	0	1	0	1	Lvl 0 IAR
0	0	0	1	1	0	Work Reg C
0	0	0	1	1	1	Lvl 0 LSR
0	0	1	0	0	0	Lvl 0 Reg 0
0	0	1	0	0	1	Lvl 0 Reg 1
0	0	1	0	1	0	Lvl 0 Reg 2
0	0	1	0	1	1	Lvl 0 Reg 3
0	0	1	1	0	0	Lvl 0 Reg 4
0	0	1	1	0	1	Lvl 0 Reg 5
0	0	1	1	1	0	Lvl 0 Reg 6
0	0	1	1	1	1	Lvl 0 Reg 7
0	1	0	0	0	0	Address Compare Reg
0	1	0	0	0	1	Spare
0	1	0	0	1	0	CIAR
0	1	0	0	1	1	Console Data Reg
0	1	0	1	0	0	Spare
0	1	0	1	0	1	Lvl 1 IAR
0	1	0	1	1	0	Work Reg D
0	1	0	1	1	1	Lvl 1 LSR
0	1	1	0	0	0	Lvl 1 Reg 0
0	1	1	0	0	1	Lvl 1 Reg 1
0	1	1	0	1	0	Lvl 1 Reg 2
0	1	1	0	1	1	Lvl 1 Reg 3
0	1	1	1	0	0	Lvl 1 Reg 4
0	1	1	1	0	1	Lvl 1 Reg 5
0	1	1	1	1	0	Lvl 1 Reg 6
0	1	1	1	1	1	Lvl 1 Reg 7
1	0	0	0	0	0	Work Reg 0
1	0	0	0	0	1	Data Buffer
1	0	0	0	1	0	Spare
1	0	0	0	1	1	Spare
1	0	0	1	0	0	Spare
1	0	0	1	0	1	Lvl 2 IAR
1	0	0	1	1	0	Work Reg E
1	0	0	1	1	1	Lvl 2 LSR
1	0	1	0	0	0	Lvl 2 Reg 0
1	0	1	0	0	1	Lvl 2 Reg 1
1	0	1	0	1	0	Lvl 2 Reg 2
1	0	1	0	1	1	Lvl 2 Reg 3
1	0	1	1	0	0	Lvl 2 Reg 4
1	0	1	1	0	1	Lvl 2 Reg 5
1	0	1	1	1	0	Lvl 2 Reg 6
1	0	1	1	1	1	Lvl 2 Reg 7
1	1	0	0	0	0	Work Reg 1
1	1	0	0	0	1	Work Reg 4
1	1	0	0	1	0	Spare
1	1	0	0	1	1	Spare
1	1	0	1	0	0	Spare
1	1	0	1	0	1	Lvl 3 IAR
1	1	0	1	1	0	Work Reg F
1	1	0	1	1	1	Lvl 3 LSR
1	1	1	0	0	0	Lvl 3 Reg 0
1	1	1	0	0	1	Lvl 3 Reg 1
1	1	1	0	1	0	Lvl 3 Reg 2
1	1	1	0	1	1	Lvl 3 Reg 3
1	1	1	1	0	0	Lvl 3 Reg 4
1	1	1	1	0	1	Lvl 3 Reg 5
1	1	1	1	1	0	Lvl 3 Reg 6
1	1	1	1	1	1	Lvl 3 Reg 7

I/O Interface Select. This function, with bit 14 equal to 0, disables and logically isolates the interrupt and cycle steal request lines on the I/O interface from the channel. When bit 14 equals 1, these lines are enabled.

Storage to Register.

Storage to Storage. This function specifies the direction of data transfer for storage and local storage functions.

Set System ID. This function sets the system model number into bits 14–15 of register 0 of the current active level. Bits 0–13 are set to zeros.

If this function is specified, all other functions in the parameter field are ignored. The system ID for the 4953 processor is 03 (hex) and register 0 is set as follows:

Register 0

0000 0000 0000 00	1	1
0 … 13	*14*	*15*

Disable.

Enable. This function disables or enables parity generation checking. See functions *Storage Select* and *I/O Interface Select.*

Program Check Conditions

Privilege Violate. Privileged instruction.

Disable (DIS)

DIS ubyte

Operation code	*Function*	*Parameter*
0 1 1 0 0	0 1 1	
0 … 4	*5 … 7*	*8 … 15*

The facilities designated by one bits in the parameter field are disabled. The bits in the parameter field have the following significance:

Bit	*Facility*
8	Not used
9	Not used
10	Not used
11	Not used
12	Not used
13	Not used
14	Not used
15	Summary mask

Note. Bits not used must be set to zero to avoid future code obsolescence.

Indicators

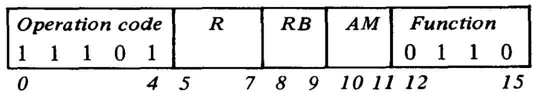
No indicators are changed.

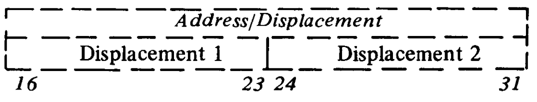
Program Check Conditions

Privilege Violate. Privileged instruction.

Divide Word (DW)

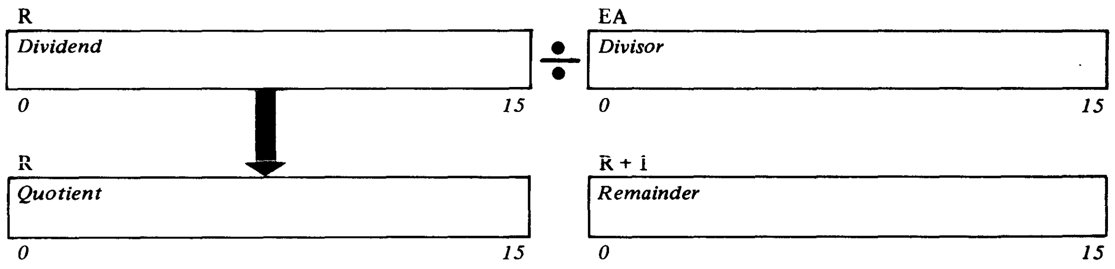
DW addr4,reg





A divide operation is performed between the word dividend contained in the register specified by the R field and the word divisor at the location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.) The one word quotient replaces the contents of the specified register. The one word remainder is placed in the register specified by R+1.

The R field wraps from 7 to 0; that is, if R specifies register 7, registers 7 and 0 are used.



Indicators

Overflow. Cleared, then turned on if division by zero is attempted, or if the quotient cannot be represented in one word. If overflow occurs, the remaining indicators and the contents of the specified registers are undefined.

Carry. Cleared, then turned on (together with the overflow indicator) if the overflow was caused by an attempt to divide by zero.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Enable (EN)

EN ubyte

Operation code	*Function*	*Parameter*
0 1 1 0 0	0 1 0	
0 4	5 7	8 15

The facilities designated by one bits in the parameter field are enabled.

The bits in the parameter field have the following significance:

Bit	*Facility*
8	Not used
9	Not used
10	Not used
11	Not used
12	Not used
13	Not used
14	Not used
15	Summary mask

Note. Bits not used must be set to zero to avoid future code obsolescence.

Indicators

No indicators are changed.

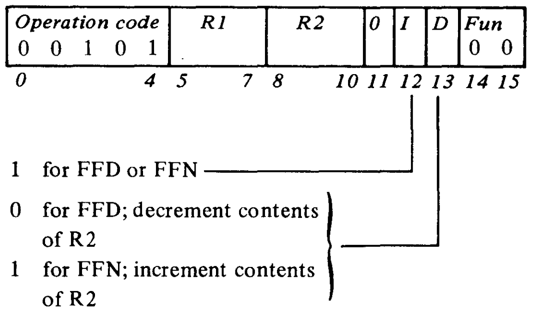
Program Check Conditions

Privilege Violate. Privileged instruction.

Fill Byte Field and Decrement (FFD)

Fill Byte Field and Increment (FFN)

FFD	reg,(reg)
FFN	reg,(reg)


Operation code
0 0 1 0 1
R1
R2
0
I
D
Fun
0 0
0
4
5
7
8
10
11
12
13
14
15
1 for FFD or FFN
0 for FFD; decrement contents of R2
1 for FFN; increment contents of R2

This instruction fills all bytes of a field in main storage with the same bit configuration in each byte. Register 7 contains the number of bytes to be filled (field length). If a field length of zero is specified, the instruction is a no-op. The register specified by R1 contains, in bits 8–15, the byte used to fill the field. The register specified by R2 contains the starting address of the field in main storage.

After each byte in the field is filled:

1. The address in R2 is either incremented or decremented, determined by bit 13 of the instruction. This permits filling the field in either direction.
2. The length count in R7 is decremented.

The operation ends when the specified field length has been filled (contents of R7 equal zero). At this time, the address in R2 has been updated and points to the byte adjacent to the end of the field.

Bits 11 and 15 of the instruction are not used and must be set to zero to avoid future code obsolescence.

See *Move Byte Field and Decrement (MVFD)* and *Move Byte Field and Increment (MVFN)* for other versions of this machine instruction.

Note. Variable field length instructions can be interrupted. When this occurs and the interrupted level resumes operation, the processor treats the uncompleted instruction as a new instruction with the remaining byte count specified in register 7.

Indicators

Carry. Unchanged.

Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect that result of the last byte moved.

Program Check Conditions

Invalid Storage Address. Operand. The instruction is terminated.

Operate I/O (IO)

Refer to Chapter 4 for a detailed description concerning the operation of this instruction.

IO longaddr

Operation code		*R2*	*X*	*Function*
0 1 1 0 1	0 0 0			1 1 0 0
0 4	*5 7*	*8 10*	*11*	*12 15*

0 = direct address
1 = indirect address (bit 11)

Address
16 31

An effective main storage address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals zero, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:

Bit 11=0 (Direct Address). The result from step 1 is the effective address.

Bit 11=1 (Indirect Address). The result from step 1 is the address of the main storage location that contains the effective address.

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

The effective address specifies the location of a two-word control block, called the immediate device control block (IDCB). The IDCB contains the command, device address, and a one-word immediate data field:

IDCB (immediate device control block)

Command field	*Device address field*
0 7	*8 15*

Immediate data field
16 31

The immediate data field serves two purposes:

1. For direct program control (DPC) operations, it holds the data transferred to or from the I/O device.
2. For cycle steal operations, it holds the address of the device control block (DCB).

Refer to Chapter 4 for additional information.

Indicators

Even, Carry, and Overflow. Changed to reflect the condition code. See Branch on Condition Code (BCC) or Branch on Not Condition Code (BNCC) instructions.

Negative and Zero. These indicators are not changed.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Privilege Violate. Privileged instruction.

Specification Check. Even byte boundary violation (indirect address or operand address).

Interchange Registers (IR)

IR reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			0 0 1 1 1
0 4	*5 7*	*8 10*	*11 15*

The contents of the register specified by the R1 and R2 fields are interchanged.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the operand moved from R1 to R2.

Program Check Conditions

No program checks occur.

Jump Unconditional (J)

J jdisp
  jaddr

Operation code	R	Word displacement
0 1 0 1 0		
0 4	5 7	8 15
	Zero	

Bit 8 of the word displacement field is propagated left seven bit positions and a zero is appended at the low order end, resulting in a 16-bit word. (Word displacement is converted to a byte displacement.) This value is added to the updated value of the instruction address register becoming the address of the next instruction to be fetched.

Note. The hardware format of this instruction is identical to the format used for the Branch Indexed Short (BXS) instruction.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Effective address.

Specification Check. Even byte boundary violation (branch address).

Jump and Link (JAL)

JAL jdisp,reg
  jaddr,reg

Operation code	R	Word displacement
1 0 0 1 1		
0 4	5 7	8 15

The updated value of the instruction address register (the location of the next sequential instruction) is stored into the register specified by the R field. Bit 8 of the word displacement field is propagated left by seven bit positions and a zero is appended at the low order end, resulting in a 16-bit word. (The word displacement is converted to a byte displacement.) This value is added to the updated contents of the instruction address register, and the result is stored in the instruction address register, becoming the address of the next instruction to be fetched.

Indicators

No indicators are changed.

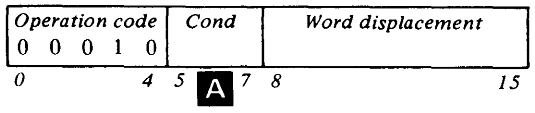
Program Check Conditions

Invalid Storage Address. Effective address. The instruction is terminated. Branching does not occur, but the storing of the updated instruction address into the register specified by the R field still occurs.

Jump On Condition (JC)

Mnemonic	*Operand syntax*	*Instruction name*	*Condition field bits (see* A *)*
JC	cond,jdisp cond,jaddr	Jump on Condition	Any value listed below

Extended Mnemonic	*Operand syntax*	*Instruction name*	*Condition field bits (see* A *)*
JE	jdisp jaddr	Jump on Equal	000
JOFF	jdisp jaddr	Jump if Off	000
JZ	jdisp jaddr	Jump on Zero	000
JMIX	jdisp jaddr	Jump if Mixed	001
JP	jdisp jaddr	Jump on Positive	001
JON	jdisp jaddr	Jump if On	010
JN	jdisp jaddr	Jump on Negative	010
JEV	jdisp jaddr	Jump on Even	011
JLT	jdisp jaddr	Jump on Arithmetically Less Than	100
JLE	jdisp jaddr	Jump on Arithmetically Less Than or Equal	101
JLLE	jdisp jaddr	Jump on Logically Less Than or Equal	110
JCY	jdisp jaddr	Jump on Carry	111
JLLT	jdisp jaddr	Jump on Logically Less Than	111



This instruction tests the condition of the various indicators set by a previously executed instruction (for example: an arithmetic, compare, test bit, or test word type of instruction).

If the condition tested is met, bit 8 of the word displacement field is propagated left by seven bit positions and a zero is appended at the low-order end resulting in a 16-bit word. (Word displacement is converted to a byte displacement.) This value is added to the updated value of the instruction address register, becoming the address of the next instruction to be fetched. If the condition tested is not met, the next sequential instruction is fetched.

For additional information about the indicator settings for the various conditions, see Chapter 2.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Effective address.

Jump On Count (JCT)

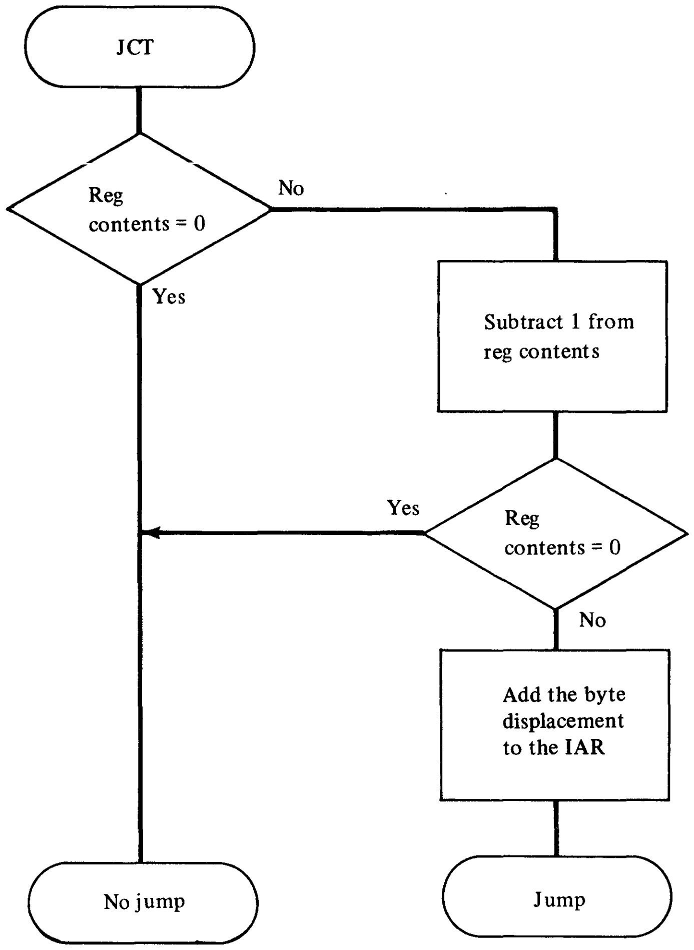
JCT	jdisp,reg
	jaddr,reg

Operation code	*R*	*Word displacement*
1 0 1 1 1		
0 4	*5 7*	*8 15*

This instruction tests the contents of the register specified by the R field.

If the register contents are not zero, the contents are decremented by one. If the register contents are still not zero, the word displacement is converted to a byte displacement and added to the updated contents of the updated instruction address register (IAR). This value indicates the location of the next instruction to be fetched.

If the register contents are zero when initially tested, no decrementing occurs. In this case, or when the register contents are zero after decrementing, the next sequential instruction is fetched.


JCT
Reg contents = 0
No
Yes
Subtract 1 from reg contents
Yes
Reg contents = 0
No
Add the byte displacement to the IAR
No jump
Jump

Note. When the register contents are not zero, the word displacement is converted to a byte displacement as follows. Bit 8 of the word displacement field is propagated left by seven bit positions, and a zero is appended at the low-order end. This results in a 16-bit word that has been doubled in magnitude.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Effective address. The jump does not occur, but the contents of the register specified by the R field are still decremented by one.

Jump On Not Condition (JNC)

Mnemonic	*Operand syntax*	*Instruction name*	*Condition field bits (see* **A** *)*
JNC	cond,jdisp cond,jaddr	Jump on Not Condition	Any value listed below

Extended Mnemonic	*Operand syntax*	*Instruction name*	*Condition field bits (see* **A** *)*
JNE	jdisp jaddr	Jump on Not Equal	000
JNOFF	jdisp jaddr	Jump if Not Off	000
JNZ	jdisp jaddr	Jump on Not Zero	000
JNMIX	jdisp jaddr	Jump on Not Mixed	001
JNP	jdisp jaddr	Jump on Not Positive	001
JNON	jdisp jaddr	Jump if Not On	010
JNN	jdisp jaddr	Jump on Not Negative	010
JNEV	jdisp jaddr	Jump on Not Even	011
JGE	jdisp jaddr	Jump on Arith-metically Greater Than or Equal	100
JGT	jdisp jaddr	Jump on Arith-metically Greater Than	101
JLGT	jdisp jaddr	Jump on Logically Greater Than	110
JLGE	jdisp jaddr	Jump on Logically Greater Than or Equal	111
JNCY	jdisp jaddr	Jump on No Carry	111

Operation code	*Cond*	*Word displacement*
0 0 0 1 1		
0 4	*5* **A** *7*	*8 15*

This instruction tests the condition of the various indicators set by a previously executed instruction (for example: an arithmetic, compare, test bit, or test word type of instruction).

If the condition tested is met, bit 8 of the word displacement field is propagated left by seven bit positions and a zero is appended at the low-order end resulting in a 16-bit word. (Word displacement is converted to a byte displacement.) This value is added to the updated value of the instruction address register, becoming the address of the next instruction to be fetched.

If the condition tested is not met, the next sequential instruction is fetched.

For additional information about the indicator settings for the various conditions, see Chapter 2.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Effective address.

Level Exit (LEX)

LEX [ubyte]

Operation code	*Function*	*Parameter*
0 1 1 0 0	0 0 1	
0 4	*5 7*	*8 15*

When this instruction is executed, the processor exits the current level. The in-process flag (LSR bit 9) for the current level is turned off. Next the instruction tests for (1) pending levels or outstanding priority interrupt requests, (2) the condition of the summary mask (LSR bit 11), and (3) the condition of the level mask bits.

- If pending levels or outstanding requests exist and the summary mask and level mask is enabled:
  - A branch is executed to the address contained in the IAR of the highest pending or requesting level.
  - This level then becomes the current level and processing resumes.
- If processing levels or outstanding requests exist and the summary mask is disabled:
  - The priority interrupts are not allowed.
  - The highest pending level becomes the current level and processing resumes.
  - If no levels are pending, the processor goes to the wait state.
- If no levels are pending and no interrupt requests are outstanding, the processor goes to the wait state.

For additional information on level switching, refer to Chapter 3.

Programming Note. When a level is exited by a LEX instruction and processing is to continue on a pending level, one instruction is executed on the pending level prior to sampling for a trace class interrupt.

Indicators

No indicators are changed.

Program Check Conditions

Privilege Violate. Privileged instruction.

Load Multiple and Branch (LMB)

Refer to *Stack Operations* in Chapter 2 for a detailed description concerning the operation of this instruction. The LMB instruction is used in conjunction with the Store Multiple (STM) instruction described later in this chapter.

LMB addr4

Operation code		*RB*	*AM*	*Function*
0 1 0 0 0	0 0 0			1 0 1 0
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

The contents of the registers for the current level are loaded from the stack defined by the *stack control block* pointed to by the effective address. (*Effective Address Generation* is explained in Chapter 2.) The registers to be loaded are defined by the stack entry previously stored by a Store Multiple (STM) instruction. The next instruction is fetched from the storage address contained in register 7.

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

Programming Note. If the AM field equals 01 the contents of the register specified by the RB field are incremented by 2.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Instruction word or stack control block. The instruction is terminated.

Specification Check.

1. Even byte boundary violation (indirect address, stack control block, or stack element).
2. Address in R7 is odd.

Soft Exception Trap Condition

Stack Exception. Stack is empty. If the AM field equals 01, the contents of the register specified by the RB field are incremented. The instruction is terminated.

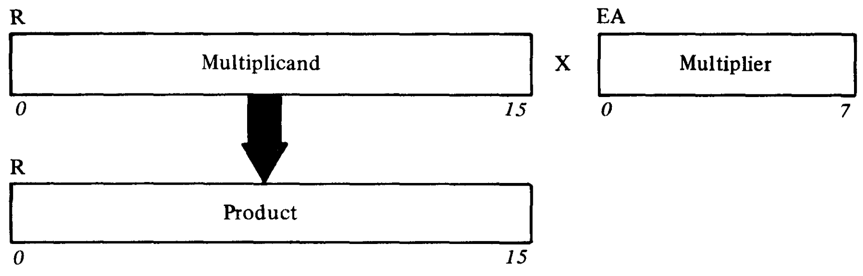
Multiply Byte (MB)

MB addr4,reg

Operation code	*R*	*RB*	*AM*	*Function*
1 1 1 0 1				0 0 0 1
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

A multiply operation is performed between the word multiplicand contained in the register specified by the R field and the byte multiplier at the location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.) The word product replaces the contents of the register.



Indicators

Carry. Reset.

Overflow. Cleared, then turned on if the result cannot be represented in 16 bits. If overflow occurs, the contents of the specified register are undefined.

Even, Negative, and Zero. Set to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. If the AM field equals 01, the contents of the register specified by the RB field are incremented.

Specification Check. Even Byte boundary violation (indirect address).

Multiply Doubleword (MD)

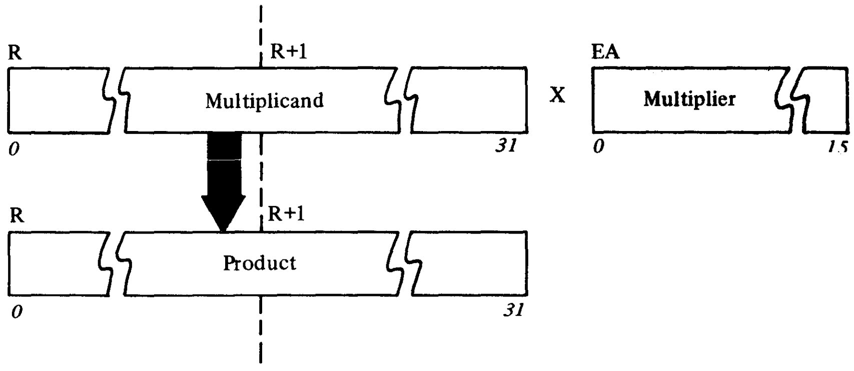
MD addr4,reg

Operation code	R	RB	AM	Function
1 1 1 0 1				1 0 0 1
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

A multiply operation is performed between the doubleword multiplicand contained in the registers specified by the R field and R+1 and the word multiplier at the location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.) The doubleword product replaces the contents of the registers with the least significant word in R+1.

The R field wraps from 7 to 0; that is, if R specifies register 7, registers 7 and 0 are used.



Programming Note. If AM=01, the register specified by the RB field is incremented by 2.

Indicators

Carry. Reset.

Overflow. Cleared, then turned on if the result cannot be represented in 32 bits. If overflow occurs, the contents of the specified registers are undefined.

Even, Negative, and Zero. Set to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Move Address (MVA)

Storage Address to Register Format

MVA addr4,reg

Operation code	R	RB	AM	Function
0 1 0 0 0				0 1 0 0
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

The effective address is loaded into the register specified by the R field. (*Effective Address Generation* is explained in Chapter 2.)

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the operand loaded into the register specified by the R field.

Program Check Conditions

Invalid Storage Address. Second Instruction word.

Specification Check. Even byte boundary violation (indirect address).

Storage Immediate Format

MVA raddr,addr4

Format without appended word for effective addressing (AM = 00 or 01)

Operation code		RB	AM	Function
0 1 0 0 0	0 0 0			0 0 0 0
0 4	5 7	8 9	10 11	12 15

Immediate
16 31

Format with appended word for effective addressing (AM = 10 or 11)

Operation code		RB	AM	Function
0 1 0 0 0	0 0 0			0 0 0 0
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

Immediate
32 47

The operand in the immediate field replaces the contents of the location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.)

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

The immediate operand is not changed.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

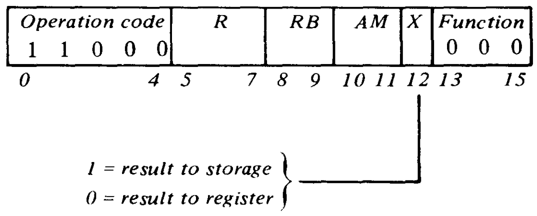
Invalid Storage Address. Instruction word or operand.

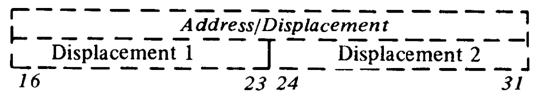
Specification Check. Even byte boundary violation (indirect address or operand address).

Move Byte (MVB)

Register/Storage Format

MVB reg,addr4
addr4,reg





A byte is moved between the least significant byte of the register specified by the R field and the location specified by the effective address in main storage. (*Effective Address Generation* is explained in Chapter 2.) Bit 12 of the instruction specifies the direction of the move:

Bit 12 = 0. The byte is moved from storage to register. The high-order bit of the byte (sign) is propagated to the eight high order bits of the register. This permits the Compare Byte Immediate (CBI) instruction to be used for byte compare operations. The operand in storage is unchanged.

Bit 12 = 1. The byte is moved from register to storage. The contents of the register specified by the R field are not changed.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the operand moved.

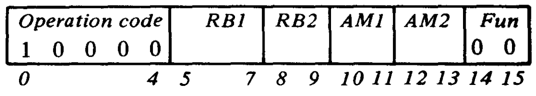
Program Check Conditions

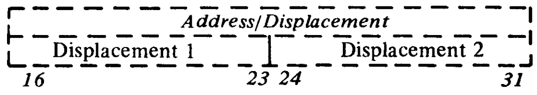
Invalid Storage Address. Instruction word or operand.

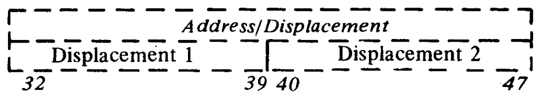
Specification Check. Even byte boundary violation (indirect address).

Storage/Storage Format

MVB addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. (*Effective Address Generation* is explained in Chapter 2.) A byte is moved from operand 1 to operand 2. Operand 1 is unchanged.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the byte moved.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

Specification Check. Even byte boundary violation (indirect address).

Move Byte Immediate (MVBI)

MVBI byte,reg

Operation code	R	Immediate
0 0 0 0 1		
0 4	5 7	8 15

The register specified by the R field is loaded with the immediate operand.

The immediate field of the instruction forms the operand to be loaded. The immediate field is expanded to a sixteen bit operand by propagating the sign bit value through the high order bit positions; this operand is loaded into the register specified by the R field.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the operand loaded into the register.

Program Check Conditions

No program checks occur.

Move Byte and Zero (MVBZ)

MVBZ addr4,reg

Operation code	R	RB	AM	Function
1 1 0 0 0				0 1 0 1
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

The byte specified by the effective address is loaded into the least significant byte of the register specified by the R field. (*Effective Address Generation* is explained in Chapter 2.) The high order bit of the byte (sign) is propagated to the eight high order bits within the register.

The byte specified by the effective address is then set to zeros.

Bit 12 of the instruction is not used and must be set to zero to avoid future code obsolescence.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the operand loaded into the register.

Program Check Conditions.

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address).

Move Doubleword (MVD)

Register/Storage Format

MVD addr4,reg
reg,addr4

Operation Code	R	RB	AM	X	Function
1 1 0 1 0					0 0 0
0 4	5 7	8 9	10 11	12	13 14 15

1 = result to storage
0 = result to register

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

A doubleword is moved between the contents of the register pair specified by the R field (R and R+1) and the doubleword location specified by the effective address in main storage. (*Effective Address Generation* is explained in Chapter 2.) The source operand is unchanged.

The R field wraps from 7 to 0; that is, if R specifies register 7, registers 7 and 0 are used.

Bit 12 of the instruction specifies the direction of the move:

Bit 12 = 0. The doubleword is moved from storage to the register pair.
Bit 12 = 1. The doubleword is moved from the register pair to storage.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the operand moved.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Storage/Storage Format

MVD addr5,addr4

Operation code	RB1	RB2	AM1	AM2	Fun
1 0 0 1 0					0 0
0 4	5 7	8 9	10 11	12 13	14 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

Address/Displacement	
Displacement 1	Displacement 2
32 39	40 47

The address arguments generate the effective addresses of two operands in main storage. (*Effective Address Generation* is explained in Chapter 2.) A doubleword is moved from operand 1 to operand 2. Operand 1 is unchanged.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the doubleword moved.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

Specification Check. Even byte boundary violation (indirect address or operand address).

Move Doubleword and Zero (MVDZ)

MVDZ addr4,reg

Operation code	R	RB	AM	Function
1 1 0 1 0				0 1 0 1
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

The doubleword specified by the effective address is loaded into the register pair specified by the R field (R and R+1). (*Effective Address Generation* is explained in Chapter 2.) The R field wraps from 7 to 0; that is, if R specifies register 7, registers 7 and 0 are used.

The doubleword specified by the effective address is then set to zeros.

Bit 12 of the instruction is not used and must be set to zero to avoid future code obsolescence.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the operand loaded into the register pair.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

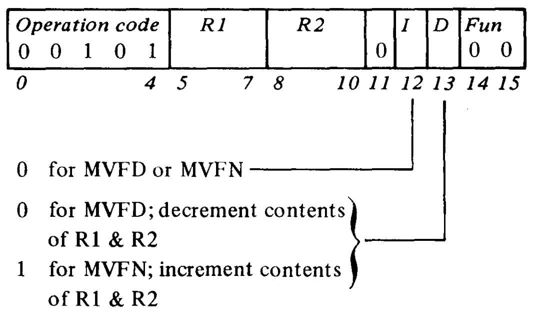
Specification Check. Even byte boundary violation (indirect address or operand address).

MVFD
MVFN

Move Byte Field and Decrement (MVFD)

Move Byte Field and Increment (MVFN)

MVFD	(reg),(reg)
MVFN	(reg),(reg)


Operation code
0 0 1 0 1
R1
R2
0
I
D
Fun
0 0
0 4 5 7 8 10 11 12 13 14 15
0 for MVFD or MVFN
0 for MVFD; decrement contents of R1 & R2
1 for MVFN; increment contents of R1 & R2

This instruction moves a specified number of bytes (one byte at a time) from one storage location to another. Register 7 contains the number of bytes to be moved (field length). If a field length of zero is specified, the instruction is a no-op. The register specified by R1 contains the address of operand 1; the register specified by R2 contains the address of operand 2. Operand 1 is moved to operand 2.

After each byte is moved:

1. The addresses in R1 and R2 are either incremented or decremented, determined by bit 13 of the instruction. This allows the field to be moved in either direction.
2. The length count in R7 is decremented.

The operation ends when the specified field length has been filled (contents of R7 equal zero). At this time, the addresses in R1 and R2 have been updated and point to the next operands.

Bits 11 and 15 of the instructions are not used and must be set to zero to avoid future code obsolescence.

See *Fill Byte Field and Decrement (FFD)* and *Fill Byte Field and Increment (FFN)* for other versions of this machine instruction.

Note. Variable field length instructions can be interrupted. When this occurs and the interrupted level resumes operation, the processor treats the uncompleted instruction as a new instruction with the remaining count specified in register 7.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result of the last byte moved.

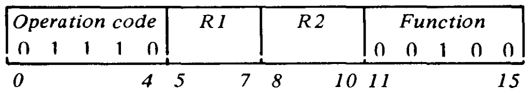
Program Check Conditions

Invalid Storage Address. Operand. The instruction is terminated.

Move Word (MVW)

Register/Register Format

MVW reg,reg



The contents of the register specified by the R1 field replace the contents of the register specified by the R2 field. The contents of the register specified by the R1 field are unchanged.

Indicators

Carry and Overflow. Unchanged.

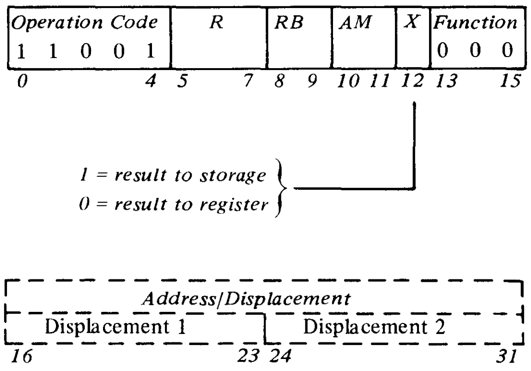
Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

No program checks occur.

Register/Storage Format

MVW reg,addr4
addr4,reg



A word is moved between the contents of the register specified by the R field and the location specified by the effective address in main storage. (*Effective Address Generation* is explained in Chapter 2.) The source operand is unchanged.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the operand moved.

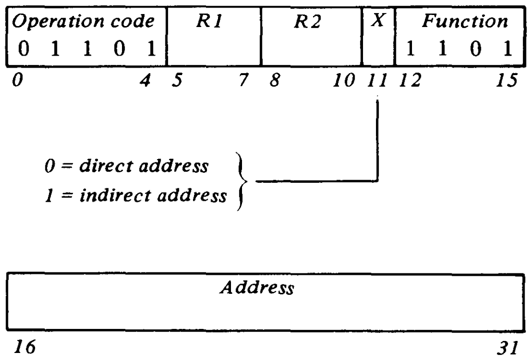
Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Register to Storage Long Format

MVW reg,longaddr



The contents of the register specified by the R1 field are stored into the main storage location specified by an effective address. This effective address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals zero, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:
   Bit 11 = 0 (direct address). The result from step 1 is the effective address.
   Bit 11 = 1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result stored from the register specified by the R1 field.

Program Check Conditions

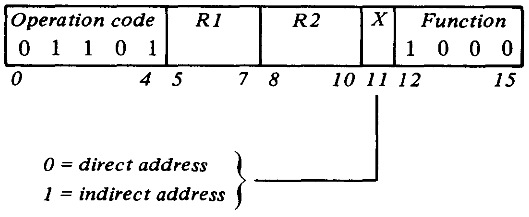
Invalid Storage Address. Instruction word or operand.

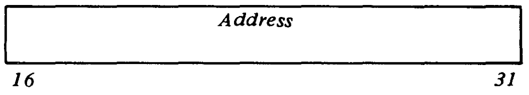
Specification Check. Even byte boundary violation (indirect address or operand address).

MVW

Storage to Register Long Format

MVW longaddr,reg





The register specified by the R1 field is loaded with the contents of the main storage location specified by an effective address. This effective address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals zero, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:
   Bit 11 = 0 (direct address). The result from step 1 is the effective address.
   Bit 11 = 1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result loaded into the register specified by the R1 field.

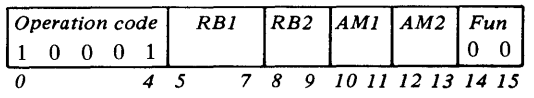
Program Check Conditions

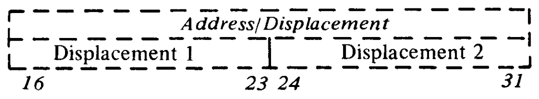
Invalid Storage Address. Instruction word or operand.

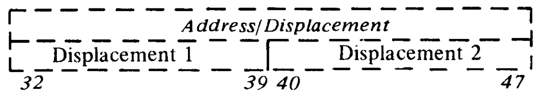
Specification Check. Even byte boundary violation (indirect address or operand address).

Storage/Storage Format

MVW addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. (*Effective Address Generation* is explained in Chapter 2.) A word is moved from operand 1 to operand 2. Operand 1 is unchanged.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the word moved.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

Specification Check. Even byte boundary violation (indirect address or operand address).

Move Word Immediate (MVWI)

Storage to Register Format

MVWI word,reg

Operation code	R	RB	AM	Function
0 1 0 0 0				0 1 0 0
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

The effective address value is loaded into the register specified by the R field. (*Effective Address Generation* is explained in Chapter 2.) This value is equal to the value of *word* as specified by the programmer.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the operand loaded into the register specified by the R field.

Program Check Conditions

Invalid Storage Address. Second instruction word.

Specification Check. Even byte boundary violation (indirect address).

Storage Immediate Format

MVWI word,addr4

Format without appended word for effective addressing (AM = 00 or 01)

Operation code		RB	AM	Function
0 1 0 0 0	0 0 0			0 0 0 0
0 4	5 7	8 9	10 11	12 15

Immediate
16 31

Format with appended word for effective addressing (AM = 10 or 11)

Operation code		RB	AM	Function
0 1 0 0 0	0 0 0			0 0 0 0
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

Immediate
32 47

The operand in the immediate field replaces the contents of the location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.)

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

The immediate operand is not changed.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

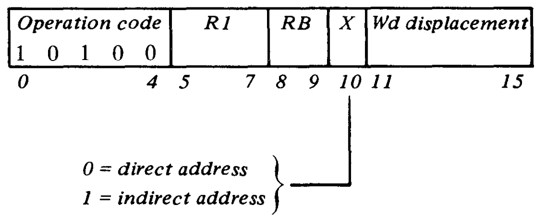
Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Move Word Short (MVWS)

Register to Storage Format

MVWS reg,shortaddr



The contents of the register specified by R1 are stored into the main storage location specified by the effective address. The contents of the register are unchanged.

The effective address is generated as follows:

1. The five bit unsigned integer (word displacement) is doubled in magnitude (converted to a byte displacement).
2. The result from step 1 is added to the contents of the base register (RB) to form a main storage address.
3. Instruction bit 10 is tested for direct or indirect addressing:

   Bit 10 = 0 (direct address). The result from step 2 is the effective address.

   Bit 10 = 1 (indirect address). The result from step 2 is the address of the main storage location that contains the effective address.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the operand stored into main storage.

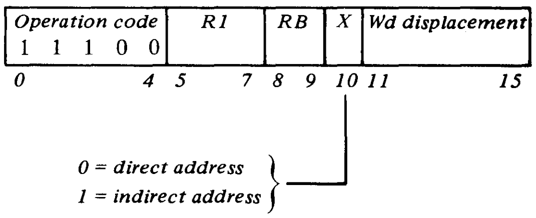
Program Check Conditions

Invalid Storage Address. Operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Storage to Register Format

MVWS shortaddr,reg



The contents of the main storage location specified by the effective address are loaded into the register specified by the R1 field. The contents of the main storage location remain unchanged.

The effective address is generated as follows:

1. The five bit unsigned integer (word displacement) is doubled in magnitude (converted to a byte displacement).
2. The result from step 1 is added to the contents of the base register (RB) to form a main storage address.
3. Instruction bit 10 is tested for direct or indirect addressing:

   Bit 10 = 0 (direct address). The result from step 2 is the effective address.

   Bit 10 = 1 (indirect address). The result from step 2 is the address of the main storage location that contains the effective address.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the operand loaded into the register specified by the R1 field.

Program Check Conditions

Invalid Storage Address. Operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Move Word and Zero (MVWZ)

MVWZ addr4,reg

Operation code	R	RB	AM	Function
1 1 0 0 1				0 1 0 1
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

The word specified by the effective address is loaded into the register specified by the R field. (*Effective Address Generation* is explained in Chapter 2.)

The word specified by the effective address is then set to zeros.

Bit 12 of the instruction is not used and must be set to zero to avoid future code obsolescence.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the results of the operand loaded.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

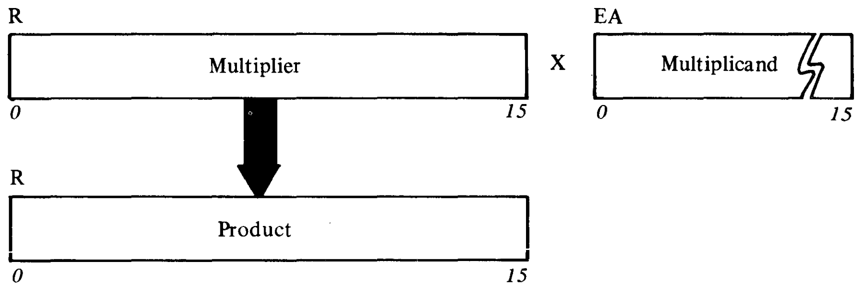
Multiply Word (MW)

MW addr4,reg

Operation code	R	RB	AM	Function
1 1 1 0 1				0 1 0 1
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

A multiply operation is performed between the word multiplier contained in the register specified by the R field and the word multiplicand at the location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.) The word product replaces the contents of the register.



Indicators

Carry. Unchanged

Overflow. Cleared, then turned on if the result cannot be represented in 16 bits. If overflow occurs, the contents of the specified register are undefined.

Even, Negative, and Zero. Set to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

No Operation (NOP)

NOP

Operation code		
0 1 0 1 0	0 0 0	0 0 0 0 0 0 0 0
0 4	5 7	8 15

The hardware format of this instruction is identical to the format used for the Branch Indexed Short (BXS) and Jump Unconditional (J) instructions. When bits 5–15 are all zeros, the instruction performs no operation.

Indicators

No indicators are changed.

Program Check Conditions

No program checks occur.

And Word Immediate (NWI)

NWI word,reg[,reg]

Operation code	R1	R2	Function
0 1 1 1 1			0 0 0 0 0
0 4	5 7	8 10	11 15

Immediate
16 31

The immediate field is ANDed bit by bit with the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field. The contents of the register specified by R1 are unchanged unless R1 and R2 specify the same register.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

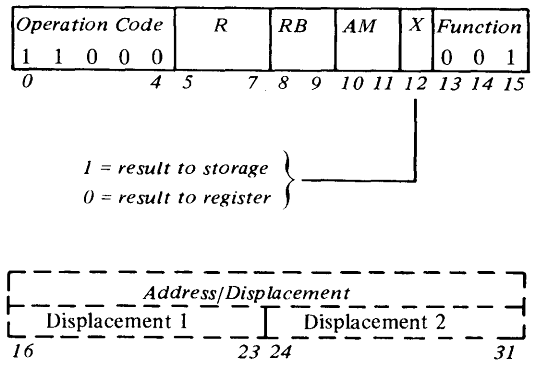
Program Check Conditions

Invalid Storage Address. Instruction word.

OR Byte (OB)

Register/Storage Format

OB reg,addr4
addr4,reg



A logical OR operation is performed between the least significant byte of the register specified by the R field and the location specified by the effective address in main storage. (*Effective Address Generation* is explained in Chapter 2.) Bit 12 of the instruction specifies the destination of the result. The source operand is unchanged. Also, when going from storage to register, bits 0 through 7 of the register are unchanged.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

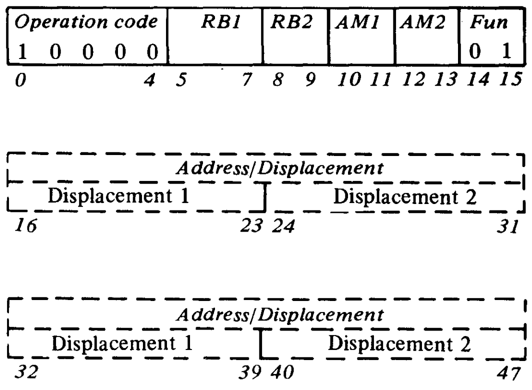
Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address).

Storage/Storage Format

OB addr5,addr4



The address arguments generate the effective addresses of two operands in main storage. (*Effective Address Generation* is explained in Chapter 2.) A one byte logical OR operation is performed between operand 1 and operand 2. The result replaces operand 2.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

Specification Check. Even byte boundary violation (indirect address).

OR Doubleword (OD)

Register/Storage Format

OD addr4,reg
reg,addr4

Operation Code	R	RB	AM	X	Function
1 1 0 1 0					0 0 1
0 4	*5 7*	*8 9*	*10 11*	*12*	*13 14 15*

1 = result to storage
0 = result to register

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

A logical OR operation is performed between the contents of the register pair specified by the R field (R and R+1) and the doubleword in main storage specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.) Bit 12 of the instruction specifies the destination of the result. The source operand is unchanged.

If the R field equals 7, register 7 and register 0 are used.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the results of the OR operation.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or direct address).

Storage/Storage Format

OD addr5,addr4

Operation code	RB1	RB2	AM1	AM2	Fun
1 0 0 1 0					0 1
0 4	*5 7*	*8 9*	*10 11*	*12 13*	*14 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

Address/Displacement	
Displacement 1	Displacement 2
32 39	*40 47*

The address arguments generate the effective addresses of two operands in main storage. (*Effective Address Generation* is explained in Chapter 2.) A doubleword logical OR operation is performed between operand 1 and operand 2. The result replaces operand 2. Operand 1 is unchanged.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result of the OR operation.

Program Check Conditions

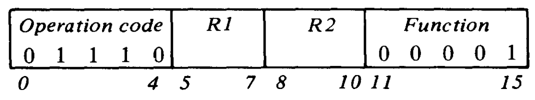
Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

Specification Check. Even byte boundary violation (indirect address or operand address).

OR Word (OW)

Register/Register Format

OW reg,reg



The contents of the register specified by the R1 field are ORed bit by bit with the contents of the register specified by the R2 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field remain unchanged unless R1 and R2 specify the same register.

Indicators

Carry and Overflow. Unchanged.

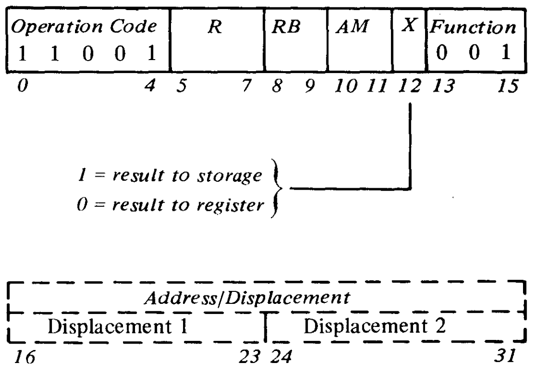
Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

No program checks occur.

Register/Storage Format

OW reg,addr4
addr4,reg



A logical OR operation is performed between the contents of the register specified by the R field and the location specified by the effective address in main storage. (See *Effective Address Generation* in Chapter 2.) Bit 12 of the instruction specifies the destination of the result. The source operand is unchanged.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result of the OR operation.

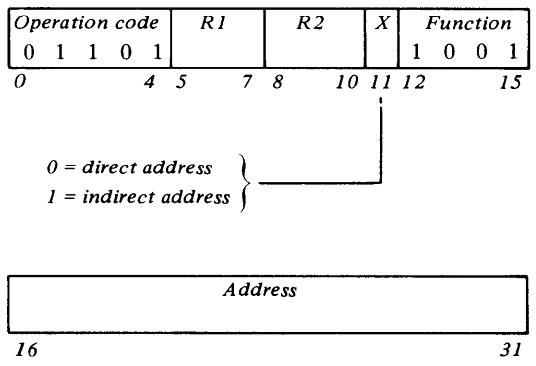
Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Storage to Register Long Format

OW longaddr,reg



A logical OR operation is performed between the contents of the main storage location specified by an effective address and the contents of the register specified by the R1 field. The result is placed in the register specified by the R1 field.

The effective main storage address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals zero, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0 (direct address). The result from step 1 is the effective address.

   Bit 11-1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result loaded into the register specified by the R1 field.

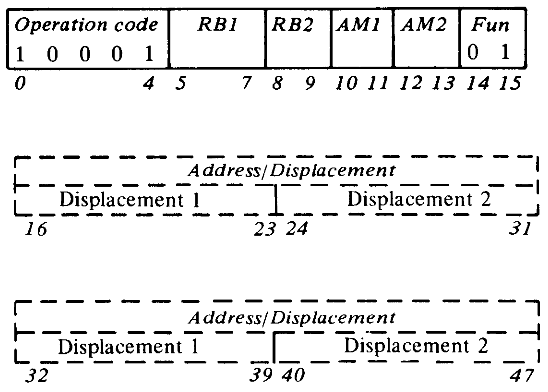
Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Storage/Storage Format

OW addr5,addr4



The address arguments generate the effective addresses of two operands in main storage. (See *Effective Address Generation* in Chapter 2.) A one word logical OR operation is performed between operand 1 and operand 2. The result replaces operand 2. Operand 1 is unchanged.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

Specification Check. Even byte boundary violation (indirect address or operand address).

OR Word Immediate (OWI)

Register Immediate Format

OWI word,reg[,reg]

Operation code	*R1*	*R2*	*Function*
0 1 1 1 1			0 0 0 1 1
0 4	*5 7*	*8 10*	*11 15*

Immediate
16 31

The immediate field is ORed bit by bit with the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field. The contents of the register specified by R1 are unchanged unless R1 and R2 specify the same register.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Storage Immediate Format

OWI word,addr4

Format without appended word for effective addressing (AM = 00 or 01)

Operation code		*RB*	*AM*	*Function*
0 1 0 0 0	0 0 0			1 1 0 0
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Immediate
16 31

Format with appended word for effective addressing (AM = 10 or 11)

Operation code		*RB*	*AM*	*Function*
0 1 0 0 0	0 0 0			1 1 0 0
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

Immediate
32 47

A logical OR operation is performed between the immediate field and the contents of the main storage location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.) The result replaces the contents of the storage location.

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence. The immediate operand is unchanged.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result of the OR operation.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Pop Byte (PB)

Refer to *Stack Operations* in Chapter 2 for additional information about the operation of this instruction and the associated stack control block.

PB addr4,reg

Operation code	R	RB	AM	Function
1 1 1 0 1				0 0 1 1
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

The top element of a byte stack is popped from the stack and loaded into the least significant byte of the register specified by the R field. The stack is defined by the stack control block pointed to by the effective address. (*Effective Address Generation* is explained in Chapter 2.)

Programming Note. If AM equals 01, the register specified by the RB field is incremented by two.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Instruction word, stack control block, or operand.

Specification Check. Even byte boundary violation (indirect address or stack control block).

Soft Exception Trap Condition

Stack Exception. Stack is empty. If AM equals 01, the contents of the register specified by the RB field are incremented.

Pop Doubleword (PD)

Refer to *Stack Operations* in Chapter 2 for additional information about the operation of this instruction and the associated stack control block.

PD addr4,reg

Operation code	R	RB	AM	Function
1 1 1 0 1				1 0 1 1
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

The top element of a doubleword stack is popped from the stack and loaded into the register pair specified by the R field (R and R+1). The stack is defined by the stack control block pointed to by the effective address. (*Effective Address Generation* is explained in Chapter 2.)

If the R field equals 7, registers 7 and 0 are used.

Programming Note. If AM equals 01, the register specified by the RB field is incremented by two.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Instruction word, stack control block, or operand.

Specification Check. Even byte boundary violation (indirect address, stack control block, or stack element).

Soft Exception Trap Condition

Stack Exception. Stack is empty. If the AM field equals 01, the contents of the register specified by the RB field are incremented.

Push Byte (PSB)

Refer to *Stack Operations* in Chapter 2 for additional information about the operation of this instruction and the associated stack control block.

PSB reg,addr4

Operation code	R	RB	AM	Function
1 1 1 0 1				0 0 0 0
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

The least significant byte of the register specified by the R field is pushed into the stack. The stack is defined by the stack control block pointed to by the effective address. (*Effective Address Generation* is explained in Chapter 2.)

Programming Note. If AM equals 01, the register specified by the RB field is incremented by two.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Instruction word, stack control block, or operand.

Specification Check. Even byte boundary violation (indirect address or stack control block).

Soft Exception Trap Condition

Stack Exception. Stack is full. If AM equals 01, the contents of the register specified by the RB field are incremented.

Push Doubleword (PSD)

Refer to *Stack Operations* in Chapter 2 for additional information about the operation of this instruction and the associated stack control block.

PSD reg,addr4

Operation code	R	RB	AM	Function
1 1 1 0 1				1 0 0 0
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

The doubleword operand contained in the register pair specified by the R field (R and R+1) is pushed into the stack. The stack is defined by the stack control block pointed to by the effective address. (*Effective Address Generation* is explained in Chapter 2.)

If the R field equals 7, registers 7 and 0 are used.

Programming Note. If AM equals 01, the register specified by the RB field is incremented by two.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Instruction word, stack control block, or operand.

Specification Check. Even byte boundary violation (indirect address, stack control block, or stack element).

Soft Exception Trap Condition

Stack Exception. Stack is full. If the AM field equals 01, the contents of the register specified by the RB field are incremented.

Push Word (PSW)

Refer to *Stack Operations* in Chapter 2 for additional information about the operation of this instruction and the associated stack control block.

PSW reg,addr4

Operation code	*R*	*RB*	*AM*	*Function*
1 1 1 0 1				0 1 0 0
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

The word operand contained in the register specified by the R field is pushed into the stack. The stack is defined by the stack control block pointed to by the effective address. (*Effective Address Generation* is explained in Chapter 2.)

Programming Note. If AM equals 01, the register specified by the RB field is incremented by two.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Instruction word, stack control block, or operand.

Specification Check. Even byte boundary violation (indirect address, stack control block, or stack element).

Soft Exception Trap Condition

Stack Exception. Stack is full. If the AM field equals 01, the contents of the register specified by the RB field are incremented.

Pop Word (PW)

Refer to *Stack Operations* in Chapter 2 for additional information about the operation of this instruction and the associated stack control block.

PW addr4,reg

Operation code	*R*	*RB*	*AM*	*Function*
1 1 1 0 1				0 1 1 1
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

The top element of a word stack is popped from the stack and loaded into the register specified by the R field. The stack is defined by the stack control block pointed to by the effective address. (*Effective Address Generation* is explained in Chapter 2.)

Programming Note. If AM equals 01, the register specified by the RB field is incremented by two.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Instruction word, stack control block, or operand.

Specification Check. Even byte boundary violation (indirect address, stack control block, or stack element).

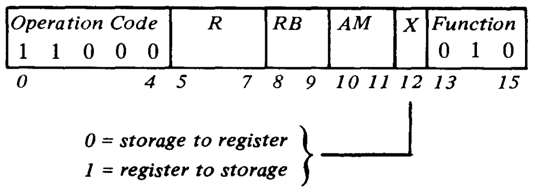
Soft Exception Trap Condition

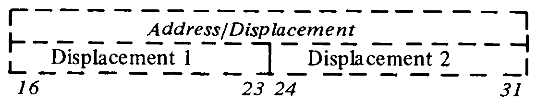
Stack Exception. Stack is empty. If the AM field equals 01, the contents of the register specified by the RB field are incremented.

Reset Bits Byte (RBTB)

Register/Storage Format

RBTB addr4,reg
reg,addr4





This instruction operates either:

1. Storage to register (instruction bit 12 equals 0) or
2. Register to storage (instruction bit 12 equals 1).

Storage to Register. The specified bits are reset in the least significant byte of the register specified by the R1 field. The bit positions turned off correspond to the bit positions containing one-bits in the main storage byte location specified by the effective address. The remaining bits in the low-order byte of the register are unchanged. Also, bits 0–7 of the register and the storage operand are unchanged.

Register to Storage. The specified bits are reset in the main storage byte location specified by the effective address. The bits turned off correspond to the bit positions containing one-bits in the least significant byte of the register specified by the R field. The remaining bits in the storage location are unchanged. The register operand is unchanged.

Note. Effective Address Generation is explained in Chapter 2.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

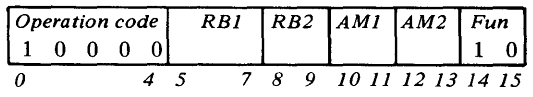
Program Check Conditions

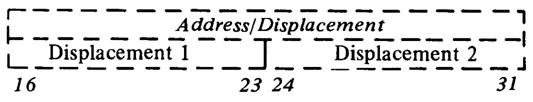
Invalid Storage Address. Instruction word or operand.

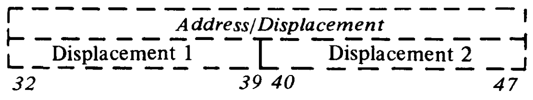
Specification Check. Even byte boundary violation (indirect address).

Storage/Storage Format

RBTB addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. (*Effective Address Generation* is explained in Chapter 2.) The bit positions containing one-bits in byte operand 1 determine the bit positions turned off in byte operand 2. The remaining bits in operand 2 are unchanged. The result replaces operand 2. Operand 1 is unchanged.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

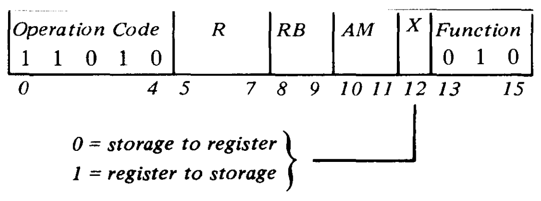
Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

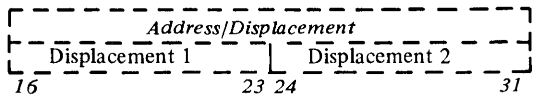
Specification Check. Even byte boundary violation (indirect address).

Reset Bits Doubleword (RBTD)

Register/Storage Format

RBTD addr4,reg
reg,addr4





This instruction operates either:

1. Storage to register (instruction bit 12 equals 0) or
2. Register to storage (instruction bit 12 equals 1)

Storage to Register. The specified bits are reset in the register pair specified by the R field (R and R+1). The bit positions turned off correspond to the bit positions containing one-bits in the doubleword main storage location specified by the effective address. The remaining bits in the register pair are unchanged. The storage operand is unchanged.

Register to Storage. The specified bits are reset in the doubleword main storage location specified by the effective address. The bit positions turned off correspond to the bit positions containing one-bits in the register pair specified by the R field (R and R+1). The remaining bits in the storage operand are unchanged. The register operand is unchanged. If the R field equals 7, registers 7 and 0 are used.

Note. Effective Address Generation is explained in Chapter 2.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

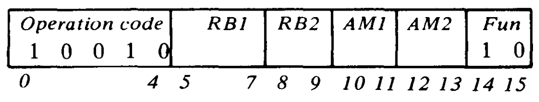
Program Check Conditions

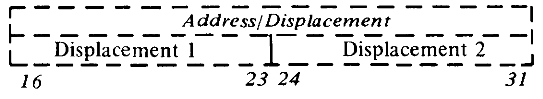
Invalid Storage Address. Instruction word or operand.

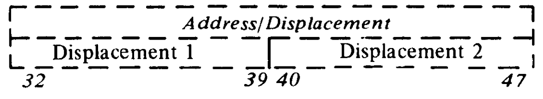
Specification Check. Even byte boundary violation (indirect address or operand address).

Storage/Storage Format

RBTD addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. (*Effective Address Generation* is explained in Chapter 2.) The bit positions containing one-bits in doubleword operand 1 determine the bit positions turned off in doubleword operand 2. The remaining bits in operand 2 are unchanged. The result replaces operand 2. Operand 1 is unchanged.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

Specification Check. Even byte boundary violation (indirect address or operand address).

Reset Bits Word (RBTW)

Register/Register Format

RBTW reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			0 0 0 0 0
0 4	*5 7*	*8 10*	*11 15*

The bit positions containing one-bits in the register specified by the R1 field determine the bit positions turned off in the register specified by the R2 field. The remaining bits in the register specified by the R2 field are unchanged. The contents of the register specified by the R1 field are unchanged unless R1 and R2 specify the same register.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

No program checks occur.

Register/Storage Format

RBTW addr4,reg
reg,addr4

Operation Code	*R*	*RB*	*AM*	*X*	*Function*
1 1 0 0 1					0 1 0
0 4	*5 7*	*8 9*	*10 11*	*12*	*13 15*

0 = storage to register
1 = register to storage

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

This instruction operates either:

1. Storage to register (instruction bit 12 equals 0) or
2. Register to storage (instruction bit 12 equals 1)

Storage to Register. The specified bits are reset in the register specified by the R field. The bit positions turned off correspond to the bit positions containing one-bits in the main storage word location specified by the effective address. The remaining bits in the register are unchanged. The storage operand is unchanged.

Register to Storage. The specified bits are reset in the main storage word location specified by the effective address. The bit positions turned off correspond to the bit positions containing one-bits in the register specified by the R field. The remaining bits in the storage operand are unchanged. The register operand is unchanged.

Note. Effective Address Generation is explained in Chapter 2.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

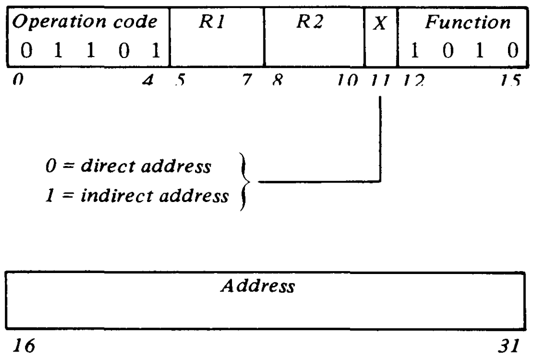
Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Storage to Register Long Format

RBTW longaddr,reg



The bit positions containing one-bits in the main storage word location specified by the effective address determine the bit positions turned off in the register specified by the R1 field. The remaining bits in the register specified by the R1 field are unchanged. The storage operand is unchanged.

The effective address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field to form a main storage address. If the R2 field equals zero, no register contributes to the address generation. The contents of R2 are not changed.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11 = 0 (direct address). The result from step 1 is the effective address.

   Bit 11 = 1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

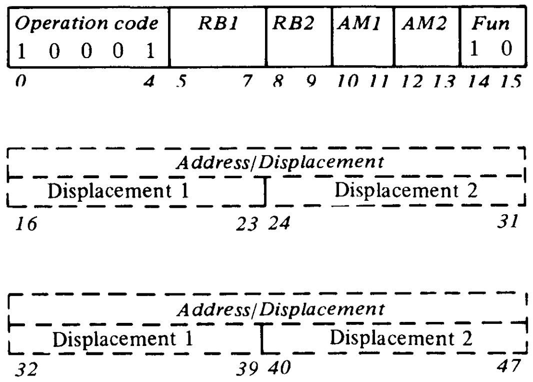
Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Storage/Storage Format

RBTW addr5,addr4



The address arguments generate the effective addresses of two operands in main storage. (*Effective Address Generation* is explained in Chapter 2.) The bit positions containing one-bits in word operand 1 determine the bit positions turned off in word operand 2. The remaining bits in operand 2 are unchanged. The result replaces operand 2. Operand 1 is unchanged.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

Specification Check. Even byte boundary violation (indirect address or operand address).

Reset Bits Word Immediate (RBTWI)

Register Immediate Long Format

RBTWI word,reg[,reg]

Operation code	R1	R2	Function
0 1 1 1 1			0 0 1 0 0
0 4	5 7	8 10	11 15

Immediate
16 31

The bit positions containing one-bits in the immediate field determine the bit positions to be reset. These bit positions are reset in the operand from the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field.

Example:

Contents of immediate field	0000 0000 0000 1111
Contents of R1 register	0101 0101 0101 0101
Result in R2 register	0101 0101 0101 0000

The contents of the register specified by the R1 field are unchanged unless R1 and R2 specify the same register.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word.

Storage Immediate Format

RBTWI word,addr4

Format without appended word for effective addressing (AM = 00 or 01)

Operation code		RB	AM	Function
0 1 0 0 0	0 0 0			1 1 0 1
0 4	5 7	8 9	10 11	12 15

Immediate
16 31

Format with appended word for effective addressing (AM = 10 or 11)

Operation code		RB	AM	Function
0 1 0 0 0	0 0 0			1 1 0 1
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

Immediate
32 47

The bit positions containing one-bits in the immediate field determine the bit positions turned off in the main storage location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.) The immediate operand is unchanged.

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Subtract Byte (SB)

SB reg,addr4
addr4,reg

Operation code	R	RB	AM	X	Function
1 1 0 0 0					1 1 1
0 4	5 7	8 9	10 11	12	13 15

1 = result to storage
0 = result to register

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

A subtract operation is performed between the least significant byte of the register specified by the R field and the location specified by the effective address in main storage. (See *Effective Address Generation* in Chapter 2.) Bit 12 of the instruction specifies the destination of the result. The source operand and high-order byte of the register are unchanged.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the byte. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one byte; i.e., if the difference is less than -2^7 or greater than $+2^7-1$.

If an overflow occurs, the result contains the correct low-order eight bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address).

Subtract Carry Indicator (SCY)

SCY reg

Operation code		R2	Function
0 1 1 1 0	0 0 0		0 0 0 1 0
0 4	5 7	8 10	11 15

The value of the carry indicator on entry is subtracted from the contents of the register specified by the R2 field. The result is placed in the register specified by the R2 field. Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

Programming Note. This instruction can be used when subtracting multiple word operands. See *Indicators – Multiple Word Operands* in Chapter 2.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the word. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one word: i.e., if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even. Unchanged.

Negative. Changed to reflect the result.

Zero. If on at entry, changed to reflect the result. If off at entry, it remains off.

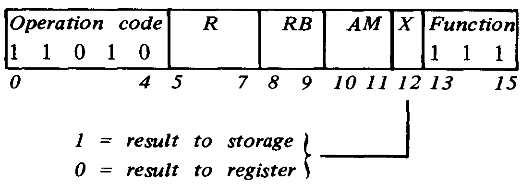
Program Check Conditions

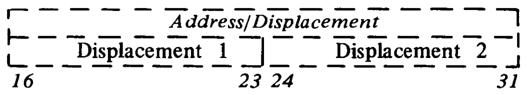
No program checks occur.

Subtract Doubleword (SD)

Register/Storage Format

SD reg,addr4
addr4,reg





A subtract operation is performed between the register pair specified by the R field (R and R+1) and the doubleword in main storage specified by the effective address. (See *Effective Address Generation* in Chapter 2.) Bit 12 of the instruction specifies the destination of the result. The source operand is unchanged.

If the R field equals 7, register 7 and register 0 are used.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the doubleword. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in the doubleword; i.e., if the difference is less than -2^{31} or greater than $+2^{31}-1$.

If an overflow occurs, the result contains the correct low-order 32 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

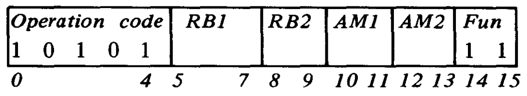
Program Check Conditions

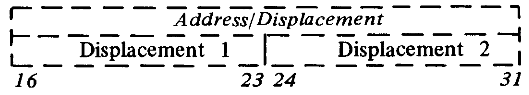
Invalid Storage Address. Instruction word or operand.

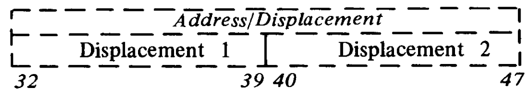
Specification Check. Even byte boundary violation (indirect address or operand address).

Storage/Storage Format

SD addr5,addr4







The address arguments generate the effective addresses of two operands in main storage. (See *Effective Address Generation* in Chapter 2.) Doubleword operand 1 is subtracted from doubleword operand 2. The result replaces operand 2. Operand 1 is unchanged.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the doubleword. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in the doubleword; i.e., if the difference is less than -2^{31} or greater than $+2^{31}-1$.

If an overflow occurs, the result contains the correct low-order 32 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

Specification Check. Even byte boundary violation (indirect address or operand address).

Set Console Data Lights (SECON)

SECON reg

Operation code		R2	Function
0 1 1 1 1	0 0 0		1 0 0 0 0
0 4	5 7	8 10	11 15

The contents of the register specified by R2 are stored in the console data lights. The contents of the register are unchanged.

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

If the Programmer Console is not installed, the instruction performs no operation.

Indicators

All indicators are unchanged.

Program Check Conditions

Privilege Violate. Privileged instruction.

Set Interrupt Mask Register (SEIMR)

SEIMR addr4

Operation code		RB	AM	Function
0 1 0 1 1	0 0 0			0 0 0 0
0 4	5 7	8 9	10 11	12 15

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

Bits 0–3 of the word location in main storage specified by the effective address are loaded into the interrupt mask register. (*Effective Address Generation* is explained in Chapter 2.) Bits 4–15 of the word in main storage are not used. The contents of main storage are unchanged.

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

The mask is represented in a bit significant manner as follows:

Mask bit	*Interrupt level*
0	0
1	1
2	2
3	3

A mask bit set to "1" indicates that the level is enabled.
A mask bit set to "0" indicates that the level is disabled.

Indicators

All indicators are unchanged.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Privilege Violate. Privileged instruction.

Specification Check. Even byte boundary violation (indirect address or operand address).

Set Indicators (SEIND)

SEIND reg

Operation code		*R2*	*Function*
0 1 1 1 0	0 0 0		0 1 1 1 1
0 4	*5 7*	*8 10*	*11 15*

Bits 0–4 of the register specified by the R2 field are loaded into bits 0–4 of the current level status register (indicators). Bits 5–15 of the register specified by R2 are ignored. Bits 5–15 of the level status register are unchanged.

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

The following table shows the indicator bits of the level status register (LSR):

LSR bit	*Indicator*
0	Even
1	Carry
2	Overflow
3	Negative
4	Zero

Indicators

Changed as specified by the R2 register.

Program Check Conditions

No program checks occur.

Set Level Block (SELB)

Execution of the SELB instruction can cause the processor to change levels. Also, the processor may exit supervisor state. For additional information concerning the processor action when executing this instruction, refer to *Program Controlled Level Switching* in Chapter 3.

SELB reg,addr4

Operation code	*R*	*RB*	*AM*	*Function*
0 1 0 1 1				0 1 1 0
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

This instruction loads a level status block (LSB) from 11 words of main storage beginning with the location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.) The contents of the storage locations are not changed.

The register specified by the R field contains the binary encoded level of the LSB to be loaded. The binary encoded level is placed in bits 14–15 of the register. Bits 1–13 are not used and must be zero to avoid future code obsolescence.

Bit 0 of the register specified by the R field is the inhibit trace (IT) interrupt bit. If bit 0 is a one and the trace bit (bit 10) in the LSR of the target LSB is a one, then both the Set Level Block instruction and the instruction pointed to by the IAR in the target LSB are executed before trace interrupts are allowed.

If bit 0 is zero and the trace bit in the LSR of the target LSB is a one, the Set Level Block instruction is executed and then trace interrupts are allowed.

The target LSB is defined by either (1) the effective address, if the in-process bit is set to one in the LSR of the target LSB and the specified R field level is higher than or equal to the current level, or (2) the currently active LSB when condition (1) is not met.

Level Status Block Format

EA	IAR
	Zeros
	LSR
	Register 0
	Register 1
	Register 2
	Register 3
	Register 4
	Register 5
	Register 6
EA+20 (+14 hex)	Register 7

EA=effective address

Format of Register Specified by R in Instruction

IT		*X X*
	0 0 0 0 0 0 0 0 0 0 0 0 0	
0	*1 13*	*14 15*

Level 0	*0 0*
Level 1	*0 1*
Level 2	*1 0*
Level 3	*1 1*

Programming Notes.

1. The Set Level Block instruction with the IT bit equal to one should be used to return from the trace interrupt routine and from a class interrupt routine when the instruction causing the interrupt is to be reexecuted. This is necessary to prevent a double trace of the instruction.
2. If the Set Level Block instruction sets the current level in-process bit to zero and the current level trace bit to one, no trace interrupt occurs as the level is exited.
3. The registers and LSR for the current level are not changed if the specified R field level is other than the current level.
4. If the AM field equals 01, the contents of the register specified by the RB field are incremented by 2.

Indicators

All indicators are unchanged if the specified level is other than the current level.

Program Check Conditions

Invalid Storage Address. Instruction word or level status block. The instruction is terminated.

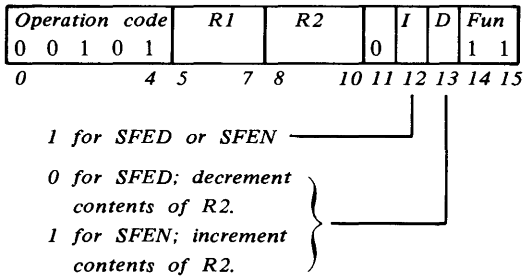
Privilege Violate. Privileged instruction.

Specification Check. Even byte boundary violation (indirect address or level status block address). The instruction is terminated.

Scan Byte Field Equal and Decrement (SFED)

Scan Byte Field Equal and Increment (SFEN)

SFED	reg,(reg)
SFEN	reg,(reg)



This instruction compares a field in main storage against a single byte contained in a register. This comparison is made one byte at a time. Register 7 contains the number of bytes to be compared. This number is decremented after each byte is compared.

The register specified by R1 contains, in bits 8–15, the single byte of operand 1. The register specified by R2 contains the starting address of operand 2. Operand 1 is subtracted from operand 2, but neither operand is changed.

After each byte is compared, the address in R2 is incremented or decremented (determined by bit 13 of the instruction). The operation terminates when either:

1. An equal condition is detected, or
2. The number of bytes specified in register 7 has been compared.

When an equality occurs, the address in the register specified by R2 points to the next operand to be compared, but the count in R7 is not updated.

Bit 11 of the instruction is not used and must be set to zero to avoid future code obsolescence.

See Compare Byte Field Equal and Decrement (CFED) and Compare Byte Field Equal and Increment (CFEN) for other versions of this machine instruction.

Notes.

1. Variable field length instructions can be interrupted. When this occurs and the interrupted level resumes operation, the processor treats the uncompleted instruction as a new instruction with the remaining byte count specified in register 7.
2. If the specified count in R7 is zero, the instruction performs no operation (no-op).

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the byte. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one byte; i.e., if the difference is less than -2^7 or greater than $+2^7-1$.

Even, Negative, and Zero. Changed to reflect the result.

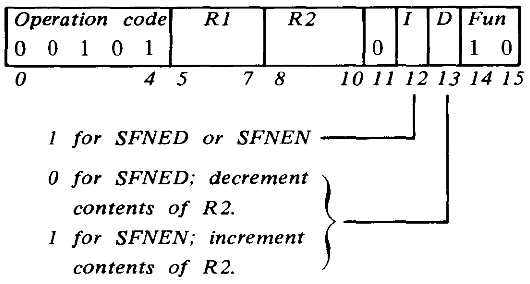
Program Check Conditions

Invalid Storage Address. Operand. The instruction is terminated.

Scan Byte Field Not Equal and Decrement (SFNED)

Scan Byte Field Not Equal and Increment (SFNEN)

SFNED	reg,(reg)
SFNEN	reg,(reg)



This instruction compares a field in main storage against a single byte contained in a register. This comparison is made one byte at a time. Register 7 contains the number of bytes to be compared. This number is decremented after each byte is compared.

The register specified by R1 contains, in bits 8–15, the single byte of operand 1. The register specified by R2 contains the starting address of operand 2. Operand 1 is subtracted from operand 2, but neither operand is changed. After each byte is compared, the address in R2 is incremented or decremented (determined by bit 13 of the instruction). The operation terminates when either:

1. An unequal condition is detected, or
2. The number of bytes specified in register 7 has been compared.

When an inequality occurs, the address in the register specified by R2 points to the next operand to be compared, but the count in R7 is not updated.

Bit 11 of the instruction is not used and must be set to zero to avoid future code obsolescence.

See Compare Byte Field Not Equal and Decrement (CFNED) and Compare Byte Field Not Equal and Increment (CFNEN) for other versions of this machine instruction.

Notes.

1. Variable field length instructions can be interrupted. When this occurs and the interrupted level resumes operation, the processor treats the uncompleted instruction as a new instruction with the remaining byte count specified in register 7.
2. If the specified count in R7 is zero, the instruction performs no operation (no-op).

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the byte. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one byte; i.e., if the difference is less than -2^7 or greater than $+2^7-1$.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Operand. The instruction is terminated.

Shift Left Circular (SLC)

Immediate Count Format

SLC cnt16,reg

Operation code	R	Count	Function
0 0 1 1 0			0 0 0
0 4	*5 7*	*8 12*	*13 15*

The bits in the register specified by the R field are shifted left by the number of bit positions specified in the count field. The bits shifted out of the high-order bit (bit 0) re-enter at the low-order bit (bit 15). A count of zero causes no shifting to take place.

Although the register to be shifted contains only 16 bits, shift count values of 0–31 may be specified. Shift counts greater than 16 provide an effective shift of modulo 16.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the final contents of the register.

Program Check Conditions

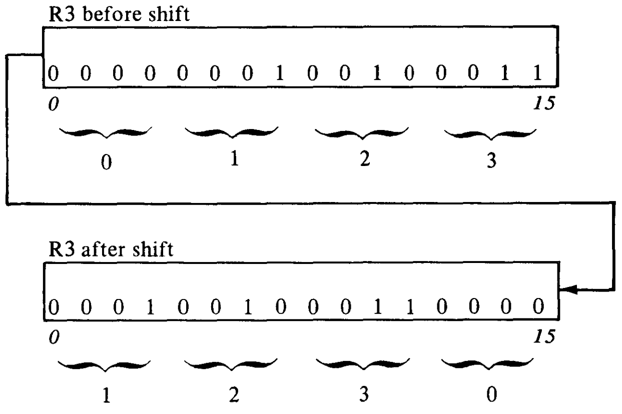
No program checks occur.

Example:

Instruction

Operation code	R	Count	Function
0 0 1 1 0	0 1 1	0 0 1 0 0	0 0 0
0 4	*5 7*	*8 12*	*13 15*

R3 Count = 4



Count in Register Format

SLC reg,reg

Operation code	R1	R2	Function
0 1 1 1 0			1 0 0 0 0
0 4	*5 7*	*8 10*	*11 15*

The bits in the register specified by the R1 field are shifted left by the number of bits specified by the shift count. This count is obtained from bits 8 through 15 of the register specified by the R2 field.

The contents of the register specified by the R2 field are unchanged unless the R1 and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the register to be shifted contains only 16 bits, shift count values of 0–255 may be specified. Shift counts greater than 16 provide an effective shift of modulo 16.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the final contents of the register specified by the R1 field.

Program Check Conditions

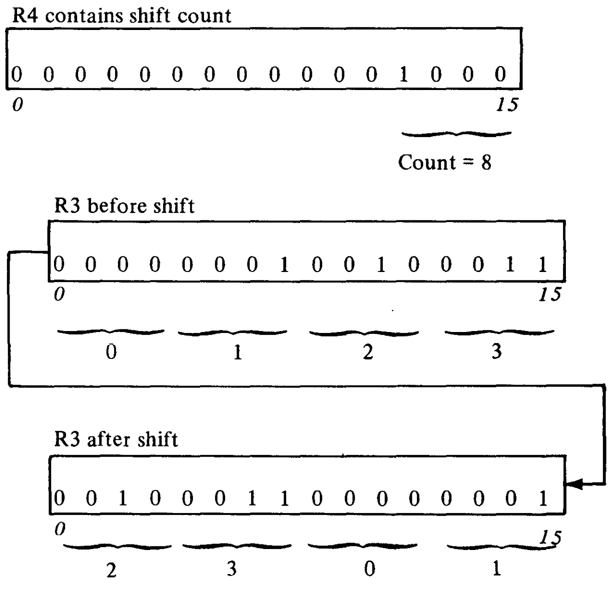
No program checks occur.

Example:

Instruction

Operation code	R1	R2	Function
0 1 1 1 0	0 1 1	1 0 0	1 0 0 0 0
0 4	*5 7*	*8 10*	*11 15*

R3 R4



Shift Left Circular Double (SLCD)

Immediate Count Format

SLCD cnt31,reg

Operation code	*R*	*Count*	*Function*
0 0 1 1 0			1 0 0
0 4	*5 7*	*8 12*	*13 15*

The bits in the register pair specified by the R field and R+1 are shifted left by the number of bit positions specified in the count field.

Within the register pair, the register specified by the R field contains the high-order word (bits 0–15); the register specified by R+1 contains the low-order word (bits 16–31). The bits shifted out of the high-order bit (bit 0) re-enter at the low-order bit (bit 31).

If the count is zero, no shifting occurs. If the R field equals 7, registers 7 and 0 are used for the register pair.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the final contents of the two registers.

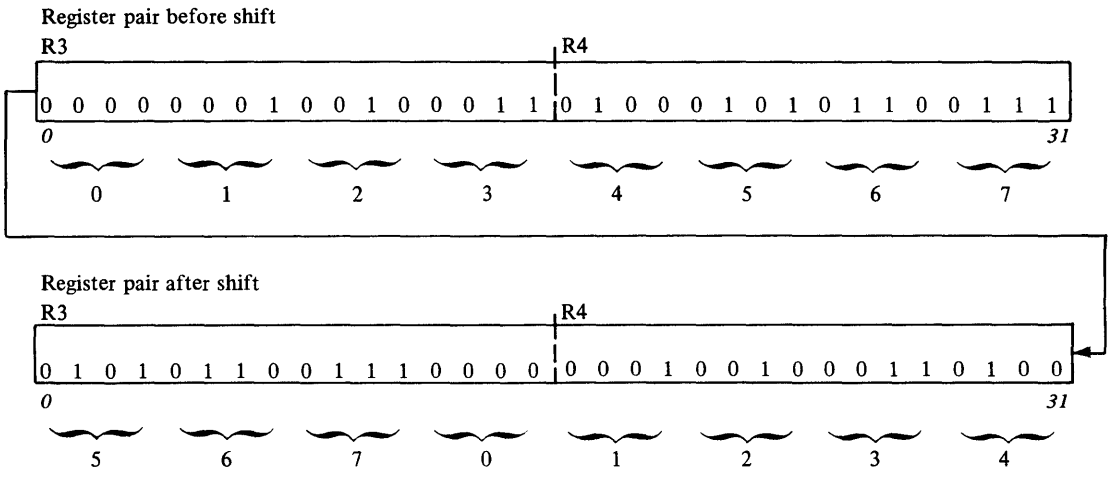
Program Check Conditions

No program checks occur.

Example:

Instruction

Operation code	*R1*	*Count*	*Function*
0 0 1 1 0	0 1 1	1 0 1 0 0	1 0 0
0 4	*5 7*	*8 12*	*13 15*
	R3	Count = 20	



Count in Register Format

SLCD reg,reg

Operation code	R1	R2	Function
0 1 1 1 0			1 0 1 0 0
0 4	*5 7*	*8 10*	*11 15*

The bits in the register pair specified by the R1 field and R1+1 are shifted left by the number of bits specified by the shift count. This count is obtained from bits 8 through 15 of the register specified by the R2 field.

Within the register pair, the register specified by the R1 field contains the high-order word (bits 0–15); the register specified by R1+1 contains the low-order word (bits 16–31). The bits shifted out of the high-order bit (bit 0) re-enter at the low-order bit (bit 31).

If the count is zero, no shifting occurs. If the R1 field equals 7, registers 7 and 0 are used for the register pair.

The contents of the register specified by the R2 field are unchanged unless the R1 (or R1 + 1) and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the registers to be shifted represent 32 bits, shift count values of 0–255 may be specified. Shift count values greater than 32 provide an effective shift of modulo 32.

Indicators

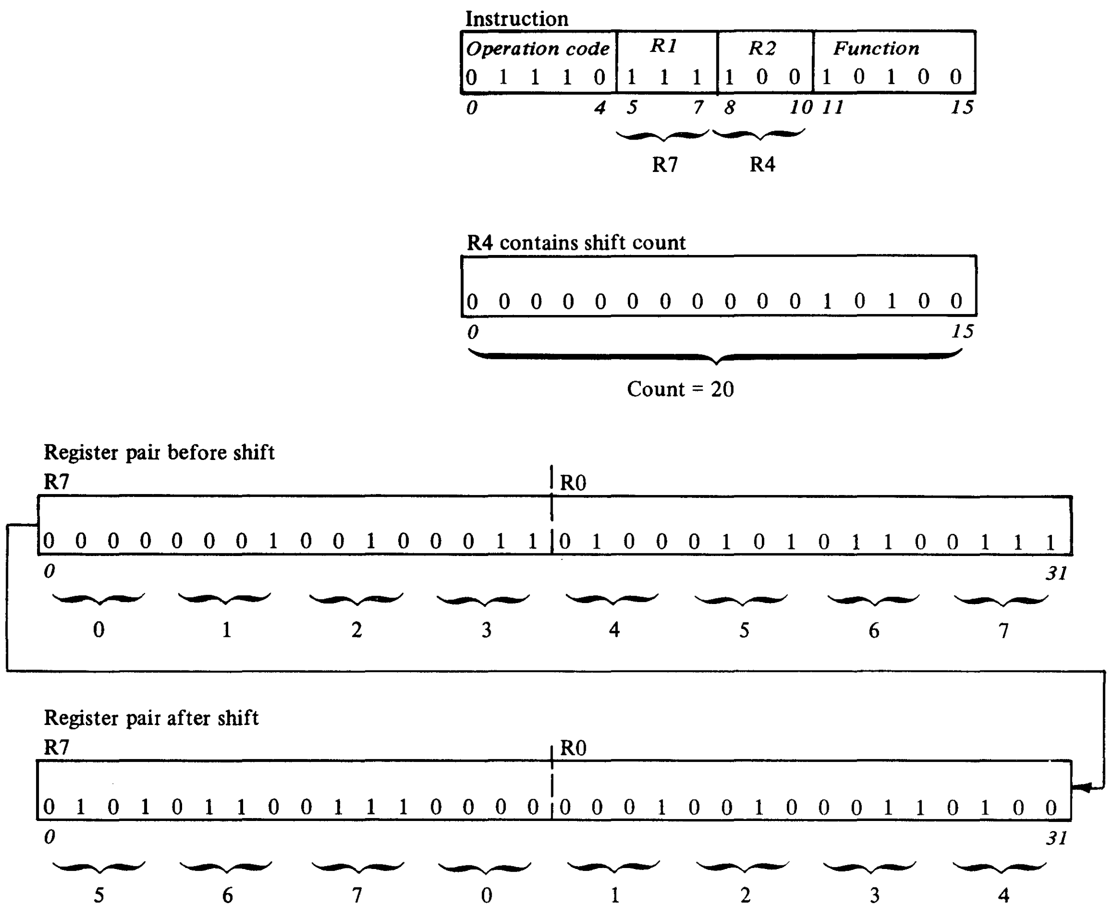
Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the final contents of the two registers.

Program Check Conditions

No program checks occur.

Example:



Shift Left Logical (SLL)

Immediate Count Format

SLL cnt16,reg

Operation code	*R*	*Count*	*Function*
0 0 1 1 0			0 0 1
0 4	*5 7*	*8 12*	*13 15*

The bits in the register specified by the R field are shifted left by the number of bit positions specified in the count field. The vacated low-order bit positions of the register are set to zero. A count of zero causes no shifting to take place.

Although the register to be shifted contains only 16 bits, shift count values of 0–31 may be specified. Shift counts greater than 17 provide an effective shift of 17.

Indicators

Carry. Set to reflect the last bit shifted out of bit 0. If the count is zero, the carry indicator is reset.

Overflow. First reset, then set to a one if the most significant bit in the register (bit 0) has changed during the operation.

Even, Carry, and Overflow. Changed to reflect the final contents of the register.

Program Check Conditions

No program checks occur.

Count in Register Format

SLL reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			1 0 0 0 1
0 4	*5 7*	*8 10*	*11 15*

The bits in the register specified by the R1 field are shifted left by the number of bits specified by the shift count. This count is obtained from bits 8–15 of the register specified by the R2 field. The vacated low-order bits of the register specified by the R1 field are set to zero.

The contents of the register specified by the R2 field are unchanged unless the R1 and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the register shifted contains only 16 bits, shift count values of 0–255 may be specified. Shift counts greater than 17 provide an effective shift of 17.

Indicators

Carry. Set to reflect the last bit shifted out of bit 0. If the count is zero, the carry indicator is reset.

Overflow. First reset, then set to a one if the most significant bit in the register (bit 0) has changed during the operation.

Even, Negative, and Zero. Changed to reflect the final contents of the register specified by the R1 field.

Program Check Conditions

No program checks occur.

Shift Left Logical Double (SLLD)

Immediate Count Format

SLLD cnt31,reg

Operation code	*R*	*Count*	*Function*
0 0 1 1 0			1 0 1
0 4	*5 7*	*8 12*	*13 15*

The bits in the register pair specified by the R field and R+1 are shifted left by the number of bit positions specified in the count field. The vacated low-order bits of the register pair are set to zero.

Within the register pair, the register specified by the R field contains the high-order word (bits 0–15); the register specified by R+1 contains the low-order word (bits 16–31).

If the shift count is zero, no shifting occurs. If the R field equals 7, registers 7 and 0 are used for the register pair.

Indicators

Carry. First reset, then set to reflect the last bit shifted out of the most significant bit position (bit 0) in the register pair.

Overflow. First reset, then set to one if the most significant bit position in the register pair (bit 0) has changed during the operation.

Even, Negative, and Zero. Changed to reflect the final contents of the two registers.

Program Check Conditions

No program checks occur.

Count in Register Format

SLLD reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			1 0 1 0 1
0 4	*5 7*	*8 10*	*11 15*

The bits in the register pair specified by the R1 field and R1+1 are shifted left by the number of bit positions specified by the shift count. This count is obtained from bits 8 through 15 of the register specified by the R2 field. The vacated low-order bit positions of the register pair are set to zero.

Within the register pair, the register specified by the R1 field contains the high-order word (bits 0–15); the register specified by R1+1 contains the low-order word (bits 16–31).

If the shift count is zero, no shifting occurs. If the R1 field equals 7, registers 7 and 0 are used for the register pair.

The contents of the register specified by the R2 field are unchanged unless the R1 (or R1+1) and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the registers to be shifted represent 32 bits, shift count values of 0–255 may be specified. Shift counts greater than 33 provide an effective shift of 33.

Indicators

Carry. First reset, then set to reflect the last bit shifted out of the most significant bit position (bit 0) in the register pair.

Overflow. First reset, then set to a one if the most significant bit in the register pair (bit 0) has changed during the operation.

Even, Negative, and Zero. Changed to reflect the final contents of the two registers.

Program Check Conditions

No program checks occur.

Shift Left and Test (SLT)

SLT reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			1 1 0 0 1
0 4	*5 7*	*8 10*	*11 15*

The bits in the register specified by the R1 field are shifted left. The vacated low-order bit positions of the register are set to zero.

Shifting continues until either one of the following occurs:

1. The number of bits specified by the shift count have been shifted. This count is obtained from bits 8 through 15 of the register specified by the R2 field. No shifting occurs if the shift count is zero.
2. A one-bit is shifted from the high-order bit (bit 0) to the carry indicator. In this case, the remaining shift count is loaded into bits 8 through 15 of the register specified by the R2 field.

Bits 0 through 7 of the register specified by the R2 field are unchanged; these bits must be set to zero to avoid future code obsolescence.

If the R1 and R2 fields specify the same register, the bits in the register are shifted as specified and, when shifting is complete, the remaining shift count replaces the shifted result.

Although the register to be shifted contains only 16 bits, shift count values of 0–255 may be specified.

Indicators

Carry. Set to reflect the last bit shifted out of bit 0. If the count is zero, the carry indicator is reset.

Overflow. First reset, then set to a one if the most significant bit in the register (bit 0) has changed during the operation.

Even, Negative, and Zero. Changed to reflect the final contents of the register specified by the R2 field.

Program Check Conditions

No program checks occur.

Shift Left and Test Double (SLTD)

SLTD reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			1 1 1 0 1
0 4	*5 7*	*8 10*	*11 15*

The bits in the register pair specified by the R1 field and R1+1 are shifted left. The vacated low-order bit positions of the register pair are set to zero.

Shifting continues until either one of the following occurs:

1. The number of bits specified by the shift count have been shifted. This count is obtained from bits 8 through 15 of the register specified by the R2 field. No shifting occurs if the shift count is zero.
2. A one-bit is shifted from the high-order bit to the carry indicator. In this case, the remaining shift count is loaded into bits 8 through 15 of the register specified by the R2 field.

Bits 0 through 7 of the register specified by the R2 field are unchanged; these bits must be set to zero to avoid future code obsolescence.

Within the register pair, the register specified by the R1 field contains the high-order word (bits 0–15); the register specified by R1+1 contains the low-order word (bits 16–31). If the R1 field equals 7, registers 7 and 0 are used for the register pair.

If the R1 (or R1+1) and R2 fields specify the same register, the bits in the register are shifted as specified and, when shifting is complete, the remaining shift count replaces the shifted result.

Although the registers to be shifted contain only 32 bits, shift count values of 0–255 may be specified.

Indicators

Carry. Set to reflect the last bit shifted out of bit 0. If the count is zero, the carry indicator is reset.

Overflow. First reset, then set to a one if the most significant bit in the register (bit 0) has changed during the operation.

Even, Negative, and Zero. Changed to reflect the final contents of the register specified by the R2 field.

Program Check Conditions

No program checks occur.

Shift Right Arithmetic (SRA)

Immediate Count Format

SRA cnt16,reg

Operation code	*R*	*Count*	*Function*
0 0 1 1 0			0 1 1
0 4	5 7	8 12	13 15

The bits in the register specified by the R field are shifted right by the number of bit positions specified in the count field. The value of the sign (the high-order bit) is entered into the vacated high-order bit positions of the register specified by the R field. If the shift count is zero, no shifting takes place.

Although the register to be shifted contains only 16 bits, shift count values of 0–31 may be specified. Shift counts greater than 16 provide an effective shift of 16.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the final contents of the register.

Program Check Conditions

No program checks occur.

Count in Register Format

SRA reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			1 0 0 1 1
0 4	5 7	8 10	11 15

The bits in the register specified by the R1 field are shifted right by the number of bit positions specified by the shift count. This count is obtained from bits 8 through 15 of the register specified by the R2 field. The value of the sign (the high-order bit) is entered into the vacated high-order bit positions of the register specified by the R1 field. If the shift count is zero, no shifting takes place.

The contents of the register specified by the R2 field are unchanged unless the R1 and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the register to be shifted is 16 bits, shift count values of 0–255 may be specified. Shift counts greater than 16 provide an effective shift of 16.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the final contents of the register specified by the R1 field.

Program Check Conditions

No program checks occur.

Shift Right Arithmetic Double (SRAD)

Immediate Count Format

SRAD cnt31,reg

Operation code	R	Count	Function
0 0 1 1 0			1 1 1
0 4	5 7	8 12	13 15

The bits in the register pair specified by the R field and R+1 are shifted right by the number of bit positions specified in the count field. The value of the sign (the high-order bit) is entered into the vacated high-order bit positions of the register pair.

Within the register pair, the register specified by the R field contains the high-order word (bits 0–15); the register specified by R+1 contains the low-order word (bits 16–31).

If the shift count is zero, no shifting occurs. If the R field equals 7, registers 7 and 0 are used for the register pair.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the final contents of the register pair.

Program Check Conditions

No program checks occur.

Count in Register Format

SRAD reg,reg

Operation code	R1	R2	Function
0 1 1 1 0			1 0 1 1 1
0 4	5 7	8 10	11 15

The bits in the register pair specified by the R1 field and R1+1 are shifted right by the number of bit positions specified by the shift count. This count is obtained from bits 8 through 15 of the register specified by the R2 field. The value of the sign (the high-order bit) is entered into the vacated high-order bit positions of the register pair.

Within the register pair, the register specified by the R1 field contains the high-order word (bits 0–15); the register specified by R1+1 contains the low-order word (bits 16–31).

If the shift count is zero, no shifting occurs. If the R field equals 7, registers 7 and 0 are used for the register pair.

The contents of the register specified by the R2 field are unchanged unless the R1 (or R1+1) and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the registers to be shifted represent 32 bits, shift count values of 0–255 may be specified. Shift counts greater than 32 provide an effective shift of 32.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the final contents of the register pair.

Program Check Conditions

No program checks occur.

Shift Right Logical (SRL)

Immediate Count Format

SRL cnt16,reg

Operation code	*R*	*Count*	*Function*
0 0 1 1 0			0 1 0
0 4	*5 7*	*8 12*	*13 15*

The bits in the register specified by the R field are shifted right by the number of bit positions specified in the count field. The vacated high-order bit positions of the register are set to zero. A count of zero causes no shifting to take place.

Although the register to be shifted contains only 16 bits, shift count values of 0–31 may be specified. Shift counts greater than 16 provide an effective shift of 16.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the final contents of the register.

Program Check Conditons

No program checks occur.

Count in Register Format

SRL reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			1 0 0 1 0
0 4	*5 7*	*8 10*	*11 15*

The bits in the register specified by the R1 field are shifted right by the number of bit positions specified by the shift count. This count is obtained from bits 8 through 15 of the register specified by the R2 field. The vacated high-order bit positions of the register specified by the R1 field are set to zero. A count of zero causes no shifting to take place.

The contents of the register specified by the R2 field are unchanged unless the R1 and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the register to be shifted contains only 16 bits, shift count values of 0–255 may be specified. Shift counts greater than 16 provide an effective shift of 16.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the final contents of the register specified by the R1 field.

Program Check Conditions

No program checks occur.

Shift Right Logical Double (SRLD)

Immediate Count Format

SRLD cnt31,reg

Operation code	*R*	*Count*	*Function*
0 0 1 1 0			1 1 0
0 4	*5 7*	*8 12*	*13 15*

The bits in the register pair specified by the R field and R+1 are shifted right by the number of bit positions specified in the count field. The vacated high-order bits of the register pair are set to zero.

Within the register pair, the register specified by the R field contains the high-order word (bits 0–15); the register specified by R+1 contains the low-order word (bits 16–31).

If the shift count is zero, no shifting occurs. If the R field equals 7, registers 7 and 0 are used for the register pair.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the final contents of the register.

Program Check Conditions

No program checks occur.

Count in Register Format

SRLD reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			1 0 1 1 0
0 4	*5 7*	*8 10*	*11 15*

The bits in the register pair specified by the R1 field and R1+1 are shifted right by the number of bit positions specified by the shift count. This count is obtained from bits 8 through 15 of the register specified by the R2 field. The vacated high-order bits of the register pair are set to zero.

Within the register pair, the register specified by the R1 field contains the high-order word (bits 0–15); the register specified by R1+1 contains the low-order word (bits 16–31).

If the shift count is zero, no shifting occurs. If the R1 field equals 7, registers 7 and 0 are used for the register pair.

The contents of the register specified by the R2 field are unchanged unless the R1 (or R1+1) and R2 fields specify the same register. In this case, the register contents are shifted as specified.

Although the registers to be shifted represent 32 bits, shift count values of 0–255 may be specified. Shift counts greater than 32 provide an effective shift of 32.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the final contents of the register pair.

Program Check Conditions

No program checks occur.

Store Multiple (STM)

Refer to *Stack Operations* in Chapter 2 for additional information about the operation of this instruction. The STM instruction is used in conjunction with the Load Multiple and Branch (LMB) instruction described previously in this chapter.

STM reg,addr4[,abcnt]

Format without appended word for effective addressing (AM = 00 or 01)

Operation code		*RB*	*AM*	*Function*
0 1 0 0 0	0 0 0			1 0 0 0
0 4	*5 7*	*8 9*	*10 11*	*12 15*

RL	*N*
16 18	*19 31*

Format with appended word for effective addressing (AM = 10 or 11)

Operation code		*RB*	*AM*	*Function*
0 1 0 0 0	0 0 0			1 0 0 0
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

RL	*N*
32 34	*35 47*

The STM instruction stores the contents of a specified number of registers for the current level into a stack. This stack is defined by the stack control block pointed to by the effective address. (Effective Address Generation is explained in Chapter 2.)

The RL field specifies the last register to be stored. Register 7 is stored first, then registers 0 through the register specified by RL. If RL specifies register 7, only register 7 is stored.

The N field specifies the number of words to be allocated in the stack as a dynamic work area. A value of zero is valid.

The new top element address of the stack (incremented by two) is loaded into the last register stored; that is, the register specified by RL. This address points to the low storage end of the dynamic work area (or the last register stored if N=0).

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

Programming Note. If the AM field equals 01, the contents of the register specified by the RB field are incremented by 2.

Indicators

No indicators are changed.

Program Check Conditions

Invalid Storage Address. Instruction word or stack control block. The instruction is terminated.

Specification Check. Even byte boundary violation (indirect address, stack control block, or stack element).

Soft Exception Trap Condition

Stack Exception.

1. Stack is full.
2. Stack cannot contain the number of words to be stored; that is:
   a. Number of words specified by the N field, plus
   b. The number of registers to be moved, plus
   c. One control word.

If the AM field equals 01, the contents of the register specified by the RB field are incremented.

Stop (STOP)

STOP [ubyte]

Operation code	Function	Parameter
0 1 1 0 0	1 0 0	
0 4	5 7	8 15

The parameter field is ignored by the hardware, and may be used for software flags or indicators.

This instruction is executed only when the Programmer Console is installed and the Mode switch is in the Diagnostic position. Otherwise this instruction performs no operation (no-op). The processor enters the stop state following execution of this instruction. The indicators are unchanged.

Indicators

No indicators are changed.

Program Check Conditions

No program checks occur.

Supervisor Call (SVC)

Execution of this instruction causes a *class interrupt*. Additional information appears in Chapter 3.

SVC ubyte

Operation code	Function	Parameter
0 1 1 0 0	0 0 0	
0 4	5 7	8 15

The instruction address register is incremented by two. The current level status block (LSB) is stored starting at the main storage location specified by the contents of the SVC LSB pointer that resides in main storage location 0010 hexadecimal. The instruction also causes the following events:

- The summary mask (LSR bit 11) is disabled.
- Supervisor state (LSR bit 8) is turned on.
- Trace (LSR bit 10) is turned off.

The parameter field (bits 8–15) is under control of the Programming System. This field is loaded into the low-order byte of register 1. The high-order byte of register 1 is set to zero.

Subsequently, the contents of main storage location 0012 hexadecimal (SVC start instruction address) are loaded into the instruction address register, becoming the address of the next instruction to be fetched.

Indicators

No indicators are changed.

Program Check Conditions

Specification Check. Even byte boundary violation (LSB or SIA pointers).

Subtract Word (SW)

Register/Register Format

SW reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			0 1 0 1 0
0 4	*5 7*	*8 10*	*11 15*

The contents of the register specified by the R1 field are subtracted from the contents of the register specified by the R2 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field remain unchanged unless R1 and R2 specify the same register.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the register. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one word; i.e., if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

No program checks occur.

Register/Storage Format

SW reg,addr4
addr4,reg

Operation code	*R*	*RB*	*AM*	*X*	*Function*
1 1 0 0 1					1 1 1
0 4	*5 7*	*8 9*	*10 11*	*12*	*13 15*

Bit 12 (X): *1 = result to storage*, *0 = result to register*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

A subtract operation is performed between the register specified by the R field and the location specified by the effective address in main storage. (See *Effective Address Generation* in Chapter 2.) Bit 12 of the instruction specifies the destination of the result. The source operand is unchanged.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the word. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one word; i.e., if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

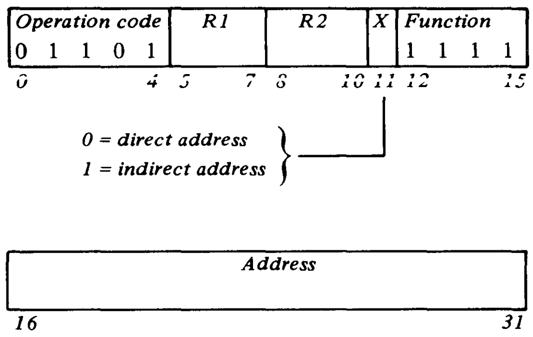
Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Storage to Register Long Format

SW longaddr,reg



The contents of the main storage word location specified by an effective address are subtracted from the contents of the register specified by the R1 field. The result is placed in the register specified by the R1 field.

The effective main storage address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals zero, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0 (direct address). The result from step 1 is the effective address.

   Bit 11=1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the word. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one word; i.e., if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

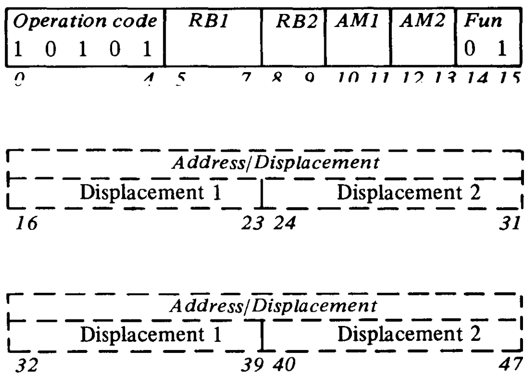
Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Storage/Storage Format

SW addr5,addr4



The address arguments generate the effective addresses of two operands in main storage. (See *Effective Address Generation* in Chapter 2.) Word operand 1 is subtracted from word operand 2. The result replaces operand 2.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the word. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one word; i.e., if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand. The instruction is terminated. If AM1 equals 01 and the operand 2 effective address is invalid, RB1 is incremented.

Specification Check. Even byte boundary violation (indirect address or operand address).

Subtract Word with Carry (SWCY)

SWCY reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			0 1 0 1 1
0 4	*5 7*	*8 10*	*11 15*

If the carry indicator is on at entry (denoting a borrow), a positive one is subtracted from the contents of the register specified by the R2 field. Then the contents of R1 are subtracted from the intermediate result. If the carry indicator is off at entry, the contents of R1 are subtracted from the contents of the register specified by R2. The contents of the register specified by the R1 field are unchanged unless R1 and R2 specify the same register. The final result replaces the contents of the register specified by the R2 field.

Programming Note. This instruction can be used when subtracting multiple word operands. See *Indicators – Multiple Word Operands* in Chapter 2.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the register. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one word; i.e., if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even. Unchanged.

Zero. If on at entry, set to reflect the result. If off at entry, remains off.

Negative. Changed to reflect the result.

Program Check Conditions

No program checks occur.

Subtract Word Immediate (SWI)

Register Immediate Long Format

SWI word,reg[,reg]

Operation code	*R1*	*R2*	*Function*
0 1 1 1 1			0 0 0 1 0
0 4	*5 7*	*8 10*	*11 15*

Immediate
16 31

The immediate field is subtracted from the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are unchanged unless R1 and R2 specify the same register.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the register. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one word; i.e., if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word.

Storage Immediate Format

SWI word,addr4

Format without appended word for effective addressing (AM = 00 or 01)

Operation code		*RB*	*AM*	*Function*
0 1 0 0 0	0 0 0			1 1 1 0
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Immediate
16 31

Format with appended word for effective addressing (AM = 10 or 11)

Operation code		*RB*	*AM*	*Function*
0 1 0 0 0	0 0 0			1 1 1 0
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

Immediate
32 47

The immediate field is subtracted from the contents of the main storage location specified by the effective address. (See *Effective Address Generation* in Chapter 2.) The result replaces the contents of the storage location specified by the effective address.

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

The immediate operand is unchanged.

Indicators

Carry. Turned on by the detection of a borrow beyond the high-order bit position of the word. If no borrow is detected, the carry indicator is reset.

Overflow. Cleared, then turned on if the difference cannot be represented in one word; i.e., if the difference is less than -2^{15} or greater than $+2^{15}-1$.

If an overflow occurs, the result contains the correct low-order 16 bits of the difference; the carry indicator contains the complement of the high-order (sign) bit.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Test Bit (TBT)

TBT (reg,bitdisp)

Operation code	*R*	*Fun*	*Bit displacement*
0 1 0 0 1		0 0	
0 4	*5 7*	*8 9*	*10 15*

The bit displacement is added to the byte address contained in the register specified by the R field to form an effective bit address. The bit displacement field is an unsigned six-bit binary integer. The bit at the effective bit address is tested, and the zero and negative indicators are set to reflect the result.

Indicators

Zero and Negative. First reset, then set as follows:

Value of tested bit	*Indicators Zero*	*Negative*
0	1	0
1	0	1

Even, Carry, and Overflow. Unchanged.

Program Check Conditions

Invalid Storage Address. Operand.

Test Bit and Reset (TBTR)

TBTR (reg,bitdisp)

Operation code	*R*	*Fun*	*Bit displacement*
0 1 0 0 1		1 0	
0 4	*5 7*	*8 9*	*10 15*

The bit displacement is added to the byte address contained in the register specified by the R field to form an effective bit address. The bit displacement field is an unsigned six-bit integer.

The bit at the effective address is tested, and the zero and negative indicators are set to reflect the result.

Following the preceding test, the addressed bit is *unconditionally set to zero.*

Indicators

Zero and Negative. First reset, then set as follows:

Value of tested bit	*Indicators Zero*	*Negative*
0	1	0
1	0	1

Even, Carry, and Overflow. Unchanged.

Program Check Conditions

Invalid Storage Address. Operand.

Test Bit and Set (TBTS)

TBTS (reg,bitdisp)

Operation code	R	Fun	Bit displacement
0 1 0 0 1		0 1	
0 4	5 7	8 9	10 15

The bit displacement is added to the byte address contained in the register specified by the R field to form an effective bit address. The bit displacement field is an unsigned six-bit binary integer.

The bit at the effective address is tested, and the zero and negative indicators are set to reflect the result.

Following the preceding test, the addressed bit is *unconditionally set to one.*

Indicators

Zero and Negative. First reset, then set as follows:

Value of	*Indicators*	
tested bit	*Zero*	*Negative*
0	1	0
1	0	1

Even, Carry, and Overflow. Unchanged.

Program Check Conditions

Invalid Storage Address. Operand.

Test Bit and Invert (TBTV)

TBTV (reg,bitdisp)

Operation code	R	Fun	Bit displacement
0 1 0 0 1		1 1	
0 4	5 7	8 9	10 15

The bit displacement is added to the byte address contained in the register specified by the R field to form an effective bit address. The bit displacement field is an unsigned six-bit binary integer.

The bit at the effective address is tested, and the zero and negative indicators are set to reflect the result.

Following the preceding test, the addressed bit is *unconditionally inverted.*

Indicators

Zero and Negative. First reset, then set as follows:

Value of	*Indicators*	
tested bit	*Zero*	*Negative*
0	1	0
1	0	1

Even, Carry, and Overflow. Unchanged.

Program Check Conditions

Invalid Storage Address. Operand.

Test Word Immediate (TWI)

Register Immediate Long Format

TWI word,reg

Operation code	*R1*		*Function*
0 1 1 1 1		0 0 0	0 0 1 1 1
0 4	*5 7*	*8 10*	*11 15*

Mask
16 31

The contents of the register specified by the R1 field are tested against the mask contained in the immediate word of the instruction. The contents of the register specified by the R1 field are not changed.

Mask bits set to one select the bits to be tested in the register.

Example:

Mask	0000 0000 0101 1100
Register	0000 0000 0011 0101
Selected bits	0 1 01

The selected bits are tested for the following: (1) all bits zero, (2) all bits ones, or (3) a combination of one and zero bits (mixed). The zero and negative indicators are set to reflect the result as shown under *Indicators.*

Instruction bits 8 through 10 are not used and must be set to zero to avoid future code obsolescence.

Indicators

Zero and Negative. Reset, then set as follows:

	Indicators	
Selected bits	*Zero*	*Negative*
All zeros*	1	0
All ones	0	1
Mixed	0	0 (positive)

*Also applies when the mask bits are all zeros.

Even, Carry, and Overflow. Unchanged.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Storage Immediate Format

TWI word,addr4

Format without appended word for effective addressing (AM = 00 or 01)

Operation code		*RB*	*AM*	*Function*
0 1 0 0 0	0 0 0			1 0 1 1
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Mask
16 31

Format with appended word for effective addressing (AM = 10 or 11)

Operation code		*RB*	*AM*	*Function*
0 1 0 0 0	0 0 0			1 0 1 1
0 4	*5 7*	*8 9*	*10 11*	*12 15*

Address/Displacement	
Displacement 1	Displacement 2
16 23	*24 31*

Mask
32 47

The contents of the storage location specified by the effective address are tested against the mask in the immediate word of the instruction. (*Effective Address Generation* is explained in Chapter 2.) Both operands remain unchanged.

Mask bits set to one select the bits to be tested in the storage operand.

Example:

Mask	0000 0000 0000 1110
Storage operand	0000 0000 0101 1110
Selected bits	111

The selected bits are tested for the following: (1) all bits zeros, (2) all bits ones, or (3) a combination of one and zero bits (mixed). The zero and negative indicators are set to reflect the result as shown under *Indicators.*

Bits 5–7 of the instruction are not used and must be set to zero to avoid future code obsolescence.

Indicators

Zero and Negative. Reset, then set as follows:

	Indicators	
Selected bits	*Zero*	*Negative*
All zeros*	1	0
All ones	0	1
Mixed	0	0 (positive)

*Also applies when the mask bits are all zeros.

Even, Carry, and Overflow. Unchanged.

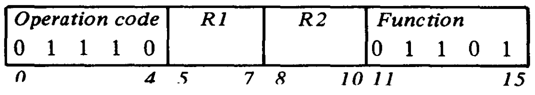
Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Invert Register (VR)

VR reg[,reg]



The contents of the register specified by the R1 field are ones complemented. The result is placed in the register specified by the R2 field. The contents of the register specified by the R1 field are unchanged.

Indicators

Carry and Overflow. Unchanged.

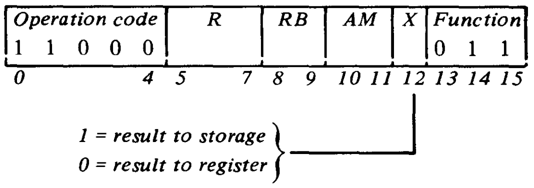
Even, Negative, and Zero. Changed to reflect the result.

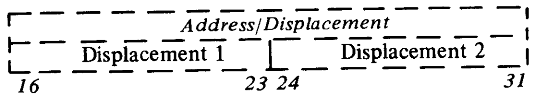
Program Check Conditions

No program checks occur.

Exclusive OR Byte (XB)

XB reg,addr4
addr4,reg





A logical *exclusive OR* operation is performed between the least significant byte of the register specified by the R field and the main storage location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.) Bit 12 of the instruction specifies the destination of the result. The source operand is unchanged. Also, when going from storage to register, bits 0–7 of the register are unchanged.

Example of Exclusive OR Byte:

Register contents	0000 1010 1100 0011
Storage operand	0110 0101
Result	1010 0110

Rule: Either but not both bits.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result of the exclusive OR operation.

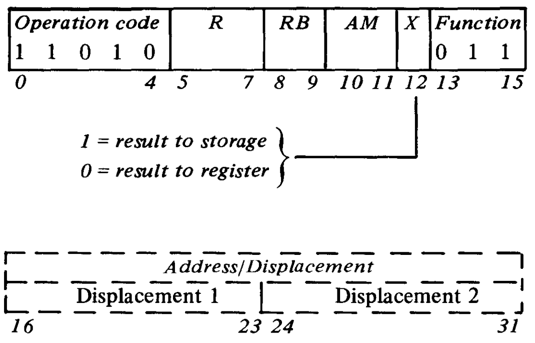
Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address).

Exclusive OR Doubleword (XD)

XD reg,addr4
addr4,reg



A logical *exclusive OR* operation is performed between the contents of the register pair specified by the R field (R and R+1) and the doubleword in main storage specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.) Bit 12 of the instruction specifies the destination of the result. The source operand is unchanged.

If the R field equals 7, registers 7 and 0 are used as the register pair.

Example of Exclusive OR Doubleword:

Register pair contents	0000 0000 1010 1100 0000 0000 1110 1111
Storage operand	0000 0000 1101 0011 0000 0000 1101 0000
Result	0000 0000 0111 1111 0000 0000 0011 1111

Rule: Either but not both bits.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result of the exclusive OR operation.

Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Exclusive OR Word (XW)

Register/Register Format

XW reg,reg

Operation code	*R1*	*R2*	*Function*
0 1 1 1 0			0 0 0 1 1
0 4	5 7	8 10	11 15

The contents of the register specified by the R1 field are *exclusive ORed* bit by bit with the contents of the register specified by the R2 field. The result is placed in the register specified by the R2 field. The contents of the register specified by R1 are unchanged unless R1 and R2 specify the same register.

Example of Exclusive OR Word:

Register contents (R1)	1111 0000 1010 0000
Register contents (R2)	0011 1111 0111 1111
Result	1100 1111 1101 1111

Rule: Either but not both bits.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result of the exclusive OR operation.

Program Check Conditions

No program checks occur.

Register/Storage Format

XW reg,addr4
addr4,reg

Operation code	*R*	*RB*	*AM*	*X*	*Function*
1 1 0 0 1					0 1 1
0 4	5 7	8 9	10 11	12	13 15

1 = result to storage
0 = result to register

Address/Displacement	
Displacement 1	Displacement 2
16 23	24 31

A logical *exclusive OR* operation is performed between the contents of the register specified by the R field and the main storage location specified by the effective address. (*Effective Address Generation* is explained in Chapter 2.) Bit 12 of the instruction specifies the destination of the result. The source operand is unchanged.

Example of Exclusive OR Word:

Register contents (R)	1111 0000 1010 0000
Storage operand	0011 1111 0111 1111
Result	1100 1111 1101 1111

Rule: Either but not both bits.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result of the exclusive OR operation.

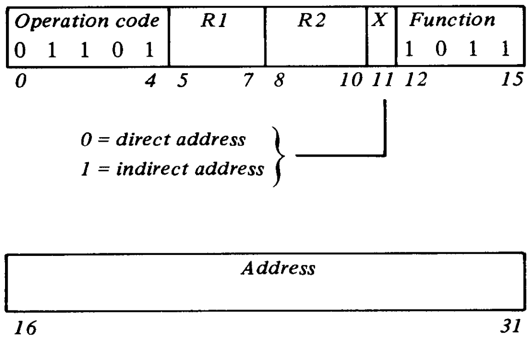
Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Storage to Register Long Format

XW longaddr,reg



A logical *exclusive OR* operation is performed between the contents of the register specified by the R1 field and the contents of the main storage word location specified by the effective address. The result is placed in the register specified by the R1 field.

The effective main storage address is generated as follows:

1. The address field is added to the contents of the register specified by the R2 field. If the R2 field equals zero, no register contributes to the address generation.
2. Instruction bit 11 is tested for direct or indirect addressing:

   Bit 11=0 (direct address). The result from step 1 is the effective address.

   Bit 11=1 (indirect address). The result from step 1 is the address of the main storage location that contains the effective address.

Example of Exclusive OR Word:

Register contents (R1)	1111 0000 1010 0000
Storage operand	0011 1111 0111 1111
Result	1100 1111 1101 1111

Rule: Either but not both bits.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result loaded into the register specified by the R1 field.

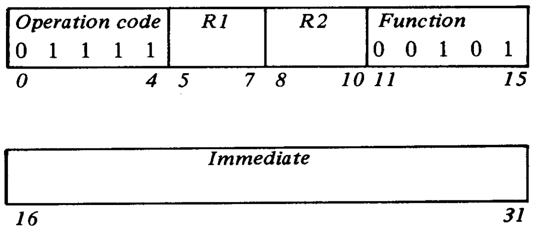
Program Check Conditions

Invalid Storage Address. Instruction word or operand.

Specification Check. Even byte boundary violation (indirect address or operand address).

Exclusive OR Word Immediate (XWI)

XWI word,reg[,reg]



The immediate field is *exclusive ORed* bit by bit with the contents of the register specified by the R1 field. The result is placed in the register specified by the R2 field. The contents of the register specified by R1 are unchanged unless R1 and R2 specify the same register.

Example of Exclusive OR Word:

Register contents (R1)	1111 0000 1010 0000
Immediate operand	0011 1111 0111 1111
Result	1100 1111 1101 1111

Rule: Either but not both bits.

Indicators

Carry and Overflow. Unchanged.

Even, Negative, and Zero. Changed to reflect the result.

Program Check Conditions

Invalid Storage Address. Instruction word.

Appendix A. Instruction Execution Times

This appendix contains instruction execution times.

The instructions are in alphabetic sequence based on assembler mnemonics. Figure A-1 is the additional time required when executing register/storage instructions or storage/storage instructions and assembler syntax for addressing modes.

When running in Stop on Address Mode, instruction execution time is increased by 7.8 microseconds per instruction.

Key to symbols for tables in this appendix:

Symbol	*Meaning*
AM1	Additional time for addressing mode 1 (see Figure A-1).
AM2	Additional time for addressing mode 2 (see Figure A-1).
CL	Current level.
CT	The count value at the beginning of instruction execution.
RL	Limit register (LMB and STM instructions).
RS	Additional addressing-mode time for register/storage instructions (see Figure A-1).
SL	Selected level.
*	Indirect address.

Note 1. The instruction is interruptable.

Note 2. If CT equals zero, use 5.4 for total time.

Note 3. Add 0.6 to the time if AM=01.

Instructions that use addressing mode (AM) for effective address generation require additional time that must be added to the base time for execution.

- RS–the additional time for register/storage instructions

AM	*Time (microseconds)*
00	2.0
01	2.6
10 RB=0	2.6
10 RB≠0	3.2
11 RB=0	3.2
11 RB≠0	6.2

- AM1, AM2–the additional time for storage/storage instructions

AM1	*AM2*	*Time (microseconds)*
00	00	3.6
00	01	4.2
00	10	5.4
00	11	6.2
01	00	4.2
01	01	4.8
01	10	6.0
01	11	6.8
10	00	5.4
10	01	6.0
10	10	7.2
10	11	8.0
11	00	6.2 (If RB1=00, add 0.2)
11	01	6.8 (If RB1=00, add 0.2)
11	10	8.0
11	11	8.8

For each AM field=11, add 1.6 if its corresponding RB≠0.

- Assembler syntax for address modes

Assembler Syntax		*Address Modes*
addr4	*addr5*	(see Note 1)
(reg$^{0-3}$)	(reg)	00
(reg$^{0-3}$)+	(reg)+	01
addr	addr	10
(reg$^{1-3}$,waddr)	(reg$^{1-7}$,waddr)	10
addr*	addr*	11
displ(reg$^{1-3}$,disp2)*	displ(reg$^{1-7}$,disp2)*	11
disp(reg$^{1-3}$)*	disp(reg$^{1-7}$)*	11
(reg$^{1-3}$)*	(reg$^{1-7}$)*	11
(reg$^{1-3}$,disp)*	(reg$^{1-7}$,disp)*	11

Note 1. Register/storage instructions use assembler syntax *addr4* for address mode (AM).

Storage/storage instructions use assembler syntax:

(1) *addr5* for address mode for operand 1 (AM1), and
(2) *addr4* for address mode for operand 2 (AM2).

Figure A-1. Additional instruction times for addressing mode and assembler syntax for addressing modes

Mnemonic	*Instruction name*	*Syntax*	*Execution time (microseconds)*		
AB	Add Byte	reg,addr4	11.0+RS		Note 3
		addr4,reg	10.6+RS		Note 3
ABI	Add Byte Immediate	byte,reg	4.2		
ACY	Add Carry Register	reg	4.8		
AD	Add Double Word	reg,addr4	9.4+RS		Note 3
		addr4,reg	8.2+RS		Note 3
		addr5,addr4	11.2+AM1+AM2		
AW	Add Word	reg,reg	4.2		
		reg,addr4	5.8+RS		
		addr4,reg	5.2+RS		
		addr5,addr4	6.8+AM1+AM2		
		longaddr,reg	7.2		
		longaddr*,reg	7.8		
AWCY	Add Word With Carry	reg,reg	4.8		
AWI	Add Word Immediate	word,reg[,reg]	5.4		
		word,addr4	6.6+RS		
B	Branch Unconditional	longaddr	4.8		
		longaddr*	5.4		
BAL	Branch and Link	longaddr,reg	6.6		
		longaddr*,reg	7.2		
BALS	Branch and Link Short	(reg,jdisp)*	5.4		
		(reg)*	5.4		
		addr*	5.4		
BALX	Branch and Link External		See BAL		
			Test Condition		
			No Branch	*Branch*	
BC	Branch on Condition	cond,longaddr	4.2	6.0	
		cond,longaddr*	4.2	6.6	
			Test Condition		
			No Branch	*Branch*	
BCC	Branch on Condition Code	cond,longaddr	4.2	6.0	
		cond,longaddr*	4.2	6.6	
BCY	Branch on Carry		See BC		
BE	Branch on Equal		See BC		
BER	Branch on Error		See BNCC		
BEV	Branch on Even		See BC		
BGE	Branch on Arithmetically Greater Than or Equal		See BNC		
BGT	Branch on Arithmetically Greater Than		See BNC		
BLE	Branch on Arithmetically Less Than or Equal		See BC		
BLGE	Branch on Logically Greater Than or Equal		See BNC		
BLGT	Branch on Logically Greater Than		See BNC		
BLLE	Branch on Logically Less Than or Equal		See BC		
BLLT	Branch on Logically Less Than		See BC		
BLT	Branch on Arithmetically Less Than		See BC		
BMIX	Branch if Mixed		See BC		
BN	Branch on Negative		See BC		
			Test Condition		
			No Branch	*Branch*	
BNC	Branch on Not Condition	cond,longaddr	4.2	6.0	
		cond,longaddr*	4.2	6.6	

Mnemonic	*Instruction name*	*Syntax*	*Execution time (microseconds)*		
			Test Condition *No Branch*	*Branch*	
BNCC	Branch on Not Condition Code	cond,longaddr	4.2	6.0	
		cond,longaddr*	4.2	6.6	
BNCY	Branch on No Carry		See BNC		
BNE	Branch on Not Equal		See BNC		
BNER	Branch if Not Error		See BCC		
BNEV	Branch on Not Even		See BNC		
BNMIX	Branch if Not Mixed		See BNC		
BNN	Branch on Not Negative		See BNC		
			Test Condition *No Branch*	*Branch*	
BNOV	Branch on Not Overflow	longaddr	3.6	5.4	
		longaddr*	3.6	6.0	
BNP	Branch on Not Positive		See BNC		
BNZ	Branch on Not Zero		See BNC		
BOFF	Branch if Off		See BC		
BON	Branch if ON		See BC		
			Test Condition *No Branch*	*Branch*	
BOV	Branch on Overflow	longaddr	3.6	5.4	
		longaddr*	3.6	6.0	
BP	Branch on Positive		see BC		
BX	Branch external	vcon	See B		
BXS	Branch Indexed Short	(reg1-7,jdisp)	4.2		
		(reg1-7)	4.2		
		addr	4.2		
BZ	Branch on Zero		See BC		
CB	Compare Byte	addr4,reg	8.8+RS		Note 3
		addr5,addr4	7.2+AM1+AM2		(Note)
	Note. 7.8 if operand 2 is at an odd address.				
CBI	Compare Byte Immediate	byte,reg	3.6		
CD	Compare Double Word	addr4,reg	8.2+RS		Note 3
		addr5,addr4	9.8+AM1+AM2		
CFED	Compare Byte Field Equal and Decrement	(reg),(reg)	8.4+(7.8 x CT)		Note 1 Note 2
CFEN	Compare Byte Field Equal and Increment	(reg),(reg)	8.4+(7.8 x CT)		Note 1 Note 2
CFNED	Compare Byte Field Not Equal and Decrement	(reg),(reg)	8.4+(7.8 x CT)		Note 1 Note 2
CFNEN	Compare Byte Field Not Equal and Increment	(reg),(reg)	8.4+(7.8 x CT)		Note 1 Note 2
CMR	Complement Register	reg[,reg]	4.8		
CPCL	Copy Current Level	reg	5.4		
CPCON	Copy Console Data Buffer	reg	4.8		
CPIMR	Copy Interrupt Mask Register	addr4	6.0+RS		
CPIPF	Copy In-Process Flags	addr4	9.4+RS		
CPLB	Copy Level Block	reg,addr4	23.4+RS	SL=CL	
			24.6+RS	SL≠CL	
CPLSR	Copy Level Status Register	reg	6.0		
CPPSR	Copy Processor Status and Reset	addr4	5.8+RS		
CW	Compare Word	reg,reg	4.2		
		addr4,reg	5.8+RS		
		addr5,addr4	5.8+AM1+AM2		

Mnemonic	*Instruction name*	*Syntax*	*Execution time (microseconds)*		
CWI	Compare Word Immediate	word,reg	5.4		
		word,addr4	5.8+RS		
DB	Divide Byte	addr4,reg	7.0+RS	Minimum	
			88.6+RS	Maximum	
DD	Divide Double Word	addr4,reg	7.0+RS	Minimum	
			159.4+RS	Maximum	
DIAG	Diagnose	ubyte	4.8	Minimum	
			10.8	Maximum	
DIS	Disable	ubyte	7.2		(Note)
	Note. 5.4 if bit 15 of parameter field equals zero.				
DW	Divide Word	addr4,reg	7.0+RS	Minimum	
			86.8+RS	Maximum	
EN	Enable	ubyte	7.2		(Note)
	Note. 5.4 if bit 15 of parameter field equals zero.				
FFD	Fill Byte Field and Decrement	reg,(reg)	8.4+(3.6 x CT)		Note 1 Note 2
FFN	Fill Byte Field and Increment	reg,(reg)	8.4+(3.6 x CT)		Note 1 Note 2
IO	Operate I/O	longaddr	9.5 Typical		(Note)
			16.8 Halt I/O		
		longaddr*	10.1 Typical		(Note)
			17.4 Halt I/O		
	Note. The range within which the OIO and OIO Indirect Instructions may be successfully completed is 9.1 min/9.7 min to 28.9 max/29.5 max respectively. The minimum times do not reflect any delays which may be induced by the attachment and/or device to which the command is directed.				
IR	Interchange Registers	reg,reg	4.8		
J	Jump Unconditional	jdisp	4.2		
		jaddr	4.2		
JAL	Jump and Link	jdisp,reg	5.4		
		jaddr,reg	5.4		
			Test Condition		
			No Branch	*Branch*	
JC	Jump on Condition	cond,jdisp	4.2	6.0	
		cond,jaddr	4.2	6.0	
			CT=0 CT=1	CT>1	
JCT	Jump on Count	jdisp,reg	5.4 4.8	6.6	
		jaddr,reg	5.4 4.8	6.6	
JCY	Jump on Carry		See JC		
JE	Jump on Equal		See JC		
JEV	Jump on Even		See JC		
JGE	Jump on Arithmetically Greater Than or Equal		See JNC		
JGT	Jump on Arithmetically Greater Than		See JNC		
JLE	Jump on Arithmetically Less Than or Equal		See JC		
JLGE	Jump on Logically Greater Than or Equal		See JNC		
JLGT	Jump on Logically Greater Than		See JNC		
JLLE	Jump on Logically Less Than or Equal		See JC		

Mnemonic	*Instruction name*	*Syntax*	*Execution time (microseconds)*		
JLLT	Jump on Logically Less Than		See JC		
JLT	Jump on Arithmetically Less Than		See JC		
JMIX	Jump if Mixed		See JC		
JN	Jump on Negative		See JC		
			Test Condition		
			No Branch	*Branch*	
JNC	Jump on Not Condition	cond,jdisp	4.2	6.0	
		cond,jaddr	4.2	6.0	
JNCY	Jump on No Carry		See JNC		
JNE	Jump on Not Equal		See JNC		
JNEV	Jump on Not Even		See JNC		
JNMIX	Jump if Not Mixed		See JNC		
JNN	Jump on Not Negative		See JNC		
JNP	Jump on Not Positive		See JNC		
JNZ	Jump on Not Zero		See JNC		
JOFF	Jump if Off		See JC		
JON	Jump if On		See JC		
JP	Jump on Positive		See JC		
JZ	Jump on Zero		See JC		
LEX	Level Exit	[ubyte]	8.8	Minimum	
			19.6	Maximum	
LMB	Load Multiple and Branch	addr4	17.2+RS		(Note)
	Note. For each specified register from 0 through 6, add an additional 3.2 per register.				
MB	Multiply Byte	addr4,reg	11.2+RS	Minimum	
			26.2+RS	Maximum	
MD	Multiply Doubleword	addr4,reg	10.0+RS	Minimum	
			98.2+RS	Maximum	
MVA	Move Address	addr4,reg	5.2+RS		
		addr5,addr4	5.4+RS		
MVB	Move Byte	reg,addr4	6.6+RS		Note 3
		addr4,reg	8.2+RS		Note 3
		addr5,addr4	5.4+AM1+AM2		
MVBI	Move Byte Immediate	byte,reg	3.6		
MVBZ	Move Byte and Zero	addr4,reg	8.8+RS		Note 3
MVD	Move Doubleword	reg,addr4	8.4+RS		Note 3
		addr4,reg	7.6+RS		Note 3
		addr5,addr4	8.4+AM1+AM2		
MVDZ	Move Doubleword and Zero	addr4,reg	9.6+RS		Note 3
MVFD	Move Byte Field and Decrement	(reg),(reg)	8.4+(5.4 x CT)		Note 1 Note 2
MVFN	Move Byte Field and Increment	(reg),(reg)	8.4+(5.4 x CT)		Note 1 Note 2
MVW	Move Word	reg,reg	4.2		
		reg,addr4	5.4+RS		
		addr4,reg	5.2+RS		
		addr5,addr4	4.8+AM1+AM2		
		reg,longaddr	7.6		
		reg,longaddr*	8.2		
		longaddr,reg	7.2		
		longaddr*,reg	7.8		
MVWI	Move Word Immediate	word,reg	5.2		
		word,addr4	5.4+RS		

Mnemonic	*Instruction name*	*Syntax*	*Execution time (microseconds)*		
MVWS	Move Word Short	reg,shortaddr	6.0		
		reg,shortaddr*	6.6		
		shortaddr,reg	5.4		
		shortaddr*,reg	6.2		
MVWZ	Move Word and Zero	addr4,reg	6.2+RS		
MW	Multiply Word	addr4,reg	9.4+RS	Minimum	
			40.6+RS	Maximum	
NOP	No Operation		4.2		
NWI	And Word Immediate	word,reg[,reg]	5.4		
OB	OR Byte	reg,addr4	8.8+RS		Note 3
		addr4,reg	8.8+RS		Note 3
		addr5,addr4	7.8+AM1+AM2		
OD	OR Doubleword	reg,addr4	9.0+RS		Note 3
		addr4,reg	7.6+RS		Note 3
		addr5,addr4	9.8+AM1+AM2		
OW	OR Word	reg,reg	4.2		
		reg,addr4	5.8+RS		
		addr4,reg	5.2+RS		
		addr5,addr4	6.0+AM1+AM2		
		longaddr,reg	7.2		
		longaddr*,reg	7.8		
OWI	OR Word Immediate	word,reg[,reg]	5.4		
		word,addr4	6.6+RS		
PB	Pop Byte	addr4,reg	11.2+RS		
PD	Pop Doubleword	addr4,reg	10.4+RS		
PSB	Push Byte	reg,addr4	10.6+RS		
PSD	Push Doubleword	reg,addr4	11.0+RS		
PSW	Push Word	reg,addr4	9.8+RS		
PW	Pop Word	addr4,reg	9.2+RS		
RBTB	Reset Bits Byte	reg,addr4	8.8+RS		Note 3
		addr4,reg	8.8+RS		Note 3
		addr5,addr4	8.0+AM1+AM2		
RBTD	Reset Bits Doubleword	reg,addr4	9.0+RS		Note 3
		addr4,reg	8.0+RS		Note 3
		addr5,addr4	9.8+AM1+AM2		
RBTW	Reset Bits Word	reg,reg	4.2		
		reg,addr4	5.8+RS		
		addr4,reg	5.8+RS		
		addr5,addr4	6.6+AM1+AM2		
		longaddr,reg	7.2		
		longaddr*,reg	7.8		
RBTWI	Reset Bits Word Immediate	word,reg[,reg]	5.4		
		word,addr4	6.6+RS		
SB	Subtract Byte	reg,addr4	11.0+RS		Note 3
		addr4,reg	10.6+RS		Note 3
SCY	Subtract Carry Indicator	reg	4.8		
SD	Subtract Doubleword	reg,addr4	9.4+RS		Note 3
		addr4,reg	8.8+RS		Note 3
		addr5,addr4	11.2+AM1+AM2		
SECON	Set Console Data Lights	reg	6.0		
SEIMR	Set Interrupt Mask Register	addr4	8.0+RS		
SEIND	Set Indicators	reg	6.0		
SELB	Set Level Block	reg,addr4	24.4	Minimum	
			43.6	Maximum	
SFED	Scan Byte Field Equal and Decrement	reg,(reg)	8.4+(6.0 x CT)		Note 1 Note 2
SFEN	Scan Byte Field Equal and Increment	reg,(reg)	8.4+(6.0 x CT)		Note 1 Note 2

Mnemonic	*Instruction name*	*Syntax*	*Execution time (microseconds)*		
SFNED	Scan Byte Field Not Equal and Decrement	reg,(reg)	8.4+(6.0 x CT)		Note 1 Note 2
SFNEN	Scan Byte Field Not Equal and Increment	reg,(reg)	8.4+(6.0 x CT)		Note 1 Note 2
SLC	Shift Left Circular	cnt16,reg	6.6	Minimum	
			7.8	Maximum	
		reg,reg	6.6	Minimum	
			9.0	Maximum	
SLCD	Shift Left Circular Double	cnt31,reg	15.0	Minimum	
			18.0	Maximum	
		reg,reg	15.0	Minimum	
			18.0	Maximum	
SLL	Shift Left Logical	cnt16,reg	5.4	Minimum	
			12.0	Maximum	
		reg,reg	5.4	Minimum	
			10.8	Maximum	
SLLD	Shift Left Logical Double	cnt31,reg	6.6	Minimum	
			18.6	Maximum	
		reg,reg	6.6	Minimum	
			18.6	Maximum	
SLT	Shift Left and Test	reg,reg	7.2	Minimum	
			37.2	Maximum	
SLTD	Shift Left and Test Double	reg,reg	7.8	Minimum	
			105.6	Maximum	
SRA	Shift Right Arithmetic	cnt16,reg	6.6	Minimum	
			10.2	Maximum	
		reg,reg	6.6	Minimum	
			10.2	Maximum	
SRAD	Shift Right Arithmetic Double	cnt31,reg	10.8	Minimum	
			16.8	Maximum	
		reg,reg	10.8	Minimum	
			16.8	Maximum	
SRL	Shift Right Logical	cnt16,reg	5.4	Minimum	
			9.0	Maximum	
		reg,reg	5.4	Minimum	
			9.0	Maximum	
SRLD	Shift Right Logical Double	cnt31,reg	10.2	Minimum	
			16.2	Maximum	
		reg,reg	10.2	Minimum	
			16.2	Maximum	
STM	Store Multiple	reg,addr4[,abcnt]	21.8+RS		(Note)
	Note. For each specified register from 0 through 6, add an additional 3.0 per register.				
STOP	Stop	[ubyte]	4.2		(Note)
	Note. Time given is for execution as a No-Op.				
SVC	Supervisor Call	ubyte	26.0		
SW	Subtract Word	reg,reg	4.2		
		reg,addr4	6.2+RS		
		addr4,reg	5.8+RS		
		addr5,addr4	6.8+AM1+AM2		
		longaddr,reg	7.8		
		longaddr*,reg	8.4		
SWCY	Subtract Word With Carry	reg,reg	4.8		
SWI	Subtract Word Immediate	word,addr4	6.6+RS		
		word,reg[,reg]	5.4		
TBT	Test Bit	(reg,bitdisp)	8.4		
TBTR	Test Bit and Reset	(reg,bitdisp)	9.6		
TBTS	Test Bit and Set On	(reg,bitdisp)	9.6		

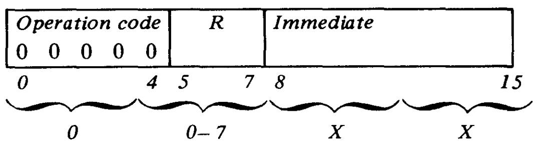
Mnemonic	Instruction name	Syntax	Execution time (microseconds)		
TBTV	Test Bit and Invert	(reg,bitdisp)	9.6		
TWI	Test Word Under Mask Immediate	word,reg	7.2		
			7.8	All bits = 1	
		word,addr4	7.0+RS		
			7.6+RS	All bits = 1	
VR	Invert Register	reg[,reg]	4.8		
XB	Exclusive OR Byte	reg,addr4	8.8+RS		Note 3
		addr4,reg	8.8+RS		Note 3
XD	Exclusive OR Doubleword	reg,addr4	9.0+RS		Note 3
		addr4,reg	8.2+RS		Note 3
XW	Exclusive OR Word	reg,reg	4.2		
		reg,addr4	5.8+RS		
		addr4,reg	5.2+RS		
		longaddr,reg	7.2		
		longaddr*,reg	7.8		
XWI	Exclusive OR Word Immediate	word,reg[,reg]	5.4		

Appendix B. Instruction Formats

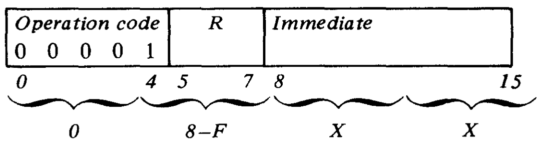
The following instruction formats are shown in ascending sequence based on operation code. Bits zero through four of the first instruction word comprise the operation code field. Bit combinations are shown for each operation code along with the hexadecimal representation.

Some instructions contain a function field that modifies the operation code to form individual instructions within a group. Each chart shows the function field bit combinations in hexadecimal and in ascending sequence. The assembler mnemonic, assembler syntax, and instruction name are listed for the individual instructions. The asterisk shown with the assembler syntax indicates indirect addressing.

Refer to Chapter 2, *Effective Address Generation,* for a description of the Address Mode (AM) appended words.

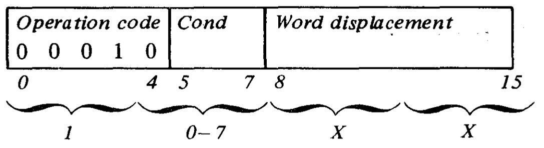


0	0–7	X	X	ABI	byte,reg	Add Byte Immediate



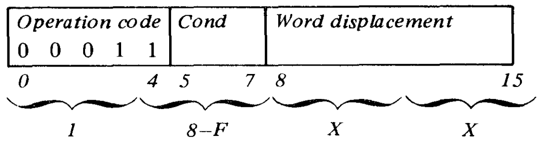
0	8–F	X	X	MVBI	byte,reg	Move Byte Immediate

1xxx
2xxx



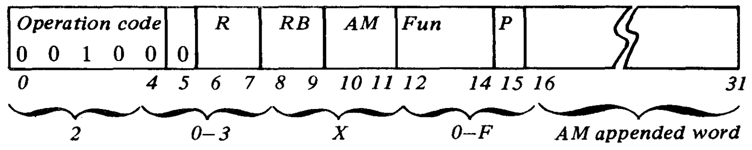
1	0–7	X	X	JC	cond,jdisp	Jump on Condition
				JC	cond,jaddr	Jump on Condition

Extended mnemonics:
JCY, JE, JEV, JLE, JLLE, JLLT, JLT, JMIX,
JN, JOFF, JON, JP, JZ

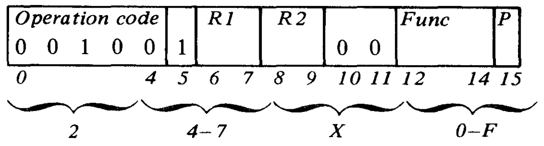


1	8–F	X	X	JNC	cond,jdisp	Jump on Not Condition
				JNC	cond,jaddr	Jump on Not Condition

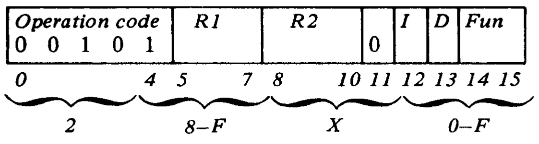
Extended mnemonics:
JGE, JGT, JLGE, JLGT, JNCY, JNE, JNEV,
JNMIX, JNN, JNP, JNZ



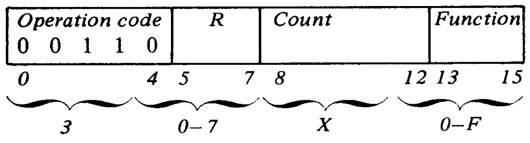
2	0–3	X	X	Illegal operation code (invalid function)



2	4–7	X	X	Illegal operation code (invalid function)

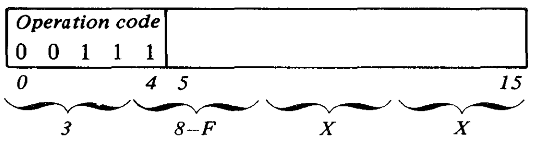


2	8-F	X	0	MVFD	(reg),(reg)	Move Byte Field and Decrement
			1	(unused)		
			2	CFNED	(reg),(reg)	Compare Byte Field Not Equal and Decrement
			3	CFED	(reg),(reg)	Compare Byte Field Equal and Decrement
			4	MVFN	(reg),(reg)	Move Byte Field and Increment
			5	(unused)		
			6	CFNEN	(reg),(reg)	Compare Byte Field Not Equal and Increment
			7	CFEN	(reg),(reg)	Compare Byte Field Equal and Increment
			8	FFD	reg,(reg)	Fill Byte Field and Decrement
			9	(unused)		
			A	SFNED	reg,(reg)	Scan Byte Field Not Equal and Decrement
			B	SFED	reg,(reg)	Scan Byte Field Equal and Decrement
			C	FFN	reg,(reg)	Fill Byte Field and Increment
			D	(unused)		
			E	SFNEN	reg,(reg)	Scan Byte Field Not Equal and Increment
			F	SFEN	reg,(reg)	Scan Byte Field Equal and Increment

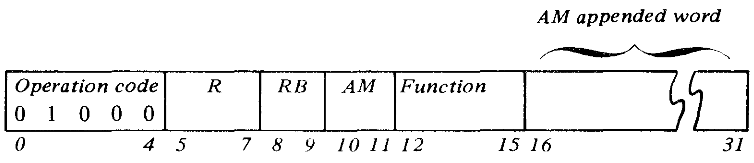


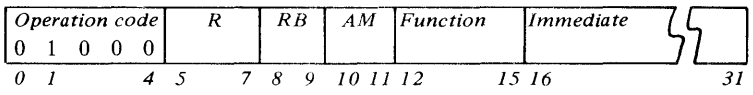
3	0–7	X	0,8	SLC	cnt16,reg	Shift Left Circular
			1,9	SLL	cnt16,reg	Shift Left Logical
			2,A	SRL	cnt16,reg	Shift Right Logical
			3,B	SRA	cnt16,reg	Shift Right Arithmetic
			4,C	SLCD	cnt31,reg	Shift Left Circular Double
			5,D	SLLD	cnt31,reg	Shift Left Logical Double
			6,E	SRLD	cnt31,reg	Shift Right Logical Double
			7,F	SRAD	cnt31,reg	Shift Right Arithmetic Double

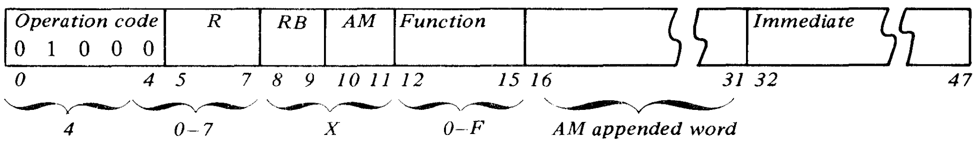
3xxx
4xxx



3	8–F	X	X	Illegal operation code (program check condition)

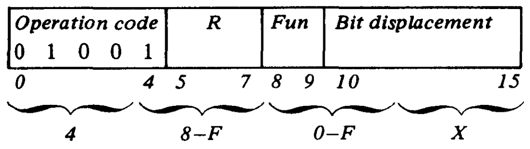




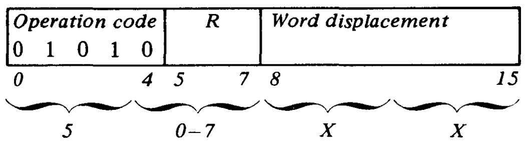


4	0–7	X	0	MVA	addr,addr4	Move Address
			0	MVWI	word,addr4	Move Word Immediate
			1	(invalid)		
			2	(invalid)		
			3	(invalid)		
			4	MVA	addr4,reg	Move Address (Note 1)
			4	MVWI	word,reg	Move Word Immediate (Note 1)
			5	(invalid)		
			6	(invalid)		
			7	(invalid)		
			8	STM	reg,addr4[,abcnt]	Store Multiple
			9	AWI	word,addr4	Add Word Immediate
			A	LMB	addr4	Load Multiple and Branch (Note 1)
			B	TWI	word,addr4	Test Word Under Mask Immediate
			C	OWI	word,addr4	OR Word Immediate
			D	RBTWI	word,addr4	Reset Bits Word Immediate
			E	SWI	word,addr4	Subtract Word Immediate
			F	CWI	word,addr4	Compare Word Immediate

Note 1. Use format without immediate field.

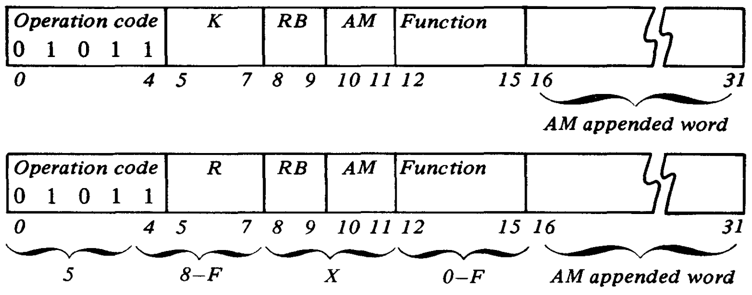


4	8–F	0–3	X	TBT	(reg,bitdisp)	Test Bit
		4–7		TBTS	(reg,bitdisp)	Test Bit and Set On
		8–B		TBTR	(reg,bitdisp)	Test Bit and Reset
		C–F		TBTV	(reg,bitdisp)	Test Bit and Invert



5	0	0	0	NOP	No Operation	
	0	X	X	J	jdisp	Jump Unconditional
				J	jaddr	Jump Unconditional
	1–7	X	X	BXS	(reg$^{1-7}$,jdisp)	Branch Indexed Short
				BXS	(reg$^{1-7}$)	Branch Indexed Short
				BXS	addr	Branch Indexed Short

5xxx

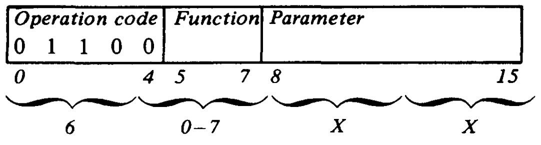


5 8–F X	0	SEIMR	addr4	Set Interrupt Mask Register
	1	(Note 2)		
	2	(Note 3)		
	3	(Note 4)		
	4	(Note 3)		
	5	(invalid)		
	6	SELB	reg,addr4	Set Level Status Block
	7	(invalid)		
	8	CPIMR	addr4	Copy Interrupt Mask Register
	9	(Note 2)		
	A	(Note 3)		
	B	(Note 4)		
	C	(Note 3)		
	D	CPIPF	addr4	Copy In-Process Flags
	E	CPLB	reg,addr4	Copy Level Block
	F	CPPSR	addr4	Copy Processor Status and Reset

Note 2. Supervisor state: program check, invalid function
Problem state: program check, privilege violate

Note 3. Supervisor state: No-op
Problem state: program check, privilege violate

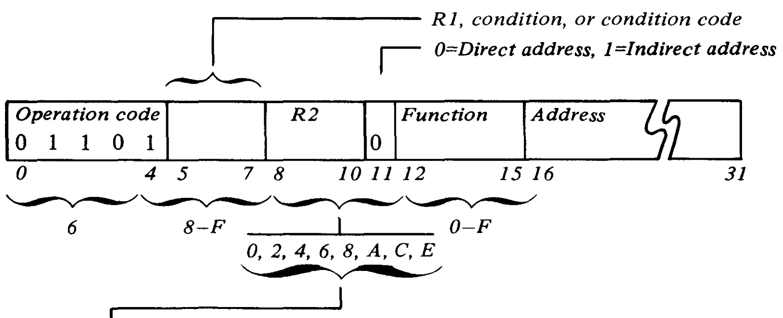
Note 4. Supervisor state: soft exception trap, invalid function
Problem state: program check, privilege violate



6 0 X X	SVC	ubyte	Supervisor Call
1	LEX	[ubyte]	Level Exit
2	EN	ubyte	Enable
3	DIS	ubyte	Disable
4	STOP	[ubyte]	Stop
5	DIAG	ubyte	Diagnose
6	(Note 5)		
7	(invalid)		

Note 5. Supervisor state: No-op
Problem state: program check, privilege violate

6xxx



6 8–F			
0	BC	cond,longaddr	Branch on Condition (Note 6)
1	BNC	cond,longaddr	Branch on Not Condition (Note 7)
2	B	longaddr	Branch Unconditional (Note 8)
3	BAL	longaddr,reg	Branch and Link (Note 9)
4	BCC	cond,longaddr	Branch on Condition Code (Note 10)
5	BNCC	cond,longaddr	Branch on Not Condition Code (Note 11)
6	BOV	longaddr	Branch on Overflow
7	BNOV	longaddr	Branch on Not Overflow
8	MVW	longaddr,reg	Move Word
9	OW	longaddr,reg	OR Word
A	RBTW	longaddr,reg	Reset Bits Word
B	XW	longaddr,reg	Exclusive OR Word
C	IO	longaddr	Operate I/O
D	MVW	reg,longaddr	Move Word
E	AW	longaddr,reg	Add Word
F	SW	longaddr,reg	Subtract Word

Note 6. Extended mnemonics: BCY, BE, BEV, BLE, BLLE, BLLT, BLT, BMIX, BN, BOFF, BON, BP, BZ

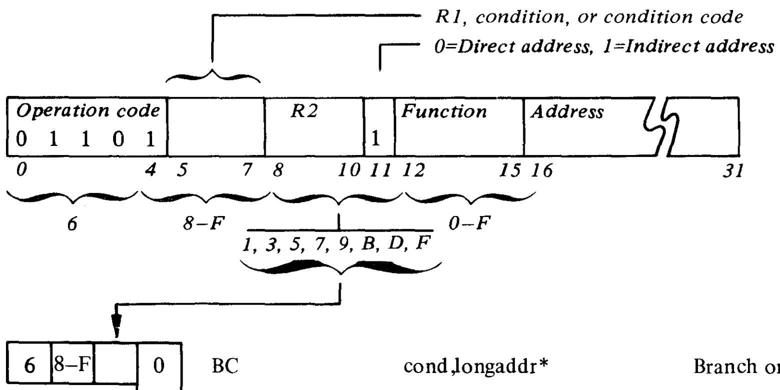
Note 7. Extended mnemonics: BGE, BGT, BLGE, BLGT, BNCY, BNE, BNEV, BNMIX, BNN, BNOFF, BNON, BNP, BNZ

Note 8. Extended mnemonic: BX

Note 9. Extended mnemonic: BALX

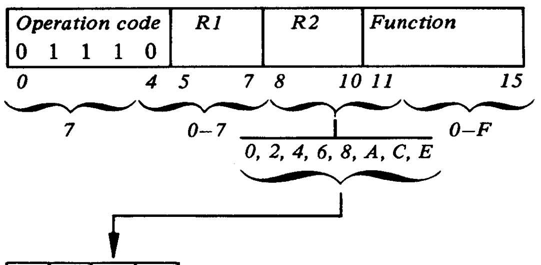
Note 10. Extended mnemonic: BNER

Note 11. Extended mnemonic: BER

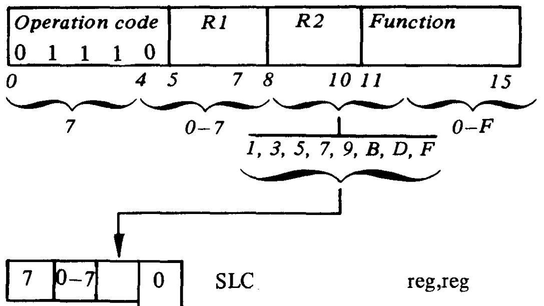


0	BC	cond,longaddr*	Branch on Condition
1	BNC	cond,longaddr*	Branch on Not Condition
2	B	longaddr*	Branch Unconditional
3	BAL	longaddr*,reg	Branch and Link
4	BCC	cond,longaddr*	Branch on Condition Code
5	BNCC	cond,longaddr*	Branch on Not Condition Code
6	BOV	longaddr*	Branch on Overflow
7	BNOV	longaddr*	Branch on Not Overflow
8	MVW	longaddr*,reg	Move Word
9	OW	longaddr*,reg	OR Word
A	RBTW	longaddr*,reg	Reset Bits Word
B	XW	longaddr*,reg	Exclusive OR Word
C	IO	longaddr*	Operate I/O
D	MVW	reg,longaddr*	Move Word
E	AW	longaddr*,reg	Add Word
F	SW	longaddr*,reg	Subtract Word

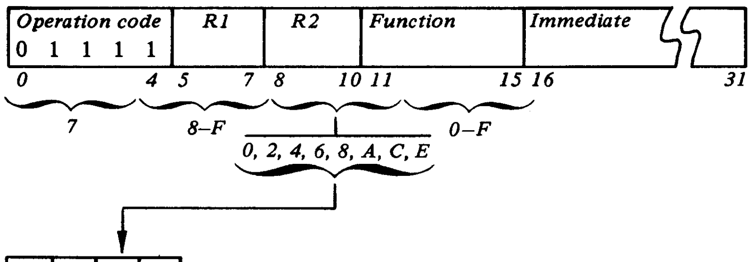
7xxx



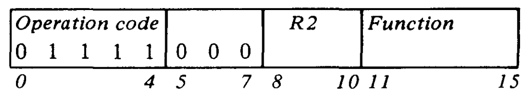
7 0–7			
0	RBTW	reg,reg	Reset Bits Word
1	OW	reg,reg	OR Word
2	SCY	reg	Subtract Carry Indicator
3	XW	reg,reg	Exclusive OR Word
4	MVW	reg,reg	Move Word
5	CW	reg,reg	Compare Word
6	CMR	reg[,reg]	Complement Register
7	IR	reg,reg	Interchange Registers
8	AW	reg,reg	Add Word
9	AWCY	reg,reg	Add Word With Carry
A	SW	reg,reg	Subtract Word
B	SWCY	reg,reg	Subtract Word With Carry
C	ACY	reg	Add Carry Register
D	VR	reg[,reg]	Invert Register
E	CPLSR	reg	Copy Level Status Register
F	SEIND	reg	Set Indicators

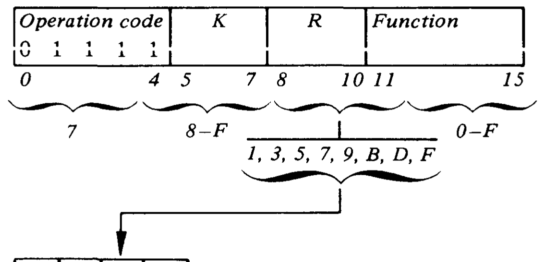


0	SLC	reg,reg	Shift Left Circular
1	SLL	reg,reg	Shift Left Logical
2	SRL	reg,reg	Shift Right Logical
3	SRA	reg,reg	Shift Right Arithmetic
4	SLCD	reg,reg	Shift Left Circular Double
5	SLLD	reg,reg	Shift Left Logical Double
6	SRLD	reg,reg	Shift Right Logical Double
7	SRAD	reg,reg	Shift Right Arithmetic Double
8	(invalid)		
9	SLT	reg,reg	Shift Left and Test
A	(invalid)		
B	(invalid)		
C	(invalid)		
D	SLTD	reg,reg	Shift Left and Test Double
E	(invalid)		
F	(invalid)		



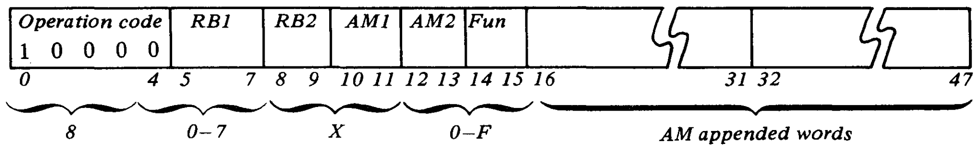
0	NWI	word,reg[,reg]	And Word Immediate
1	AWI	word,reg[,reg]	Add Word Immediate
2	SWI	word,reg[,reg]	Subtract Word Immediate
3	OWI	word,reg[,reg]	OR Word Immediate
4	RBTWI	word,reg[,reg]	Reset Bits Word Immediate
5	XWI	word,reg[,reg]	Exclusive OR Word Immediate
6	CWI	word,reg	Compare Word Immediate
7	TWI	word,reg	Test Word Under Mask Immediate
8	(invalid)		
9	(invalid)		
A	(invalid)		
B	(invalid)		
C	(invalid)		
D	(invalid)		
E	(invalid)		
F	(invalid)		



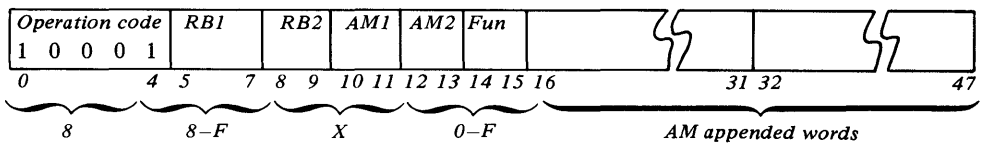


7 8–F	0	SECON	reg	Set Console Data Lights
	1	(invalid)		
	2	(Note 12)		
	3	(invalid)		
	4	(invalid)		
	5	(invalid)		
	6	(invalid)		
	7	(invalid)		
	8	CPCON	reg	Copy Console Data Buffer
	9	CPCL	reg	Copy Current Level
	A	(Note 12)		
	B	(invalid)		
	C	(invalid)		
	D	(invalid)		
	E	(invalid)		
	F	(invalid)		

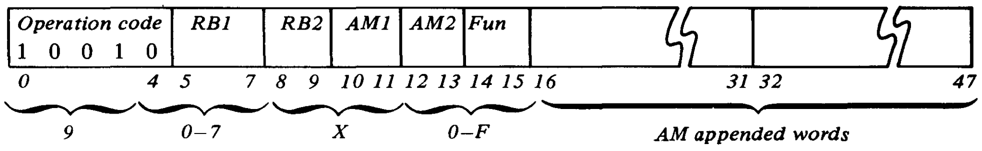
Note 12. Supervisor state: No-op
Problem state: program check, privilege violate



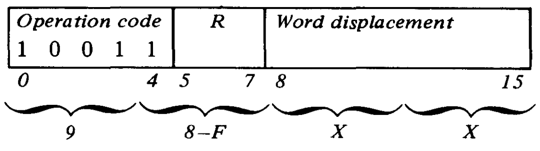
8 0–7 X	0,4 8,C	MVB	addr5,addr4	Move Byte
	1,5 9,D	OB	addr5,addr4	OR Byte
	2,6 A,E	RBTB	addr5,addr4	Reset Bits Byte
	3,7 B,F	CB	addr5,addr4	Compare Byte



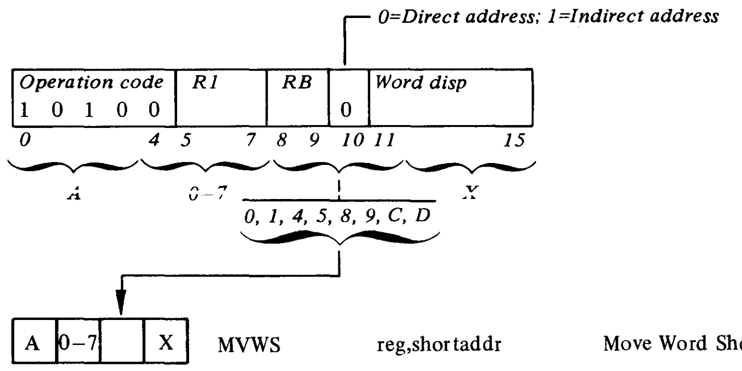
8	8–F	X	0,4 8,C	MVW	addr5,addr4	Move Word
			1,5 9,D	OW	addr5,addr4	OR Word
			2,6 A,E	RBTW	addr5,addr4	Reset Bits Word
			3,7 B,F	CW	addr5,addr4	Compare Word



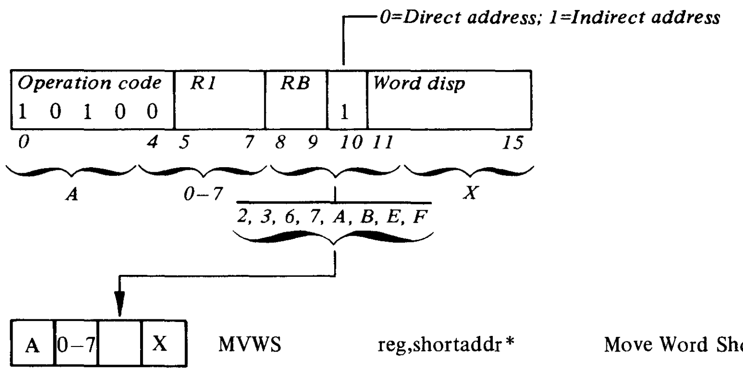
9	0–7	X	0,4 8,C	MVD	addr5,addr4	Move Double Word
			1,5 9,D	OD	addr5,addr4	OR Double Word
			2,6 A,E	RBTD	addr5,addr4	Reset Bits Double Word
			3,7 B,F	CD	addr5,addr4	Compare Double Word



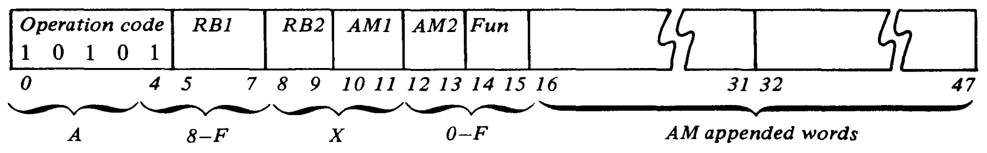
9	8–F	X	X	JAL	jdisp,reg	Jump and Link
				JAL	jaddr,reg	Jump and Link



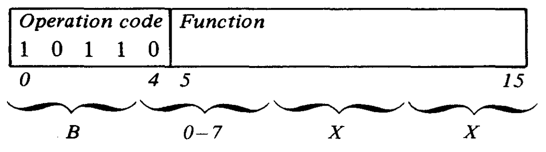
MVWS reg,shortaddr Move Word Short



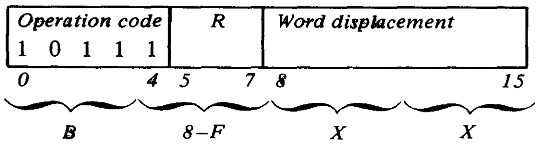
MVWS reg,shortaddr* Move Word Short



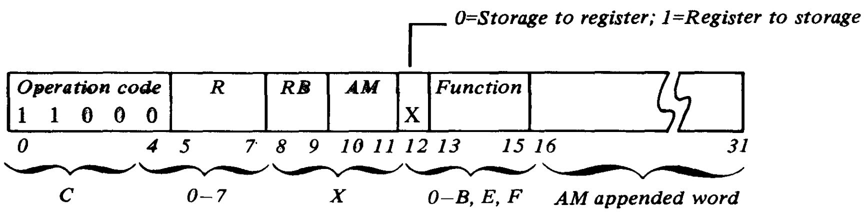
A	8–F	X	0,4 8,C	AW	addr5,addr4	Add Word
			1,5 9,D	SW	addr5,addr4	Subtract Word
			2,6 A,E	AD	addr5,addr4	Add Double Word
			3,7 B,F	SD	addr5,addr4	Subtract Double Word



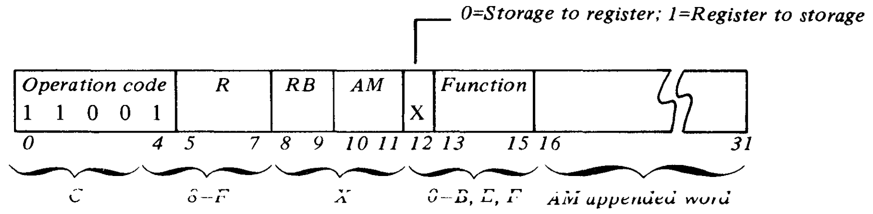
B	0–7	X	X	Unsupported operation code (Soft exception trap condition)



B 8–F X X	JCT	jdisp,reg	Jump on Count
	JCT	jaddr,reg	Jump on Count

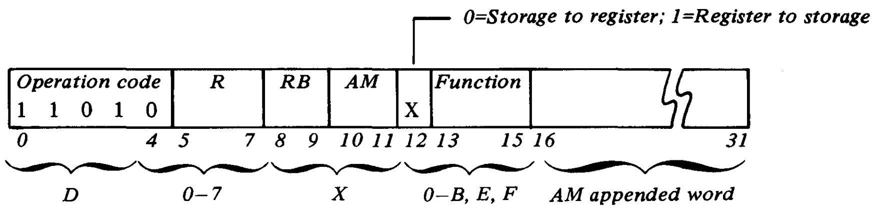


C 0–7 X	0	MVB	addr4,reg	Move Byte
	1	OB	addr4,reg	OR Byte
	2	RBTB	addr4,reg	Reset Bits Byte
	3	XB	addr4,reg	Exclusive OR Byte
	4	CB	addr4,reg	Compare Byte
	5	MVBZ	addr4,reg	Move Byte and Zero
	6	AB	addr4,reg	Add Byte
	7	SB	addr4,reg	Subtract Byte
	8	MVB	reg,addr4	Move Byte
	9	OB	reg,addr4	OR Byte
	A	RBTB	reg,addr4	Reset Bits Byte
	B	XB	reg,addr4	Exclusive OR Byte
	E	AB	reg,addr4	Add Byte
	F	SB	reg,addr4	Subtract Byte

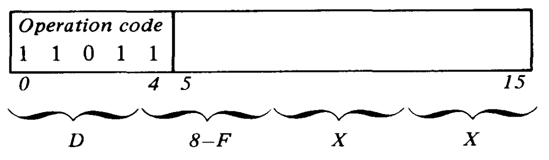


C 8–F X 0	MVW	addr4,reg	Move Word
1	OW	addr4,reg	OR Word
2	RBTW	addr4,reg	Reset Bits Word
3	XW	addr4,reg	Exclusive OR Word
4	CW	addr4,reg	Compare Word
5	MVWZ	addr4,reg	Move Word and Zero
6	AW	addr4,reg	Add Word
7	SW	addr4,reg	Subtract Word
8	MVW	reg,addr4	Move Word
9	OW	reg,addr4	OR Word
A	RBTW	reg,addr4	Reset Bits Word
B	XW	reg,addr4	Exclusive OR Word
E	AW	reg,addr4	Add Word
F	SW	reg,addr4	Subtract Word

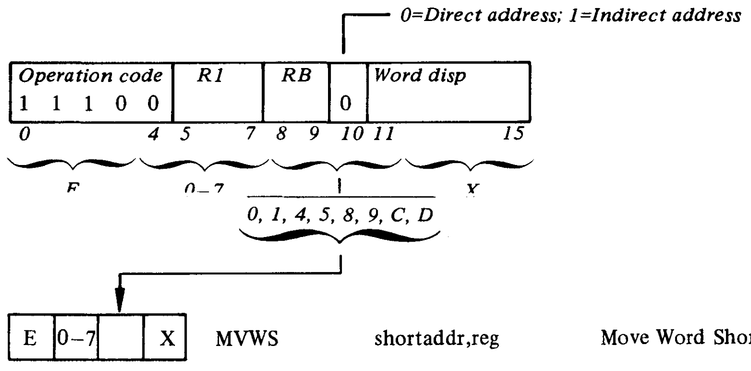
Dxxx



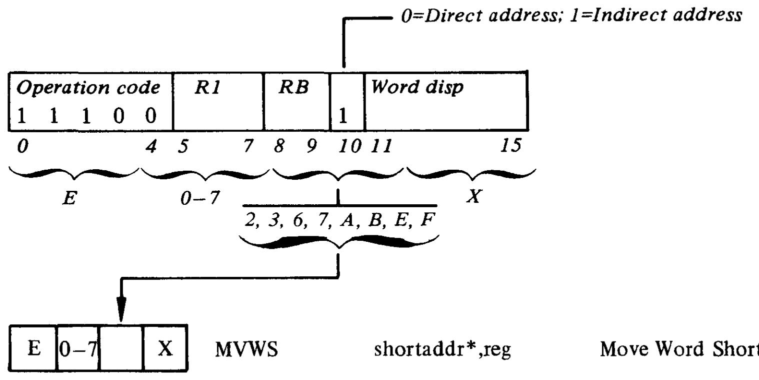
D	0–7	X	0	MVD	addr4,reg	Move Double Word
			1	OD	addr4,reg	OR Double Word
			2	RBTD	addr4,reg	Reset Bits Double Word
			3	XD	addr4,reg	Exclusive OR Double Word
			4	CD	addr4,reg	Compare Double Word
			5	MVDZ	addr4,reg	Move Double Word and Zero
			6	AD	addr4,reg	Add Double Word
			7	SD	addr4,reg	Subtract Double Word
			8	MVD	reg,addr4	Move Double Word
			9	OD	reg,addr4	OR Double Word
			A	RBTD	reg,addr4	Reset Bits Double Word
			B	XD	reg,addr4	Exclusive OR Double Word
			E	AD	reg,addr4	Add Double Word
			F	SD	reg,addr4	Subtract Double Word



D	8–F	X	X	Illegal operation code (Program check condition)

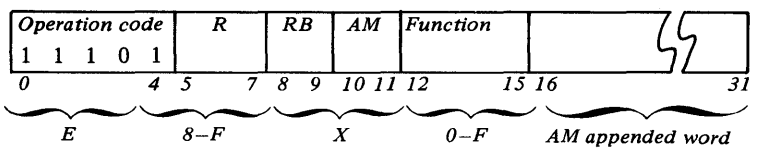


MVWS shortaddr,reg Move Word Short

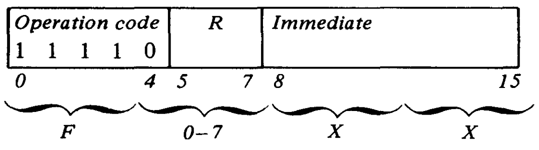


MVWS shortaddr*,reg Move Word Short

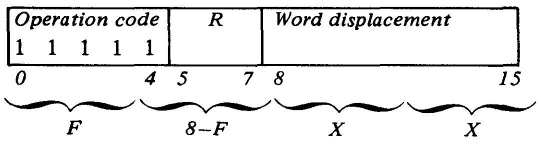
Exxx
Fxxx



E	8–F	X	0	PSB	reg,addr4	Push Byte
			1	MB	addr4,reg	Multiply Byte
			2	DB	addr4,reg	Divide Byte
			3	PB	addr4,reg	Pop Byte
			4	PSW	reg,addr4	Push Word
			5	MW	addr4,reg	Multiply Word
			6	DW	addr4,reg	Divide Word
			7	PW	addr4,reg	Pop Word
			8	PSD	reg,addr4	Push Double Word
			9	MD	addr4,reg	Multiply Double Word
			A	DD	addr4,reg	Divide Double Word
			B	PD	addr4,reg	Pop Doubleword
			C	(invalid)		
			D	(invalid)		
			E	(invalid)		
			F	(invalid)		



F	0–7	X	X	CBI	byte,reg	Compare Byte Immediate



F	8–F	X	X	BALS	(reg,jdisp)*	Branch and Link Short
				BALS	(reg)*	Branch and Link Short
				BALS	addr*	Branch and Link Short

Appendix C. Assembler Syntax

Coding Notes

1. Data flow, when it modifies a field, is always from left to right.
2. Registers used in effective address calculations are always in parentheses.
3. An address specification followed by an asterisk indicates indirect addressing. Here, the effective address is the contents of the addressed storage location.
4. The (reg)+ format indicates that, after use, the contents of reg are increased by the number of bytes addressed.
5. AM indicates address mode.

Legend for Machine Instruction Operands

abcnt	An absolute value or expression representing the size of a work storage area to be allocated by the Store Multiple (STM) instruction. The value you code must be an even number in the range 0–16382.
addr	An address value. Code an absolute or relocatable expression in the range 0–65535.
addr4	An address value that you code in one of the following forms:

(reg$^{0-3}$)	The effective address is the contents of the reg $^{0-3}$. (AM=00)
(reg$^{0-3}$) +	The effective address is the contents of the register reg$^{0-3}$. After an instruction uses it, the contents of the register are increased by the number of bytes addressed by the instruction. (AM-01)
addr	The effective address is the value of addr, unless the instruction and addr are within the range of the same USING statement. If they are, the assembler computes the effective address as a displacement (-32768 to +32767 or 0 to 65535) from the base register, which must be reg $^{1-3}$. (AM=10)
addr*	The effective address is the contents of storage at the address defined by addr, unless the instruction and addr are within the domain and range of the same USING statement. If they are, the assembler computes the effective address as the contents of storage at the address defined by a displacement (0 255) from the base register, which must be reg $^{1-3}$. (AM=11)
(reg$^{1-3}$,waddr)	The effective address is the contents of the register reg$^{1-3}$, added to the value of waddr. (AM=10)
displ (reg$^{1-3}$,disp 2)*	The effective address is calculated as follows: The contents of the register reg are added to the value of the displacement disp2 to form an address. The contents of that storage location are added to the value of displ to form the effective address. (AM=11)
disp (reg$^{1-3}$)*	The effective address is the contents of storage at the address defined by the contents of reg$^{1-3}$, added to the value of disp. (AM=11)
(reg $^{1-3}$)*	The effective address is the contents of storage at the address defined by the contents of reg$^{1-3}$. (AM=11)
(reg$^{1-3}$,disp)*	The contents reg$^{1-3}$ are added to disp, forming an address. The contents of storage at that address form the effective address. (AM=11)

For the byte addressing, the effective address can be even or odd. For word or doubleword addressing, the effective address must be even.

addr5	An address value that you code in one of the following forms:

(reg)	The effective address is the contents of the register reg. (AM=00)
(reg) +	The effective address is the contents of the register reg. After an instruction uses it, the contents of the register are increased by the number of bytes addressed by the instruction. (AM=01)
addr	The effective address is the value of addr, unless the instruction and addr are within the domain and range of the same USING statement. If they are, the assembler computes the effective address as a displacement (-32768 to +32767 or 0 to 65535) from the base register, which must be reg $^{1-7}$. (AM=10)

addr*	The effective address is the contents of storage at the address defined by addr, unless the instruction and addr are within the domain and range of the same USING statement. If they are, the assembler computes the effective address as the contents of storage at the address defined by a displacement (0–255) from the base register, which must be reg $^{1-7}$. (AM=11)
(reg$^{1-7}$,waddr)	The effective address is the contents of reg$^{1-7}$, added to the value of waddr. (AM=10)
displ(reg$^{1-7}$,disp2)*	The effective address is calculated as follows: The contents of the register reg$^{1-7}$ are added to the value of the displacement disp2 to form an address. The contents of that storage location are added to the value of displ to form the effective address. (AM=11)
disp(reg$^{1-7}$)*	The effective address is the contents of storage at the address defined by the contents of reg$^{1-7}$, added to the value of disp. (AM=11)
(reg$^{1-7}$)*	The effective address is the contents of storage at the address defined by the contents of reg$^{1-7}$. (AM=11)
(reg$^{1-7}$,disp)*	The contents of reg$^{1-7}$ are added to disp, forming an address. The contents of storage at that address form the effective address. (AM=11)

For byte addressing, the effective address can be even or odd. For word or doubleword addressing, the effective address must be even.

bitdisp	A displacement into a bit field. Code an absolute value or expression in the range 0–63.
byte	A byte value. Code an absolute value or expression in the range -128 to +127 or 0 to 255.
cnt16	A single word (one register) shift count. Code an absolute value or expression in the range 0–16.
cnt31	A doubleword (register pair) shift count. Code an absolute value or expression in the range 0–31.
cond	A condition code value. Code an absolute value or expression in the range 0–7.
disp	A byte address displacement. Code an absolute value or expression in the range 0–255.
freg	A floating-point register. Code either a predefined floating register symbol (FR0–FR3) or a symbol that is equated to the desired register number (0, 1, 2, or 3). Symbols are equated with EQUR statements, which must precede the instruction using the register symbol.
jaddr	The address of an instruction that is within -256 to +254 bytes of the byte following a jump instruction. Code a relocatable expression.
jdisp	A displacement from the byte following a jump instruction. Code an absolute value or expression in the range -256 to +254.
longaddr	An address value that you code in one of the following forms:

addr	The effective address is the value of addr, unless the instruction and addr are within the domain and range of the same USING statement. If they are, the assembler computes the effective address as a displacement (-32768 to +32767 or 0 to 65535) from the base register, which must be reg$^{1-7}$
addr*	The effective address is the contents of storage at the address defined by addr, unless the instruction and addr are within the domain and range of the same USING statement. If they are, the assembler computes the effective address as the contents of storage at the address defined by a displacement (-32768 to +32767 or 0 to 65535) from the base register, which must be reg$^{1-7}$.
(reg$^{1-7}$,waddr)	The effective address is the contents of reg$^{1-7}$, added to the value of waddr.
(reg$^{1-7}$,waddr)*	The contents of the reg$^{1-7}$, plus waddr, form an address. The contents of storage at that location form the effective address.
(reg$^{1-7}$)	The effective address is the contents of the register reg$^{1-7}$
(reg$^{1-7}$)*	The effective address is the contents of storage at the address defined by the contents of reg$^{1-7}$.

raddr	An address value. Code a relocatable expression in the range 0–65535.
reg	A general-purpose register. Code either a predefined register symbol (R0–R7) or a symbol that is equated to the desired register number (0, 1, 2, 3, 4, 5, 6, or 7). Symbols are equated with EQUR statements, which must precede the instruction using the register symbol.
reg$^{0-3}$	A general-purpose register. Code either a predefined register symbol (R0–R3) or a symbol that is equated to the desired register number (0, 1, 2, or 3). Symbols are equated with EQUR statements, which must precede the instruction using the register symbol.

reg^{1-3}	A general-purpose register. Code either a predefined register symbol (R1–R3) or a symbol that is equated to the desired register number (1, 2, or 3). Symbols are equated with EQUR statements, which must precede the instruction using the register symbol.
reg^{1-7}	A general-purpose register. Code either a predefined register symbol (R1–R7) or a symbol that is equated to the desired register number (1, 2, 3, 4, 5, 6, or 7). Symbols are equated with EQUR statements, which must precede the instructions using the register symbol.
shortaddr	An address value that you code in one of the following forms:

(reg^{0-3},wdisp)	The effective address is the value of wdisp added to the contents of reg^{0-3}.
(reg^{0-3},wdisp)*	The effective address is the contents of storage at the address defined by the value of wdisp added to the contents of reg^{0-3}.
(reg^{0-3})	The effective address is the contents of (reg^{0-3}).
(reg^{0-3})*	The effective address is the contents of storage at the address defined by the contents of reg^{0-3}.
addr	To use this form, the instruction and addr must be in the domain and range of the same USING statement. The assembler computes a displacement (0–62) and register combination that refers to the requested location.
addr*	Same as addr, except the assembler computes the effective address as the contents of storage at the address defined by a displacement (0–62) and register combination.

Note: For addr and addr*, the base register must be reg^{0-3}.

ubyte	An unsigned byte value or mask. Code an absolute value or expression in the range 0–255.
vcon	An ordinary symbol that is defined externally from the current source program.
waddr	A one-word address value. Code an absolute or relocatable expression in the range -32768 to +32767 or 0 to 65535.
wdisp	An even byte address displacement. Code an absolute value or expression in the range 0–62.
word	A word value. Code an absolute value or expression in the range -32768 to +32767 or 0 to 65535.

register symbol (R
0

Appendix D. Numbering Systems and Conversion Tables

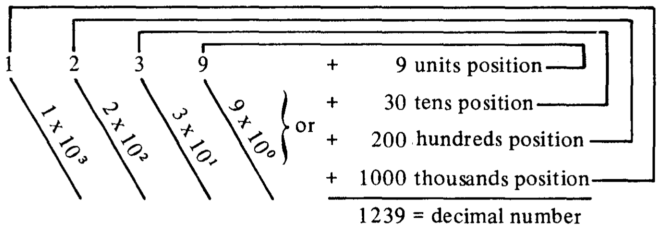
Binary and Hexadecimal Number Notations

Binary Number Notation

A binary number system, such as is used in Series/1, uses a base of two. The concept of using a base of two can be compared with the base of ten (decimal) number system.

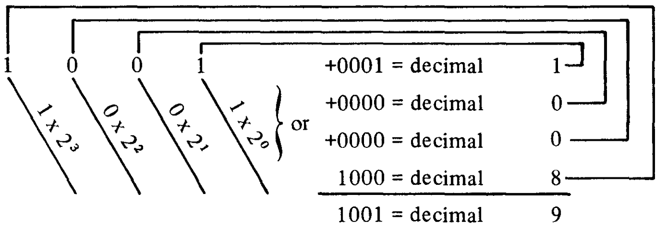
Decimal number	*Binary number*
0	= 0
1	= 1
2	= 10
3	= 11
4	= 100
5	= 101
6	= 110
7	= 111
8	= 1000
9	= 1001

Example of a decimal number:



As shown above, the decimal number system allows counting to ten in each position from units to tens to hundreds to thousands, etc. The binary system allows counting to two in each position. Register displays in the Series/1 are in binary form: a bit light on is a 1; a bit light off is a 0.

Example of a binary number:



Hexadecimal Number System

It has been noted that binary numbers require about three times as many positions as decimal numbers to express the equivalent number. This is not much of a problem to the computer; however, in talking and writing or in communicating with the computer, these binary numbers are bulky. A long string of 1's and 0's cannot be effectively transmitted from one individual to another. Some shorthand method is necessary.

The hexadecimal number system fills this need. Because of the simple relationship of hexadecimal to binary, numbers can be converted from one system to another by inspection. The base or radix of the hexadecimal system is 16. This means there are 16 symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. The letters A, B, C, D, E, and F represent the 10-base system values of 10, 11, 12, 13, 14, and 15, respectively.

Four binary positions are equivalent to one hexadecimal position. The following table shows the comparable values of the three number systems.

Decimal	*Binary*	*Hexadecimal*
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

At this point, all 16 symbols have been used, and a carry to the next higher position of the number is necessary. For example:

Decimal	*Binary*	*Hexadecimal*
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
19	0001 0011	13
20	0001 0100	14
21	0001 0101	15

–and so on–

Remember that as far as the internal circuitry of the computer is concerned, it understands only binary. But an operator can look at a series of lights on the computer console showing binary 1's and 0's, for example: 0001 1110 0001 0011, and say that the lights represent the hexadecimal value 1E13, which is easier to state than the string of 1's and 0's.

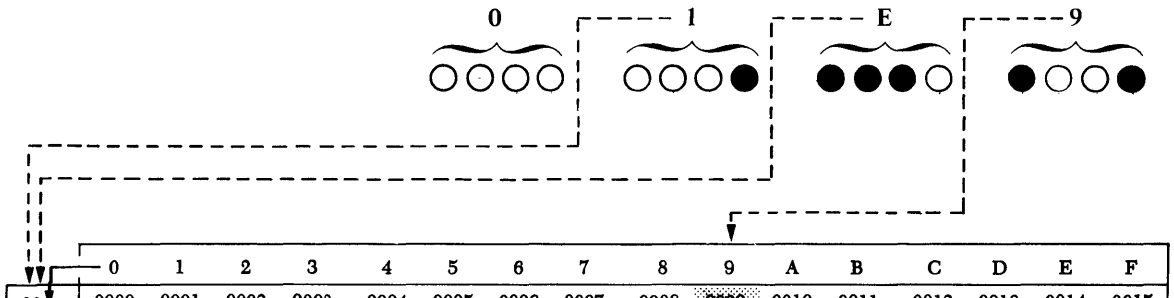
Hexadecimal–Decimal Conversion Tables

The table in this appendix provides for direct conversion of decimal and hexadecimal number in these ranges:

Hexadecimal	*Decimal*
000 to FFF	0000 to 4095

For numbers outside the range of the table, add the following values to the tables figures:

Hexadecimal	*Decimal*
1000	4096
2000	8192
3000	12288
4000	16384
5000	20480
6000	24576
7000	28672
8000	32768



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00_	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
01_	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
02_	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
03_	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
04_	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
05_	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
06_	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
07_	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
08_	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
09_	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A_	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B_	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C_	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D_	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E_	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F_	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255
10_	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
11_	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
12_	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
13_	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
14_	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
15_	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
16_	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
17_	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
18_	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
19_	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A_	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B_	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C_	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D_	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E_	0480	0481	0482	0483	0484	0485	0486	0487	0488	$\boxed{0489}$	0490	0491	0492	0493	0494	0495
1F_	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20_	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
21_	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
22_	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
23_	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
24_	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
25_	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
26_	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
27_	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
28_	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
29_	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A_	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B_	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C_	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D_	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E_	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F_	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
30_	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
31_	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
32_	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
33_	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
34_	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
35_	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
36_	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
37_	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
38_	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
39_	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A_	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B_	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C_	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D_	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E_	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F_	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
40_	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
41_	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
42_	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
43_	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
44_	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
45_	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
46_	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
47_	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
48_	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
49_	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A_	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B_	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C_	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D_	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E_	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F_	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
50_	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
51_	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
52_	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
53_	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
54_	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
55_	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
56_	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
57_	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
58_	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
59_	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A_	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B_	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C_	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D_	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E_	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F_	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
60_	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
61_	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
62_	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
63_	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
64_	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
65_	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
66_	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
67_	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
68_	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
69_	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A_	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B_	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C_	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D_	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E_	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F_	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791
70_	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
71_	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
72_	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
73_	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
74_	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
75_	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
76_	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
77_	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
78_	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
79_	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A_	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B_	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C_	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D_	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E_	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F_	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
80_	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
81_	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
82_	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
83_	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
84_	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
85_	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
86_	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
87_	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
88_	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
89_	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A_	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B_	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C_	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D_	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E_	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F_	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
90_	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
91_	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
92_	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
93_	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
94_	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
95_	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
96_	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
97_	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
98_	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
99_	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A_	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B_	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C_	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D_	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E_	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F_	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A0_	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A1_	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A2_	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A3_	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A4_	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A5_	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A6_	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A7_	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A8_	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A9_	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA_	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB_	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC_	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD_	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE_	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF_	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B0_	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B1_	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B2_	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B3_	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B4_	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B5_	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B6_	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B7_	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B8_	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B9_	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA_	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB_	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC_	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD_	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE_	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF_	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C0_	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C1_	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C2_	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C3_	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C4_	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C5_	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C6_	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C7_	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C8_	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C9_	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA_	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB_	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC_	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD_	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE_	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF_	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
D0_	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D1_	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D2_	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D3_	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D4_	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D5_	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D6_	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D7_	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D8_	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D9_	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA_	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB_	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC_	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD_	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE_	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF_	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E0_	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E1_	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E2_	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E3_	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E4_	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E5_	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E6_	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E7_	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E8_	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E9_	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA_	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB_	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC_	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED_	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE_	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF_	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F0_	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F1_	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F2_	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F3_	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F4_	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F5_	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F6_	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F7_	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F8_	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F9_	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA_	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB_	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC_	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD_	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE_	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF_	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

Powers of Two Table

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.0625
32	5	0.03125
64	6	0.01562 5
128	7	0.00781 25
256	8	0.00390 625
512	9	0.00195 3125
1,024	10	0.00097 65625
2,048	11	0.00048 82812 5
4,096	12	0.00024 41406 25
8,192	13	0.00012 20703 125
16,384	14	0.00006 10351 5625
32,768	15	0.00003 05175 78125
65,536	16	0.00001 52587 89062 5
131,072	17	0.00000 76293 94531 25
262,144	18	0.00000 38146 97265 625
524,288	19	0.00000 19073 48632 8125
1,048,576	20	0.00000 09536 74316 40625
2,097,152	21	0.00000 04768 37158 20312 5
4,194,304	22	0.00000 02384 18579 10156 25
8,388,608	23	0.00000 01192 09289 55078 125
16,777,216	24	0.00000 00596 04644 77539 0625
33,554,432	25	0.00000 00298 02322 38769 53125
67,108,864	26	0.00000 00149 01161 19384 76562 5
134,217,728	27	0.00000 00074 50580 59692 38281 25
268,435,456	28	0.00000 00037 25290 29846 19140 625
536,870,912	29	0.00000 00018 62645 14923 09570 3125
1,073,741,824	30	0.00000 00009 31322 57461 54785 15625
2,147,483,648	31	0.00000 00004 65661 28730 77392 57812 5
4,294,967,296	32	0.00000 00002 32830 64365 38696 28906 25
8,589,934,592	33	0.00000 00001 16415 32182 69348 14453 125
17,179,869,184	34	0.00000 00000 58207 66091 34674 07226 5625
34,359,738,368	35	0.00000 00000 29103 83045 67337 03613 28125
68,719,476,736	36	0.00000 00000 14551 91522 83668 51806 64062 5
137,438,953,472	37	0.00000 00000 07275 95761 41834 25903 32031 25
274,877,906,944	38	0.00000 00000 03637 97880 70917 12951 66015 625
549,755,813,888	39	0.00000 00000 01818 98940 35458 56475 83007 8125
1,099,511,627,776	40	0.00000 00000 00909 49470 17729 28237 91503 90625
2,199,023,255,552	41	0.00000 00000 00454 74735 08864 64118 95751 95312 5
4,398,046,511,104	42	0.00000 00000 00227 37367 54432 32059 47875 97656 25
8,796,093,022,208	43	0.00000 00000 00113 68683 77216 16029 73937 98828 125
17,592,186,044,416	44	0.00000 00000 00056 84341 88608 08014 86968 99414 0625
35,184,372,088,832	45	0.00000 00000 00028 42170 94304 04007 43484 49707 03125
70,368,744,177,664	46	0.00000 00000 00014 21085 47152 02003 71742 24853 51562 5
140,737,488,355,328	47	0.00000 00000 00007 10542 73576 01001 85871 12426 75781 25
281,474,976,710,656	48	0.00000 00000 00003 55271 36788 00500 92935 56213 37890 625
562,949,953,421,312	49	0.00000 00000 00001 77635 68394 00250 46467 78106 68945 3125
1,125,899,906,842,624	50	0.00000 00000 00000 88817 84197 00125 23233 89053 34472 65625
2,251,799,813,685,248	51	0.00000 00000 00000 44408 92098 50062 61616 94526 67236 32812 5
4,503,599,627,370,496	52	0.00000 00000 00000 22204 46049 25031 30808 47263 33618 16406 25
9,007,199,254,740,992	53	0.00000 00000 00000 11102 23024 62515 65404 23631 66809 08203 125
18,014,398,509,481,984	54	0.00000 00000 00000 05551 11512 31257 82702 11815 83404 54101 5625
36,028,797,018,963,968	55	0.00000 00000 00000 02775 55756 15628 91351 05907 91702 27050 78125
72,057,594,037,927,936	56	0.00000 00000 00000 01387 77878 07814 45675 52953 95851 13525 39062 5
144,115,188,075,855,872	57	0.00000 00000 00000 00693 88939 03907 22837 76476 97925 56762 69531 25
288,230,376,151,711,744	58	0.00000 00000 00000 00346 94469 51953 61418 88238 48962 78381 34765 625
576,460,752,303,423,488	59	0.00000 00000 00000 00173 47234 75976 80709 44119 24481 39190 67382 8125
1,152,921,504,606,846,976	60	0.00000 00000 00000 00086 73617 37988 40354 72059 62240 69595 33691 40625
2,305,843,009,213,693,952	61	0.00000 00000 00000 00043 36808 68994 20177 36029 81120 34797 66845 70312 5
4,611,686,018,427,387,904	62	0.00000 00000 00000 00021 68404 34497 10088 68014 90560 17398 83422 85156 25
9,223,372,036,854,775,808	63	0.00000 00000 00000 00010 84202 17248 55044 34007 45280 08699 41711 42578 125
18,446,744,073,709,551,616	64	0.00000 00000 00000 00005 42101 08624 27522 17003 72640 04349 70855 71289 0625

2^n	n
18,446,744,073,709,551,616	64
36,893,488,147,419,103,232	65
73,786,976,294,838,206,464	66
147,573,952,589,676,412,928	67
295,147,905,179,352,825,856	68
590,295,810,358,705,651,712	69
1,180,591,620,717,411,303,424	70
2,361,183,241,434,822,606,848	71
4,722,366,482,869,645,213,696	72
9,444,732,965,739,290,427,392	73
18,889,465,931,478,580,854,784	74
37,778,931,862,957,161,709,568	75
75,557,863,725,914,323,419,136	76
151,115,727,451,828,646,838,272	77
302,231,454,903,657,293,676,544	78
604,462,909,807,314,587,353,088	79
1,208,925,819,614,629,174,706,176	80
2,417,851,639,229,258,349,412,352	81
4,835,703,278,458,516,698,824,704	82
9,671,406,556,917,033,397,649,408	83
19,342,813,113,834,066,795,298,816	84
38,685,626,227,668,133,590,597,632	85
77,371,252,455,336,267,181,195,264	86
154,742,504,910,672,534,362,390,528	87
309,485,009,821,345,068,724,781,056	88
618,970,019,642,690,137,449,562,112	89
1,237,940,039,285,380,274,899,124,224	90
2,475,880,078,570,760,549,798,248,448	91
4,951,760,157,141,521,099,596,496,896	92
9,903,520,314,283,042,199,192,993,792	93
19,807,040,628,566,084,398,385,987,584	94
39,614,081,257,132,168,796,771,975,168	95
79,228,162,514,264,337,593,543,950,336	96
158,456,325,028,528,675,187,087,900,672	97
316,912,650,057,057,350,374,175,801,344	98
633,825,300,114,114,700,748,351,602,688	99
1,267,650,600,228,229,401,496,703,205,376	100
2,535,301,200,456,458,802,993,406,410,752	101
5,070,602,400,912,917,605,986,812,821,504	102
10,141,204,801,825,835,211,973,625,643,008	103
20,282,409,603,651,670,423,947,251,286,016	104
40,564,819,207,303,340,847,894,502,572,032	105
81,129,638,414,606,681,695,789,005,144,064	106
162,259,276,829,213,363,391,578,010,288,128	107
324,518,553,658,426,726,783,156,020,576,256	108
649,037,107,316,853,453,566,312,041,152,512	109
1,298,074,214,633,706,907,132,624,082,305,024	110
2,596,148,429,267,413,814,265,248,164,610,048	111
5,192,296,858,534,827,628,530,496,329,220,096	112
10,384,593,717,069,655,257,060,992,658,440,192	113
20,769,187,434,139,310,514,121,985,316,880,384	114
41,538,374,868,278,621,028,243,970,633,760,768	115
83,076,749,736,557,242,056,487,941,267,521,536	116
166,153,499,473,114,484,112,975,882,535,043,072	117
332,306,998,946,228,968,225,951,765,070,086,144	118
664,613,997,892,457,936,451,903,530,140,172,288	119
1,329,227,995,784,915,872,903,807,060,280,344,576	120
2,658,455,991,569,831,745,807,614,120,560,689,152	121
5,316,911,983,139,663,491,615,228,241,121,378,304	122
10,633,823,966,279,326,983,230,456,482,242,756,608	123
21,267,647,932,558,653,966,460,912,964,485,513,216	124
42,535,295,865,117,307,932,921,825,928,971,026,432	125
85,070,591,730,234,615,865,843,651,857,942,052,864	126
170,141,183,460,469,231,731,687,303,715,884,105,728	127
340,282,366,920,938,463,463,374,607,431,768,211,456	128

Appendix E. Character Codes

Decimal	*Hex*	*Binary*	*EBCDIC*	*ASCII*	*Eight bit data inter-change*	*PTTC/EBCD*	*PTTC/ Correspondence*
0	00	0000 0000	NUL	NUL	NUL		
1	01	0001	SOH	SOH	NUL	space	space
2	02	0010	STX	STX		1	1,]
3	03	0011	ETX	ETX	@		
4	04	0100	PF	EOT		2	2
5	05	0101	HT	ENQ	space		
6	06	0110	LC	ACK			
7	07	0111	DEL	BEL		3	3
8	08	1000		BS		4	5
9	09	1001	RLF	HT			
10	0A	1010	SMM	LF	P (even parity)		
11	0B	1011	VT	VT	P (odd parity)	5	7
12	0C	1100	FF	FF	0 (even parity)		
13	0D	1101	CR	CR	0 (odd parity)	6	6
14	0E	1110	SO	SO		7	8
15	0F	1111	SI	SI			
16	10	0001 0000	DLE	DLE		8	4
17	11	0001	DC1	DC1			
18	12	0010	DC2	DC2	H (even parity)		
19	13	0011	TM	DC3	H (odd parity)	9	0
20	14	0100	RES	DC4	((even parity)		
21	15	0101	NL	NAK	((odd parity)	0	z
22	16	0110	BS	SYN		Ⓓ (EOA)	Ⓓ (EOA),9
23	17	0111	IL	ETB			
24	18	1000	CAN	CAN			
25	19	1001	EM	EM			
26	1A	1010	CC	SUB			
27	1B	1011	CU1	ESC	X		
28	1C	1100	IFS	FS		upper case	upper case
29	1D	1101	IGS	GS	8		⩞
30	1E	1110	IRS	RS			
31	1F	1111	IUS	US		Ⓒ (EOT)	Ⓒ (EOT)
32	20	0010 0000	DS	space		@	t
33	21	0001	SOS	!	EOT		
34	22	0010	FS	"	D (even parity)		
35	-23	0011		#	D (odd parity)	/	x
36	24	0100	BYP	$	S (even parity)		
37	25	0101	LF	%	S (odd parity)	s	n
38	26	0110	ETB	&		t	u
39	27	0111	ESC	'			
40	28	1000		(			
41	29	1001		)		u	e
42	2A	1010	SM	*		v	d
43	2B	1011	CU2	+	T		
44	2C	1100		,		w	k
45	2D	1101	ENQ	-	4		
46	2E	1110	ACK	.			
47	2F	1111	BFL	/		x	c
48	30	0011 0000		0	form feed		
49	31	0001		1	form feed	y	l
50	32	0010	SYN	2		z	h
51	33	0011		3	L		
52	34	0100	PN	4			
53	35	0101	RS	5	,		
54	36	0110	UC	6			

Decimal	*Hex*	*Binary*	*EBCDIC*	*ASCII*	*Eight bit data interchange*	*PTTC/EBCD*	*PTTC/ Correspondence*
55	37	0011 0111	EOT	7		Ⓢ (SOA), comma	b
56	38	1000		8			
57	39	1001		9			
58	3A	1010		:	\ (even parity)		
59	3B	1011	CU3	;	\ (odd parity)	index	index
60	3C	1100	DC4	<	< (even parity)		
61	3D	1101	NAK	=	< (odd parity)	Ⓑ (EOB)	
62	3E	1110		>			
63	3F	1111	SUB	?			
64	40	0100 0000	space	@		N ,-	!
65	41	0001		A	EOA		
66	42	0010		B	B (even parity)		
67	43	0011		C	B (odd parity)	i	m
68	44	0100		D	" (even parity)		
69	45	0101		E	" (odd parity)	k	
70	46	0110		F		l	v
71	47	0111		G			
72	48	1000		H			
73	49	1001		I		m	'
74	4A	1010	¢	J		n	r
75	4B	1011	.	K	R		
76	4C	1100	<	L		o	i
77	4D	1101	(	M	2		
78	4E	1110	+	N			
79	4F	1111	]	O		p	a
80	50	0101 0000	&	P	line feed		
81	51	0001		Q	line feed	q	o
82	52	0010		R		r	s
83	53	0011		S	J		
84	54	0100		T			
85	55	0101		U	*		
86	56	0110		V			
87	57	0111		W		$	w
88	58	1000		X			
89	59	1001		Y			
90	5A	1010	!	Z	Z (even parity)		
91	5B	1011	$	[	Z (odd parity)	CRLF	CRLF
92	5C	1100	*	\	: (even parity)		
93	5D	1101	)	]	: (odd parity)	backspace	backspace
94	5E	1110	;	^		idle	idle
95	5F	1111	¬	–			
96	60	0110 0000	-		ACK		
97	61	0001	/	a		&	j
98	62	0010		b		a	g
99	63	0011		c	F		
100	64	0100		d		b	
101	65	0101		e	&		
102	66	0110		f			
103	67	0111		g		c	f
104	68	1000		h		d	p
105	69	1001		i			
106	6A	1010	¦	j	V (even parity)		
107	6B	1011	,	k	V (odd parity)	e	
108	6C	1100	%	l	6 (even parity)		
109	6D	1101	–	m	6 (odd parity)	f	q
110	6E	1110	>	n		g	comma
111	6F	1111	?	o			
112	70	0111 0000		p		h	/
113	71	0001		q	shift out		
114	72	0010		r	N (even parity)		
115	73	0011		s	N (odd parity)	i	y
116	74	0100		t	. (even parity)		

Decimal	*Hex*	*Binary*	*EBCDIC*	*ASCII*	*Eight bit data inter-change*	*PTTC/EBCD*	*PTTC/ Correspondence*
117	75	0111 0101		u	. (odd parity)		
118	76	0110		v		Ⓨ ,period	-
119	77	0111		w			
120	78	1000		x			
121	79	1001		y			
122	7A	1010	:	z		horiz tab	tab
123	7B	1011	#	{	↑		
124	7C	1100	@	|		lower case	lower case
125	7D	1101	’	}	>		
126	7E	1110	=	~			
127	7F	1111	”	DEL		delete	
128	80	1000 0000					
129	81	0001	a		SOM	space	space
130	82	0010	b		A (even parity)	=	±, [
131	83	0011	c		A (odd parity)		
132	84	0100	d		! (even parity)	<	@
133	85	0101	e		! (odd parity)		
134	86	0110	f				
135	87	0111	g			;	#
136	88	1000	h		X-ON	:	%
137	89	1001	i				
138	8A	1010					
139	8B	1011			Q	%	&
140	8C	1100					
141	8D	1101			1	’	¢
142	8E	1110				>	*
143	8F	1111					
144	90	1001 0000			horiz tab	*	$
145	91	0001	j		horiz tab		
146	92	0010	k				
147	93	0011	l		I	(	)
148	94	0100	m				
149	95	0101	n		)	)	Z
150	96	0110	o			Ⓓ (EOA),”	(
151	97	0111	p				
152	98	1000	q				
153	99	1001	r				
154	9A	1010			Y (even parity)		
155	9B	1011			Y (odd parity)		
156	9C	1100			9 (even parity)	upper case	upper case
157	9D	1101			9 (odd parity)		
158	9E	1110					
159	9F	1111				Ⓒ (EOT)	Ⓒ (EOT)
160	A0	1010 0000			WRU (even)	¢	T
161	A1	0001	~		WRU (odd)		
162	A2	0010	s				
163	A3	0011	t		E	?	X
164	A4	0100	u				
165	A5	0101	v		%	S	N
166	A6	0110	w			T	U
167	A7	0111	x				
168	A8	1000	y				
169	A9	1001	z			U	E
170	AA	1010			U (even parity)	V	D
171	AB	1011			U (odd parity)		
172	AC	1100			5 (even parity)	W	K
173	AD	1101			5 (odd parity)		
174	AE	1110					
175	AF	1111				X	C
176	B0	1011 0000					
177	B1	0001			return	Y	L
178	B2	0010			M (even parity)	Z	H

Decimal	*Hex*	*Binary*	*EBCDIC*	*ASCII*	*Eight bit data interchange*	*PTTC/EBCD*	*PTTC/ Correspondence*
179	B3	1011 0011			M (odd parity)		
180	B4	0100			- (even parity)		
181	B5	0101			- (odd parity)		
182	B6	0110					
183	B7	0111				Ⓢ (SOA), |	B
184	B8	1000					
185	B9	1001					
186	BA	1010					
187	BB	1011			]	index	index
188	BC	1100					
189	BD	1101			=	Ⓑ (EOB)	
190	BE	1110					
191	BF	1111					
192	C0	1100 0000	{		EOM (even)	Ⓝ ,–	
193	C1	0001	A		EOM (odd)		
194	C2	0010	B				
195	C3	0011	C		C	J	M
196	C4	0100	D				
197	C5	0101	E		#	K	
198	C6	0110	F			L	V
199	C7	0111	G				
200	C8	1000	H				
201	C9	1001	I		X-OFF	M	"
202	CA	1010			S (even parity)	N	R
203	CB	1011			S (odd parity)		
204	CC	1100	⑀		3 (even parity)	O	I
205	CD	1101			3 (odd parity)		
206	CE	1110	⑂				
207	CF	1111				P	A
208	D0	1101 0000	}				
209	D1	0001	J		vertical tab	Q	O
210	D2	0010	K		K (even parity)	R	S
211	D3	0011	L		K (odd parity)		
212	D4	0100	M		+ (even parity)		
213	D5	0101	N		+ (odd parity)		
214	D6	0110	O				
215	D7	0111	P			!	W
216	D8	1000	Q				
217	D9	1001	R				
218	DA	1010					
219	DB	1011			[	CRLF	CRLF
220	DC	1100					
221	DD	1101			;	backspace	backspace
222	DE	1110				idle	idle
223	DF	1111			PAD		
224	E0	1110 0000	\				
225	E1	0001			bell	+	J
226	E2	0010	S		G (even parity)	A	G
227	E3	0011	T		G (odd parity)		
228	E4	0100	U		, (even parity)	B	+
229	E5	0101	V		, (odd parity)		
230	E6	0110	W				
231	E7	0111	X			C	F
232	E8	1000	Y			D	P
233	E9	1001	Z				
234	EA	1010					
235	EB	1011			W	E	
236	EC	1100	⑁				
237	ED	1101			7	F	Q
238	EE	1110				G	comma
239	EF	1111					

Decimal	*Hex*	*Binary*	*EBCDIC*	*ASCII*	*Eight bit data inter- change*	*PTTC/EBCD*	*PTTC/ Correspondence*
240	F0	1111 0000	0		shift in (even)	H	?
241	F1	0001	1		shift in (odd)		
242	F2	0010	2				
243	F3	0011	3		U	I	Y
244	F4	0100	4				
245	F5	0101	5		/		
246	F6	0110	6			Ⓨ, ¬	—
247	F7	0111	7				
248	F8	1000	8				
249	F9	1001	9				
250	FA	1010	LVM		⇐ (even parity)	horiz tab	tab
251	FB	1011			⇐ (odd parity)		
252	FC	1100			? (even parity)	lower case	lower case
253	FD	1101			? (odd parity)		
254	FE	1110					
255	FF	1111			$\frac{\text{delete}}{\text{rub out}}$	delete	

Appendix F. Carry and Overflow Indicators

This appendix explains the meaning of the carry and overflow indicators for signed and unsigned numbers. Examples for setting these indicators are also provided.

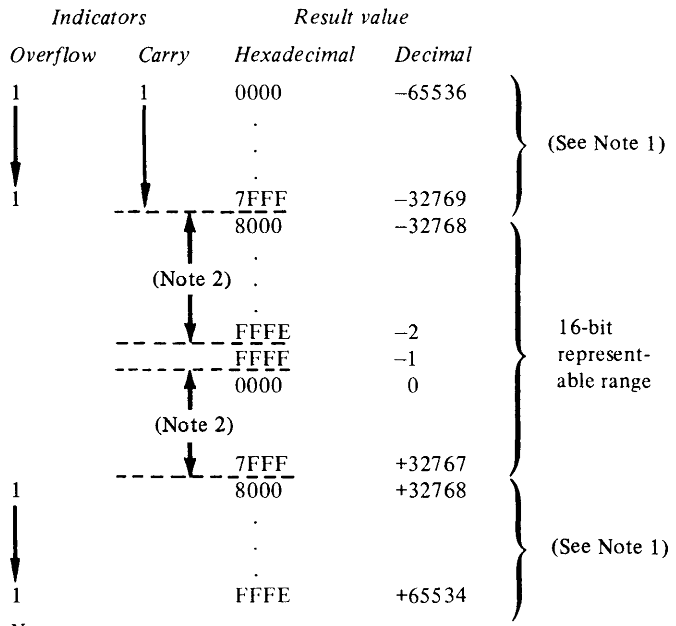
Signed Numbers

For signed addition and subtraction, the overflow indicator signals a result that exceeds the representation capability of the system for the result operand size. When overflow is indicated, the carry indicator and the resulting operand together form a valid result with the carry indicator being the most significant bit. For addition, the carry indicator is the sign (high-order bit) of this result. For subtraction, the carry indicator is the complement of the sign (high-order bit) of the result. A negative result appears in two's complement form. When no overflow is indicated, the carry indicator provides no information about the result.

Figure F-1 shows how the carry and overflow indicators are set for an add operation when using 16-bit operands. Figure F-2 provides the same information for a subtract operation.

SIGNED NUMBERS

ADD OPERATION–All possible results (16-bit example)



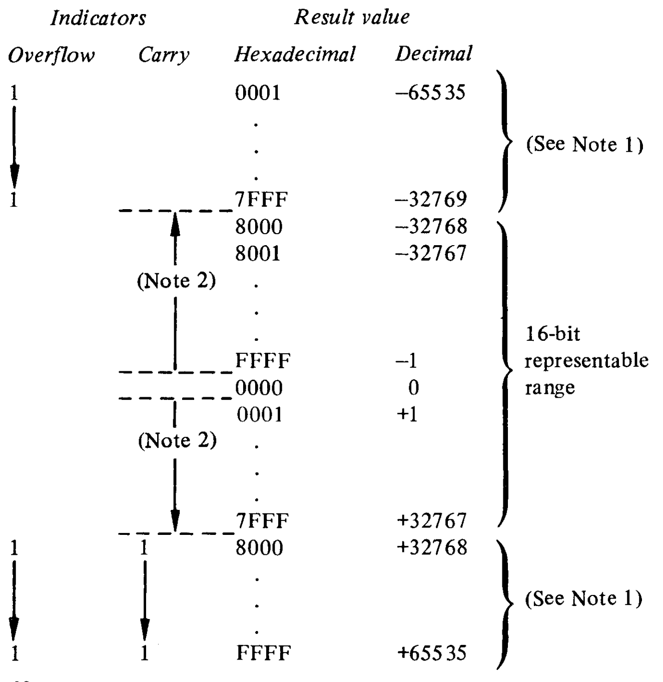
Notes.

1. When overflow occurs, the carry indicator and the result together form a valid 17-bit signed number, of which the carry is the sign, and the result is the magnitude. A negative result is in two's complement from. When no overflow occurs, no useful information is provided by the carry indicator.
2. The carry indicator may be on or off depending on the operands.

Figure F-1. All possible results of an add operation regarding the operands as signed 16-bit numbers

SIGNED NUMBERS

SUBTRACT OPERATION–All possible results (16-bit example)



Indicators		*Result value*		
Overflow	*Carry*	*Hexadecimal*	*Decimal*	
1		0001	–65535	(See Note 1)
		.		
		.		
		.		
1		7FFF	–32769	
		8000	–32768	16-bit representable range
		8001	–32767	
	(Note 2)	.		
		.		
		.		
		FFFF	–1	
		0000	0	
		0001	+1	
	(Note 2)	.		
		.		
		.		
		7FFF	+32767	
1	1	8000	+32768	(See Note 1)
		.		
		.		
		.		
1	1	FFFF	+65535	

Notes.

1. When overflow occurs, the carry indicator and the result form a valid 17-bit signed number, of which the carry is the complement of the correct sign, and the result is the magnitude. A negative result is in two's complement form. When no overflow occurs, no useful information is provided by the carry indicator.
2. The carry indicator may be on or off depending on the operands.

Figure F-2. All possible results of a subtract operation regarding the operands as signed 16-bit numbers

Unsigned Numbers

For unsigned addition and subtraction, the carry indicator signals that:

1. On an add instruction, a carry out of the high-order bit position has occurred (result exceeds result operand size). The carry indicator and the resulting operand together form a valid result of which the carry indicator is the most significant bit.
2. On a subtract operation, a borrow beyond the high-order bit position has occurred. A borrow during a subtract operation is defined as either of the following:
   –No carry is generated out of the high-order bit position when a two's complement of the subtrahend and add is performed to accomplish the subtract operation.
   –The most significant digit of the minuend must be made larger to generate a difference of zero or one when subtracting the most significant digit of the subtrahend; for example, 1 subtracted from 0.

When a borrow is signalled on a subtract operation, the result is in two's complement form.

The overflow indicator provides no useful information about unsigned operations.

Figure F-3 shows how the carry and overflow indicators are set for an add operation when using 16-bit operands. Figure F-4 provides the same information for a subtract operation.

UNSIGNED NUMBERS

ADD OPERATION—All possible results (16-bit example)

Indicators		*Result value*		
Overflow	*Carry*	*Hexadecimal*	*Decimal*	
(Note 2)		0000	0	16-bit representable range
		.		
		.		
		7FFF	32767	
		8000	32768	
		.		
		.		
		FFFE	65534	
		FFFF	65535	
	1	0000	65536	17-bit range using carry bit (See Note 1)
	↓	.		
	↓	.		
	↓	7FFF	98303	
	↓	8000	98304	
	↓	.		
	↓	.		
	1	FFFE	131070	

Notes.

1. With the carry indicator on, the result and carry form a valid 17-bit unsigned number of which the carry is the most significant bit.
2. The overflow indicator may be set; however, it provides no useful information.

Figure F-3. All possible results of an add operation regarding the operands as unsigned 16-bit numbers

UNSIGNED NUMBERS

SUBTRACT OPERATION—All possible results (16-bit example)

Indicators		*Result value*		
Overflow	*Carry*	*Hexadecimal*	*Decimal*	
(Note 2)	1	0001	−65535	17-bit negative range (See Note 1)
	↓	.		
	↓	.		
	↓	.		
	↓	7FFF	−32769	
	↓	8000	−32768	
	↓	8001	−32767	
	↓	.		
	↓	.		
	↓	.		
	1	FFFF	−1	
		0000	0	16-bit representable range
		0001	+1	
		.		
		.		
		7FFF	+32767	
		8000	+32768	
		.		
		.		
		.		
		FFFF	+65535	

Notes.

1. With carry (borrow) on, the result and carry indicator form a valid 17-bit negative number of which the carry is the sign and result is the magnitude in normal two's complement form.
2. The overflow indicator may be set; however, it provides no useful information.

Figure F-4. All possible results of a subtract operation regarding the operands as unsigned 16-bit numbers

Carry Indicator Setting

The carry indicator is used to signal overflow of the result when operands are presented as (unsigned) numbers. (The machine does not regard the numbers as either signed or unsigned, but performs the designated operation (add or subtract) on the values presented. The programmer must interpret the condition of the result for the number representation involved.) The machine detects the carry condition during the operation in two ways:

1. Add operation – when a carry out of the high-order bit position of the result operand occurs.
2. Subtract operation – when a borrow beyond the high-order bit position of the result operand occurs.

Add Operation Examples

A four-bit operand size is used in the following examples. Note that the unsigned number range for this operand is 0 to 15. No other unsigned number values may be represented for this size operand.

- Addition (carry indicator is not set)

```
Desired operation:   6 + 9 = 15
Machine operation:   Augend   0110
                     Addend   1001
                              ----
                     Result   1111
High-order bit carry = 0
```

The result fits as an unsigned number. The carry indicator is not set (C=0).

- Addition (carry indicator is set)

```
Desired operation:   15 + 1 = 16
Machine operation:   Augend   1111
                     Addend   0001
                              ----
                     Result   0000
High-order bit carry = 1
```

The result does not fit as an unsigned number. The carry indicator is set (C=1).

- Addition (carry indicator is set)

```
Desired operation:   15 + 15 = 30
Machine operation:   Augend   1111
                     Addend   1111
                              ----
                     Result   1110
High-order bit carry = 1
```

Result does not fit as an unsigned number. The carry indicator is set (C=1).

Note. The result of adding the two largest numbers can be contained in the operand size and the carry indicator. The carry indicator represents the most significant bit.

Subtract Operation Examples

The processor performs subtraction by using the complement addition method. The second operand is complemented (two's complement) then an add operation is performed. This is actually a three-way add operation between the minuend, the subtrahend (one's complement), and a constant of one. To provide the correct carry (borrow) indication for the subtraction, the carry result of the complement add operation must be inverted to determine the carry indicator setting. The following examples use a four-bit operand with an unsigned number range of 0 to 15.

- Subtract (carry indicator is not set)

```
Desired operation:     15 - 1 = 14
Machine operation:     Minuend     1111
                       Subtrahend  1110  one's complement
                       Constant       1  for two's complement
                                   ----
                       Result      1110
High-order bit carry = 1                 invert for carry
                                         indicator
```

The result fits as an unsigned number. The carry indicator is not set (C=0).

Note. The carry indicator setting (C=0) for this subtract operation was determined by inverting the complement-add carry.

- Subtract (carry indicator is not set)

```
Desired operation:     15 - 15 = 0
Machine operation:     Minuend     1111
                       Subtrahend  0000  one's complement
                       Constant       1  for two's complement
                                   ----
                       Result      0000
High-order bit carry = 1                 invert for carry
                                         indicator
```

The result fits as an unsigned number. The carry indicator is not set (C=0).

- Subtract (carry indicator is set)

  The following two examples show the case of a negative result (subtrahend greater than minuend). This negative result cannot be represented in the operand width because all operand bits are used to represent the unsigned number. To flag this condition the carry indicator is set.

Example 1:

Desired operation:	0 - 1 = -1		
Machine operation:	Minuend	0000	
	Subtrahend	1110	one's complement
	Constant	1	for two's complement
	Result	1111	
High-order bit carry = 0			invert for carry indicator

The result does not fit as an unsigned number. The carry indicator is set (C=1).

Example 2:

Desired operation:	0 - 15 = -15		
Machine operation:	Minuend	0000	
	Subtrahend	0000	one's complement
	Constant	1	for two's complement
	Result	0001	
High-order bit carry = 0			invert for carry indicator

The result does not fit as an unsigned number. The carry indicator is set (C=1).

Note. When a negative result occurs on a subtract operation, the values may be useful to the programmer. The carry indicator and the result form a signed number. The carry indicator is the sign and the result is the number in two's complement form (see Figure F-4).

Overflow Indicator Setting

The overflow indicator is used to signal overflow of the result when the operands are presented as *signed* numbers. The machine does not regard the numbers as either signed or unsigned, but performs the designated operation (add or subtract) on the values presented. The programmer must interpret the condition of the result for the number representation involved. The machine detects this condition by inspection of any carry into and out of the high-order bit (sign position) of the result operand during the operation. The overflow indicator is set (O = 1) for the two cases where the carries disagree:

1. A carry into, but no carry out of the sign position.
2. No carry into, but a carry out of the sign position.

The overflow indicator is not set (O = 0) for the remaining two cases where the carries agree:

1. A carry into and out of the sign position.
2. No carry into and no carry out of the sign position.

Examples

A four-bit operand size is used in the following examples. Note that the signed number range for a four-bit operand is -8 to +7. No other signed number values may be represented.

- Addition (overflow indicator is not set)

Desired operation: +5 + (+2) = +7

Machine operation:	Augend	0101	
	Addend	0010	
	Result	0111	

Carry into sign position = 0

Carry out of sign position = 0 carries agree

The result fits as a signed number. The overflow indicator is not set (O = 0).

Desired operation: -4 + (-4) = -8

Machine operation:	Augend	1100	two's complement
	Addend	1100	two's complement
	Result	1000	two's complement

Carry into sign position = 1

Carry out of sign position = 1 carries agree

The result fits as a signed number. The overflow indicator is not set (O = 0).

- Addition (overflow indicator is set)

Desired operation: +4 + (+4) = +8

Machine operation:	Augend	0100	
	Addend	0100	
	Result	1000	

Carry into sign position = 1

Carry out of sign position = 0 carries disagree

The result does not fit as a signed number. The overflow indicator is set (O = 1).

Desired operation: -4 + (-5) = -9

Machine operation:	Augend	1100	two's complement
	Addend	1011	two's complement
	Result	0111	

Carry into sign position = 0

Carry out of sign position = 1 carries disagree

The result does not fit as a signed number. The overflow indicator is set (O = 1).

- Subtraction (overflow indicator is not set)

Desired operation: +7 - (+2) = +5

Machine operation:	Minuend	0111	
	Subtrahend	1101	one's complement
	Constant	1	for two's complement
	Result	0101	

Carry into sign position = 1

Carry out of sign position = 1 carries agree

The result fits as a signed number. The overflow indicator is not set (O = 0).

Desired operation: +5 - (-1) = +6

Note. -1 is equal to 1111

Machine operation:	Minuend	0101	
	Subtrahend	0000	one's complement
	Constant	1	for two's complement
	Result	0110	

Carry into sign position = 0

Carry out of sign position = 0 carries agree

The result fits as a signed number. The overflow indicator is not set (O = 0).

- Subtraction (overflow indicator is set).

Desired operation: +7 - (-2) = +9

Note. -2 is equal to 1110

Machine operation:	Minuend	0111	
	Subtrahend	0001	one's complement
	Constant	1	for two's complement
	Result	1001	

Carry into sign position = 1

Carry out of sign position = 0 carries disagree

The result does not fit as a signed number. The overflow indicator is set (O = 1).

Desired operation: -3 - (+6) = -9

Machine operation:	Minuend	1101	two's complement
	Subtrahend	1001	one's complement
	Constant	1	for two's complement
	Result	0111	

Carry into sign position = 0

Carry out of sign position = 1 carries disagree

The result does not fit as a signed number. The overflow indicator is set (O = 1).

Appendix G. Reference Information

This appendix contains the following reference information:

- Condition codes
- General registers
- Interrupt status byte
- Level status register (LSR)
- Processor status word (PSW)

Condition Codes

I/O Instruction Condition Codes

These codes are reported during execution of an Operate I/O instruction.

Condition code (CC) value	*LSR position*				
	Even	*Carry*	*Overflow*	*Reported by*	*Meaning*
0	0	0	0	channel	Device not attached
1	0	0	1	device	Busy
2	0	1	0	device	Busy after reset
3	0	1	1	chan/dev	Command reject
4	1	0	0	device	Intervention required
5	1	0	1	chan/dev	Interface data check
6	1	1	0	controller	Controller busy
7	1	1	1	chan/dev	Satisfactory

Interrupt Condition Codes

These condition codes are reported by the device or controller during priority interrupt acceptance.

Condition code (CC) value	*LSR position*				
	Even	*Carry*	*Overflow*	*Reported by*	*Meaning*
0	0	0	0	controller	Controller end
1	0	0	1	device	Program controlled interrupt (PCI)
2	0	1	0	device	Exception
3	0	1	1	device	Device end
4	1	0	0	device	Attention
5	1	0	1	device	Attention and PCI
6	1	1	0	device	Attention and exception
7	1	1	1	device	Attention and device end

General Registers

R or RB field value*	*Register selected*
000	Register 0
001	Register 1
010	Register 2
011	Register 3
100	Register 4
101	Register 5
110	Register 6
111	Register 7

*The RB field sometimes contains only the two low-order bits. In this case, registers 4 through 7 cannot be specified.

Interrupt Status Byte (ISB)

DPC Devices

Bits	*Contents*
0	Device status available
1	Delayed command reject
2	Device dependent
3	Device dependent
4	Device dependent
5	Device dependent
6	Device dependent
7	Device dependent

Cycle Steal Devices

Bits	*Contents*
0	Device status available
1	Delayed command reject
2	Incorrect length record
3	DCB specification check
4	Storage data check
5	Invalid storage address
6	(not used, always zero)
7	Interface data check

Level Status Register (LSR)

Bit	*Contents*
0	Even indicator
1	Carry indicator
2	Overflow indicator
3	Negative result indicator
4	Zero result indicator
5	(not used, always zero)
6	(not used, always zero)
7	(not used, always zero)
8	Supervisor state
9	In process
10	Trace
11	Summary mask
12	(not used, always zero)
13	(not used, always zero)
14	(not used, always zero)
15	(not used, always zero)

Processor Status Word (PSW)

Bit	*Contents*
0	Specification check
1	Invalid storage address
2	Privilege violate
3	(not used, always zero)
4	Invalid function
5	(not used, always zero)
6	Stack exception
7	(not used, always zero)
8	Storage parity check
9	(not used, always zero)
10	CPU control check
11	I/O check
12	Sequence indicator
13	Auto-IPL
14	(not used, always zero)
15	Power/thermal warning

Index

Index of Instructions by Format

Index of Instructions by Name

IBM Series/1 Model 3 4953 Processor
and Processor Features Description
GA34-0022-1

READER'S COMMENT FORM

Cut or Fold Along Line

YOUR COMMENTS, PLEASE . . .

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. All comments and suggestions become the property of IBM.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or to the IBM branch office serving your locality.

Corrections or clarifications needed:

Page Comment

What is your occupation?____
Number of latest Technical Newsletter (if any) concerning this publication: _______________________
Please indicate your name and address in the space below if you wish a reply.

____

____

____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A.
(Elsewhere, an IBM office or representative will be happy to forward your comments.)

GA34-0022-1

Your comments, please . . .

This manual is part of a library that serves as a reference source for IBM systems. Your comments on the other side of this form will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Cut Along Line

Fold

Fold

First Class
Permit 40
Armonk
New York

Business Reply Mail
No postage stamp necessary if mailed in the U.S.A.

IBM Corporation
Systems Publications, Dept 27T
P.O. Box 1328
Boca Raton, Florida 33432

Fold

Fold

IBM

International Business Machines Corporation
General Systems Division
5775D Glenridge Drive N.E.
P.O. Box 2150, Atlanta, Georgia 30301
(U.S.A. only)

IBM®

International Business Machines Corporation

General Systems Division
5775D Glenridge Drive N.E.
P. O. Box 2150
Atlanta, Georgia 30301
(U.S.A. only)

IBM Series/1 Model 3 4953 Processor and Processor Features Description (S1-01) Printed in U.S.A. GA34-0022-1

GA34-0022-1