IBM Field Engineering Education
Student Self-Study Course

SYSTEM/360

Introductory Programming
Book 3 – Fixed Point Binary Operations

Preface

This is Book 3 of the System/360 Introductory Programming Student Self-Study Course.

Course Contents

	Book 1:	Introduction	R23-2933
	Book 2:	Program Control and Execution	R23-2950
●	Book 3:	Fixed Point Binary Operations	R23-2957
	Book 4:	Branching, Logical and Decimal Operations	R23-2958
	Book 5:	Input/Output Operations	R23-2959

Prerequisites

- Systems experience (1400 series with tapes, 7000 series with tapes) or a basic computer concepts course.
- Books 1 and 2 of this Introductory Programming course.

Instructions to the student and advisor

- This course is to be used by the student in accordance with the procedure in the Instructions to the Student section in Book 1 of this course.
- The course is to be administered in accordance with the procedure in the System/360 Introductory Programming Administrator Guide, Form #R23-2972.

This edition, R23-2957-1 is a minor revision of the preceding edition, but it does not obsolete R23-2957-0. Numerous changes of a minor nature have been made throughout the manual.

Issued to: ______________________

Branch Office: ______________ No: ________

Department: ______________________

Address: ______________________

If this manual is mislaid, please return it to the above address.

Copies of this and other IBM publications can be obtained through IBM Branch Offices. Address comments concerning the content of this publication to: IBM, FE Education Planning, Dept. 911, Poughkeepsie, N. Y., 12602

How to use this book

There are five sections to this text. At the beginning of each section, is a list of Learning Objectives which you will be expected to learn as a result of studying that particular section. Instead of having review questions at the end of each section, this book has a programming exercise in the last section and review questions for the entire book. You can evaluate your understanding of the book as you do this exercise. You will go through this book in a serial fashion. That is, you will not be expected to skip or branch around. The answer to each frame is in the next frame. You may find it helpful to use a standard IBM card to cover the answers as you read the frames.

Periodically, as you go through this book, you will be directed to study areas of the System/360 Principles of Operation manual. This will help you to become familiar with the manual so that it may be used as reference material at a later date.

THE CONTENTS OF THIS BOOK

This book deals mainly with the fixed point arithmetic instructions of the System/360. These instructions are part of the Standard Instruction set and are standard on models 30 - 70 of the System/360. The fixed point arithmetic instructions use both the (1) Storage-to-Register concept and the (2) Register-to-Register concept. These instructions also assume that the operands are in the Binary Data format.

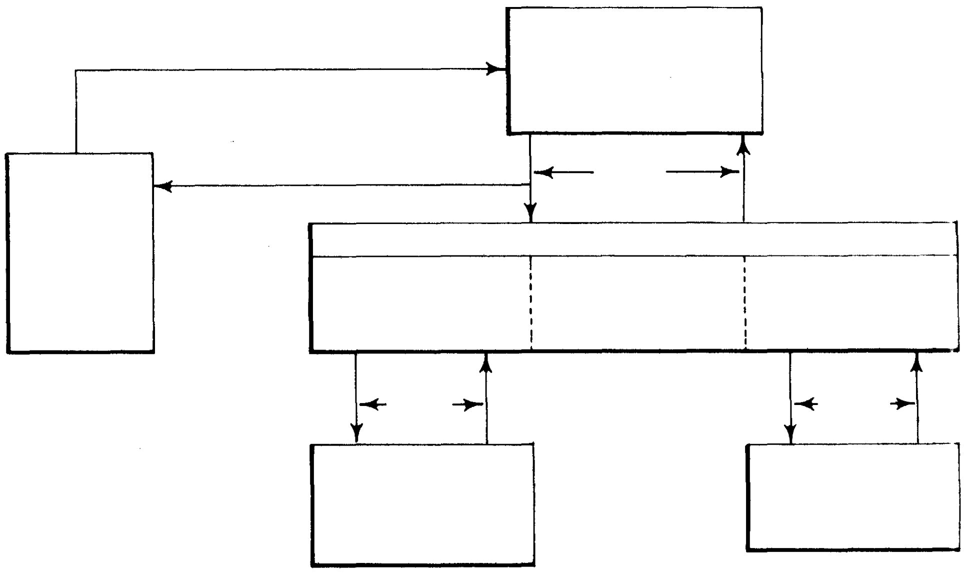
System/360 Fixed Point Binary Operations

SECTION I LEARNING OBJECTIVES

At the end of this review section. you should be able to:

1. State the names of the System/360 CPU data flow blocks and lines.
2. State that fixed point data fields are of halfword, word or doubleword lengths.
3. State that fixed point operands are of halfword or word lengths and are addressed by their high-order byte.
4. State that fixed point instructions are of the RR, RX, or RS format and are one or two halfwords in length.
5. State that negative binary operands appear in complement form.
6. Add and subtract binary operands.
7. Determine when a fixed point overflow occurs.
8. State the function of the Op code bits.

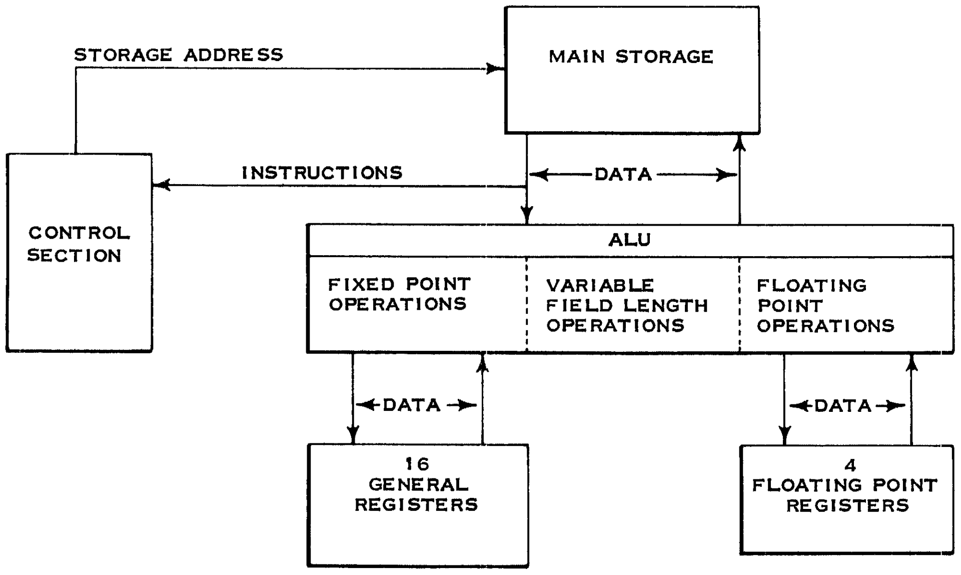
Review of Data and Instruction Formats



Shown above are the blocks that make up the System/360 CPU as well as main storage.

1. Identify the blocks as to:
   - main storage
   - control section
   - general registers
   - ALU
   - floating point registers
   - fixed length operations
   - variable length operations
   - floating point operations

2. Identify the lines as to:
   - addresses
   - instructions
   - data



For use as accumulators, the programmer has available general registers ___ through ____.

For use as base registers, the programmer may use general registers ___ through ____.

For use as index registers, the programmer has available general registers ___ through ____.

0, 15
1, 15
1, 15

When the programmer specifies general register 0 as a base register or an index register: (Circle <u>one</u> of the following.)

a. The contents of register 0 are added to the displacement.

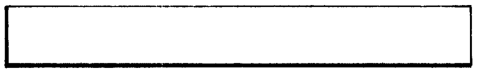
b. The contents of register 0 are ignored in generating the effective storage address.

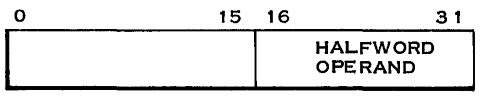
c. A program interrupt will occur.

---

b; The effective base or index address will be all zeros.

Number the bit positions of the general register below and indicate where a halfword operand would be placed.





Fixed length operands are processed using which of the following concepts: (Circle one or more.)

a. Register-to-register

b. Storage-to-register

c. Storage-to-storage

---

a, b

A fixed length operand in main storage is addressed by its ________ (leftmost/rightmost) byte location.

---

leftmost

The specified address of a fixed length operand must be divisible by the number of ______ in the field or a __________ exception will occur.

---

bytes
specification

A specification exception will cause a p______ i______.

---

program interrupt

Fixed length operands are in a ______ (binary/decimal) format.

---

binary

The three sizes of fixed length data are:

1. __________

2. __________

3. __________

---

1. halfword
2. word
3. doubleword

The leftmost bit of a binary operand is the s____ position while the remaining bits are the i______.

---

sign
integer

Positive binary numbers are represented in their _____ form with a ___ (0/1) in the high-order bit position.

---

true
0

Show the decimal value of + 1 as a halfword binary operand.

---

0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 1

Negative binary numbers are represented in their __________ form with a ___ (0/1) in the high-order bit position.

---

complement
1

Show the decimal value of -1 as a halfword binary operand.

---

1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1

In System/360, can a negative binary number be represented in true form? ________.

No; The only way the System/360 knows that a binary operand is negative is by examining the high-order bit. If that bit is 1, the machine assumes that it is a complement (negative) number.

On an "add" instruction involving two binary operands with <u>unlike</u> signs, does one of the operands have to be complemented before adding? _____

No; If the operands have unlike signs, this means that one of them is negative and therefore is already in complement form.

Add the following operands.

```
        Sign          Integer
         /          /
        ↙    ______↙______
     0   1 0 1 1 0 0 1        1st Operand
   + 1   0 0 0 1 1 1 1        2nd Operand
   -------------------
```

---

1 1 1 0 1 0 0 0

On a subtract instruction involving two binary operands with <u>unlike</u> signs, does one of the operands have to be complemented before adding? ______

Yes

Because negative binary numbers in the System/360 can only be represented in complement form, sign analysis does not apply to System/360 fixed point instructions. It boils down to this:

1. On an "add" instruction, the two operands are true added. That is, the machine does not need to complement (or re-complement in the case of negative numbers) either operand before adding.

2. On a "subtract" instruction, one of the operands is complemented (or re-complemented if negative) and then the two operands are <u>added</u>.

Subtract the following operands.

$$\begin{array}{rl} \overset{\text{Sign}}{0}\ \ \overbrace{0\ 1\ 1\ 0\ 1\ 1\ 1}^{\text{Integer}} & \text{1st Operand} \\ -\ \underline{1\ \ 1\ 1\ 1\ 1\ 1\ 1\ 1} & \text{2nd Operand} \end{array}$$

---

0 0 1 1 1 0 0 0; The 2nd operand (even though already in complement form) had to be re-complemented before the two operands were added.

Whenever the largest negative or positive number is exceeded as a result of a binary operation, a f______ p______ o________ will occur.

---

fixed point overflow

The System/360 detects a fixed point overflow whenever the carry out of the sign position __________ (does/does not) agree with the carry out of the high-order bit of the integer.

---

does not

Do the following binary problems. Show all work, indicating complementing when necessary. Also indicate whether or not a fixed point overflow will occur.

$$\begin{array}{lll} \text{a.} & \text{ADD} & \overset{\text{Sign}}{0}\ \overbrace{1\ 1\ 1\ 1\ 0\ 0\ 1}^{\text{Integer}} \quad \text{1st Operand} \\ & & \underline{0\ 0\ 0\ 0\ 1\ 1\ 1\ 1} \quad \text{2nd Operand} \end{array}$$

______________ (Overflow/No Overflow)

(Frame continued on next page.)

```
b.    ADD  1 0 0 0 1 0 0 1     1st Operand

           0 0 1 0 0 1 1 0     2nd Operand
           _______________

                               __________ (Overflow/No Overflow)


c.    SUBTRACT
           1 1 1 1 1 1 1 1     1st Operand

           0 0 1 1 0 1 1 1     2nd Operand
           _______________

                               __________ (Overflow/No Overflow)


d.    SUBTRACT
           1 0 0 0 1 0 0 0     1st Operand

           0 1 0 1 0 0 0 0     2nd Operand
           _______________

                               __________ (Overflow/No Overflow)
```

```
a.    ADD          0 1 1 1 1 0 0 1

                   0 0 0 0 1 1 1 1
                   _______________
                    ↶
                   1 0 0 0 1 0 0 0     Overflow
```

The fixed point overflow is due to the fact that there was a carry into the sign position and no carry out of it.

```
b.    ADD          1 0 0 0 1 0 0 1

                   0 0 1 0 0 1 1 0
                   _______________

                   1 0 1 0 1 1 1 1     No Overflow
```

(Answers continued on next page.)

```
c.   SUBTRACT     1 1 1 1 1 1 1 1     1 1 1 1 1 1 1 1
                                   =
                 -0 0 1 1 0 1 1 1  +  1 1 0 0 1 0 0 1
                 ----------------     ---------------
                                   C 1 1 0 0 1 0 0 0

                                      No Overflow
```

Notice that because the instruction is "subtract," the 2nd operand was complement added to the 1st operand. Also, note that the carry out of the sign bit position did <u>not</u> result in a fixed point overflow. This is because there was also a carry into the sign position.

```
d.   SUBTRACT     1 0 0 0 1 0 0 0     1 0 0 0 1 0 0 0
                                   =
                  0 1 0 1 0 0 0 0  +  1 0 1 1 0 0 0 0
                  ---------------     ---------------
                                   C  0 0 1 1 1 0 0 0

                                      Overflow
```

The 2nd operand was again complement added to the 1st operand. A fixed point overflow did occur this time because there was a carry out of the sign position and there was no carry into it.

In the previous examples of binary arithmetic, you were working with an eight-bit (1 byte) number. As you know, fixed point binary arithmetic in the System/360 uses either halfword or word operands. The principles of determining when to complement or how to detect a fixed point overflow still apply, regardless of the length of the operands.

Whenever a fixed point overflow is detected, a p________ i________ may occur depending on the program mask in the PSW.

---

program interrupt

If the program mask in the PSW allows the program interrupt, the fixed point overflow exception will be noted in the i__________ c_____ of the "old" PSW.

---

interruption code

Fixed point instructions use both halfword and word binary data as operands. These operands may be processed with both the storage-to-register and the register-to-register concepts. The Op code of the instruction will determine which size operand and which processing concept to use. Let's see what you remember about Op codes and instruction formats.

Instructions are a multiple of __________ in length.

---

halfwords

The five instruction formats are:

____ ____ ____ ____ ____ ____ ____ ____ ____ ____

RR, RX, RS, SI, SS
(In any order)

The first byte of every instruction is the ____ ______. The instruction format is indicated by bits ___ and ___ of the Op code.

Op code
0, 1

An RR format is indicated by a ____ in bits 0 and 1 of the Op code. The RR format instruction is one __________ in length.

00
halfword

Indicate the fields of the RR format.

OP CODE	R1	R2

The R1 field usually contains the address of a _________ __________.
This register contains the _____ (1st/2nd) operand.

---

general register
1st

The results of the instruction (such as add or subtract) will usually replace the _____ (1st/2nd) operand.

1st

If bits 0 and 1 of the Op code are 01, an ___ ___ format is indicated.

RX

An RX format instruction is _____ ___________ in length.

---

two halfwords

Indicate the fields of the RX format.

OP CODE	R1	X2	B2	D2

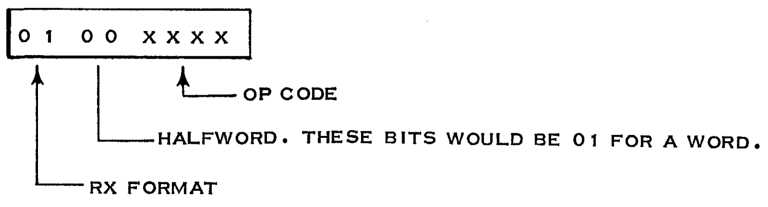
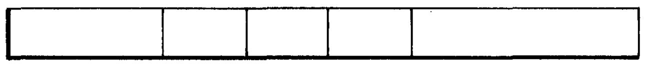
In the RX format, the 2nd operand is located in ______ _________.

---

main storage

Binary operands in main storage may be one __________ or one ______ in length.

---

halfword word	In the RX format, the length of the 2nd operand is indicated by bits ____ and ____ of the Op code.
2 3	 0 1 0 0 X X X X OP CODE HALFWORD. THESE BITS WOULD BE 01 FOR A WORD. RX FORMAT
	The location of the 2nd operand in the RX format is indicated by the ____, ____ and ____ fields.
X2, B2, D2	The effective address of the 2nd operand is generated by ____.
adding the contents of the index and base registers to the displacement.	The generated "effective address" is ____ bits long. For a halfword operand, this address must be divisible by _____. For a word operand, this address must be divisible by ______.
24 two four	If the X2 or B2 fields contain zero, the contents of reg 0 _____ (are/are not) used in generating the effective storage address.
are not	While bits 0 - 3 of the Op code indicate the instruction format and type of data, bits 4 - 7 indicate the specific ____________________________.
operation or instruction such as add, subtract, etc.	Indicate the fields of the RS format. 

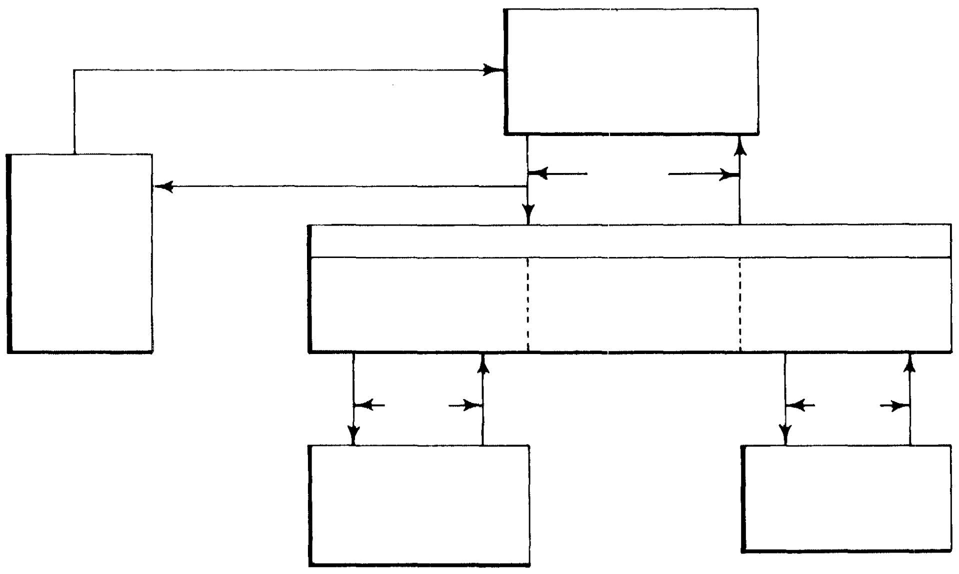
OP CODE	R1	R3	B2	D2

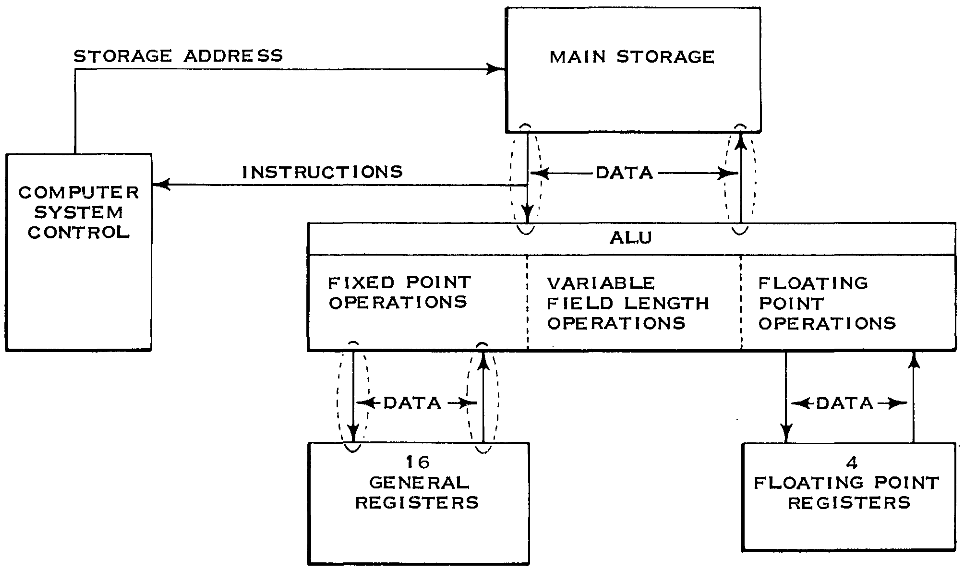
The RR and RX formats are the two with which you will be most concerned while learning the fixed point instructions, although there are a few RS type instructions. Basic to any type of data processing is the ability to add and subtract. If the System/360 had only one type and length of data, it could possibly get by with one "add" instruction and one "subtract" instruction. However, as you have learned, the System/360 has fixed as well as variable length and binary as well as decimal data formats.

In the fixed length binary format, it can even have two different operand lengths, halfwords and words. The halfword operand format can be processed storage to register, while the word operand format can be processed both storage to register and register to register. At this time, we are only concerned with the fixed point instructions. These instructions deal with fixed length binary operands.

Write in the names of the data flow blocks and lines.

Circle the lines upon which fixed point data will flow.





Go to the IBM System/360 Principles of Operation manual and briefly study the following areas of the Fixed Point Arithmetic section.

Data Format
Number Representation
Condition Code
Instruction Format
Instructions (Study the list only. Do not study the explanation of the individual instructions.)

Before studying these fixed point instructions, let's be sure you know how data can be put in the binary format. The next few pages will cover the converting of data to and from the binary format.

System/360 Fixed Point Binary Operations

SECTION II LEARNING OBJECTIVES

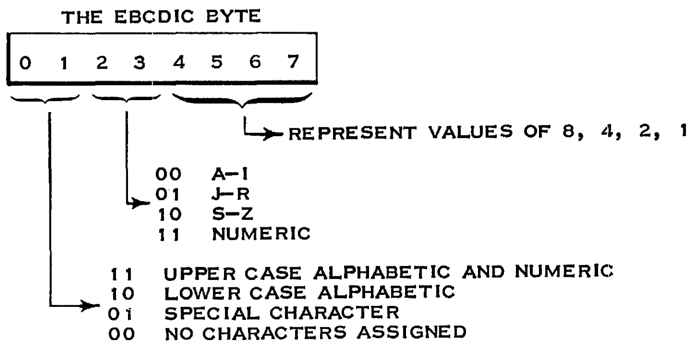
At the end of this section, you should be able to do the following when given mnemonics of PACK, UNPK, CVB and CVD.

1. State instruction length and format.
2. State location and format of operands.
3. Determine the result and where it will be located.
4. State effect on condition code.
5. State which program checks are possible.

Converting Data To/From Binary

	You have demonstrated a knowledge of binary data formats. You know that positive numbers are represented in true form and that negative numbers are represented in complement form. You also know that these binary numbers appear in main storage as halfwords or full-words. You are probably wondering, however, how data from a punched card gets into main storage in these binary formats. You should know the standard card code (hollerith). So let's start at that point.
	To punch the decimal number 1234 in an IBM card would require ______ columns.
four	Each column of a card read into a System/360 usually occupies one b______ of main storage.
byte	Data from a card reader is usually represented in main storage in the Extended _________ _________ _________ Interchange Code.
Binary Coded Decimal (BCD)	The Extended Binary Coded Decimal Interchange Code is usually called __________. The code uses _____ bits to represent a card column.
EBCDIC 8	The 8 bits of EBCDIC have two parts: zone and numeric. The zone part consists of bits __________ and the numeric part consists of bits ________.

0-3
4-7



Go to the Principles of Operation manual and refer to the EBCDIC chart in the Logical Operations area of the System Structure section.

Notice that the dark areas indicate card punching and that the circled numbers refer to notes on the bottom of the chart. The notes show some special card punching combinations.

As can be seen on the EBCDIC chart, a numeric "1" punch would be represented by a combination of 8 bits in EBCDIC as ______________.

11110001

The letter A (12 and 1 hole punches) would be represented as __________.

11000001

The character "J" is represented on a card by an____ zone punch and a __ digit punch. It is represented in main storage as the following byte: ___________.

11
1
11010001

A lower case "j" would be represented in a card by a digit punch of 1 and zone punches of ___ and ___. It would be represented in storage as ___________.

12
11
10010001

The special character % would be represented by a ___ zone punch and digit punches of ___ and ___. It would be represented in storage as ___________.

0
8
4
01101100

To get the bit combination 01101100 into storage would require a zone punch of ___ and digit punches of ___ and ___.

0
8
4

A blank column on a card would be represented in storage as _________. (Refer to note 5 on the chart.)

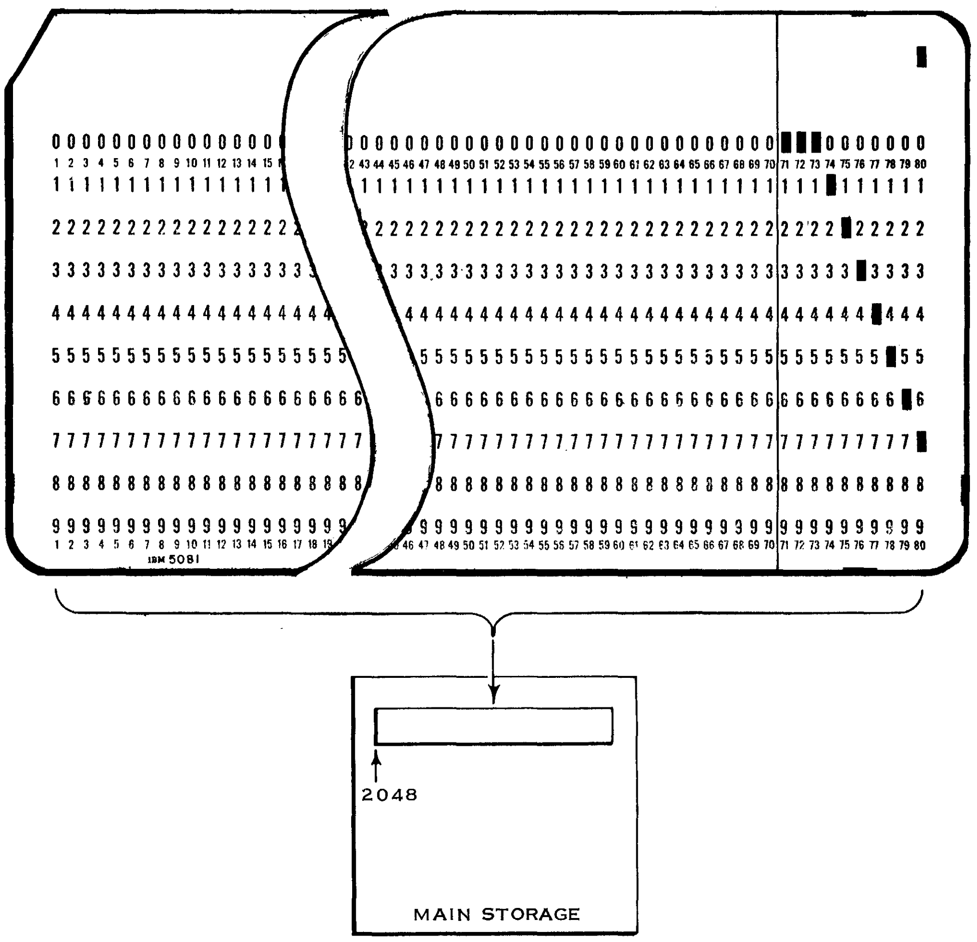
01000000

To get a bit combination of 00000000 would require zone punches of ____, ____ and digit punches of ___, ___, ___. (Refer to note 1)

12
0
9
8
1

Usually cards are punched in the standard hollerith card code. That is, only decimal and alphabetic information is punched in the card. Then after the data is brought into storage, it can be converted (via instructions) to binary and processed with the fixed point instructions.

Given the following card record:

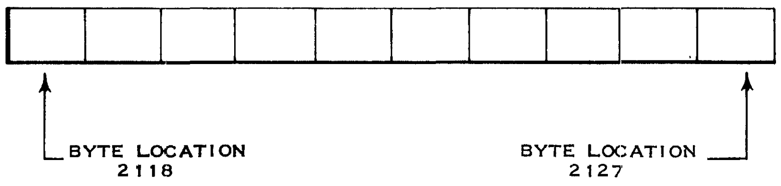


The card contains +0001234567 in columns 71-80. Assuming that the entire card record has been read into main storage starting at location 2048, columns 71-80 will be in byte locations _____ through ______.

2118, 2127;
locations 2048-2117
would contain card
columns 1-70

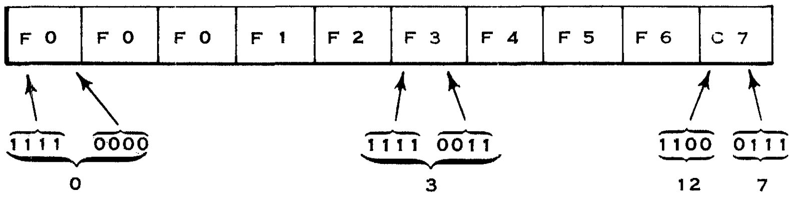
Numeric fields in EBCDIC are said to be in the ______ (zoned/packed) decimal format.

Show (in hex) the zoned decimal data from columns 71-80 of the card.



---

zoned



The sign of a zoned decimal data field is in bits ____ of the ____ (low/high) order byte.

PACK INSTRUCTION

0-3
low

Decimal data must be in the packed format before it can be converted to binary. Zoned decimal fields can be changed to the packed decimal format by an instruction called "_______."

"pack"

Packed decimal data consists of two _________ per byte with the low-order byte containing one digit and the ________. The sign of a packed decimal field is in bits _____ of the low-order byte.

---

digits
sign
4-7

Show (in hex) how the data in columns 71-80 would look if it were packed into eight bytes.

00	00	00	00	12	34	56	7C

The instruction "pack" is of the SS format. Label the fields.

SS FORMAT

OP CODE	L1	L2	B1	D1	B2	D2

Read the description of the "pack" instruction in the Decimal Arithmetic section of your Principles of Operation manual.

In the "pack" instruction, the 2nd operand contains the _______ (packed/zoned) decimal data.

---

zoned

The low-order byte of the 1st operand receives the low-order byte from the zoned data field. The zone bits of this byte are _______________ (assumed to be the sign/ignored).

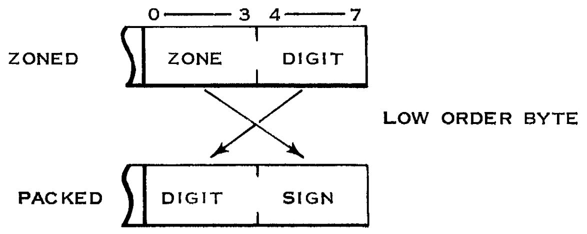
---

assumed to be the sign

The zone and digits bits from the low-order byte of the zoned decimal field are __________ before being placed in the 1st operand.

---

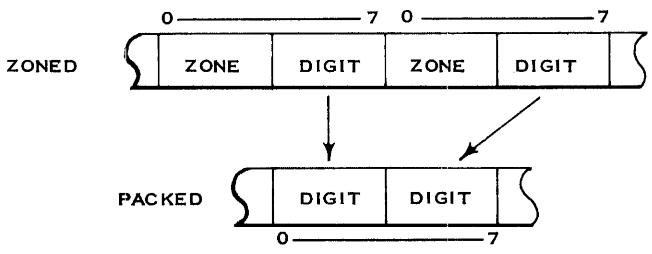
reversed or swapped as shown below



Each remaining byte of the 1st operand receives the ______ (zone/digit) bits from two successive bytes of the zoned decimal field.

---

digit; As shown below



The zone bits from the 2nd operand in the preceding example are ________.

---

ignored

The bytes from the zoned decimal field (2nd operand) _________ (are/are not) checked for valid sign or digit combinations.

---

are not

The zoned decimal field (2nd operand) and the resulting packed decimal field (1st operand) _____ (can/cannot) be of different lengths.

---

can

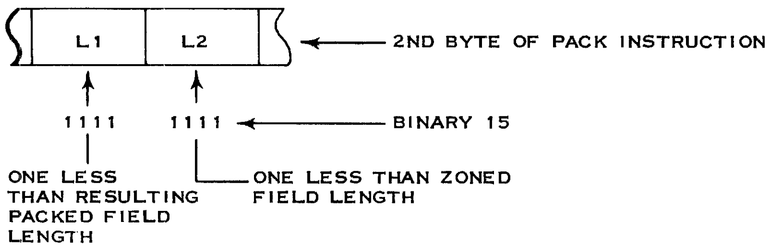
The 2nd byte of the "pack" instruction contains the ________ codes of the two operands. The number in the length code is ____________________ (equal to/one less than) the number of bytes in the operand.

---

length
one less than

The maximum number of bytes in either operand of a "pack" instruction is ____.

16; As shown below


L1
L2
2ND BYTE OF PACK INSTRUCTION
1111
1111
BINARY 15
ONE LESS THAN RESULTING PACKED FIELD LENGTH
ONE LESS THAN ZONED FIELD LENGTH

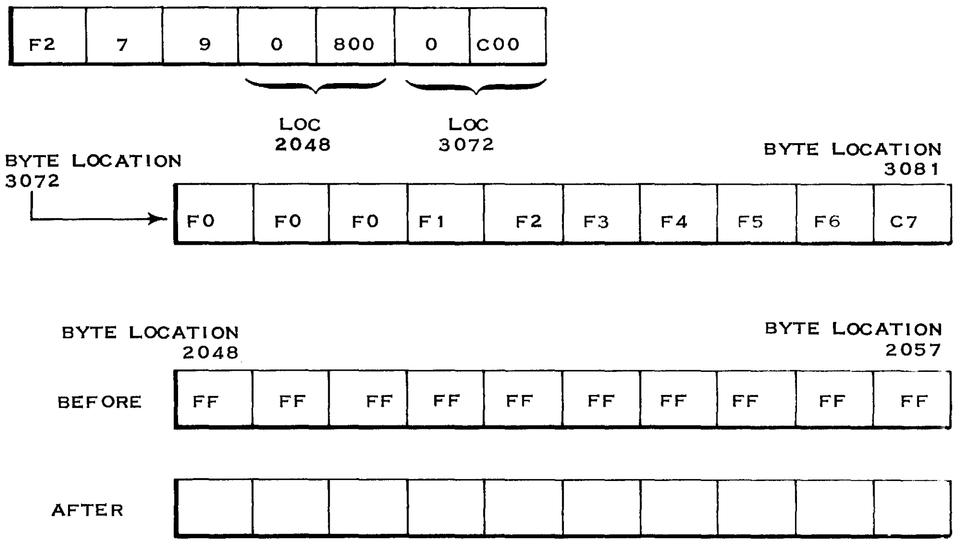
If the length codes are such that the 1st operand is long (compared to the 2nd operand), the packed decimal field will be extended with high-order ________.

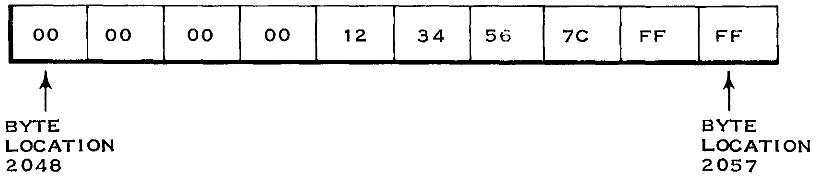
If the length codes are such that the 1st operand cannot contain all the digits from the zoned field, the remaining digits are __________.

---

zeroes
ignored

Given the following "pack" instruction, show the resulting packed decimal field. Instructions and data are shown in hex.





Just as with the previous instructions dealing with fixed length operands, the operands of the "pack" instruction are addressed by their ______ (high/low) order byte location.

---

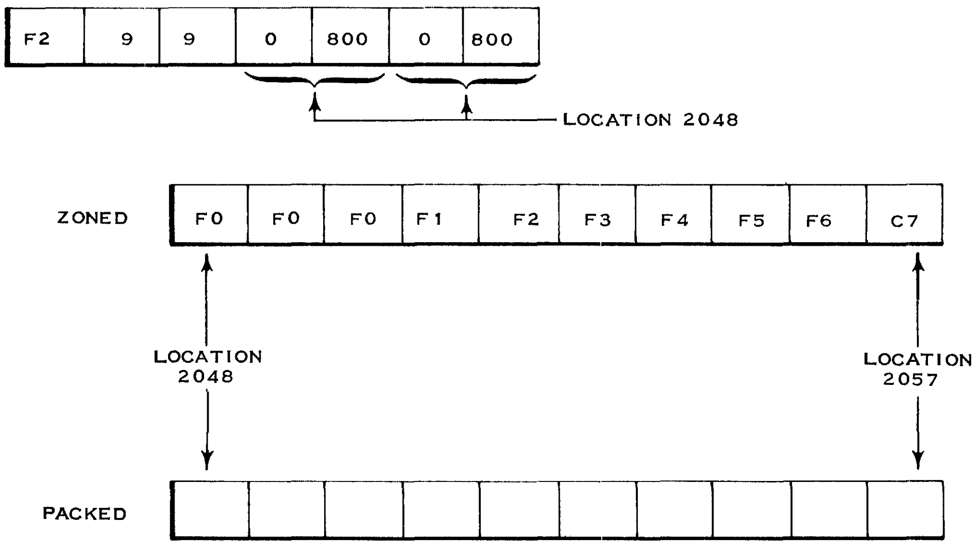
high

The address of the low-order byte of either operand in the "pack" instruction can be determined by adding its ________ code to its generated effective address.

---

length

Given the following "pack" instruction, show the resulting packed decimal field. Everything is shown in hex.



00	00	00	00	00	00	12	34.	56	7C

Notice that the original zoned data field was used to contain the resulting packed decimal field.

CONVERT TO BINARY INSTRUCTION

You have now seen how the "pack" instruction can change a zoned decimal field to a packed decimal field. The packed decimal data can now be changed to a word of binary data by use of the instruction: "convert to binary." This instruction will not only convert the data to binary, it will also load it into a general register. Read the description of the "convert to binary" instruction in the Fixed Point Arithmetic section of your Principles of Operation manual.

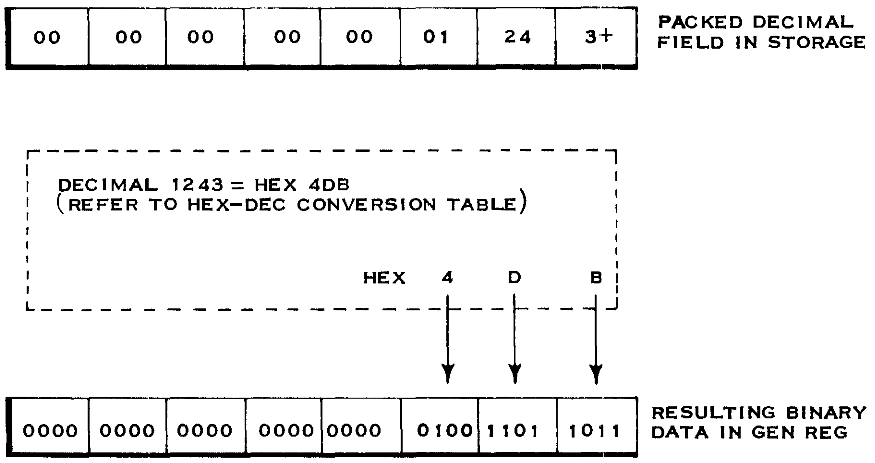
In the CVB ("convert to binary") instruction, the 2nd operand contains a __________________ (zoned decimal/packed decimal/binary) data field.

---

packed decimal	To use the CVB instruction, the packed decimal field must consist of _______ bytes. The specified address of the high-order byte must be divisible by ____ or a _____________ exception will occur.
eight 8 specification	The results of the CVB instruction will be a binary word and will be loaded into a _________ ____________.
general register	The data in the packed decimal field is checked for valid sign and digit codes. If any codes are improper, a _______ exception will be recognized.
data	0000-1001 are valid digit codes. If any of the digits of the packed decimal field are coded from 1010-1111, a _____ exception will be recognized. Valid sign codes are 1010-1111. If the sign of the packed decimal field (low-order four bits) contains any of the valid digit codes, a ______ exception will be recognized.
data data	Since a twelve hole punch is used to indicate a plus field on a card, the usual EBCDIC plus sign will be ______. (Refer to the EBCDIC chart) Since an eleven hole punch is used to indicate a minus field on a card, the usual EBCDIC minus sign will be _________.
1100; These are the zone bits for the letters A-I 1101; These are the zone bits for the letters J-R	Sometimes plus fields in a card do not have a twelve hole punch. In these cases, the expected EBCDIC plus sign would be _______.
1111; These are the zone bits for the numbers 0-9	Because decimal data may be in EBCDIC or in extended 8-bit ASCII (depending on PSW bit 12), either 1101 (EBCDIC) or 1011 (ASCII) are acceptable as minus signs. All other bit combinations of 1010-1111 are acceptable as ______ signs.
plus	If the sign of the packed decimal field is plus, the binary equivalent of the field will be loaded into a register in __________ (true/complement) form. If the sign of the packed decimal field is minus, the binary equivalent of the field will be loaded into a register in _____________ (true/complement) form.

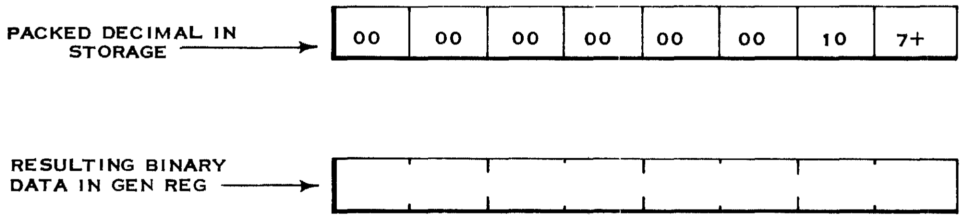
true complement

Example of conversion from packed decimal format to binary format.


00 00 00 00 00 01 24 3+
PACKED DECIMAL FIELD IN STORAGE
DECIMAL 1243 = HEX 4DB
(REFER TO HEX-DEC CONVERSION TABLE)
HEX 4 D B
0000 0000 0000 0000 0000 0100 1101 1011
RESULTING BINARY DATA IN GEN REG

In the preceding example, the conversion was made by first changing the decimal data to hex data and then the hex data to binary data. We go through the hex step simply because it makes decimal to binary conversion easier. Of course, the System/360 does not go through this hex step. It converts directly from decimal to binary.

Given the following packed decimal field, show the converted results in binary bits (not EBCDIC bits) in the general register.



0000	0000	0000	0000	0000	0000	0110	1011

Given the following packed decimal field, show the binary results in the general register.

PACKED DECIMAL IN STORAGE ⟶

00	00	00	00	00	00	10	7-

RESULTING BINARY DATA IN GEN REG ⟶

1111	1111	1111	1111	1111	1111	1001	0101

Notice that the -107 is loaded as the complement of the value 107.

If the value of the packed decimal field exceeds +2,147,483,647 or -2,147,483,648 it cannot be expressed in a binary word. When this happens, the low-order binary bits are placed in the register and a fixed point d__________ exception is recognized.

Divide! Of course, this instruction has nothing to do with division. The fixed point divide exception code is used in this case in the "old" PSW only as a convenient way of indicating what kind of programming error occurred on the CVB instruction.

While a fixed point divide exception is usually thought of as something like dividing by zero, it is used with the CVB instruction to indicate that the packed decimal number was too large for a binary word.

When the binary equivalent of the packed decimal number cannot be contained within a binary word, a ________ ________ ________ exception is recognized.

---

fixed point divide

Let's stop a minute and go back and consider reading in data from a card. You just learned how normal numerical punching can be read in and, through the use of the "pack" and "convert to binary" instructions, end up as binary data in storage.

It is also possible to use special punching in the cards. This punching will result in the direct entry of binary data into storage. The "pack" and "convert to binary" instructions will not be needed.

Since each card column comes into storage as a byte, ______ card columns would be required to bring in a binary word.

---

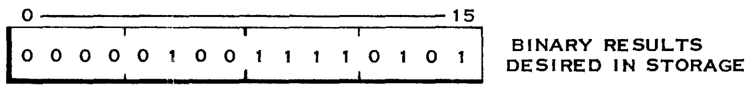
four

To bring in a binary halfword, _____ card columns are used.

---

two

To bring in the following halfword would require two card columns. Assume columns 70-71 are used. Column 70 would be punched ____________ and Column 71 would be punched ____________. (Refer to the EBCDIC chart.)

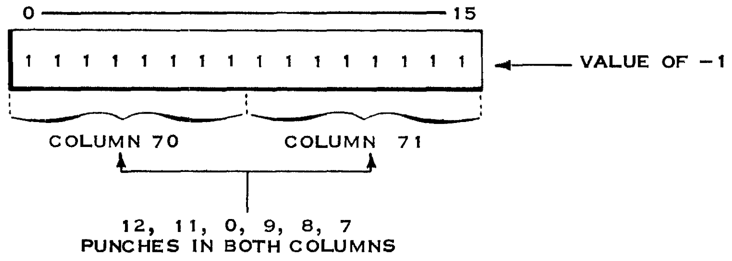


Column 70; 12, 9, 4 punches
Column 71; 5 hole punch

To bring a value of -1 in a halfword (using columns 70-71) would require the following holes to be punched in both columns 70 and 71.

____

12, 11, 0, 9, 8, 7 as shown below.



Of course to originally punch this binary information into cards would require use of the multi key on the keypunch. Binary information in main storage can be sent to a card punch unit and automatically punched out in the EBCDIC card code. Each byte of binary data would be punched in a card column as one of the 256 possible punching combinations.

Since each card column comes into main storage as a byte, an IBM card can hold ____ binary words.

20

To avoid specification exceptions later on, the beginning address of an input area for an IBM card containing 20 binary words should be divisible by _____.

---

four

In review then, data must be in the binary format to be processed with the fixed point instructions. Since each of the 256 different bit combinations in a byte can be obtained with the EBCDIC card code, it is possible to bring data into the machine in the binary format. In the event, however, that data is punched in the card, in the conventional manner (that is, decimal fields with a 12 or 11 hole punch over the units column) the data can be changed to binary via the "pack" instruction and the "convert to binary" instruction. The "pack" instruction will change a zoned decimal field (EBCDIC) in storage to a packed decimal field in storage. The "convert to binary" instruction will take a doubleword of packed decimal data and convert it to the binary format. The resulting binary word will be placed in the specified general register. In converting to binary, the packed decimal field is checked for:

1. Invalid digit and sign data codes - data exception.
2. Specified address not on a doubleword boundary - specification exception.
3. Decimal value is too large for binary word - fixed point divide exception.

CONVERT TO DECIMAL INSTRUCTION

After the data has been processed, it may be desirable to change it back to the zoned decimal format (EBCDIC). This would be necessary if we wished to print the data out in recognizable form or punch the data out in standard card code. This can be done by use of two instructions. The "convert to decimal" instruction will convert the contents of a general register to the packed decimal format and place it in main storage. This packed decimal field can then be changed to the zoned format by use of the "unpack" instruction.

Read the description of the "convert to decimal" instruction in the Fixed Point Arithmetic section and the description of the "unpack" instruction in the Decimal Arithmetic section of the Principles of Operation manual.

The first step in changing a binary result to EBCDIC is to use the CVD instruction. This instruction will change the binary word to a doubleword of _______ decimal data.

---

packed

If the address of the 2nd operand (packed decimal result) in the CVD instruction is not on a doubleword boundary, a ____________ exception will be recognized.

---

specification

The coding of the sign bits of the packed decimal result will depend on the sign of the binary word and bit position 12 of the PSW. If bit 12 of the PSW is 0, the EBCDIC plus sign of ______ or minus sign of _____ will be generated. (Refer to EBCDIC chart.)

1100
1101

If bit 12 of the PSW is set to 1, the standard EBCDIC signs will not be generated. Instead, the generated signs will be those of the extended _______ code.

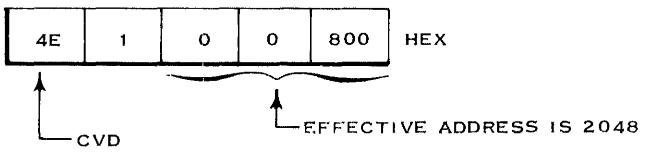
ASCII

The generated sign is placed in the ______ (low/high) order four bits of the doubleword in storage. The remaining bits of the doubleword will contain a total of ___ BCD digits.

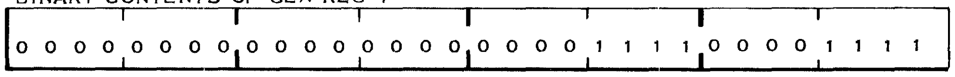
---

low
15

Given the following CVD instruction, show the resulting packed decimal field.

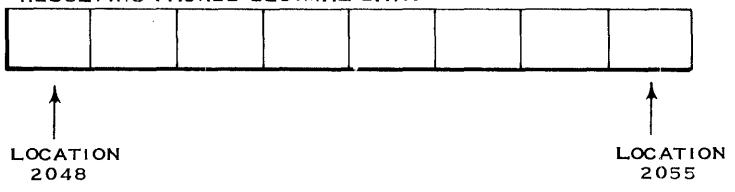


BINARY CONTENTS OF GEN REG 1



(1ST CONVERT THE BINARY TO HEX. USE THE HEX-DEC CONVERSION TABLE TO FIND THE DECIMAL RESULT.)

RESULTING PACKED DECIMAL DATA



00	00	00	00	00	03	85	5+

Show the bit structure of the 3 low-order bytes of the preceding packed decimal field.

---

0 0 0 0 0 0 1 1	1 0 0 0 0 1 0 1	0 1 0 1 1 1 0 0

Given the following binary word, show how the packed decimal doubleword would appear after using the CVD instruction.

BINARY CONTENTS OF GEN REG 1

1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	0 1 1 1 1 0 1 0	0 1 0 0 0 0 0 0

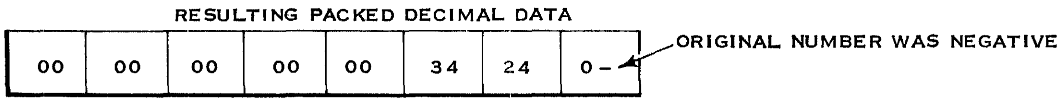
(CONVERT BINARY TO HEX. COMPLEMENT THE HEX. CONVERT COMPLEMENTED HEX TO DECIMAL.)

RESULTING PACKED DECIMAL DATA

$$\begin{tikzcd}[column sep=small]
\text{F F F F 7 A 4 0} \arrow[d] & \text{Hex representation of complement binary number in general register.} \arrow[l] \\
\text{0 0 0 0 8 5 C 0} & \text{Hex representation of the true form of the binary number.} \arrow[l]
\end{tikzcd}$$

$$\begin{tikzcd}[column sep=small]
\text{85 C0} & \text{Convert to decimal using Hex-Dec Conversion Table. Break down hex number to fit table.} \arrow[l]
\end{tikzcd}$$

Hex	=	Decimal	
5C0	=	1472	Add together
8000	=	32768	
85C0	=	34240	



UNPACK INSTRUCTION

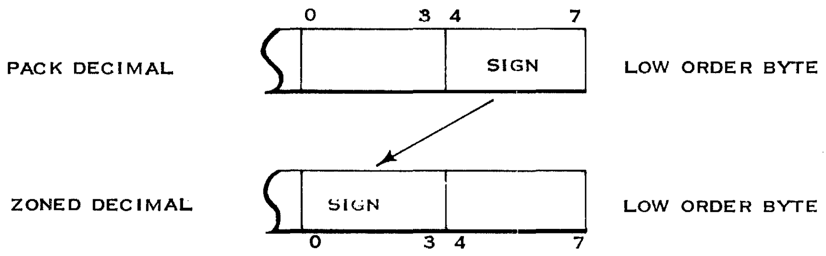
Now that the binary results of the processed data have been placed back in main storage as packed decimal data, the "unpack" instruction can be used to change the data to the zoned decimal format.

The 2nd operand of the "unpack" instruction is assumed to be in the _______ (zoned/packed) format.

packed

Bits 4-7 of the 2nd operand's low-order byte are placed unchanged in bits _____ of the 1st operand's low-order byte.

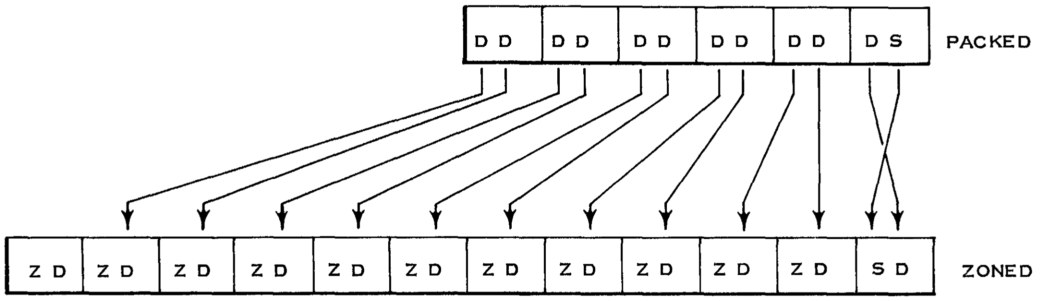
0-3; These bits represent the sign as shown below.



The remaining bits of the packed decimal field represent ___________. These digits are placed in bits ____ of the bytes of the 1st operand.

---

digits
4-7; As shown below.



Bits 0-3 of the zoned decimal field represents the zone. If PSW bit 12 is 0 (EBCDIC), zone bits of _______ will be inserted.

1111

In review, then, once data has been processed with the fixed point instructions, it can be converted back to the zoned decimal format. This would allow the data to be punched out in the standard card code or printed out in readily readable form. To convert binary data to zoned decimal data requires using the "convert to decimal" instruction and the "unpack" instruction. The "convert to decimal" instruction will take the binary contents of a general register and place it in main storage as a doubleword of packed decimal data. The "unpack" instruction will change the packed decimal data to zoned decimal data.

You should now know that data can be put into the binary format by one of the two following methods:

1. Numeric Data can be punched in the standard card code (Hollerith) and read into storage as zoned decimal data. Then by means of two instructions it can be changed to the binary format.

2. By using the 256 possible punching combinations of EBCDIC, any binary bit combination can be read into main storage.

You are now ready to study the fixed point instructions. These instructions will process data which is in the binary format.

System/360 Fixed Point Binary Operations

SECTION III LEARNING OBJECTIVES

At the end of this section, you should be able to do the following when given the mnemonic of any fixed point instruction.

1. State instruction length and format.

2. State location and format of operands.

3. Determine the result and where it will be located.

4. State effect on condition code.

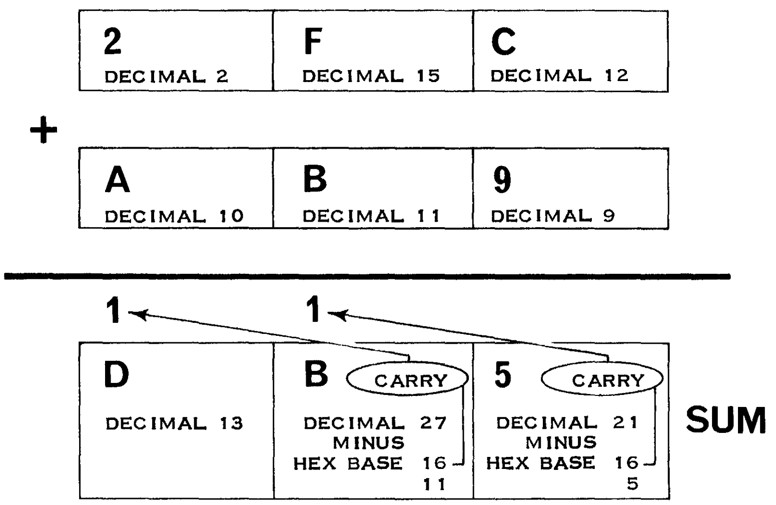
5. State which program checks are possible.

Fixed Point Instructions

Binary numbers are often shown in hexadecimal. This is done to simplify working with the binary data. Hex arithmetic is used in many of the System/360 manuals and will be used when you receive System/360 training at a plant school. Therefore, it is important that you become thoroughly familiar with hex arithmetic.

Before studying the fixed point add instructions, let's review hex addition.

Example of adding hex AB9 to hex 2FC: (Dark numbers are hex arithmetic)



Use this example to complete the following frames.

In the low-order position of the preceding example, a hex C is added to a hex 9. The result is a hex ___ and a carry of a hex ___.

5, 1

In the next position, a hex ____, ____ and ____ are added. The result is a hex ____ and a c______ of hex 1.

F, B, 1
B, carry

The high-order position shows the addition of a hex ___, ____ and ____. The result is a hex ____.

2, A, 1
D

Hex addition rule:

Any time the addition of two hex numbers results in more than F (decimal 15) the amount over (more than) decimal 16 becomes the s____ and a carry of hex ___ is added to the next position.

---

sum
1

Do the following:

1. Use the Hexadecimal-Decimal Conversion table in the Appendix of the Princples of Operation manual and convert the two decimal numbers of the addition problem to hex.
2. Add the decimal numbers.
3. Add the hex numbers.
4. Use the conversion table to check your hex sum against the decimal sum.

Addition problem:

Decimal		Hex
2849	=	
+ 1021	=	+

Okay, now that you have reviewed hex addition, let's start the fixed point add instructions.

ADD INSTRUCTIONS - ALGEBRAIC

Shown below are three instructions which can be used to add the binary data formats that you have learned.

Mnemonic	Hex Op Code	Data Flow
AH	4A	Halfword storage to register
A	5A	Fullword storage to register
AR	1A	Fullword register to register

It is assumed that you know that a mnemonic is a symbolic method of representing an Op code. Notice that the letter "A" is used to indicate an add instruction. An ending letter of "H" is used to indicate a halfword operand length while an ending letter of "R" is used to indicate an RR type instruction.

In each of the above instructions, the 2nd operand is added to the 1st operand and the sum replaces the 1st operand.

Read the description of the preceding "add" instructions in your Principles of Operation manual. These descriptions will be found in the Fixed Point Arithmetic section. Do not read the description of the "add logical" instruction. It will be covered later.

Write (using "hex") the complete instruction to add field A to field B. Assume field A is in register 5 and field B is in register 2.

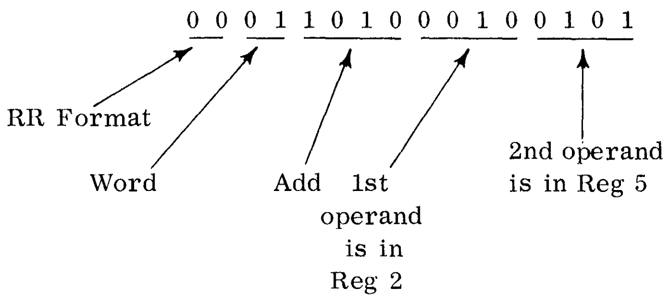
			RR FORMAT

1A	2	5

Notice that since we are adding to field B, field B (reg 2) is implied to be the 1st operand.

Write the preceding instruction in binary bit fashion.

____



Show the contents of registers 2 and 5 as a result of the preceding instruction.

Reg 2   0 0 4 8 7 A 0 1   ______________

Reg 5   F F F F A A A A   ______________

Reg 2

0 0 4 8 2 4 A B

Reg 5

F F F F A A A A

Notice that the 2nd operand is unchanged by the addition. The 1st operand (in reg 2) is replaced by the sum.

Example of how the System/360 executes the instruction using the actual binary operands.

```
Reg 2  =  0000 0000 0100 1000 0111 1010 0000 0001

Reg 5  =  1111 1111 1111 1111 1010 1010 1010 1010
          ---------------------------------------
          0000 0000 0100 1000 0010 0100 1010 1011
           ↑    ↑    ↑    ↑    ↑    ↑    ↑    ↑
           0    0    4    8    2    4    A    B
```

In the preceding example, reg 2 contained a __________ (positive/negative) number and reg 5 contained a __________ (positive/negative) number.

---

positive
negative

As a result of adding the above numbers, the sum was ________ (positive/negative).

---

positive

Since the carry bit into the sign position agreed with the carry out of the sign position, a fixed point overflow ________ (would/would not) occur.

---

would not

After an "add" instruction, the condition code is set to indicate one of the four possible arithmetic results.

Indicate the condition code setting for each of the following arithmetic results.

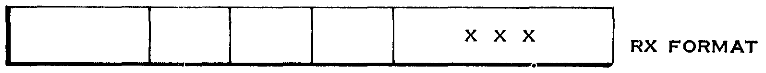
Result	Binary	Hex
Zero	______	________
⟨Zero or Negative	______	________
⟩Zero or Positive	______	________
Overflow	______	________

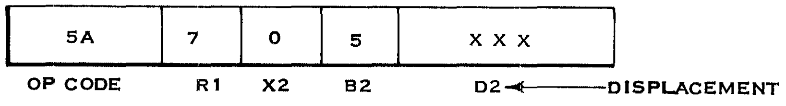
00	0
01	1
10	2
11	3

The mnemonic "A" is used to indicate a fullword add of storage to register. This instruction, whose Op code is a hex 5A, is of the ___ ___ format.

RX

Assuming that the base address is in reg 5 and that there is no index address, write the instruction that would add a fullword in storage to a fullword in reg 7.





Given the following, show the contents after execution of the preceding instruction.

	Before	After
Reg 0	E E E E E E E E	________
Reg 5	0 0 0 0 0 F 0 0	________
Reg 7	0 F 0 F 0 F 0 F	________
Storage	F F F F F F F F	________

Reg 0	E E E E E E E E
Reg 5	0 0 0 0 0 F 0 0
Reg 7	0 F 0 F 0 F 0 E
Storage	F F F F F F F F

The resulting sum which replaces the original operand in reg 7 can be determined either by converting the operands to binary and then adding, or simply by adding the hex numbers. Of course, as far as the System/360 is concerned, these are binary operands.

```
    Hex Addition                Binary Addition

   F F F F F F F F       1111 1111 1111 1111 1111 1111 1111 1111
+
   0 F 0 F 0 F 0 F       0000 1111 0000 1111 0000 1111 0000 1111
   ---------------       ---------------------------------------
   0 F 0 F 0 F 0 E       0000 1111 0000 1111 0000 1111 0000 1110
                           ↑    ↑    ↑    ↑    ↑    ↑    ↑    ↑
                           0    F    0    F    0    F    0    E
```

The preceding instruction ________ (did/did not) result in a fixed point overflow.

---

did not

The condition code setting as a result of the above instruction would be _____.

10; The final sum was positive or greater than zero.

Again, notice that even though the two operands were opposite in signs. there was no need to complement on this add instruction. This is because negative fixed point operands are already in their __________ form.

---

complement

Using the following instruction, show the contents of reg 7 and storage after instruction execution.

5A	7	0	2	X X X

	Before	After
Storage	F 0 F 0 F 0 F 0	________
Reg 7	F F F F F F F F	________

Storage	Unchanged
Reg 7	F0F0F0EF

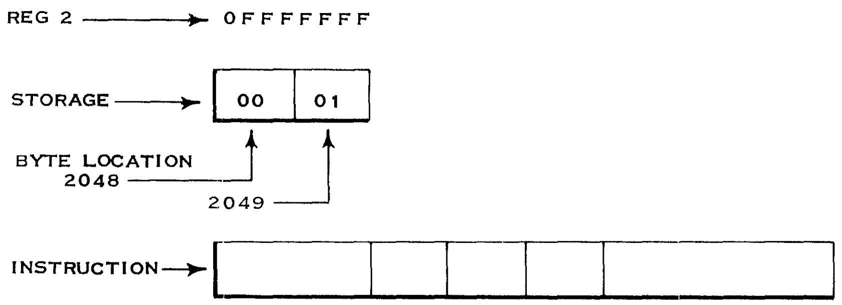
```
F F F F F F F F     1111 1111 1111 1111 1111 1111 1111  1111
F 0 F 0  F 0  F 0   1111 0000 1111 0000 1111 0000 1111  0000
F  0 F  0  F0  E F  1111 0000 1111 0000 1111 0000 1110  1111
```

Notice that the final sum was negative and as such is in "twos" complement form. The condition code setting will be ____.

01

If the operand in storage is a halfword, the Op code "4A" (mnemonic: AH) can be used. It also is of the RX format.

Write the instruction that will add the following binary operands. Assume reg 1 has a base address of 2048.



4A	2	0	1	0 0 0
OP CODE	R1	X2	B2	D2

In the add halfword instruction, the entire register contents are used. The halfword from storage is expanded to a fullword by propagating the sign bit to the left. The operands are then added and the result goes back into the register.

Show (in hex) the contents of reg 2 after adding the indicated halfword.

	Before	After
Reg 2	7 F F F F F F F	________
Storage	0 0 0 1	________

Reg 2 80000000
Storage 0001

In the preceding problem, the carry bit into the sign position does not agree with the carry bit out of the sign position. The condition code would be set to ____ indicating a _______ _______ _________.

11
fixed point overflow

Remember now, that in the AH instruction, only the storage operand is considered to be a halfword. It is expanded to a fullword by sign bit propagation before being added to the fullword in the register.

Because of the overflow in the previous "add halfword" instruction, a program interrupt might occur depending on the ________ ______ in the PSW.

---

program mask

In review then, there are three instructions to algebraically add binary operands. List their mnemonics.

__________ __________ __________

AR
A
AH

A mnemonic which ends in the letter R (such as AR) indicates an instruction of the ___ ___ format. If the mnemonic ends in the letter H (such as AH), it indicates that the second operand is a _________.

RR
halfword

The hexadecimal Op code for the mnemonic AR is 1A. Assuming that the 1st operand is in reg 0 and the 2nd operand is in reg 8, write the binary bit structure of the instruction that would add these.

____

0001 1010	0000	1000
Op Code	R1	R2

Given the following, what would be the contents of reg 0 after the instruction is executed? What would be the condition code?

Instruction
Mnemonic is AR

1A	0	8

Reg 0	Before	0 A 4 3 F 8 7 6
Reg 8	Before	0 0 0 3 2 1 F 9
Reg 0	After	____________
PSW Condition Code		____

Reg 0 = 0 A 4 7 1 A 6 F
Condition Code 1 0

The hexadecimal Op code for the mnemonic A is 5A. Assuming that the 1st operand is in reg 6 and that the 2nd operand has a displacement of zero, with a base address in reg 5 and no index factor, write the instruction (in hex) that would add these operands.

5A	6	0	5	0 0 0
MNEMONIC IS A	R1	X2	B2	D2

Referring to the preceding instruction and given the following, what will be the contents of reg 6 after the instruction is executed? What will be the condition code?

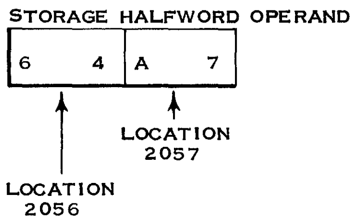
5A	6	0	5	0 0 0

1st Operand	Before	A 0 8 7 F A 7 6
2nd Operand	Before	0 7 4 A 0 2 3 7
Reg 6	After	____________
PSW Condition Code		____

Reg 6 = A 7 D 1 F C A D
Condition Code 01

To save main storage space, smaller binary numbers can be kept in main storage as halfwords. The mnemonic to add a halfword in storage to a fullword in a register is AH. The hexadecimal Op code is 4A.

Assuming that reg 12 has a base address of 2048, write (in hex) the instruction that will add the following halfword to the word in reg 5.



4A	5	0	C	0 0 8

BASE ADDRESS IS IN REG 12

Given the following, show the contents of reg 5 and the condition code after the instruction is executed.

4A	5	0	C	0 0 8

1st Operand	Before	0 7 4 A A 4 3 F
2nd Operand	Before	6 4 A 7
Reg 5	After	____________
PSW Condition Code		____

Reg 5 = 0 7 4 B 0 8 E 6
Condition Code 10

Just as there are three Op codes for algebraic addition of binary operands, there are three Op codes for algebraic subtraction.

Algebraic Subtraction

Mnemonic	Hex Op Code	Data Flow
SH	4 B	Halfword storage from register
S	5 B	Fullword storage from register
SR	1 B	Fullword register from register

Notice the similarities between the subtract Op codes above and the add Op codes below.

Algebraic Add

Mnemonic	Hex Op Code	Data Flow
AH	4 A	Halfword storage to register
A	5 A	Fullword storage to register
AR	1 A	Fullword register to register

"A" is the mnemonic for add while "S" is the mnemonic for __________.

A mnemonic ending in "H" (such as AH or SH) indicates that the second operand is a _________.

A mnemonic ending in "R" (such as AR or SR) indicates that the instruction is of the __ __ format.

---

subtract
halfword
RR

The four high-order bits of add and subtract Op codes are the same, assuming that the data flow concept and length of data are the same.

The four high-order bits of the SR instruction are 0001. The four high-order bits of the AR instruction are ______.

0001

The four low-order bits of an Op code indicate the specific operation such as add or subtract.

The four low-order bits of the SR instruction are ___________________ (the same as/different from) those of the AR instruction.

---

different from

Read the description of the following "subtract" instructions in your Principles of Operation manual. They will be found in the Fixed Point Arithmetic section. Do not read the description of the "subtract logical" instruction. It will be covered later.

Mnemonics
SR
S
SH

Write (in hex) the complete instruction that will subtract field B from field A. Both fields are binary operands. Field A is in register 0 and field B is in register 7.

1B	0	7
OP CODE	R1	R2

Notice that since we are subtracting field B (reg 7) from field A (reg 0), register 0 contains the 1st operand. Also note that register 0 can be used as an accumulator. As you have previously seen, register 0 could not be used as a base or an index register.

Because the preceding instruction says to subtract binary operands, the 2nd operand will be complemented and then ________ to the 1st operand.

---

added

1B	0	7

In the SR instruction above, the register that will have its contents complemented is register ___.

7

In the preceding instruction, the complementing of the 2nd operand (reg 7) during a binary subtract operation _______________ (does/does not) change the contents of the 2nd operand (reg 7).

---

does not; In other words, the 2nd operand will be brought out to the ALU without changing the register. In ALU, the 2nd operand is complemented and added to the 1st operand which has also been brought out to ALU. The resulting answer is then put back in the location of the 1st operand. The actual mechanics of how the ALU does the complementing or adding may vary from one model of System/360 to another. Such topics will not be covered here.

Given the following information, show the complement of the 2nd operand as well as the result that will replace the 1st operand.

Instruction

1B	4	6

Reg 4	0100 1000 0010 0001 0001 0010 0100 1000
Reg 6	0001 0010 0100 1000 1000 0100 0010 0001
Complement of 2nd operand	________________________________
Final Result in Reg 4	________________________________

1110 1101 1011 0111 0111 1011 1101 1111
0011 0101 1101 1000 1000 1110 0010 0111

In the preceding problem, there was a carry into the sign position and a carry out of it. Because of this, a fixed point overflow __________ (did/did not) occur.

---

did not

Because the sign bit of the final answer was 0, the condition code (bits 34-35 of the PSW) would be set to ____.

10; This indicates a positive result.

1B	6	6

The SR instruction will subtract the contents of one register from another. It can also be used to subtract the contents of a register from itself. In the instruction above, the contents of register 6 after instruction execution will be ________.

---

zero; The preceding instruction is a good example of how a register may be cleared out.

In the preceding example, the condition code was set to _______.

00

The SR instruction used the RR format. The S and SH instruction use the ___ ___ format. These S and SH instructions are identical to the A and AH instructions with the following exception. In the S and SH instructions, the 2nd operand (main storage) is added to the 1st operand after it (2nd operand) has been _________________.

RX
complemented

Just as in the A and AH instructions, the main storage operands specified by the S and SH instructions must reside on the correct fixed length boundaries. If not, a program interrupt will result and a ____________ exception will be indicated in the "_____" __________.

---

specification
"old" PSW

If the address of the main storage operand is not available on the particular System/360 (such as address 16,000 on an 8K machine), an <u>addressing</u> exception will cause a ____________ ____________.

---

program interrupt

The preceding types of program interrupts _______ (can/cannot) be masked.

---

cannot

If the carry out of the sign position does not agree with the carry into it in the preceding add and subtract instructions, there will be a _________ __________ ______________.

---

fixed point overflow	A fixed point overflow can cause a ________ ________. The program mask (bits 36-39) of the PSW can be used to prevent program interrupts caused by ______________________.

---

program interrupt fixed point overflow; Bit 36 of the PSW will prevent the interrupt if it contains a 0.	For the following mnemonics, indicate the instruction formats and the length of the 2nd operand.

Mnemonic	Format	Length of 2nd Operand
A R	______	________________
A	______	________________
A H	______	________________
S R	______	________________
S	______	________________
S H	______	________________

Mnemonic	Format	Length of 2nd Operand
A R	R R	Fullword
A	R X	Fullword
A H	R X	Halfword
S R	R R	Fullword
S	R X	Fullword
S H	R X	Halfword

In all the above instructions, the 1st operand is a word in length.

ADD AND SUBTRACT INSTRUCTIONS-LOGICAL

There are four more add and subtract instructions which are quite similar to the ones you have just studied. They are the "add logical" and "subtract logical" instructions. Before proceeding, read the descriptions of these instructions in your Principles of Operation manual. You will find the descriptions in the Fixed Point Arithmetic section.

Logical Add and Subtract

Mnemonic	Hex Op Code	Data Flow
A L	5 E	Fullword storage to register
A L R	1 E	Fullword register to register
S L	5 F	Fullword storage from register
S L R	1 F	Fullword register from register

To differentiate the "logical add/subtract" instructions from the "algebraic add/subtract" instructions, which you previously learned, the logical instructions include the letter ___ in their mnemonics.

L

Just like the algebraic instructions, the logical instructions denote the RR format by the ending letter of ____.

R

The length of both operands in the "logical add/subtract" instructions is always a __________. Therefore, these instructions do not use the mnemonic, ____.

---

fullword
H

AL =	5E	A =	5A
ALR =	1E	AR =	1A
SL =	5F	S =	5B
SLR =	1F	SR =	1B

The high-order four bits of the AL, ALR, SL, SLR instructions are the same as those of the A, AR, S and SR instructions, respectively. This is because the instruction formats and type of data (fullword binary) are the same. The low-order four bits are different, however, because the specific operations are different. The instruction AR calls for an algebraic add (signed numbers) while the ALR instruction calls for a logical add (unsigned numbers). Actually, the arithmetic results are the same for both algebraic add/subtract and logical add/subtract.

```
  Algebraic Add        Logical Add

   01101101             01101101
 + 00111000           + 00111000
   --------             --------
   10100101             10100101
```

Notice that the arithmetic results of the previous example are the same. The operands shown were 8 bits in length for purposes of simplicity. If the arithmetic results of algebraic and logical addition are the same, what is the difference between the two types of instructions? The difference is in the setting of the condition code (bits 34 and 35 of the PSW) and its meaning.

```
  Algebraic Add           Logical Add

   01101101                01101101
 + 00111000              + 00111000
   --------                --------
   10100101                10100101

PSW Condition Code = 11   PSW Condition Code = 01
```

In the preceding example of an algebraic add, a fixed point overflow resulted because of a carry into the sign position without a carry out of it. This overflow was indicated by a condition code of 11. A program interrupt might also occur depending on the program mask (bits 36-39) in the PSW. If the program mask prevents the interrupt, the next instruction could be a "branch on condition" instruction to test the conditio code.

In the case of the preceding logical add instruction, a fixed point overflow cannot possibly occur because there is no sign bit to consider. All that can be indicated is:

Condition Code	Meaning
00	No carry and zero result
01	No carry and a non-zero result
10	Carry and zero result
11	Carry and a non-zero result

Notice that (for the logical add/subtract instructions only) PSW bit 34 means a carry while bit 35 means non-zero.

```
Examples:   11                                00
            ↑↑                                ↑↑
   Carry ───┘└─── Non-Zero        No Carry ───┘└─── Zero
```

The resulting PSW condition code of the following "logical add" would be ____.

```
   10011100
 + 01001100
   --------
   11101000
```

---

01 The resulting PSW condition code of the following "logical add" would be ____.

```
   10010001
 + 11010001
   --------
```

---

11 In summary then:

1. The arithmetic results of "logical" and "algebraic" addition of binary operands are ____________ (identical/different).

2. The "logical add/subtract" instructions use _______ operands only.

3. A "logical add/subtract" instruction _________ (can/cannot) result in a fixed point overflow.

4. The "logical add/subtract" instructions use the letter ____ in their mnemonic.

5. The condition code settings and their meanings are as follows:

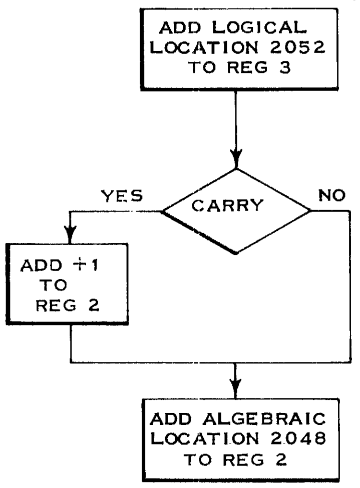
Condition Code	Algebraic	Logical
00	Zero Result	No carry, zero
01	Negative Result	No carry, non-zero
10	Positive Result	Carry, zero
11	Overflow	Carry, non-zero

identical
fullword
cannot
L

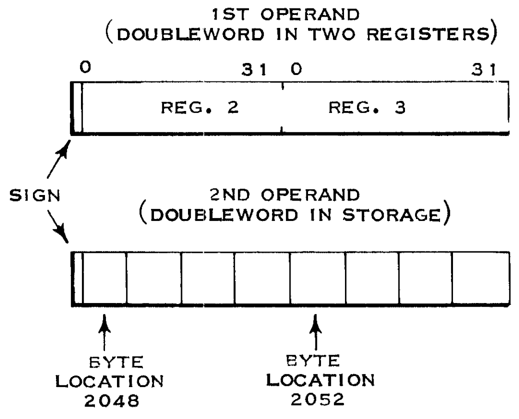
Before going on to more instructions, let's consider one use of the "logical add/subtract" instructions.

As you learned in the beginning of the "add" instruction section, only words and halfwords can be added. What happens when a programmer desires to add two doublewords? What he can do is place the high-order word of the 1st operand in one register and the low-order word in another. Then he can logically add the low-order word of the 2nd operand to the low-order word of the 1st operand. There is no fixed point overflow possible. He can then test the condition code for a carry. If a carry resulted, he can add a value of +1 to the high-order word of the 1st operand. In any case, the last step would be to algebraically add the high-order word of the 2nd operand to the high-order word of the 1st operand. The following flowchart and sample program should illustrate this more clearly.

FLOW CHART


ADD LOGICAL LOCATION 2052 TO REG 3
CARRY
YES
NO
ADD +1 TO REG 2
ADD ALGEBRAIC LOCATION 2048 TO REG 2

OPERANDS


1ST OPERAND (DOUBLEWORD IN TWO REGISTERS)
0
31
0
31
REG. 2
REG. 3
SIGN
2ND OPERAND (DOUBLEWORD IN STORAGE)
BYTE LOCATION 2048
BYTE LOCATION 2052

PROGRAM (IN HEX)

ASSUME:

A. REG 1 HAS BASE ADDRESS OF 2048
B. LOCATION 2056 HAS A HALFWORD CONTAINING A VALUE OF +1

5E	3	0	1	004	ADD LOGICAL LOCATION 2052 TO REG 3

07	C	4	BRANCH ON CONDITION. ASSUME R4 CONTAINS ADDRESS OF OP CODE 5A.

4A	2	0	1	008	ADD +1 TO REG 2

5A	2	0	1	000	ALGEBRAIC ADD LOCATION 2048 TO REG 2

LOAD INSTRUCTIONS

So far you have been adding and subtracting binary operands in the general registers. You have not seen how the data was originally placed in the registers. As you know, all input data must come into main storage before it can be processed. In turn, processed data must be in main storage before it can be sent to an output unit. As a result, there must be instructions to take data out of main storage and place it in a general register and later to put the processed binary data back in storage. These instructions are the "load and store" instructions. "Load" instructions put data in a register, while "store" instructions put data back in main storage.

There are three "load" instructions which do no more than place data in a general register. These instructions have no effect on the PSW condition code and do not change the 2nd operand. Read the descriptions of the following three "load" instructions in your Principles of Operation manual. You will find the descriptions in the Fixed Point Arithmetic section.

Mnemonic	Hex Op Code	Data Flow
LR	18	Fullword register to register
L	58	Fullword storage to register
LH	48	Halfword storage to register

A "load" operation is specified by the letter ___ in its mnemonic. Just like the "add/subtract" instructions, the mnemonics of the "load" instructions to denote RR format or halfword use ending letters of ___ or ___.

L
R
H

The condition code in the PSW __________ (is/is not) changed by the LR, L, or LH instructions.

---

is not

The L and LH instructions load a register with data from main storage. The LR instruction loads a register from a register. Write the instruction (in hex) that will load reg 1 from reg 5.

18	1	5

The LH instruction loads a halfword from storage into bits _____ through ____ of a general register.

16
31

As a result of the LH instruction, bits 0-15 of the register are ________ (changed/unchanged).

---

changed

The following halfword is placed in a register by use of the LH instruction. Show (in hex) the resulting contents of the register.

Storage	A 7 B 6
Register after execution of LH instruction	________________

F F F F A 7 B 6 ; The halfword is expanded to a fullword by propagating the sign bit to the left.

In the preceding example, the result in the register is a __________ (positive/negative) number.

Negative; As a reminder, don't forget that negative binary numbers are carried in their complement form.

The two programming errors that are possible when using the L and LH instructions are ____________ and ____________ exceptions.

---

specification
addressing

You have just studied the instructions used to load data into general registers. In addition to the three previously mentioned instructions (LR, L, LH), there are also several special purpose "load" instructions. They are special in the sense that they affect the condition code and may also change the data as it's loaded. Read the following descriptions in the Fixed Point Arithmetic section of your Principles of Operation manual.

Mnemonic	Hex Op Code	Data Flow
LTR	12	Load and test
LCR	13	Load complement
LPR	10	Load positive
LNR	11	Load negative

As indicated by the last letter of their mnemonics, the four instructions you just read about use the ___ ___ format. All four of these instructions can change the ________________ (data/condition code).

RR
condition code

The only difference between the LR instruction and the "load and test" (LTR) instruction is the effect on the PSW ___________ ______.

---

condition code; By specifying the same register in the R1 and R2 fields, the LTR instruction can be used to test the contents of a register.

The LCR instruction will change the condition code and will also _________ the data.

---

complement

With the LCR instruction, the condition code shows the status of the data _____ (before/after) it was complemented.

---

after

The LPR instruction only complements ________ (positive/negative) numbers.

---

negative; The LPR instruction Loads Positive numbers into the register regardless of the original sign of the numbers.

The LNR instruction complements ________ numbers.

---

positive; The LNR instruction Loads Negative numbers into the register regardless of the original sign of the numbers.

The only positive number that cannot be complemented by either the LCR or the LNR instruction is _____

---

zero

Given the following list of mnemonics, indicate the effect (changed/unchanged) on the condition code and on the data.

Mnemonic	PSW Condition Code	Data
LR		
L		
LH		
LTR		
LCR		
LPR		
LNR		

Mnemonic	PSW Condition Code	Data
LR	Unchanged	Unchanged
L	Unchanged	Unchanged
LH	Unchanged	Unchanged
LTR	Changed	Unchanged
LCR	Changed	* All data is complemented
LPR	Changed	Negative data is complemented
LNR	Changed	* Positive data is complemented

* With the Exception of Zero

STORE INSTRUCTIONS

Besides the ability to put data into the registers with the "load" instruction, System/360 also needs the ability to put data from the registers back into main storage.

This last type of operation is accomplished by a "_______" instruction.

"store"

Read the descriptions of the following "store" instructions in the Fixed Point Arithmetic section of your Principles of Operation manual.

Mnemonic	Hex Op Code	Data Flow
ST	50	Fullword, register to storage
STH	40	Halfword, low-order of register to storage

You have learned that, as a general rule, most instructions cause the results to replace the ______ (1st/2nd) operand. The "store" instructions are an exception to the preceding rule. In the ST and STH instructions, the ______ (1st/2nd) operand replaces the _____ (1st/2nd) operand.

1st
1st
2nd

In the case of the STH instruction, the 2nd operand in main storage is replaced by bits ____ through ____ of the general register.

16
31

Just like the L and LH instructions, the ST and STH instructions _______ ____________ (change/do not affect) the condition code.

---

do not affect

A programming error that can occur on a "store" instruction, but not on a "load" instruction, is called a ________ exception. It can occur on a "store" instruction when the storage key associated with the 2nd operand and the protection key in the PSW are __________ (different/alike).

---

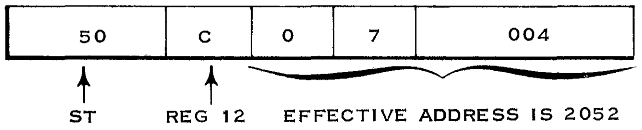
protection
different

If the protection key is zero, a protection exception ________ (can/ cannot) occur.

---

cannot

Write the instruction (in hex) that will store the contents of general register 12 in byte locations 2052-2055. Assume reg 7 contains a base address of 2048.



Notice again that storage addresses refer to the high-order (leftmost) byte location.

Two more instructions that you should learn now are the "load multiple" (LM) and "store multiple" (STM) instructions. You will find descriptions of these instructions in the Fixed Point Arithmetic section of your Principles of Operation manual. Read the descriptions and then continue with the following frames.

The LM and STM instructions are the first ones you have studied in this book that use the RS format. Label the fields of the RS format.

OP CODE	R1	R3	B2	D2

Like the L and ST instructions, the LM and STM ________________ (change/do not change) the condition code.

---

do not change

98	2	4	0	004

In the above LM instruction, byte locations 0004 thru 0015 will be loaded into register ___ through ___.

2
4

98	2	4	0	004

In the above LM instruction, register 3 will be loaded with the contents of byte locations ______ through ______.

0008
0011

90	0	F	0	000

In the above STM instruction, register 0 through ____ will be stored in byte locations 0000 through ______. (Assume the storage and protection keys match.)

15
0063

90	0	F	0	002

The above STM instruction will result in a _______________ exception.

---

specification; Address 0002 is okay for halfwords but not for fullwords. The LM and STM instructions use the entire contents (fullword) of the registers.

90	0	F	0	000

The above STM instruction will not result in a specification exception but may (depending on the keys) result in a ____________ exception.

---

protection

At this point, you should have the ability to load binary data into the registers, add and subtract this data, and store the resulting data back into main storage. Now let's forge ahead and see how this data can be multiplied and divided. The first instruction you will learn is "multiply halfword" (MH). Read its description in the Fixed Point Arithmetic section of your Principles of Operation manual and then continue with the following frames.

The "multiply halfword" instruction, like all instructions involving halfwords in main storage, has a mnemonic which ends with the letter ___.

H

In the MH instruction, the multiplicand is in a general register while a halfword in main storage is the ____________. The halfword from storage is expanded to a ___________ before the multiplication.

---

multiplier
fullword

In the MH instruction, the multiplicand is _____ bits long.

32; The entire register is multiplied by the multiplier. Normally, the register will only be holding a halfword and therefore the register's 16 high-order positions will not affect the product. The net result is that a halfword will be multiplied by a halfword.

Binary multiplication can be quite lengthy if done by hand. The following is an example of an 8-bit multiplicand being multiplied by a 4-bit multiplier.

```
          Binary
          01101011 ───► Multiplicand
    x         0111 ───► Multiplier
    ──────────────
          01101011  ⎫
         01101011   ⎬  Partial Products
        01101011    ⎪
       00000000     ⎭
    ──────────────
       01011101101 ───► Product
```

Judging from the preceding example you can see that binary multiplication is quite lengthy. If it is necessary to determine the results of a "multiply" instruction, you should convert the numbers to decimal and then multiply.

The MH instruction follows the rules of algebra. That is, if both operands are true numbers (plus) the product will be a ____________ (true/complement) number.

---

true

If both operands are complement numbers (negative), the product will be a ____________ (true/complement) number.

---

true; Multiplication of like signs always results in a positive answer. For example:
(+2) x (+7) = +14; (-2) x (-7) = +14

If multiplication of like signs results in a positive product, multiplication of unlike signs should result in a ____________ (positive/negative) product.

---

negative

If the multiplicand (1st operand) is a complement number and the multiplier (2nd operand) is a true number, the product of the MH instruction will be a ____________ (true/complement) number.

---

complement; Because of the unlike signs, the product will be negative.

As was previously stated, it is best to convert the operands to decimal numbers and then multiply if you are interested in determining the product. For instance: +2 times +7 = +14.

If the preceding were shown (for the sake of simplicity) as four-bit binary numbers, it would look like this:

```
(+7)            0111
(+2)       x    0010
           ---------
                0000
               0111
              0000
             0000
           ---------
(+14)        0001110
```

Supposing both operands were negative:

```
(-7)          1001 ────────► Twos complement of 7
(-2)      x   1110 ────────► Twos complement of 2
          ----------
              0000
             1001
            1001
           1001
          ----------
           1111110 ────────► Twos complement of 2
                             (Algebraically, this should be
                             positive number.)
```

(Frame continued on next page.)

As you can see, the binary multiplication of complement numbers does <u>not</u> produce the desired product. Obviously, the System/360 must do something internallv to the operands since the results of multiplication should be algebraically correct. The obvious solution would be to make the two negative operands true numbers and then multiply.

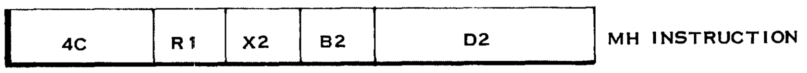
```
(-7)           1001   ------->         0111
(-2)        x  1110   ------->  x      0010
                                      -------
                                       0000
                                      0111
                                     0000
                                    0000
                                    -------
               (+14)  ------->      0001110
```

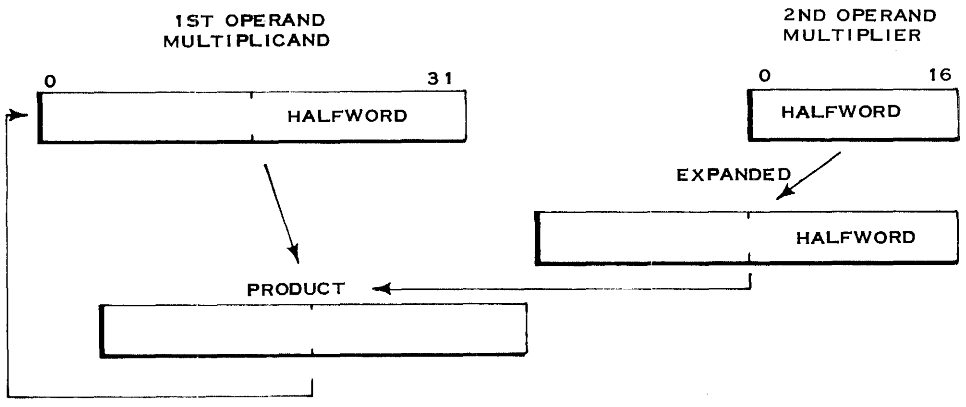
What if only one operand is negative? The solution again is to make it a true number, multiply and then complement the product.

```
  (-7)           1001   ------->         0111
x (+2)        x  0010   ------->  x      0010
                                       -------
                                         0000
                                        0111
                                       0000
  And then because the   \              0000
  original signs were     |           -------
  different, complement   >--.        0001110
  the product.            |   \
                         /     `--->  1110001
                                          + 1
                                      -------
  (-14) ------------------------->    1110010 ---> Twos complement
                                                   of 14
```

Another rule of multiplication is that the maximum number of significant bits in the product is equal to the total number of significant bits in multiplicand and multiplier.

```
For example:                0111 ---> 3 significant bits
                      x     0111 ---> 3 significant bits
                      ----------
                            0111
                           0111
                          0111
                         0000
                      ----------
                         0110001 ---> 6 significant bits
```





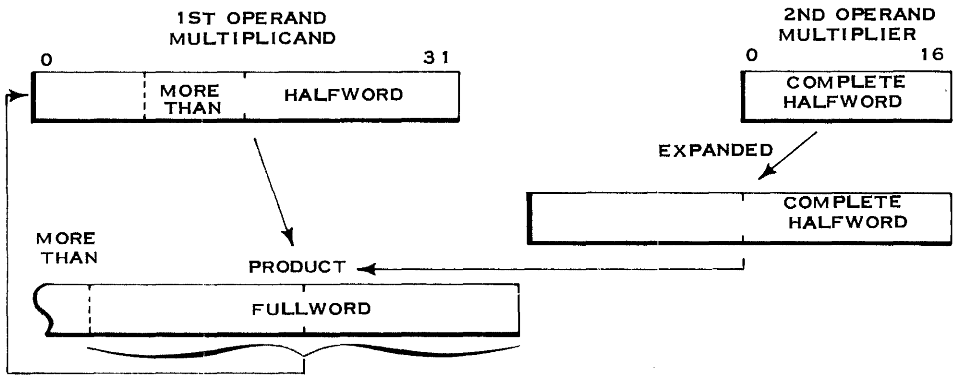
The example above shows that the MH instruction is normally used to multiply one h__________ (1st operand) by another h_________ (2nd operand). The maximum product that could result would be a f__________ and would replace the contents of the 1st operand r_________.

---

halfword
halfword
fullword
register

If the MH instruction is used with a multiplicand that has 15 significant bits and a multiplier that has 15 significant bits, the product will be in bits ___ through ____ of the register that previously held the multiplicand.

2
31



The example above shows that the 1st operand register contains more than a ___________. The 2nd operand contains a complete (16 significant bits) ___________.

If the MH (multiply halfword) instruction is used, the product will be more than ___ bits long. The entire product will not fit in the 1st operand r___________.

halfword halfword 32 register	In the preceding example, the resulting product in the 1st operand register contained only the ___ low-order bits of the actual product. The high-order bits of the actual product were ______.
32 lost	The preceding example __________ (is/is not) a normal application of the MH instruction.
is not	The product of the 32-bit multiplicand and the 16-bit multipler may exceed 32 bits but only the low-order ____ bits of the product replace the 1st operand.
32	Although the register containing the 1st operand may not contain the entire product, a fixed point overflow will not occur and the condition code remains ______________.
unchanged	If the register containing the multiplicand had been loaded with the LH instruction, the low-order 32 bits of the product ______ (will/ will not) contain all of the significant bits of the product.
will;	This is because a halfword contains only 15 integer bits. The maximum length of the product is equal to the total number of significant bits in the multiplier and multiplicand.
	In summary, the MH instruction multiplies the contents of a general register by a halfword from main storage. The low-order 32 bits of the product replace the multiplicand. No fixed point overflow is possible and the condition code remains unchanged.
	Let's now study two more binary multiply instructions. You will find the descriptions of the M and MR instructions in the Fixed Point Arithmetic section of your Principles of Operation manual.
	The mnemonic MR denotes a multiply instruction of the __ __ format. The instructions MH, M, and MR cause the ___ (1st/2nd) operand to be multipled by the ___ (1st/2nd) operand. The product of the MH, MR, or M instruction replaces the ___ (1st/2nd) operand.
RR 1st 2nd 1st	The R1 field in both the M and MR instructions must contain the address of an _____ (even/odd) numbered register. If the R1 field of an M or MR instruction has an odd address, a program interrupt will be caused by a __________ exception.
even specification	A specification exception of an M instruction can also be caused by a 2nd operand address that is not divisible by ____.

4; The 2nd operand is a fullword in storage.

Although the R1 field contains the address of an even-numbered register, the 1st operand (multiplicand) is actually in an _________ (even/odd) numbered register.

---

odd

If the R1 field of an M instruction contains a 4, the contents of register 4 are ignored and the multiplicand is brought out of register ____.

5

1C	4	7

In the above MR instruction, the multiplicand is in register ___ and the multiplier in register ___.

5
7

1C	4	4

In the above MR instruction, the multiplicand is in register ___ and the multiplier is in register ___.

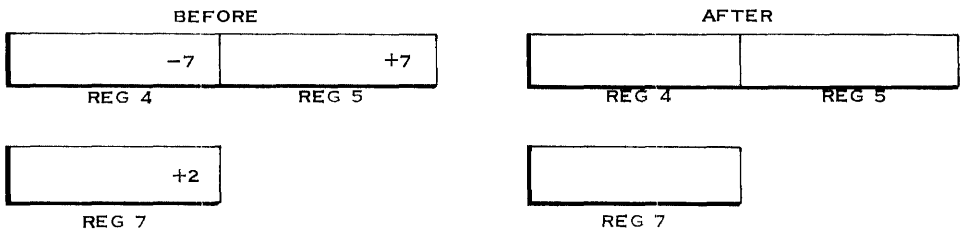
5
4

In the preceding MR instruction, both the multiplicand and the multiplier were wiped out by the product which is placed in register ___ and ___.

4
5

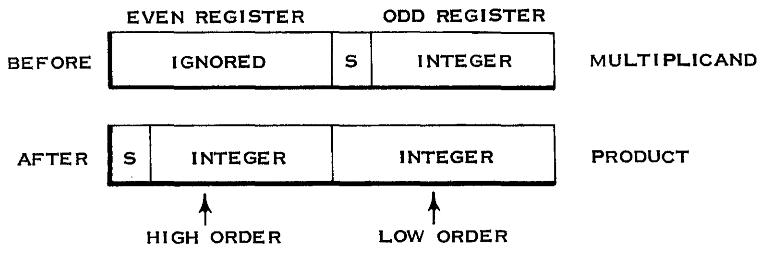
Show the register contents (expressed decimally) after the following MR instruction is executed.

1C	4	7



Reg 4 = Zero
Reg 5 = +14
Reg 7 = +2

Notice that in the preceding instruction, register 4 was zeroed out even though the product was small enough to be fitted into reg 5.



The example above shows that the product of an MR or M instruction is always developed as a doubleword with the high-order in the ________ register and the low-order in the ____ register.

DIVIDE INSTRUCTIONS

---

even
odd

Now that you have studied the instructions that multiply fixed length binary numbers, let's consider the instructions that will <u>divide</u> fixed length binary numbers. But first, let's review some information concerning division.

```
      120
12 ) 1440
```

The problem above shows a division of decimal numbers. The number 12 is called the __________ and the number 1440 is the __________. The answer is called the ____________.

---

divisor
dividend
quotient

```
12 )  1443
```

The divide problem above has a __________ of 120 and a __________ of 3.

---

quotient
remainder

The sign of the quotient follows the rules of algebra. If both the divisor and dividend have plus signs, the quotient will also have a ________ sign.

If both the divisor and dividend have negative signs, the quotient will have a _______ sign.

---

plus
plus

To illustrate the preceding rules, consider the following:

$$\begin{array}{r} +12 \\ -12\overline{)\,-144} \end{array}$$

To check, multiply the quotient and divisor.

$$+12 \times -12 = -144$$

If the divisor and dividend have opposite signs, the quotient will have a _________ sign.

---

minus

To illustrate the preceding rule, consider the following:

$$\begin{array}{r} -\ 12 \\ -12\overline{)\,+144} \end{array}$$

To check, multiply the quotient and divisor.

$$-12 \times -12 = +144$$

What about the sign of the remainder? By definition, the remainder is what is left from the dividend. As a result, the sign of the remainder should be ____________ (the same as/ different from) that of the dividend.

---

the same as

To illustrate the preceding rule, consider the following:

$$\begin{array}{r} +120 \\ -12\overline{)\,-1443} \end{array} \qquad \text{with a remainder of } -3$$

To check the above, multiply the quotient and divisor and add the remainder.

$$-12 \times +120 = -1440$$
$$-1440 + (-3) = -1443$$

Show the quotient and remainder.

$$-12\overline{)\,+1443}$$

$$\begin{array}{r@{\,}l}
 & -\ 120 \\
\cline{2-2}
-12) & +1443
\end{array}$$

with a remainder of +3

To check:

-12 x -120 = +1440
+1440 + (+3) = +1443

You are now ready to study the binary divide instructions. You will find a description of the D and DR instructions in the Fixed Point Arithmetic section of your Principles of Operation manual. Read the description and then continue with the following frames.

There are two binary divide instructions. Their mnemonics are ___ and ____.

D
DR; Notice that there is no DH instruction.

The R1 field of the D and DR instructions must contain the address of an ______ (odd/even) register or a program interrupt will be caused by a ____________ exception.

---

even
specification

The even-odd pair of registers addressed by the R1 field of the divide instruction contains a doubleword that is the _________ (divisor/dividend).

---

dividend

In the DR instruction, the R2 field has the address of the register containing the _________.

In the D instruction, the divisor is a word from _______ _____________.

---

divisor
main storage

The quotient and remainder from a D or DR instruction replaces the __________ (dividend/divisor).

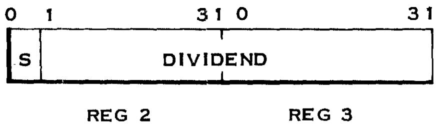
---

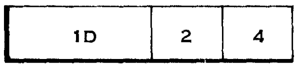
dividend

1D	2	4

In the above DR instruction, the dividend is in ______________________.

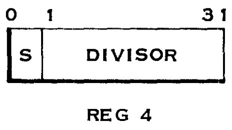
Registers 2 and 3 as shown below

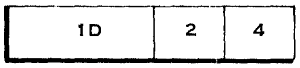




In the above DR instruction, the divisor is in ____________________.

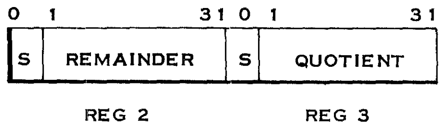
Register 4 as shown below



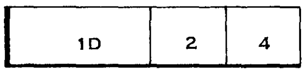


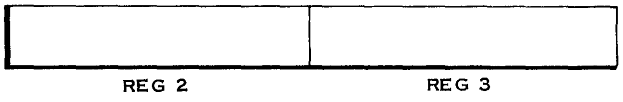
In the above instruction, the quotient will be in __________________ and the remainder will be in ____________.

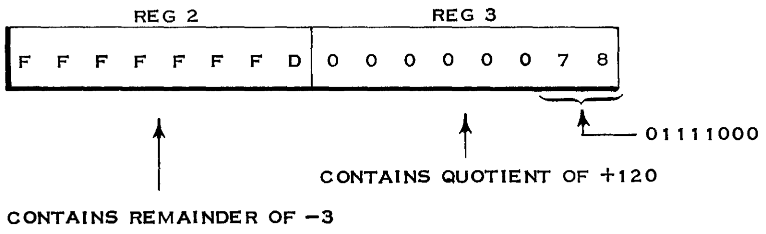
Reg 3
Reg 2 as shown below



Given the following DR instruction, show (in hex) the contents of registers 2 and 3 after the instruction has been executed. Assume the dividend is -1443 and the divisor is -12.







You have already learned some of the exceptions that can cause program interrupts. They are as follows:

1. Fixed point overflow
2. Specification
3. Addressing
4. Protection

An additional exception is fixed point divide. A fixed point divide occurs any time the quotient cannot be contained as a 32-bit signed integer.

When the divisor is zero, a program interrupt will be caused by a _______ __________ ___________ exception.

---

fixed point divide

No division takes place and the dividend is left undisturbed any time the System/360 recognizes a __________ ___________ __________ exception.

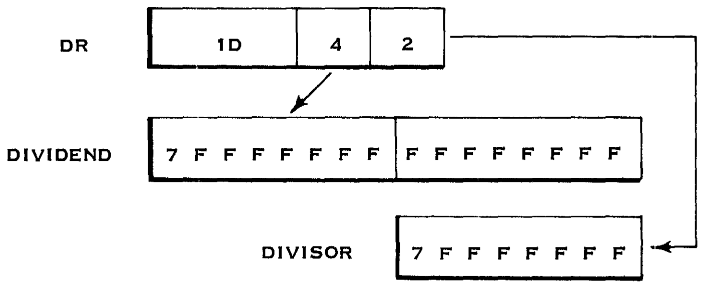
---

fixed point divide

The System/360 would recognize a divisor of ______ as a fixed point divide exception.

---

zero



In the above problem:

Will a fixed point divide be recognized? ______

Will the contents of registers 4 and 5 be changed? _____

Yes. The quotient cannot be contained in reg 5 because it is too large.
No. A fixed point divide exception will occur instead.

If that portion of the dividend that is in the even register is equal to or greater than the divisor, the system _______ (will/will not) recognize a fixed point divide exception.

COMPARE INSTRUCTIONS

---

will

You should now be in a position to load registers, do multiplication, division, addition, or subtraction, and store the results. You have two more types of instructions (compare and shift) to learn and then we will be able to see some programming examples. Let's examine the "compare" instructions first. You will find descriptions of the CR, C, and CH instructions in the Fixed Point Arithmetic section of your Principles of Operation manual. Read the descriptions and continue with the following frames.

To indicate a "compare" instruction, the mnemonic uses the letter ___. To compare a halfword in storage to the contents of a general register you would use the mnemonic ____. To compare the contents of one register to another, you would use the mnemonic ____.

C
CH
CR

The 1st and 2nd operands are ____________ (changed/unchanged) by the compare operation. The operation is used to set the PSW _________ ____________.

---

unchanged
condition code

A "compare" instruction would usually be followed by the instruction " __________ ____ __________."

"branch on condition"

If a compare operation shows that both operands are equal, the condition code would be set to _____.

00

A condition code of 01 indicates a low compare. In other words, the _______ (1st/2nd) operand is less than the ________ (1st/2nd) operand.

A condition code of 11 is impossible after a compare but a code of 10 would indicate that the ______ (1st/2nd) operand is high.

1st
2nd
1st

The comparison is algebraic. In other words, the operands are considered as signed integers. A negative operand would be _________ (less/greater) than a positive integer.

---

less

Given the following CR instruction, indicate the condition code setting.

19	4	7

Reg 4 A 0 F 1 0 F F F

Reg 7 7 F F F F F F F

PSW Condition Code ______

---

01 (or a hex 1);
The 1st operand (reg 4) is low because it is a negative number which is algebraically less than a positive number.

Given the following CH instruction, indicate the condition code setting.

49	4	0	1	00F

Reg 4 7 F F F 7 F 7 0

Main Storage 7 F F F

PSW Condition Code ______

---

10 (or a hex 2); The halfword is expanded to a fullword by sign propagation. Then the two fullword operands are algebraically compared.

So far you have studied most of the instructions in the Fixed Point Arithmetic section of your Principles of Operation manual. Shown below are most of these instructions with hex Op codes:

	Halfword (RX)	Fullword (RX)	Fullword (RR)
Load	48	58	18
Compare	49	59	19
Add	4A	5A	1A
Subtract	4B	5B	1B
Multiply	4C	5C	1C
Divide	None	5D	1D
Add Logical	None	5E	1E
Subtract Logical	None	5F	1F
Store	40	50	None

Notice!

1. The 1st hex digit for each column of instructions is the same. That is, all the halfword operations have the same 1st hex digit (4).

2. Each specific operation such as load, add and so forth, have the same last hex digit. That is, all the "multiply" instructions have an Op code ending in C.

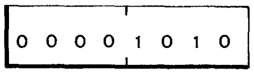
The preceding should agree with what you learned previously in the self-study book entitled "System/360 - Program Control and Execution." The Op code is summarized as follows:

```
0---------------7
|X X Y Y Z Z Z Z|  OP CODE

XX------  INSTRUCTION FORMAT (RR, RX AND SO FORTH)
YY------  TYPE OF DATA (HALFWORD, WORD AND SO FORTH)
ZZZZ----  SPECIFIC OPERATION (SUCH AS ADD, LOAD AND SO FORTH)
```

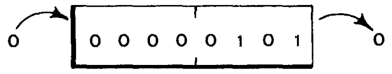
One other point to be made before continuing concerns the halfword operations. In all of the halfword instructions, with the exception of store halfword, the entire contents (fullword) of the register specified as the 1st operand is used. If this register had been initially loaded with a halfword and if all of the operations involving this register used the halfword instructions, bits 16-31 of the register would possibly contain all of the significant bits. Therefore the use of the "store halfword" instruction would store the entire accumulated data. When in doubt about the magnitude of the accumulated data, the "store" instruction should be used instead. This instruction would store the entire fullword contents of the register and four bytes of storage would be needed.

Let's now examine the "shift" instructions in System/360! The "shift" instructions only involve the general registers. Data in main storage cannot be shifted.

What do we mean by shifting? Shifting basically is moving the contents of the register to the right or to the left. For instance, assuming we have a theoretical 8-bit register, shifting would take place as follows:



If this register were shifted one place to the right it would look like this:

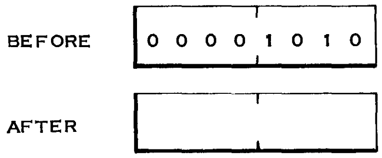


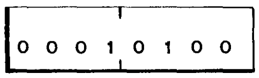
Notice that the low-order bit was shifted out. The resulting number (5) in the register is 1/2 the original number (10). Right shifting is similar to dividing by the powers of 2.

A right shift of two places is similar to dividing by 4; a right shift of three places is similar to dividing by ______ (6/8).

8

If the same theoretical 8-bit register shown below were shifted one place to the left, what would the resulting register look like?





Notice that the result (20) of the preceding problem is twice that of the original number (10). Left shifting is similar to ______________ by the powers of two.

---

multiplying

The System/360 can shift a register or a pair of registers either to the left or to the right. Furthermore, its "shift" instructions fall into two categories: algebraic and logical.

All of the "shift" instructions use the RS format. Label the fields of the RS format.

OP CODE	R1	R3	B2	D2

Read the descriptions of the following "algebraic shift" instructions in the Fixed Point Arithmetic section of your Principles of Operation manual.

Mnemonic	Hex Op Code	Data Flow
SLA	8B	Shift register to the left
SRA	8A	Shift register to the right

In the SLA instruction as in all "shift" instructions, the RS format is used but the ____ field is ignored. The register to be shifted by an SLA or SRA instruction is indicated by the ____ field.

R3
R1

The address generated by adding the base register contents and the displacement is used to ____________________ (address data/indicate number of shifts).

---

indicate number of shifts

The number of places to shift the register is indicated by the ________ low-order bits of the generated address.

---

six (6)

The maximum number of shifts is ____.

63; 111111 = 63

If the generated address is zero, the condition code will be set and the register ____________ (will/will not) be shifted.

---

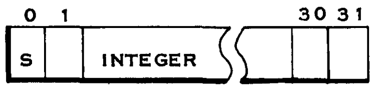
will not

The letter A in the mnemonics (SLA, SRA) indicates that the shift is ___________ (algebraic/logical). In an algebraic shift, the sign bit ________ (is/is not) shifted.

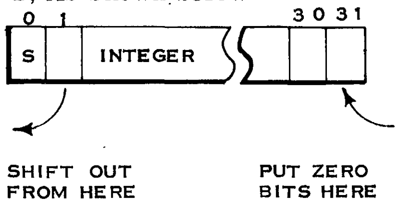
---

algebraic
is not

In the SLA instruction, the shifting is out of bit position ___ (0/1).



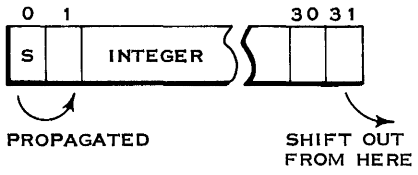
1; As shown below



In the SRA instruction, the sign bit is ____________ (shifted/propagated) to the right.

---

propagated; As shown below



Given the following SLA instruction, show (in hex) the contents of the shifted register.

8B	2	3	0	008

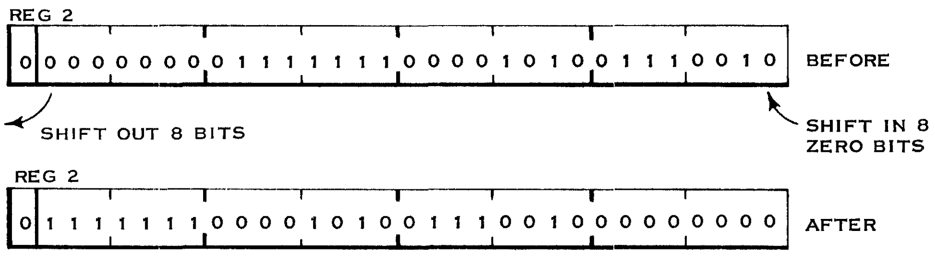
REG 2

0 0 7 F 0 A 7 2	BEFORE
	AFTER

---

7 F 0 A 7 2 0 0

The generated address was 0008. As a result, register 2 was shifted eight places to the left. Let's take a look at the preceding example again. This time, we will show the binary contents.



Notice that no significant bits were shifted out in the preceding example. If the register had been shifted 9 places, a significant bit would have been lost.

When a bit is shifted out (SLA only) that is different than the sign bit, a significant bit is lost. A _____ _________ __________ exception will result and a program interrupt may occur.

---

fixed point overflow; Notice that a program interrupt <u>may</u> occur. Remember that the fixed point overflow interrupt can be prevented by use of the program mask (bits 36-39 of PSW).

Given the following SLA instruction, show the contents of reg 2 in binary.

8B	2	3	0	F08

REG 2

1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1	BEFORE
	AFTER

1 0 1 0	0 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0	0 0 0 0

Even though the displacement was F08, the shift was only 8 places. Only the low-order six bits of the generated address determine the amount of shifting.

Did a fixed point overflow occur in the preceding example? ______

No; Since the original number was negative a fixed point overflow would be indicated by shifting out a 0 bit as opposed to a 1 bit for positive numbers.

Given the following SLA instruction, indicate the contents of the shifted register and the condition code.

8B	3	0	0	00F

REG 3

0 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0	BEFORE
								AFTER

	PSW CONDITION CODE

---

0 1 1 1	1 0 0 0	0 1 1 1	1 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

11	← FIXED POINT OVERFLOW

Notice that even though the fixed point overflow occurs with the 1st bit shifted, the entire shift of 15 places still occurs.

Let's move on to the "shift right algebraic" instruction.

Given the following SRA instruction, show the contents of the shifted register.

8A	3	2	0	00F

REG 3

1 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0	BEFORE
								AFTER

---

1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 1	1 1 1 0	0 0 0 1	1 1 1 0	0 0 0 1

Notice the propagation of the sign bit.

The condition code setting for the preceding problem would be ______.

01; This condition code reflects a negative result. Notice that a fixed point overflow cannot occur on a right shift operation no matter what bits are shifted.

Given the following SRA instruction, show the contents of the shifted register and the condition code.

8A	3	0	0	FFF

REG 3

0 1 1 0	1 1 1 1	1 0 1 0	0 0 1 1	1 1 1 0	0 0 0 1	1 1 1 1	0 0 0 1	BEFORE
								AFTER

	CONDITION CODE

0000 0000 0000 0000 0000 0000 0000 0000	REG 3

00	CONDITION CODE

Notice that a right shift of 31 or greater of a positive number will zero out a register, because the sign bit of 0 is propagated to the right.

Given the following SRA instruction, show the contents of the shifted register and the resulting condition code.

8A	4	7	0	FFF

REG 4

1000 0000 0000 0000 0000 0000 0000 0000	BEFORE
	AFTER

	CONDITION CODE

1111 1111 1111 1111 1111 1111 1111 1111	REG 4

01	CONDITION CODE

Notice that a right shift of 31 or greater of a negative number will result in a -1, because the sign bit of 1 is propagated to the right.

Besides shifting a single register, the System/360 also has the ability to shift a doubleword that resides in an even-odd pair of registers (remember the doubleword product as a result of a multiply). Read the description of the following instructions in the Fixed Point Arithmetic section of your Principles of Operation manual.

Mnemonic	Hex Op Codes	Data Flow
SLDA	8F	Shift double register to left
SRDA	8E	Shift double register to right

The SLDA and SRDA instructions are similar to the SLA and SRA instructions in that the ___ field is ignored.

The SLDA, SRDA, SLA, and SRA are also similar in that the number of shifts is determined by ______________________________.

R3
Only the low-order six bits of the generated address.

In both the SLDA and SRDA instructions, the R1 field must have the address of an __________________________.

---

even-numbered register

8F	3	4	0	001

The above SLDA instruction would result in a __________ exception.

---

specification;
Because the R1 field has an odd address.

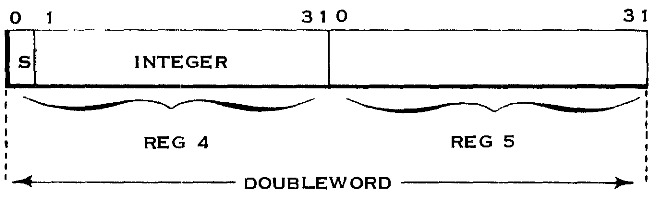
8F	4	3	0	006

In the above SLDA instruction, registers ___ and ___ will be shifted together.

4
5

In the preceding example the sign of the doubleword is in bit position ____ of register ___.

0
4 as shown below.



Given the following SLDA instruction, show (in hex) the contents of the shifted registers.

8F	4	3	0	010

REG 4	REG 5	
0 0 0 0 0 0 1 0	F 0 F 0 F F F F	BEFORE
		AFTER

REG 4	REG 5	
0 0 1 0 F 0 F 0	F F F F 0 0 0 0	A SHIFT OF 16 PLACES WAS SPECIFIED

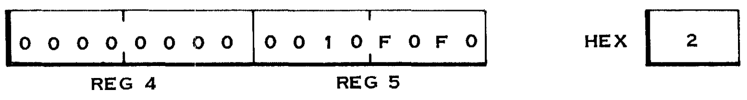
Given the following SRDA instruction, show (in hex) the contents of the shifted registers and the resulting condition code.

8E	4	3	0	010

REG 4	REG 5
0 0 0 0 0 0 1 0	F 0 F 0 F F F F

CONDITION CODE

HEX	



SHIFT INSTRUCTIONS - LOGICAL

You have finished the four "algebraic shift" instructions and are now ready to study the four "logical shift" instructions. The "logical shifts" differ from the "algebraic shifts" in that the entire register participates in the shift, the condition code is unchanged and a fixed point overflow cannot occur. You will find descriptions of the following "logical shift" instructions in the Logical Operations section of your Principles of Operation manual.

Mnemonic	Hex Op Code	Data Flow
SLL	89	Shift register left
SRL	88	Shift register right
SLDL	8D	Shift double reg left
SRDL	8C	Shift double reg right

Just like the "algebraic shifts," the "logical shift" instructions ignore the ___ field. The number of logical shifts taken is determined by the ________________________.

R3
Low-order six bits of the generated address.

Unlike the "algebraic shifts," the "logical shifts" _______ (do/do not) change the condition code.

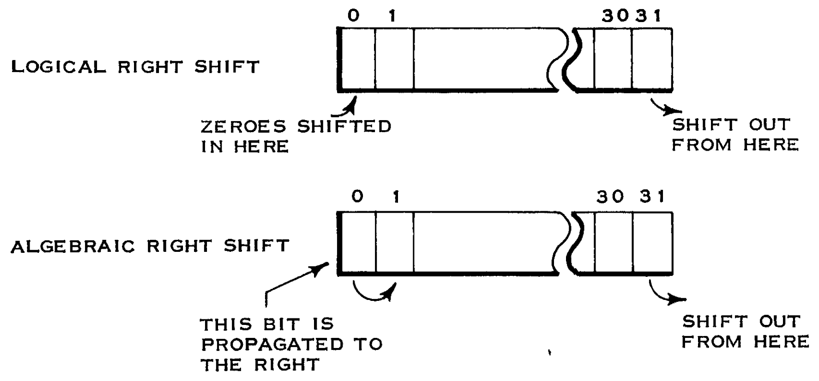
---

do not

In a "logical right shift," the sign bit is not propagated. Instead, it is shifted and zeroes are inserted in bit position ___.

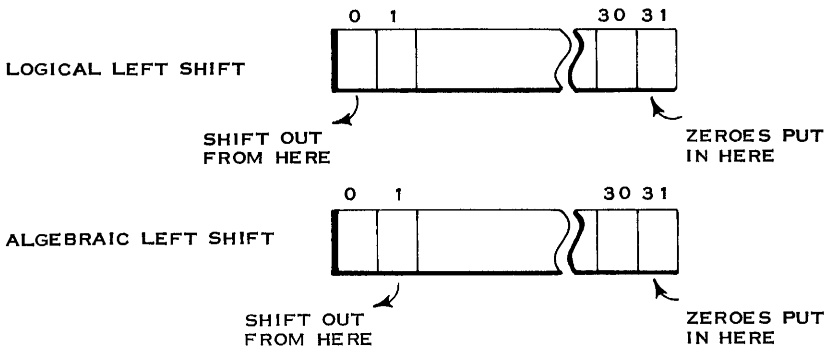
---

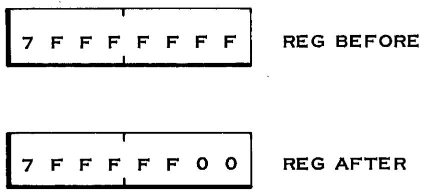
0 as shown below



In a "logical left shift" such as SLL, shifting is done out of bit position __ and zeroes are inserted into bit position ____.

0
31 as shown below





Which of the following mnemonics _______ (SLA, SRA, SLL, SRL) would have produced the results indicated above?

SLA; In this example, the condition code would have been set to 11 and a fixed point overflow occurs. If the SLL instruction had been used, shifting would have been done out of position 0 and the sign bit would have changed.

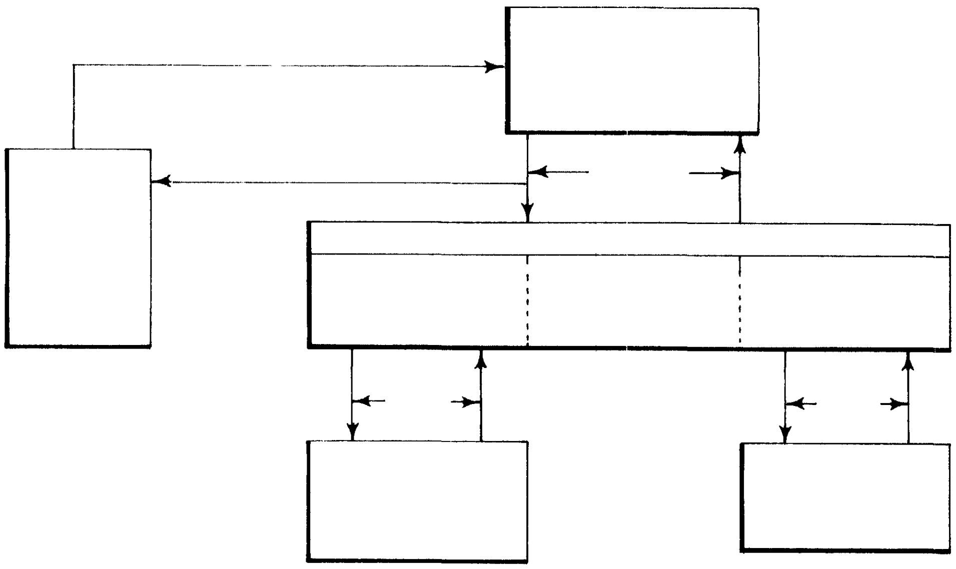
```
| A 0 0 0 F 0 0 0 |   REG BEFORE

| 0 F 0 0 0 0 0 0 |   REG AFTER
```

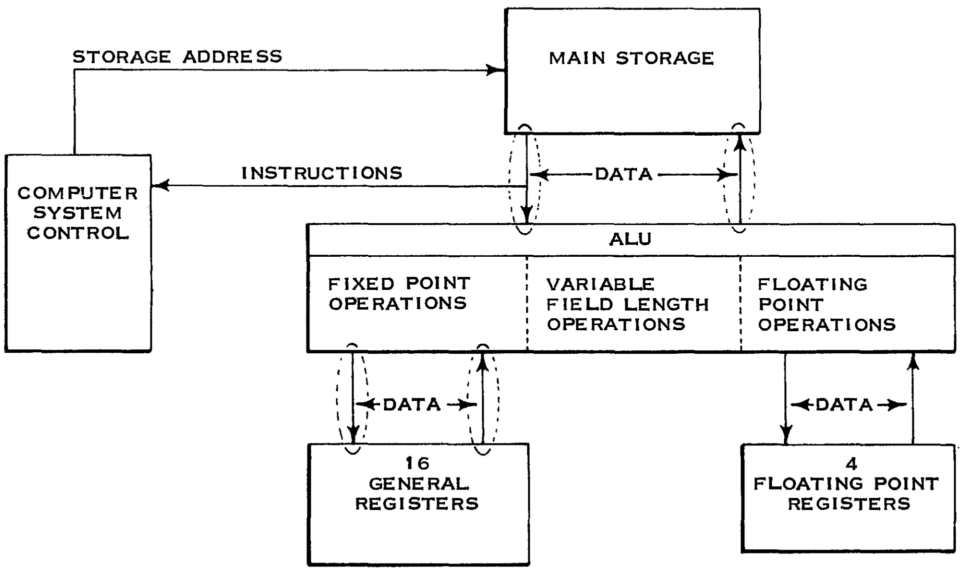
Which of the following mnemonics _____ (SLA, SRA, SLL, SRL) would have produced the results indicated above?

SLL; In this example, bit position 0 is changed.

Before going to the next section of this book, let's take a minute to review the fixed point operation's data flow.

Write in the names of the blocks and lines. Circle the lines upon which fixed point data will flow.




STORAGE ADDRESS
MAIN STORAGE
INSTRUCTIONS
DATA
COMPUTER SYSTEM CONTROL
ALU
FIXED POINT OPERATIONS
VARIABLE FIELD LENGTH OPERATIONS
FLOATING POINT OPERATIONS
DATA
DATA
16 GENERAL REGISTERS
4 FLOATING POINT REGISTERS

System/360 Fixed Point Binary Operations

SECTION IV LEARNING OBJECTIVES

At the end of this section, you should be able to use the Interruption Action chart to do the following:

Determine from the PSW interruption code, the fixed point programming exception that caused the interrupt.

Fixed Point Programming Exceptions

A programming error on the System/360 will result in a program interrupt. When the programming error is detected, the PSW is stored in byte locations 0040-0047. Once stored, this PSW is referred to as the "old" PSW. Just prior to storing this PSW, the exception code is placed in the interruption code portion (bits 16-31 of the PSW). There is an Interruption Action chart in the Interruption section and in the Appendix of your Principles of Operation manual which shows the code for the fifteen possible programming exceptions. Once the "old" PSW has been stored at location 0040, the doubleword in locations 104-111 (called the "new" PSW) is fetched and becomes the controlling ("current") PSW.

In going through the fixed point instructions, you have learned about various programming exceptions. The next few pages will be a summary of these programming exceptions. But first read the description of Fixed Point Arithmetic Exceptions in the Fixed Point Arithmetic section of your Principles of Operation manual. Use the Interruption Action chart as reference when you read the following frames.

When a program interrupt occurs, bits 16-31 of the "old" PSW receive the exception code. If the bits are coded as 00000000 00000100, a ____________ exception is indicated.

---

protection

A protection exception code indicates that the ________ key of a storage location does not match the ___________ key in the PSW.

---

storage
protection

Even though the two keys do not match, a protection exception will not occur if the protection key is ________.

---

zero

Of the instructions covered so far in this text, only the three "store" instructions (ST, STH, STM), the "convert to decimal," and the "pack" and "unpack" instructions can cause a protection exception. This is because these are the only instructions whose results are placed in main storage. The other instructions covered so far place the results in a general register.

If the interruption code in the "old" PSW (on a program interrupt) is coded as 00000000 00000101, an __________ exception is indicated.

---

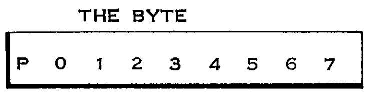
addressing	An addressing exception can occur when an instruction addresses a location of main storage that ____________________ ____________________ ____________________

Is not available on the particular System/360 installation. For instance, if a particular System/360 model 40 has a 64K main storage unit, an addressing exception will occur any time an address of 65, 536 or greater is used.

	The only instructions that cannot cause an addressing exception are those that do not address main storage, like the "shift" instructions and those of the __ __ format.
RR	If the interruption code in the "old" PSW (on a program interrupt) is coded as 00000000 00000110, a __________ exception is indicated.
specification	A specification exception occurs any time a fixed length operand in storage is addressed with an address that is not divisible by the number of ______ in the operand.
bytes	The address of a word operand in storage must be divisible by _____ or a ____________ exception is recognized.
four specification	On instructions (such as divide or multiply) in which a doubleword operand is located in a pair of adjacent registers, a specification exception will occur if the ____ (odd/even) register is addressed.
odd	If the interruption code in the "old" PSW (on a program interrupt) is coded as 00000000 00000111, a _____ exception is indicated.
data	A data exception indicates that a packed decimal operand contains invalid ______ or ______ codes.
digit sign (in either order)	Invalid digit codes are those in the range of _____ through ______.
1010 1111	Invalid sign codes are those in the range of ________ through ______.

0000 1001	Invalid sign codes are valid _____ codes. Invalid digit codes are valid _____ codes.
digit sign	The only instruction you have studied so far that can cause a data exception is the "________ ____ ________" instruction.
"convert to binary"	Sign and digit codes on a "pack" or "unpack" instruction __________ (are/are not) checked for validity.
are not	One point should be made absolutely clear at this time. Checking sign and digit codes for validity is not the same as checking a byte to ensure that the byte contains an odd number of bits set. Checking for an odd or even number of bits set in a byte is called p_______ checking.
parity	A parity error occurs whenever a byte has an ______ (odd/even) number of bits set. The parity error cannot be caused by programming. Therefore, a parity error will not cause a program interrupt. Parity errors cause ________ _________ interrupts.
even machine check	In order to have a parity checking function, every byte must consist of ________ data bits and one _______ bit.
eight parity	Normally, the parity bit is not shown with bytes. Whenever it is shown, it is the _________ (leftmost/rightmost) bit.

leftmost; As shown below.



Assume the following bytes are the low-order byte of the packed decimal operand on a CVB instruction. The parity bit is shown.

0 0 0 1 1 1 1 0 1	1 1 0 1 1 0 0 0 1	1 1 0 0 1 1 1 0 1
BYTE A	BYTE B	BYTE C

1. Which byte will cause a program interrupt? ___________
2. Which byte will not cause any interrupt? ___________
3. Which byte will cause a machine check interrupt? ______

---

1. B; because bits 0-3 (digit)and bits 4-7 (sign) are not valid codes.
2. A
3. C; because there is an even number of bits set to "1."

If the interruption code in the "old" PSW (on a program interrupt) is coded as 00000000 00001000, a ____________________________ exception is indicated.

---

fixed point overflow

A fixed point overflow can only cause a program interrupt when its corresponding mask bit in the PSW (bit 36) is set to ______ (zero/one).

---

one

On the "algebraic add/subtract" instructions, a fixed point overflow occurs whenever the carry into the sign position and the carry out of it _______ (do/do not) agree.

---

do not

Fixed point overflows cannot occur on the "________ (logical/algebraic) add/subtract" instructions.

---

logical

A fixed point overflow will occur on an "algebraic left shift" instruction whenever a bit shifted out of position 1 of the register is different from the _____ bit.

---

sign

Fixed point overflows cannot occur on the "______ (logical/algebraic) shift" instructions.

---

logical	Fixed point overflows can also occur on the "load positive" and "load complement" instructions. The overflows occur when, as a result of complementing, the carry into the sign position ______________________________.
does not agree with the carry out of the sign position.	The fixed point overflow mask bit in the PSW cannot prevent the overflow. It can only prevent the resulting program interrupt. Any time a fixed point overflow occurs, the condition code is set to _____.
11	If the interruption code in the "old" PSW (on a program interrupt) is coded as 00000000 00001001, a _______ _______ _______ exception is indicated.
fixed point divide	Division by zero will cause a _______ _______ _______ exception.
fixed point divide	A fixed point divide exception will also occur if the quotient cannot be contained within a __________ (halfword/word/doubleword).
word	A fixed point divide exception will also occur if the value of the packed decimal operand is too large to be contained as a binary word when using the instruction "c_______ to b_______."
"convert to binary"	You have just covered the fixed point instructions, including data conversions and possible programming exceptions. In the next and last section of binary operations, you will analyze a few programs using the fixed point instructions.

System/360 Fixed Point Binary Operations

	Section I:	Review of Data and Instruction Formats
	Section II:	Converting Data To/From Binary
	Section III:	Fixed Point Instructions
	Section IV:	Fixed Point Programming Exceptions
●	Section V:	Analyzing Fixed Point Programs

SECTION V LEARNING OBJECTIVES

At the end of this section, you should be able to use fixed point instructions to do the following:

Write programs, using stored data in any form (binary, zoned or packed decimal), to solve the following equations.

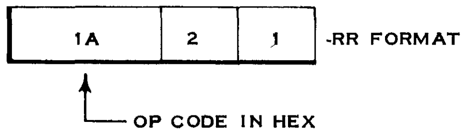
$$A + B = C$$
$$A + B - C = D$$
$$A \times B = C$$
$$A \div B = C$$
$$\frac{A \times B}{C} = D$$

Analyzing Fixed Point Programs

Notice: This section of the binary operations is very important. Your ability to learn the System/360 and ultimately, to service the system, will depend upon your understanding of the following material. The material will require much effort and concentration. Don't expect it to be easy. Use the Principles of Operation manual for reference and/or review whenever you are unsure of the details of a fixed point instruction.

Remember, now is the time and here is the place to learn.

To make the following programs easier to read, we are showing the instructions symbolically. The symbolic instruction format we will use will be similar to, but not necessarily identical to, the source language format required by the System/360 assembler program. For instance, to add the contents of register 1 to the contents of register 2, the following machine language instruction could be used.



Symbolically we will show this instruction like this:

```
AR          2, 1
```

Notice that the mnemonic of the instruction rather than its "hex" Op code will be used. The operand addresses will be separated by a comma and the 1st operand will be listed first.

Let's see if you understand the format we will be using. Write in this symbolic format the instruction that will subtract the contents of register 7 from the contents of register 5. ______________

SR 5, 7

Fine! Now how about symbolically expressing an RX format instruction. Supposing we wish to algebraically add the contents of a halfword from location 4096 to the contents of register 3.

Assuming that there is no indexing factor and that register 2 contains a base address of 2048, the machine language instruction would look like this:

4A	3	0	2	800	IN HEX
OP CODE	R1	X2	B2	D2	

Symbolically we will show this instruction like this:

$$\begin{tikzcd}
\text{AH} & 3, & 2048 & (0, & 2) \\
\text{Mnemonic} \arrow[u] & \text{R1} \arrow[u] & \text{D2} \arrow[u] & (\text{X2}, \arrow[u] & \text{B2}) \arrow[u]
\end{tikzcd}$$

Notice the use of the decimal number (2048) for the displacement rather than the machine language displacement (800). Also note that the X2 and B2 fields are in parentheses after the displacement.

Assuming that register 4 has a base address of 2048, write in the symbolic format the instruction that will subtract the halfword at location 5000 from the contents of register 5.

PROGRAM #1

SH 5, 2952 (0, 4)

Consider the following symbolic program:

```
LH     1, 2048  (0, 0)
AH     1, 2050  (0, 0)
STH    1, 2052  (0, 0)
```

Given the following data (shown in hex), show the contents of the storage area after execution of program #1.

STORAGE

00	4A	FF	FA	88	88	BEFORE

↑

BYTE
LOCATION 2048

						AFTER

00	4A	FF	FA	00	44

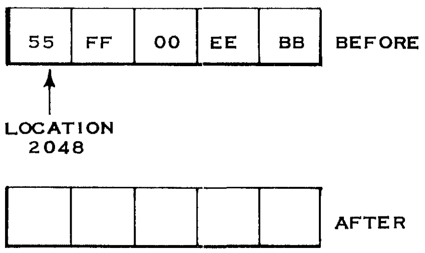
In the preceding program, the halfword from location 2048 (004A) was loaded into register 1. Then the halfword at location 2050 (FFFA) was added to it. The resulting answer was then stored as a halfword (0044) at location 2052.

PROGRAM #2

Consider the following program:

```
SR      1, 1
AH      1, 2048 (0, 1)
AR      1, 1
ST      1, 2048 (0, 0)
```

Given the following data in storage (shown in hex), show the storage contents after execution of program #2. If you have trouble analyzing program #2, continue on to the next frame and do the step-by-step analysis of the program.



00	00	AB	FE	BB

If you had the correct answer, you analyzed the program quite well. You may proceed to program #3 or you may continue with the following frames and review your solution. If you had the wrong answer, proceed with the following frames, which will analyze program #2 step-by-step.

The first instruction will cause the contents of register _____ to be subtracted from register ____.

1
1

Subtracting register 1 from itself will reduce its contents to ________.

---

zero

The second instruction will add a ________ (byte/halfword/word) from storage to register ___.

---

halfword 1	The effective address of the storage operand will be generated by adding the displacement of 2048 and the contents of register ___ which will be ______.

1
zero; Because register 0 was specified for the index register, it wasn't used in generating the storage address.

	The second instruction will add the halfword from byte locations ______ and ______ to register 1.

2048 2049	After the second instruction has been executed, register 1 will contain (in hex) __________.

000055FF	The third instruction will cause the contents of register ___ to be added to the contents of register ____.

---

1 1; In effect, this will double the contents of register 1.	After execution of the 3rd instruction, register 1 will contain (in hex) ___________.

0000ABFE as shown below

```
   000055FF     Reg 1
 + 000055FF     Reg 1
   0000ABFE
```

	The 4th instruction will cause the contents of register 1 to be stored in byte locations ______ through ______.

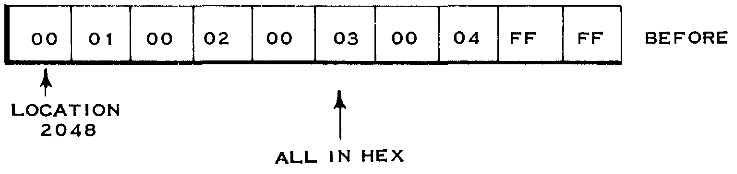
2048
2051; Notice that the 4th instruction was ST and not STH. This meant that the entire contents of the register was stored.

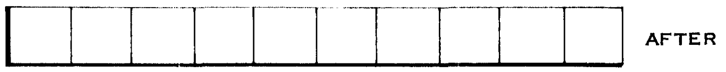
PROGRAM #3

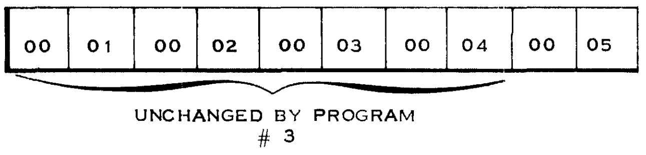
Consider the following program:

```
LH        3, 0 (0, 1)
AH        3, 2 (0, 1)
MH        3, 4 (0, 1)
SH        3, 6 (0, 1)
STH       3, 8 (0, 1)
```

Given the following data (shown in hex), and assuming register 1 contains 2048, show the storage contents after execution of program #3.







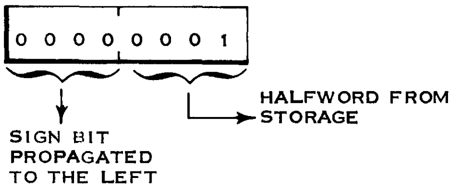
Notice the use of general register 3 in program #3 to accumulate the results of the program. The final result is then stored in main storage.

If you analyzed the program without difficulty and obtained the correct result, you may proceed to program #4. Otherwise, continue with the following step by step analysis of program #3.

The 1st instruction of program #3 loaded register 3 with the contents of byte locations _______ and ________.

2048
2049

Assume that prior to execution of the 1st instruction, register 3 contains FFFFFFFF. What will be the contents of register 3 after executing the instruction?



The 2nd instruction will cause the bytes at locations _____ and _____ to be added to the contents of register 3.

2050
2051

The contents of register 3 after executing the 2nd instruction will be (in hex) ________________.

00000003

Show the preceding answer as 32 binary bits.

____

00000000000000000000000000000011

The 3rd instruction will cause the contents in register 3 to be multiplied by the bytes in location ______ and ______.

2052
2053

After execution of the 3rd instruction register 3 will contain (in hex):

For your convenience, this is a repeat of Program #3.

```
LH      3, 1 (0, 1)
AH      3, 2 (0, 1)
MH      3, 4 (0, 1)
SH      3, 6 (0, 1)
STH     3, 8 (0, 1)
```

---

0 0 0 0	0 0 0 9

The binary multiplication is shown in the following example. The example uses only the four low-order bits as the remaining bits are zero anyway.

```
1st Operand ——>    0011
2nd Operand ——>  x 0011
                 ------
                   0011  \
                  0011    |  Partial Products
                 0000     |
                0000     /
                -------
                0001001 ——> Product
```

The 4th instruction will subtract the bytes at locations _____ and _____ from register 3.

2054
2055

After execution of the 4th instruction, register 3 will contain

---

0 0 0 0	0 0 0 5

The final instruction will cause bits ___ through ___ of register 3 to be stored in byte locations _____ and ______.

16
31
2056
2057

Consider the following program:

Assume register 9 contains the address 2048.

```
L       1, 0 (0, 9)
M       0, 4 (0, 9)
D       0, 8 (0, 9)
M       0, 12 (0, 9)
ST      1, 16 (0, 9)
```

Given the following data (shown in hex), show the storage contents after execution of program #4.

LOCATION	BEFORE				AFTER			
2048 →	00	00	00	04				
2052 →	00	00	00	02				
2056 →	00	00	00	07				
2060 →	00	00	00	10				
2064 →	11	00	00	F0				

Locations 2048-2063 are unchanged.
Locations 2064-2067 contain

00	00	00	10

If you analyzed the program without much difficulty and obtained the correct result, you may proceed to program #5. Otherwise <u>continue</u> with the following step-by-step analysis of program #4.

For your convenience, this is a repeat of Program #4.

```
L     1, 0    (0, 9)
M     0, 4    (0, 9)
D     0, 8    (0, 9)
M     0, 12   (0, 9)
ST    1, 16   (0, 9)
```

The first instruction of program #4 will cause register ___ to be loaded with a word from byte locations ______ through ______.

1
2048
2051

The condition code __________ (will/will not) be changed as a result of the first instruction.

---

will not

After execution of the first instruction register 1 will contain (in hex):

0000	0004

The second instruction of program #4 will multiply the contents of register ___ by the storage word in byte locations ______ through ______.

1
2052
2055

In the second instruction, the original contents of register 0 ____________ (are ignored/ should be zero).

---

are ignored

The product of the multiplication is developed as a doubleword. The high-order word is placed in register ___ with the low-order being placed in register ___.

0
1

The sign of the product (from the second instruction) is in bit position ___ of register ___.

0
0

After execution of the second instruction, the contents of register 0 and 1 will be: (indicate your answer in hex)

REG 0 REG 1

---

0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 8
REG 0	REG 1

As a result of the second instruction, the condition code ___________ (will/will not) be changed.

---

will not; The multiply and divide instructions do not change the condition code.

The third instruction will cause the doubleword in registers ___ and ___ to be divided.

0, 1; This doubleword is the dividend. It was the product of the previous multiply instruction.

The divisor for the third instruction comes from byte locations _____ through _____.

2056
2059

Since the third instruction has a dividend of +8 and a divisor of +7, there will be a quotient of ___ and a remainder of _____.

+1
+1

The quotient will be placed in register ____ and the remainder in register ___.

1
0

Show in hex the contents of registers 0 and 1 after executing the third instruction.

REG 0	REG 1

---

0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 1
REG 0	REG 1
REMAINDER	QUOTIENT

For your convenience, this is a repeat of Program #4.

```
L     1, 0     (0, 9)
M     0, 4     (0, 9)
D     0, 8     (0, 9)
M     0, 12    (0, 9)
ST    1, 16    (0, 9)
```

The remainder from the third instruction _______ (is/is not) ignored in executing the fourth instruction.

---

is; Only the contents of reg 1 (the previous quotient) are used as the multiplicand.

After executing the fourth instruction, the contents (in hex) of registers 0 and 1 will be:

REG 0	REG 1	
0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 1	BEFORE
		AFTER

---

0 0 0 0 0 0 0 0	0 0 0 0 0 0 1 0
REG 0	REG 1

The final instruction of program #4 will store the contents of register ___ in byte locations ______ through ______.

1
2064
2067

Did any of the instructions of program #4 change the condition code? ____

No

How many bytes of main storage were necessary to hold the five instructions of program #4? ___________

PROGRAM #5

---

20 bytes; Since all five instructions were of the RX format, each instruction was two half-words or 4 bytes in length.

Consider the following program.

Assume that the program begins at location 2048 and that general register 9 contains the base address of 2048.

```
LOCATION                      INSTRUCTION

  2048      L                  1, 256  (0, 9)
  2052      M                  0, 260  (0, 9)
  2056      LTR                0, 0
  2058      BC                 4, 18   (0, 9)          R1 field is the
  2062      BC                15, 22   (0, 9)          Mask Field
  2066      LCR                0, 0
  2068      LCR                1, 1
  2070      ST                 0, 264  (0, 9)
  2074      ST                 1, 268  (0, 9)
```

Which of the following statements is correct concerning the instruction at location 2056 (circle one):

a. This instruction does nothing useful.

b. This instruction will set the condition code according to the contents of register 0.

---

b; The purpose of the "load and test" instruction is to test the contents of a register.

The instruction at location 2058 will cause a "branch" only when the product of the previous multiply instruction is a ________ (positive/negative) number.

---

negative

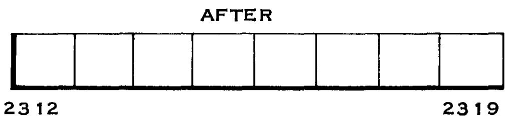
The instruction at location 2062 is a(n) ____________ (conditional/unconditional) "branch."

---

unconditional; 15 in the R1 field would be all bits (1111). This will always result in a "branch"

Given the following data in main storage show (in hex) the contents of locations 2312 through 2319 after program #5 is executed.

Location	Before
2304	0 0 0 0 0 0 0 1
2308	F F F F F F F F
2312	F F F F F F F F
2316	0 0 0 0 0 0 0 0



00	00	00	01	00	00	00	01
2312							2319

In the previous program problem, a value of +1 was multipled by a value of -1. The product would be -1. However, in the program, negative products were complemented prior to being stored. As a result, a value of +1 is stored in locations 2312-2315 and in locations 2316-2319.

PROGRAM #6

Consider the following program. Assume that register 1 has a base address of 2048.

```
Location 4096   LH       15, 0 (0, 1)
                SH       15, 2 (0, 1)
                CH       15, 4 (0, 1)
                BC     6, 2052 (0, 1)
                STH      15, 6 (0, 1)
```

Location	Data in Hex
2048	0 0 0 0
2050	F F F F
2052	0 0 1 0
2054	0 0 0 0

The subtract instruction will be executed: (Circle one of the following.)

a. Once
b. Seven times
c. Sixteen times
d. Fifteen times

c. Sixteen times

If you had the correct answer, you did fine. You may then proceed to program #7. If you didn't have the correct answer, continue with the following analysis of program #6.

	As a result of executing the first instruction of program #6, register 15 will be loaded with a value of ______.
zero	The second instruction will subtract a value of ____ from register 15.
-1; Same as hex FFFF.	After the second instruction is executed for the first time, register 15 will contain a value of _______.
+1; 0 - (-1) = +1	In effect then, the second instruction will cause a value of 1 to be ________________ (subtracted from/ added to) register 15.
added to	The third instruction will compare the contents of register 15 to a value of ______.
+16; Location 2052 has a hex 0010 which is a value of +16.	After the compare instruction has been executed the first time, the condition code will contain ____ (00/01/10/11).

01; As shown below.

<u>Condition Code After A Compare Operation</u>

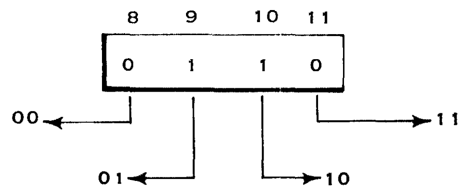
00	-	Equal
01	-	1st Operand is Low
10	-	1st Operand is High
11	-	Impossible after a Compare

The fourth instruction is a "branch on condition." The PSW condition code will be tested for which of the following settings: (Circle one or more.)

a. 00
b. 01
c. 10
d. 11

b, c; As shown below.

R1 field of "branch on condition" instruction



As a result of its R1 field, the fourth instruction is equivalent to a "branch unequal" instruction. A successful "branch" will be taken the first ____ (15/16) times this instruction is executed.

15

The sixteenth time that the fourth instruction is executed, a "branch" will not be taken because the condition code will contain ____ (00/01/10/11).

00

The sixteenth time through the program, the "store" instruction will be executed. At this time a value of _____ will be stored.

PROGRAM #7

+16; Hex 0010

This program will be written by you. Use only the instructions that you have learned so far. Don't hesitate to refer to the Principles of Operations manual.

A man borrows \$1,000 (A) from a bank. A 6% (B) service charge is added to the principle. The man agrees to pay off the debt with 12 monthly payments (C). What will his monthly payment (D) be.

Which of the following equations could be used to solve the above problem:

a. $\frac{AB + A}{C} = D$

b. $\frac{A \times B}{C} = D$

c. $\frac{A}{C} + AB = D$

---

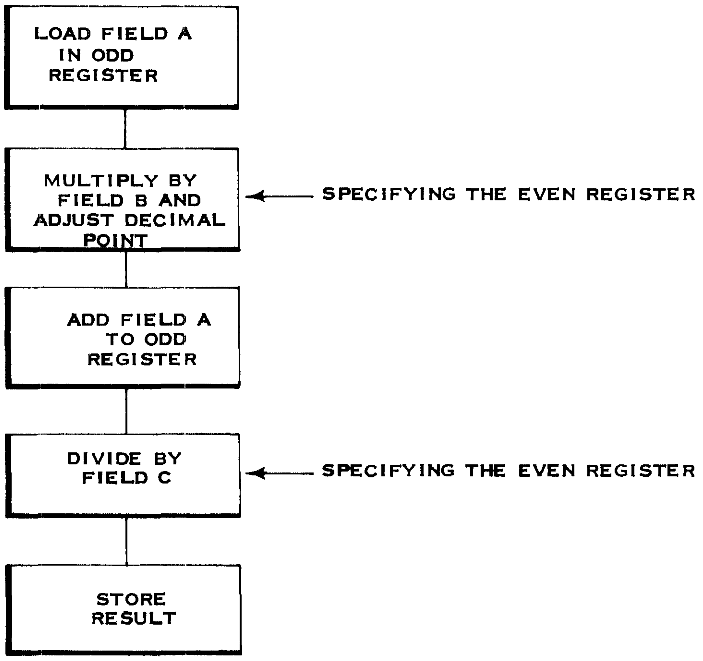
a. $\frac{AB + A}{C} = D$

Substituting the values given in the problem we have:

(1)	$\frac{\$1000 \times .06 + \$1000}{12}$	=	Monthly Payment
(2)	$\frac{\$60 + \$1000}{12}$	=	Monthly Payment
(3)	$\frac{\$1060}{12}$	=	Monthly Payment
(4)	\$88.33	=	Monthly Payment

Given the following data, draw a flowchart of the instructions necessary to solve the problem. Be sure to adjust for the decimal point after multiplication.

Field A	=	\$1000	Assume that these fields are full-word binary operands.
Field B	=	6%	
Field C	=	12	
Field D		Monthly Payment	


LOAD FIELD A IN ODD REGISTER
MULTIPLY BY FIELD B AND ADJUST DECIMAL POINT
SPECIFYING THE EVEN REGISTER
ADD FIELD A TO ODD REGISTER
DIVIDE BY FIELD C
SPECIFYING THE EVEN REGISTER
STORE RESULT

Write the necessary symbolic instructions to solve the problem. Use registers 0 and 1 for the accumulators. Assume register 8 has a base address of 2048. Note: Adjust decimal point by dividing by a +100.

Given:

LOCATION	OPERAND	COMMENT
2048	+100000	$1,000.00
2052	+6	6% (.06)
2056	+12	# of months
2060	Stored Result	$XX.XX
2064	+100	To adjust decimal point

```
L       1, 0    (0, 8)
M       0, 4    (0, 8)
D       0, 16   (0, 8)
A       1, 0    (0, 8)
D       0, 8    (0, 8)
ST      1, 12   (0, 8)
```

The reason the product was divided by +100 to adjust the decimal point is this: The decimal values are being carried as binary values with a base of two. Therefore, we can't adjust the decimal point by shifting the register.

Do you need a review? If you think that you may require a review of areas of this book, do the following:

> Read the learning objectives at the beginning of each section.
>
> You should review only those areas where you think that you cannot do what the objective indicates.

Starting on the next page is a self-evaluation quiz. It will allow you to check your overall understanding of fixed point instructions.

REVIEW QUESTIONS ON FIXED POINT BINARY OPERATION

- Use <u>only</u> the Appendix section of the Principles of Operation manual to answer these questions. When you are done, check your answers with the answers on page 113, and allow yourself five points for each correct answer. If your score is less than 80, review the areas of this text that correspond with the questions answered incorrectly.

1. Which of the following represents a decimal value of -26 as a half-word binary operand?

   a. 1000 0000 0010 0110
   b. 1111 1111 1101 1010
   c. 1000 0000 0001 1010
   d. 1111 1111 1110 0110
   e. None of the above

2. Which of the following instruction formats is used to add both half-word and word binary operands?

   a. RR
   b. RX
   c. RS
   d. SI
   e. SS

3. Columns 1 - 5 of an IBM card are punched 1, 2, 3, 4, and 5 respectively. It is desired to process this field as a binary word operand. Which of the following statements is true.

   a. The data field is automatically converted into a binary operand when read into storage. All that is necessary is to use the "load" instruction.
   b. The data field is read into storage as packed decimal data. The "convert to binary" instruction will change the data to the binary format and load the register.
   c. The data field is read into storage as zoned decimal data. The "convert to binary" instruction will change it to the binary format and load the register.
   d. The data field is read into storage as zoned decimal data. The "pack" instruction must be used to change it to packed decimal data. The "convert to binary" instruction can then be used to change it to the binary format and load the register.
   e. None of the above.

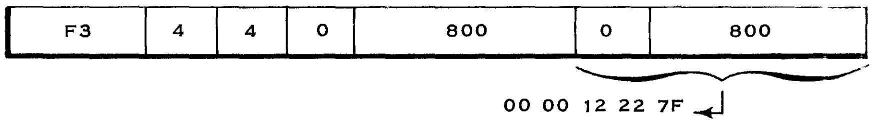
4. The "convert to decimal" instruction:

   a. Stores the contents of a register as packed decimal data into a variable length storage field.
   b. Stores the contents of a register as zoned decimal data into a fixed length storage field.
   c. Stores the contents of a register as packed decimal data into a fixed length storage field.
   d. Converts the contents of a register into packed decimal data and leaves this decimal data in the register.
   e. None of the above.

5. Which of the following programming exceptions is not possible on a "convert to binary" instruction?

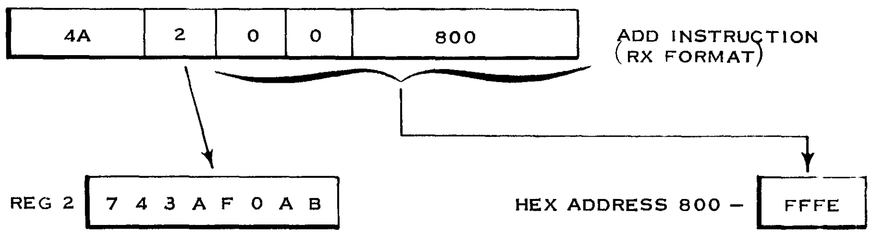
   a. Specification
   b. Addressing
   c. Data
   d. Protection
   e. None of the above

6. What is the result of the following "unpack" instruction?



   a. 01 02 02 02 F7
   b. F1 F2 F2 F2 F7
   c. F1 F2 F2 F2 7C
   d. F1 F2 F2 F2 C7
   e. None of the above.

7. Given the following fixed point "add" instruction:



   Which of the following would be the resulting contents of register 2?

   a. 7 4 3 B F 0 A 9
   b. 7 4 3 A F 0 A 9
   c. 7 4 3 A F 0 A D
   d. F F F F F 0 A 9
   e. None of the above.

8. Given the following fixed point "multiply" instruction, which of the statements is true?

```
MR INSTRUCTION  | 1C | 3 | 4 |

REG 3 ------->  0000000F
REG 4 ------->  00000000
```

   a. The "multiply" instruction will be executed and the product will be placed in registers 3 and 4.
   b. The "multiply" instruction will be executed and the product will be placed in registers 2 and 3.
   c. The "multiply" instruction will be executed and the product will be placed in register 3.
   d. The "multiply" instruction will not be executed. There will be a program interrupt because the multiplier is zero.
   e. The "multiply" instruction will not be executed. There will be a program interrupt because an odd register is being addressed as the multiplicand.

9. Which of the mnemonics represents the instruction that would cause the following result?

1st operand	-	0 0 F F E E A A
2nd operand	-	A A B B C C D D
Result	-	5 6 4 4 2 1 C D and a condition code of 10.

   a. AR
   b. AH
   c. ALR
   d. SR
   e. SLR

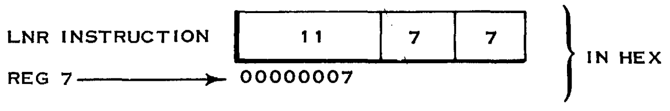
10. Which of the following statements is false?

   a. Binary operands must be converted to decimal to have a punched card output.
   b. On a fixed point "add" instruction, the signs are not analyzed. Instead, the operands are always added without complementing an operand.
   c. The arithmetic results (not including the condition code) of "algebraic add" and "logical add" operations are always the same.
   d. A fixed point overflow will not always cause an interrupt.
   e. None of the above.

11. Which of the following instructions (mnemonics) will not set the condition code?

   a. LR
   b. LTR
   c. LNR
   d. LPR
   e. LCR

12. Given the following LNR instruction, what will be the resulting contents of register 7?



a. 8 0 0 0 0 0 0 7
b. 0 0 0 0 0 0 0 7
c. F F F F F F F 9
d. F F F F F F F 8
e. None of the above.

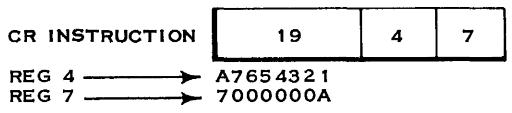
13. Which of the following is true concerning "compare" instructions?

a. The first operand is occassionally changed as a result of the comparisons.
b. The condition code is always set to one of three settings by the comparison.
c. An automatic branch will result when two operands compare equal.
d. All of the above.
e. None of the above.

14. Which of the following is true concerning the "algebraic compare" instructions (C, CH, CR)?

a. A positive operand is always higher than a negative operand.
b. If both operands are negative, the smaller absolute value is considered the higher operand.
c. A zero value always compares higher than a negative value.
d. All of the above.
e. a or c above.

15. Given the following CR instruction, what would be the resulting condition code?



a. 00
b. 01
c. 10
d. 11
e. None of the above.

16. Which of the following is true concerning the "shift" instructions?

    a. The "shift" instructions are used to adjust the decimal point of an operand.
    b. When shifting left, bit position 0 is always changed.
    c. When shifting right, bit position 0 is always propagated to the right.
    d. The number of places to be shifted is determined by the right-most bits of the generated address.
    e. All of the above.

17. Which of the following instructions would have produced the indicated result?

    Register before - A 0 F F F F F F
    Register after - F F F F F F 8 0

    a. "Shift left algebraic" seven places.
    b. "Shift left algebraic" eight places.
    c. "Shift left logical" seven places.
    d. "Shift left logical" eight places.
    e. None of the above.

18. Which of the following is true concerning the "store multiple" instruction shown below?

STM INSTRUCTION	90	7	6	0	800

    a. Registers 7 - 15 will be stored in that order.
    b. No registers will be stored because the R1 field is larger than the R3 field.
    c. Registers 6 - 15 will be stored in that order.
    d. Only registers 7 and 6 will be stored and in that order.
    e. Registers 7 - 15 and 0 - 6 will be stored in that order.

19. Which of the following programming exceptions can occur on a fixed point "add" instruction (RR format)?

    a. Specification
    b. Addressing
    c. Data
    d. Fixed Point Overflow
    e. Protection

20. Which of the following programming exceptions can be masked so that a program interrupt does not occur?

    a. Specification
    b. Addressing
    c. Data
    d. Fixed Point Overflow
    e. Protection

1.	d
2.	b
3.	d
4.	c
5.	d
6.	b
7.	b
8.	e
9.	d
10.	a
11.	a
12.	c
13.	b
14.	d
15.	b
16.	d
17.	a
18.	e
19.	d
20.	d

You have now finished the course on fixed point instructions. The next course will deal with the logical and decimal instructions. At that time, you will receive more programming problems.

Before proceeding to the next book of this System/360 Introductory Programming Course, fill out and return the Course Evaluation Sheet (located in the back of the book).

Alphabetical Index

Book 3 System/360 Fixed Point Binary Operations
Student Course Evaluation

You can make this course and all future courses more useful by answering the questions on both sides of this sheet and giving us your comments.

Do you feel that you have an adequate understanding of the learning objectives that are listed at the beginning of the following sections?

Section I:	Review of Data and Instruction Formats	Yes ☐	No ☐
Section II:	Converting Data To/From Binary	Yes ☐	No ☐
Section III:	Fixed Point Instructions	Yes ☐	No ☐
Section IV:	Fixed Point Programming Exceptions	Yes ☐	No ☐
Section V:	Analyzing Fixed Point Programs	Yes ☐	No ☐

List any technical errors you found in this book.

Comments

Please complete the information block on the opposite side. Thank you for your cooperation.
For form R23-2957-1

Field Engineering Education - Student Course Evaluation

IBM

Student Name	Man Number	B/O Number	Area Number

Student: Please review this evaluation with the person administering the course; then remove it from the book and send to the FE Education Center via IBM mail.

- Were you given a copy of this text to write in and keep? . . . . Yes ☐ No ☐
- How many hours per day were scheduled for this course? ________
- Were you interrupted during this time? . . . . Yes ☐ No ☐
- How many hours were needed to complete this course? ________
- Did you require assistance during this course? . . . . Yes ☐ No ☐
  (If your answer is yes, explain in the comments section)
- Indicate your understanding of the total course. Excellent ☐ Good ☐ Fair ☐ Poor ☐

To be completed by course administrator	Date
Reviewed by:	

To be completed by FE Education Planning	Date
Reviewed by:	

IBM Corporation
FE Education Planning
Department 911
South Road
Poughkeepsie, N. Y. 12602

TEAR HERE

FO

FOL

TEAR HERE

R23-2957-1

System/360 Intro-Book 3 Student Self-Study Course Printed in U.S.A. R23-2957-1

IBM

International Business Machines Corporation
Field Engineering Division
112 East Post Road, White Plains, N.Y. 10601