IBM

Field Engineering Education
Student Self-Study Course

SYSTEM/360

Introductory Programming
Book 4 – Branching, Logical and Decimal Operations

Preface

This is Book 4 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

Prerequisite

- Systems experience (1400 series with tapes, 7000 series with tapes) or a basic computer concepts course.
- Books 1, 2, and 3 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, R23-2958-1, is a minor revision of the preceding edition, but it does not obsolete R23-2958-0. 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 four sections to this text. At the beginning of each section, there 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 "branching, logical and decimal" instructions of the System/360. Some of these instructions are part of the Standard Instruction set and some are part of the Decimal Feature Instruction set.

System/360 Branching, Logical and Decimal Operations

- Section I: Branching Operations
  Section II: Logical Operations
  Section III: Decimal Operations
  Section IV: Analyzing Decimal Feature Programs

SECTION I LEARNING OBJECTIVES

At the end of this section, you should be able to do the following when given the mnemonic of any "branching" 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.

Branching Operations

The first section of this text deals with the "branch" instructions of the System/360. You have previously learned how the PSW is used to control the sequence of instruction fetching and execution. You have also learned how the normal sequence of instruction fetching can be changed by (a) an interrupt, (b) the "load PSW" instruction and, (c) a "branch" instruction. You have also studied one of the "branch" instructions: "branch on condition."

In the IBM System/360 Principles of Operations manual, briefly study the following areas of the Branching section.

Branching
- Normal Sequential Operation
- Branch On Condition

After completing these areas, you may proceed to the next frame.

The address of the next instruction to be fetched is contained in bits 40-63 of the ___ ___ ___.

PSW

After an instruction has been fetched, bits 0 and 1 of its Op code are used to update bits ___ through ___ of the PSW. Updating this instruction address portion (bits 40-63) of the PSW consists of increasing it by ___, ___ or ___.

40
63
2
4
6

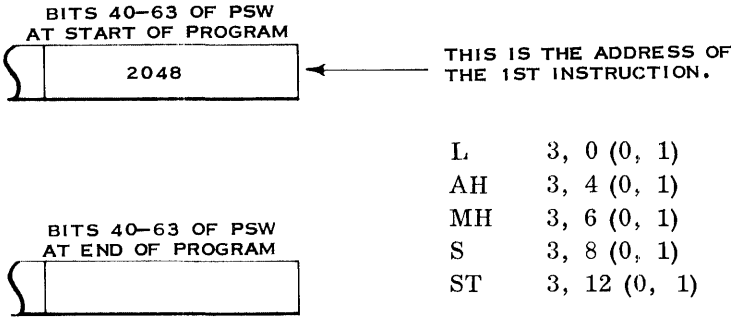
If bits 0 and 1 of the "current" instruction's Op code are 00 (RR format), the instruction address is increased by ___.

If bits 0 and 1 of the "current" instruction's Op code are 01 or 10, the instruction address is increased by ___.

If bits 0 and 1 of the "current" instruction's Op code are 11, the instruction address is increased by ___.

2
4
6

Given the following symbolic program, indicate (decimally) the contents of bits 40-63 of the PSW after the program is executed. (If necessary, use the Alphabetic List of Instructions in the Appendix of the Principles of Operation manual as a reference.)




```
L     3, 0 (0, 1)
AH    3, 4 (0, 1)
MH    3, 6 (0, 1)
S     3, 8 (0, 1)
ST    3, 12 (0, 1)
```

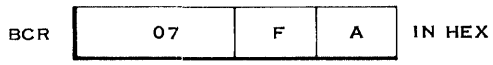
2068; Each instruction increased the address by 4.

The sequential manner of instruction fetching can be changed by means of a "branch" instruction. When a branch is taken, the address of the "branch to" location replaces _____.

BRANCH ON CONDITION INSTRUCTION - REVIEW

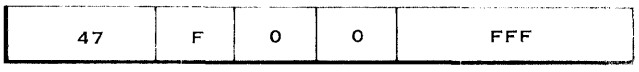
The instruction address (bits 40-63) in the PSW.

The second operand fields in all "branch" instructions indicate the "branch to" location. Given the following BCR instruction, bits 40-63 of the PSW will be replaced by bits 8-31 of register ____.



10

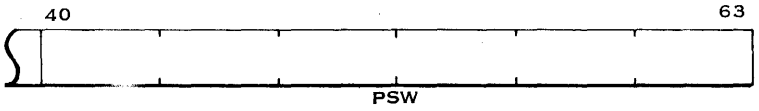
Given the following BC instruction, bits 40-63 of the PSW will be replaced by _____ (the effective generated address/ the contents of the storage area).



---

the effective generated address

Show the binary bit structure of bits 40-63 of the PSW after executing the previous instruction.



0000 0000 0000 1111 1111 1111

In the "branch on condition" instruction, the condition code is tested against the R1 field or (as it is sometimes referred to) the _____ field.

M1 or Mask

Each bit of the mask field (bits 8-11) is used to test for a specific setting of the PSW _____ _____.

---

condition code

Bit position 8 is used to test for a condition code setting of ____.

00

Bit position 9 tests for a condition code setting of ____.

01

Bit position 10 tests for a condition setting of ____ and bit position 11 is used to test for a condition code setting of ____.

10
11

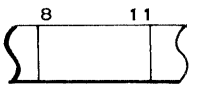
More than one condition code setting can be tested for at the same time by setting the appropriate bits of the mask field. Show the mask field bits that will test for a condition code setting of 10 or 11.

0011

Show the mask field bits that are necessary to branch on an equal or high indication after a "compare" instruction.

1010

The "branch on condition" instruction can be used as an "unconditional branch" instruction. Show the mask field bits that would accomplish this.



1111 (expressed hexadecimally as F)

The "branch on condition" instruction will never result in a branch if the mask field contains _____.

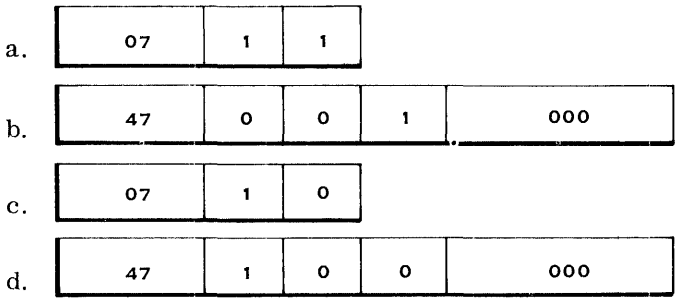
0000

If the R2 field of a BCR (not BC) instruction contains 000, a branch _____ (can/cannot) occur.

---

cannot

Which of the following "branch" instructions will not result in a branch? (Circle one or more.)



---

b. because the mask field is zero
c. because the R2 field is zero

Answers a and d above will result in a branch if the condition code is 11.

BRANCH AND LINK INSTRUCTION

Let's continue now and study another "branch" instruction. Read the description of the "branch and link" instruction in the Branching section of your Principles of Operation manual.

The mnemonic for the "branch and link" instruction is _____.

The "branch and link" instruction can be either of the RR format or the RX format. The mnemonic used for the RR "branch and link" is _____.

BAL
BALR

The BAL instruction always results in a branch. The BALR instruction will not result in a branch if the R2 field is _____.

---

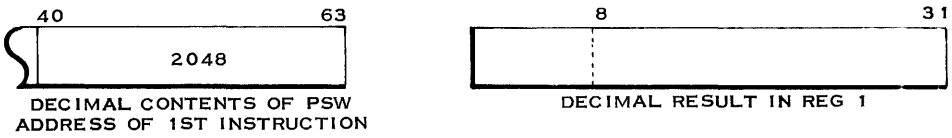
zero

In all cases (even when the R2 field is zero), bits 32-63 of the PSW are stored in the register specified by the _____ field.

R1

The address that is stored by a "branch and link" instruction is the address of the instruction that would have been executed if the branch were not taken.

Given the following symbolic program, show what bits 8-31 of register 1 will contain after the BALR instruction is executed.



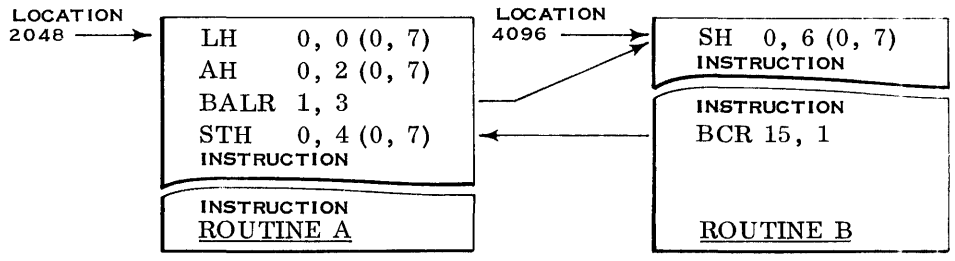

```
LH      0, 0 (0, 7)
AH      0, 2 (0, 7)
BALR    1, 3
STH     0, 4 (0, 7)
```

Reg 1 = 2058

Write the mnemonic of the instruction whose address was stored in register 1 in the previous problem. _____

STH

The reason for storing the address of the next sequential instruction during a "branch and link" operation is to provide a linkage between routines. This is illustrated as follows:



As you can see above, the "branch and link" instruction will:

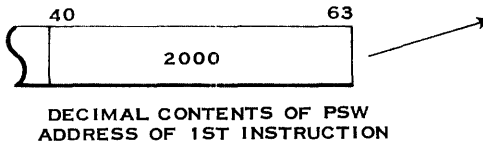
1. Cause the address of the STH instruction of routine A to be stored in register 1.
2. A branch will be taken to the SH instruction in routine B (as specified by the contents of register 3).
3. The last instruction in routine B is an unconditional branch (because the mask field contains 15) back to the STH instruction in routine A.
4. The address of the STH instruction was obtained from register 1 where it was stored from the preceding BALR instruction.

```
BALR    1, 0
AR      2, 2
```

In the above program, the BALR instruction will cause bits 32-63 of the PSW to be stored in register ___, and a branch _____ (is/is not) taken.

1
is not

The BALR instruction, with an R2 field of zero, may be used to load a base address into a general register. Examine the following program; then read the following frames.




```
BALR    12, 0
LH      1,  1000 (0, 12)
AH      1,  1002 (0, 12)
SH      1,  1004 (0, 12)
STH     1,  1006 (0, 12)
BCR     15, 12
```

Assuming that 2000 is the address of the BALR instruction, what address will be placed in register 12? _____

2002; Note that only bits 40-63 of the PSW are referred to in our discussions. Although bits 32-39 are also placed in register 12, they are ignored in generating addresses later on. Only bits 8-31 of a register are used in address generation.

Because the R2 field of the BALR instruction is zero, a branch _____ (will/will not) be taken.

---

will not

The second operand of the LH instruction will have a base address of _____ and a displacement of _____.

2002
1000

The second operands of the second through the fifth instructions will all have a base address of _____.

2002

The last instruction (BCR) will cause an "unconditional branch to" location _____.

2002

Write the mnemonic of the instruction at location 2002. _____

LH; Actually this program will never end because of the "unconditional branch back to" the LH instruction.

Thus far you have seen two main uses for the "branch and link" instruction. It can be used to:

1. Branch to some routine and automatically provide the programmer with an _____ to branch back to. Hence, the name of the instruction: "branch and link."

2. Provide the programmer with a way to load his initial base address into a g_____ r_____.

BRANCH ON COUNT INSTRUCTION

address
general register

The next "branch" instruction you will study is used to control the number of times a program loop is executed. Read the description of the "branch on count" instructions in the Branching section of your Principles of Operation manual.

Just like the "branch on condition" and the "branch and link" instructions, the "branch on count" instruction can be in two formats. List them:

___ ___, ___ ___.

RR
RX

BCT is the mnemonic for the RX format of "branch on count." The mnemonic for the RR format is _____.

BCTR

Like the BCR and BALR instructions, the BCTR instruction will not result in a branch if the R2 field contains _____.

---

zero

The "branch on count" instruction (either BCT or BCTR) will always reduce the 1st operand by a value of _____.

---

one

The "branch on count" instruction will result in a branch if the 1st operand _____ (has/has not) been reduced to zero.

---

has not

The reduction of the 1st operand occurs _____ (before/after) deciding whether to branch.

---

before

06	7	3	IN HEX

Assuming that register 7 contains a value of +1, the above "branch on count" instruction _____ (will/will not) result in a branch.

---

will not; This is because the BCT instruction will reduce register 7 by 1 before deciding whether or not to branch. This will bring the contents of register 7 to zero.

Assuming that register 7 contains a value of zero, the above "branch on count" instruction _____ (will/will not) result in a branch.

---

will; Since the register is reduced by 1 before testing for a branch, register 7 was reduced to a value of -1 and the branch did occur. In this case, the preceding instruction would have to be executed 2^{32} times before register 7 could be reduced to zero.

Examine the following program.

NOTE: ASSUME THAT BITS 32–39 OF THE PSW ARE ZERO.

```
BALR    11, 0
BALR    12, 0
LH      1,  1000 (0, 12)
AH      1,  1002 (0, 12)
SH      1,  1004 (0, 12)
BCTR    11, 12
```

Assuming that 2000 is the address of the first instruction, how many times will the above program be executed? Circle one of the following:

a. 2000
b. 4000
c. 2002
d. 2004
e. None of the above

---

c; A value of 2002 was placed in register 11 by the first instruction. Each time the last instruction (BCTR) was executed, this value was reduced by 1 and a branch was taken back to the second instruction. On the 2002nd time through the program, the contents of register 11 was reduced to zero by the BCTR instruction and a branch would not occur.

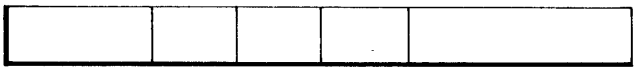
BRANCH ON INDEX HIGH INSTRUCTION

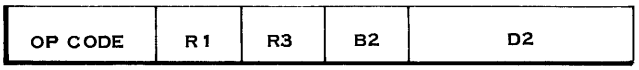
The next "branch" instruction you will learn is "branch on index high." This is a complex instruction, so take your time. Read the description of this instruction in the Branching section of your Principles of Operation manual.

The "branch on index high" instruction has a mnemonic of _____.

BXH

The BXH instruction uses the RS format. Label the fields of the RS format.





As with the other "branch" instructions you have learned, the generated storage address (B2 and D2 fields) is the _____.

"branch to" location

The R1 field in the BXH instruction is the address of the _____ operand.

Normally the number in an instruction field specifies which operand it is. For example, R1 specifies the 1st operand. However, in the case of the BXH instruction, the R3 field is used to specify the _____ operand.

---

first
second

The second operand is the R3 field register.

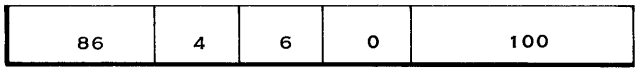
The third operand of a BXH instruction is also in a register. If the R3 field is even, the third operand is in the next odd-numbered register. That is, if the R3 field is 4, the second operand is in register ___ and the third operand is in register ___.

4
5

If the R3 field of a BXH instruction is odd, the second and third operands are in the same register. That is, if the R3 field is 5, the second operand is in register ___ and the third operand is also in register ___.

5
5

Given the following "branch on index high" instruction, indicate the locations of the three operands.



1st operand is in register ____.
2nd operand is in register ____.
3rd operand is in register ____.

4
6
7

Given the following "branch on index high" instruction, indicate the locations of the three operands.

86	3	5	0	100

1st operand is in register ____.
2nd operand is in register ____.
3rd operand is in register ____.

3
5
5

86	7	4	0	100

In the BXH instruction, the second operand is added to the 1st operand and the sum is algebraically compared to the 3rd operand. Given the above instruction, register ____ will be added to register ____ and the sum will be compared algebraically to register ____.

4
7
5

In the BXH instruction, the resulting sum replaces the first operand after being compared with the _____ (1st/2nd/3rd) operand.

3rd

Regardless of whether a branch does or does not occur, the sum of the 1st and 2nd operands always replaces the _____ (1st/2nd/3rd) operand.

1st

Given the following, indicate (in hex) the contents of the registers after execution of the BXH instruction.

86	4	6	0	100

Everything in hex

	Before	After
Register 4	00000010	_____
Register 6	FFFFFFFF	_____
Register 7	00000008	_____

Register 4 0000000F
Register 6 Unchanged
Register 7 Unchanged

In the preceding problem, a value of -1 was added to a value of +16 and the sum of +15 replaced the 1st operand.

The sum of the 1st and 2nd operands is algebraically compared with the _____ operand.

3rd

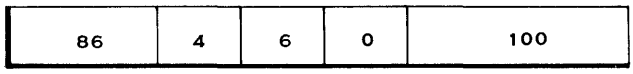
In an algebraic comparison, positive numbers are _____ (lower/higher) than negative numbers.

---

higher

In the "branch on index high" instruction, the branch occurs if the sum is higher than the _____ (1st/2nd/3rd) operand.

3rd



Register 4 00000000

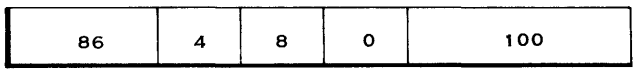
Register 6 00000001 In Hex

Register 7 00000010

In the above BXH instruction, a branch _____ (will/will not) occur.

---

will not



Register 4 00000000

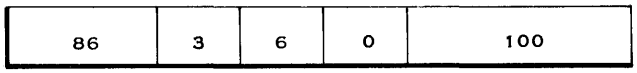
Register 8 00000010 In Hex

Register 9 00000010

In the above BXH instruction, a branch _____ (will/will not) occur.

---

will not; This is because the sum is equal to but not higher than the third operand.



Register 3 00000010

Register 6 00000001 In Hex

Register 7 00000010

In the above BXH instruction, a branch _____ (will/will not) occur.

---

will

86	3	8	0	100

Register 3 FFFFFFFF

Register 8 FFFFFFFF In Hex

Register 9 00000001

In the above BXH instruction, a branch _____ (will/will not) occur.

---

will not; The sum of registers 8 and 3 is a value of -2. This is less than the +1 in register 9.

86	3	5	0	100

Register 3 00000001

Register 5 00000001 In Hex

Register 6 00000002

In the above BXH instruction, a branch _____ (will/will not) occur.

---

will; Register 6 is not used in the preceding problem. The R3 field is odd. As a result, register 5 is used for both the 2nd and 3rd operands. The sum of registers 5 and 3 is a value of +2, which, of course, is higher than the contents of register 5.

86	3	5	0	100

Register 3 00000010

Register 5 FFFFFFFF In Hex

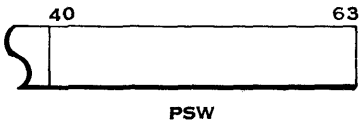
Register 6 000001FF

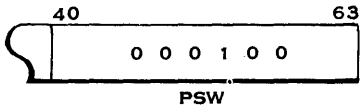
In the above BXH instruction, a branch _____ (will/will not) occur.

---

will; A 2nd operand of -1 is being added to a 1st operand value of +16 and the sum of +15 is high compared to the 3rd operand value of -1.

Show in hex the contents of bits 40-63 of the PSW after the preceding instruction has been executed.





On a successful branch, the generated storage address replaces the instruction address portion of the PSW.

BRANCH ON INDEX LOW OR EQUAL INSTRUCTION

You are now ready to study the "branch on index low or equal" instruction. It is very similar to the BXH instruction. Read the description of this instruction (BXLE) in the Branching section of your Principles of Operation manual.

BXLE is the mnemonic for the "_____ on _____ _____ or _____" instruction.

"branch on index low or equal"

The BXLE instruction is similar to the BXH instruction in that the _____ operand is added to the _____ operand and the sum is algebraically compared to the _____ operand.

2nd
1st
3rd

Indicate the location of the operands in the following BXLE instruction.

87	3	6	0	100

1st operand is in register ____.
2nd operand is in register ____.
3rd operand is in register ____.

3
6
7

When the sum of 1st and 2nd operands is higher than the 3rd operand, the BXLE instruction differs from the PXH instruction in that a branch _____ (does/does not) occur.

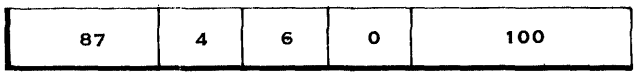
---

does not

With the BXLE instruction, a branch only occurs when the sum of 1st and 2nd operands is _____ or _____ compared with the 3rd operand.

---

low
equal



Register 4 00000008

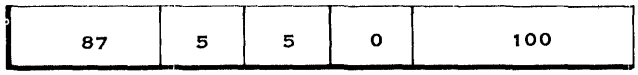
Register 6 00000001 In Hex

Register 7 00000010

In the above BXLE instruction, a branch _____ (will/will not) occur.

---

will; The sum is lower than the contents of register 7.



Register 5 00000001 In Hex

When the same register is used for both the 1st and 3rd operands, the sum is compared with the original contents of the register. In the above BXLE instruction, a branch _____ (will/will not) occur.

---

will not; In this case, the same register is used for all three operands. The 3rd operand is the original contents of reg 5. Obviously, then the System/360 will have to bring the contents of this register into ALU (Arithmetic and Logic Unit) and store it in some register so its original contents will not be lost when the 1st and 2nd operands are added together. If at a later time, this instruction is executed again, the sum from the first execution would be used as the 3rd operand.

EXECUTE INSTRUCTION

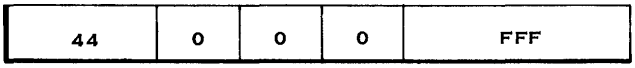
The "branch on condition, branch and link, branch on count, branch on index high, branch on index low or equal" instructions are the only actual "branch" instructions in the System/360. There is, however, another instruction called "execute" which does not change the instruction address in the PSW. However, it does cause one instruction in main storage to be executed out of sequence. That is, instead of branching from one routine to another, the "execute" instruction will cause one instruction in another routine to be executed without leaving the original routine.

Read the description of the "execute" instruction in the Branching section of your Principles of Operation manual.

EX is the mnemonic for the "_____" instruction.

"execute"

Without branching, the "execute" instruction will cause another instruction to be executed. Given the following EX instruction, indicate with six hex digits the address of the instruction to be executed.



Address of instruction to be executed = _____.

000FFF

The instruction to be executed will be executed as it is if the R1 field of the EX instruction is _____.

---

zero

Assuming that the effective generated storage address from the "execute" instruction is location 8500, write the addresses of the instructions in the sequence in which they will be executed.

Location	Instruction		Actual Sequence
2048	LH	1, 1000 (0, 2)	_____
2052	EX	0, 2000 (0, 2)	_____
2056	STH	1, 1002 (0, 2)	_____
			_____

2048
2052
8500
2056

Was the instruction at location 8500 modified in any way prior to being executed? _____

No

Why not? _____.

The R1 field of the "execute" instruction was zero.

Was the address of the instruction at location 8500 placed in bits 40-63 of the PSW? _____

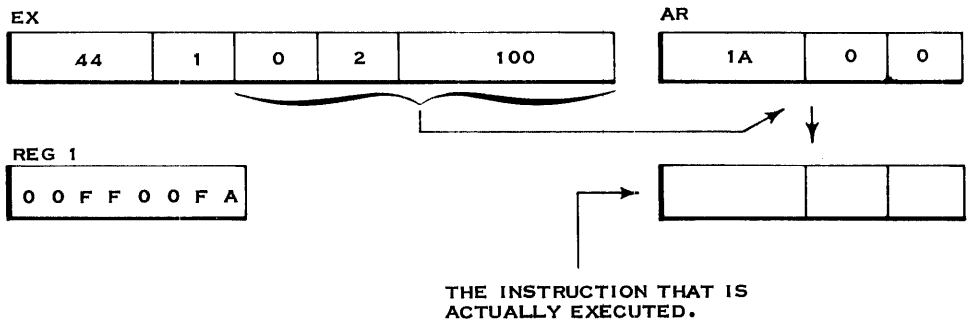
No; The "execute" instruction is not a "branch" instruction. Instead, it causes an instruction to be executed that is not in the sequence presently being executed. In the previous program example, the "execute" instruction at 2052 caused the instruction at 8500 to be executed. Then the normal sequence of instruction execution continued with the instruction at 2056.

If the R1 field of the "execute" instruction is other than zero, the low-order byte of the specified register will be ORed with the ____ (1st/2nd) byte of the instruction to be executed.

2nd

If you are not familiar with the function of ORing and ANDing bits, go to the Principles of Operation manual. Read the OR and AND examples in the Instruction Use Examples area of the Appendix.

Given the following, write the instruction that is actually executed.



1A	F	A

The instruction that was actually executed causes the contents of register ____ to be algebraically added to the contents of register ____.

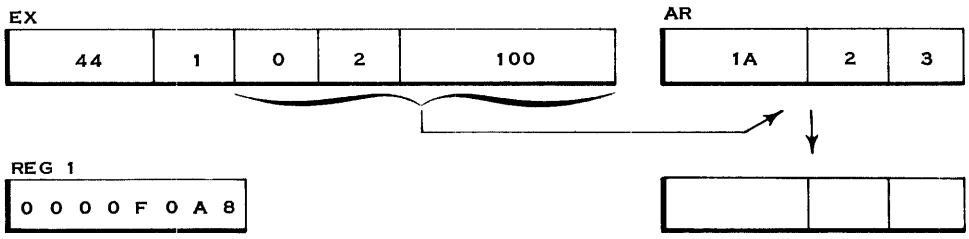
10
15

The ORing of the 2nd byte of the instruction with the low-order byte from the register is done in the ALU. As a result, the instruction in storage _____ (remains the same/is changed).

---

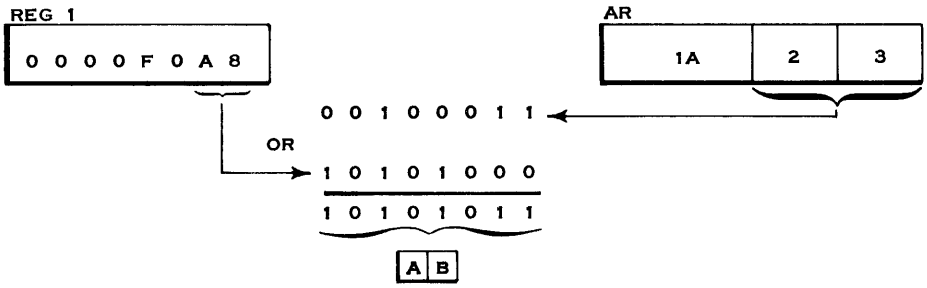
remains the same

Given the following, write the instruction that will be executed in ALU. Remember that the bytes are ORed!



1A	A	B

Bits 24-31 of register 1 are ORed with bits 8-15 of the AR instruction as shown below.



As a result of the instruction shown in the previous example, the "execute" instruction would cause (via the "add" instruction) the contents of register ___ to be added to contents of register ___.

11
10

Write the AR instruction that remains in storage as a result of the instruction in the preceding example.

AR

1A	2	3

THE INSTRUCTION IN STORAGE IS NOT CHANGED

There are three programming interrupts possible with an "execute" instruction.

1. Specification exception
2. Addressing exception
3. Execute exception

A specification exception can occur on an "execute" instruction if the generated effective address of the instruction to be executed is _____ (odd/even).

---

odd; Remember that all instructions must start on an even address.

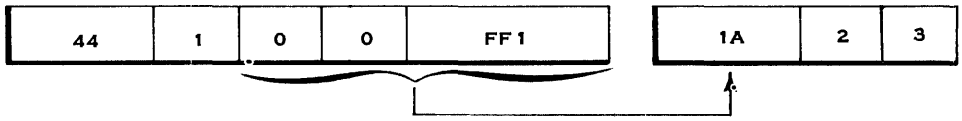
An addressing exception can occur on an "execute" instruction if the generated effective address of the instruction to be executed _____ (is/is not) available on the particular System/360 installation.

---

is not

An execute exception can occur on an "execute" instruction if the system is directed to another "_____" instruction.

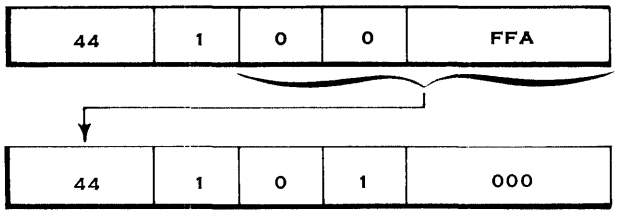
"execute"



The above "execute" instruction will result in a(n) _____ exception.

---

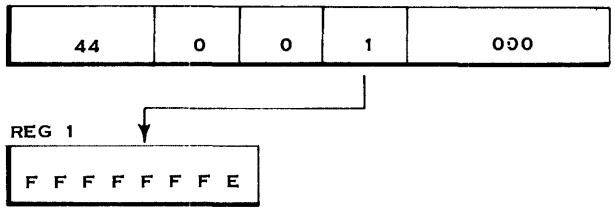
specification; The generated address is odd.



The above "execute" instruction will result in a(n) _____ exception.

---

execute



The above "execute" instruction will probably result in a(n) _____ exception.

---

addressing

Choose the correct branching instruction mnemonic from the list on the right and write it next to the proper instruction name.

NAME		MNEMONIC	
Branch on Condition	=	BC BCR	BXH
Branch and Link	=	____ ____	BCTR
Branch on Count	=	____ ____	BALR
Branch on Index High	=	____	EX
Branch on Index Low or Equal	=	____	BXLE
Execute	=	____	BCT
			BAL

BC BCR
BAL BALR
BCT BCTR
BXH
BXLE
EX

Go to the IBM System/360 Principles of Operation manual and study the following areas of the Branching section.

Decision-Making
Instruction Formats
Branching Instructions
Branching Exceptions

SECTION II LEARNING OBJECTIVES

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

Logical Operations

This section of the text covers the group of instructions known as the logical operations. These "logical" instructions allow you to move data around in storage, compare alphameric data, AND and OR two data fields, and also allow you to do other miscellaneous operations. The "logical" instructions use all give instruction formats and work with both fixed and variable length data fields. Read the following introductory material in the Logical Operations section of your Principles of Operation manual.

Logical Operations
Data Format
Condition Code
Instruction Format
Instructions

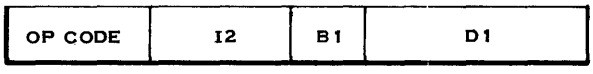
A number of the instructions in the Logical Operations section are not covered in this section of your self-study text. They include the four "logical shift" instructions (which you learned earlier in your text on Fixed Point Binary Operations) and the two "edit" instructions which are part of the Decimal Feature on System/360. The two "edit" instructions are covered later in this self-study text in the Decimal Operations section.

MOVE INSTRUCTION

Read the description of the following instructions in the Logical Operations section of your Principles of Operation manual.

Mnemonic	Descriptive Title
MVI	Move Immediate
MVC	Move Characters

The "move immediate" instruction uses the SI format. Label the fields of the SI format.



Operands that are carried in the instruction itself are called _____ operands.

---

immediate

In the MVI instruction, the immediate operand is one _____ long and is the _____ (1st/2nd) operand. The instruction will move the byte of immediate data to main _____.

---

byte
2nd
storage

The MVI instruction will cause the byte in main storage to be replaced by a byte from the _____. The main storage address _____ (does/does not) have to be even.

---

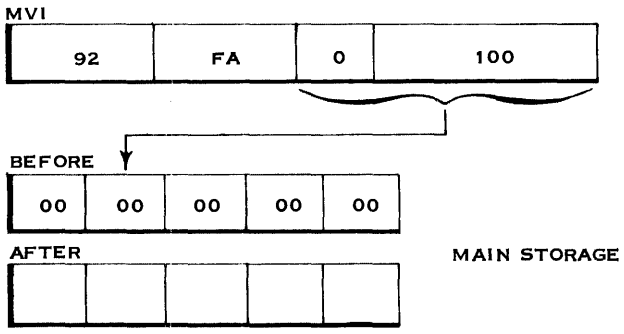
instruction
does not; Any byte in main storage can be addressed.

The MVI instruction can move _____ (more than one/only one) byte of data.

---

only one; Only the immediate byte in the instruction.

Given the following (in hex) show the contents of main storage after the MVI instruction is executed.



00	FA	00	00	00

MVI is the mnemonic for the move immediate. MVC is the mnemonic for _____ _____.

---

move characters

The MVC instruction uses the SS format because both operands are _____ (fixed/variable) length fields in main storage.

---

variable

The MVC instruction moves bytes (characters) from one area of main storage to another. The number of characters is determined by the 2nd byte of the MVC instruction. This byte is called the _____ field.

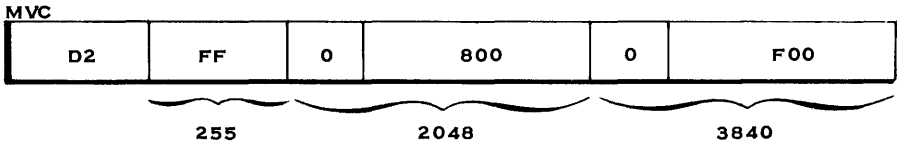
L1

The length code (L1) can represent a count of from zero to _____. Since the number of bytes in a field is equal to the length code +1, a length code of zero would mean _____ byte.

255
one

A length code of 255 in the MVC instruction would cause _____ bytes to be moved.

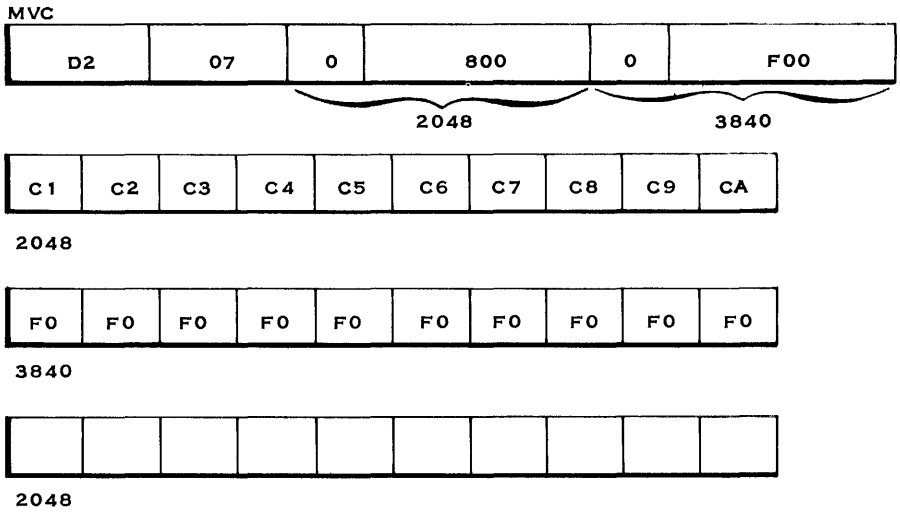
256

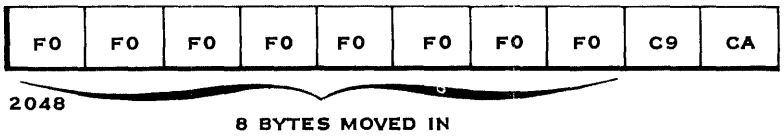


As in most System/360 operations, the 1st operand receives the results. In the above MVC instruction, _____ bytes would be moved from location _____ to location _____.

256
3840
2048

Given the following, show the contents of the indicated storage area after the MVC instruction is executed. Everything is shown in hex.





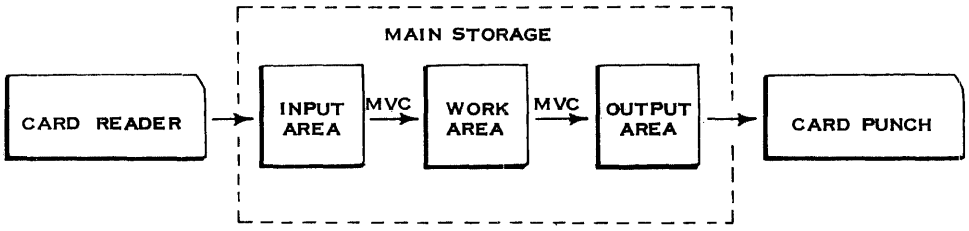
The bytes moved by the MVC instruction: (Circle one of the following)

a. Must be packed decimal data.
b. Must be in the halfword signed binary format.
c. Can be in any format.
d. Must be EBCDIC characters.

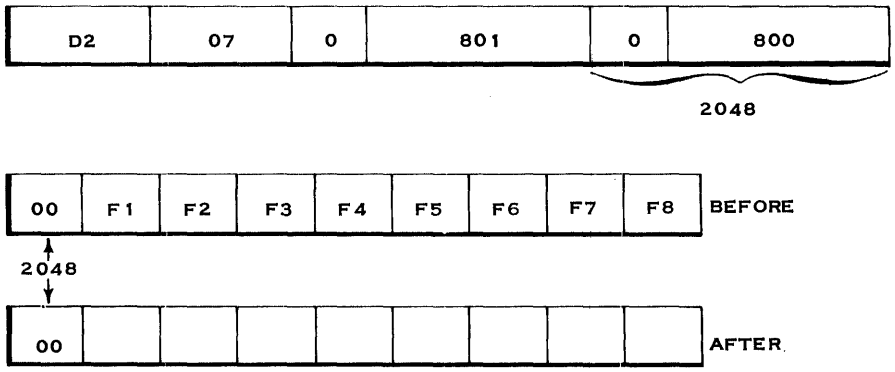
---

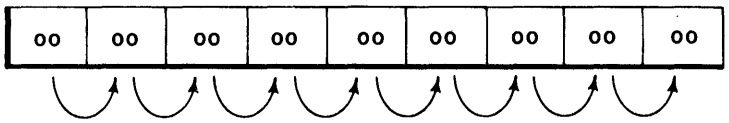
c; The bytes that are moved are not checking for coding. The title of the instruction (move characters) implies, however, that this instruction could be used to move EBCDIC characters from one area of storage to another. For instance, data could be moved from an input area to a work area without being changed. After the data has been processed in a work area, it could be moved to an output area.

The following drawing illustrates this point.



Because the bytes are moved one at a time, the MVC instruction can also be used to propagate one character throughout an area. Given the following, show the resulting storage contents.





In the preceding problem, the following occurred:

1st; The instruction looks at the first location (2048), finds the 00 and moves it to 2049 where it replaces the F1.

2nd; The instruction looks at the next location (2049), finds the 00 and moves it to 2050.

3rd; The leftmost byte continues to be moved (propagated) to the right.

MOVE NUMERICS INSTRUCTION

Read the description of the "move numerics" instruction in the Logical Operations section of your Principles of Operation manual.

MVN is the mnemonic for the "_____ _____" instruction.

"move numerics"

The MVC instruction moved the entire byte from the 2nd operand.

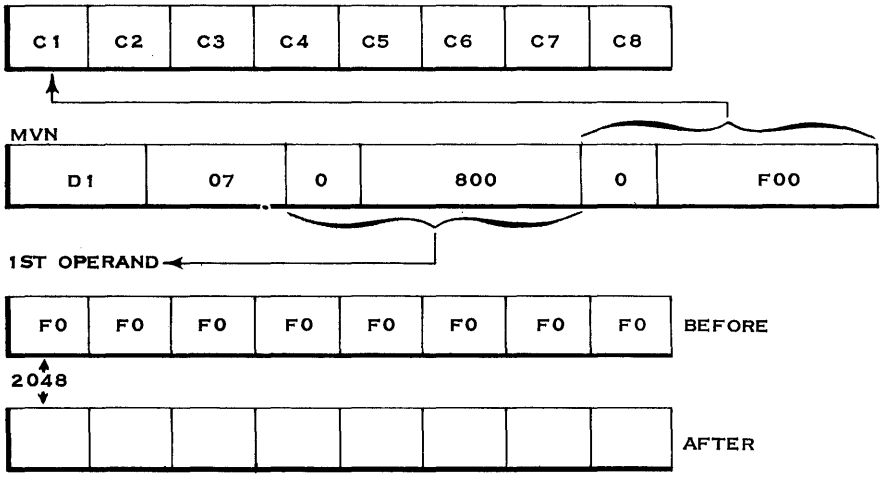
The MVN instruction only moves bits _____ (0-3/4-7).

4-7

The MVN instruction moves bits 4-7 of each byte from the _____ (1st/2nd) operand to bits 4-7 of the _____ (1st/2nd) operand.

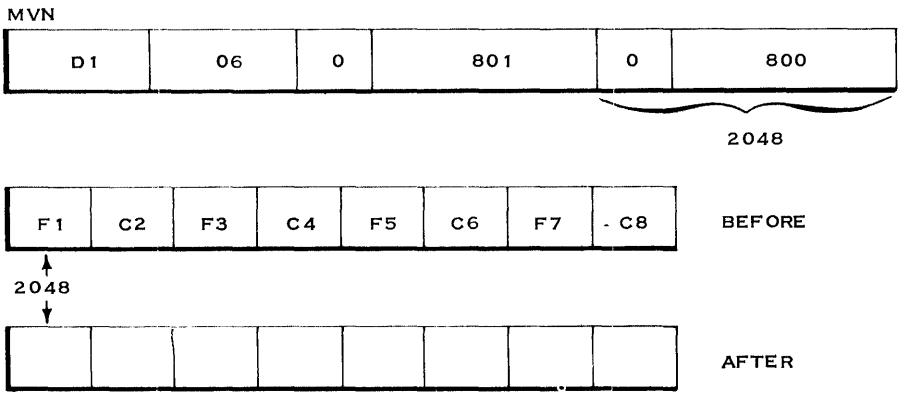
2nd
1st

Given the following, show the resulting contents of the 1st operand.



F1	F2	F3	F4	F5	F6	F7	F8

Given the following, show the resulting contents of the main storage area.



F1	C1	F1	C1	F1	C1	F1	C1

The numeric portion of the leftmost byte was moved (propagated) to the right, byte by byte, 7 times.

MOVE ZONES INSTRUCTION

Read the description of the "move zones" instruction in the Logical Operations section of your Principles of Operation manual.

MVZ is the mnemonic for the "_____ _____" instruction.

"move zones"

The MVN instruction moved bits 4–7 of each byte from the 2nd operand to the 1st operand.

The MVZ instruction moves bits ___ through ___.

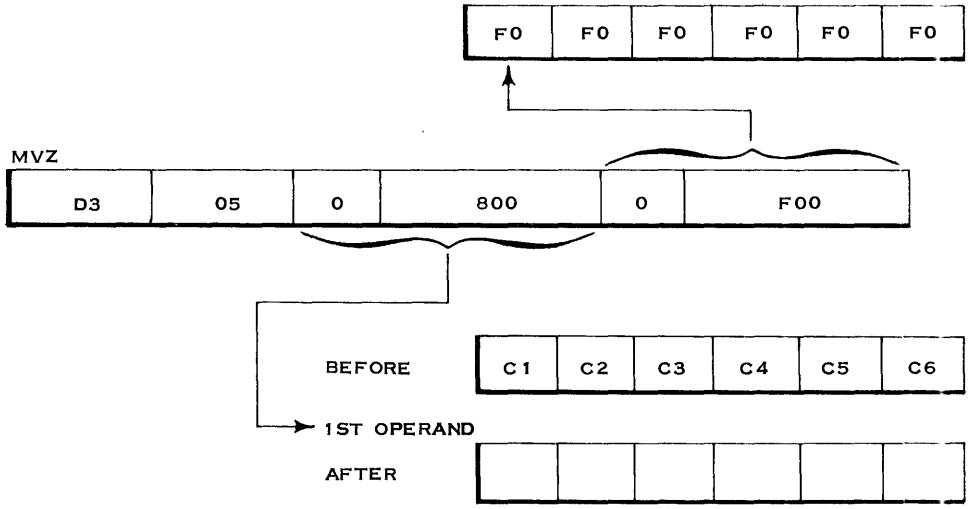
0, 3

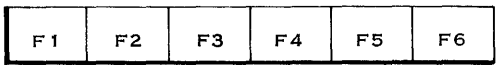
Just like the MVC and MVN instructions, the MVZ instruction processes the data field from left-to-right, one _____ at a time.

---

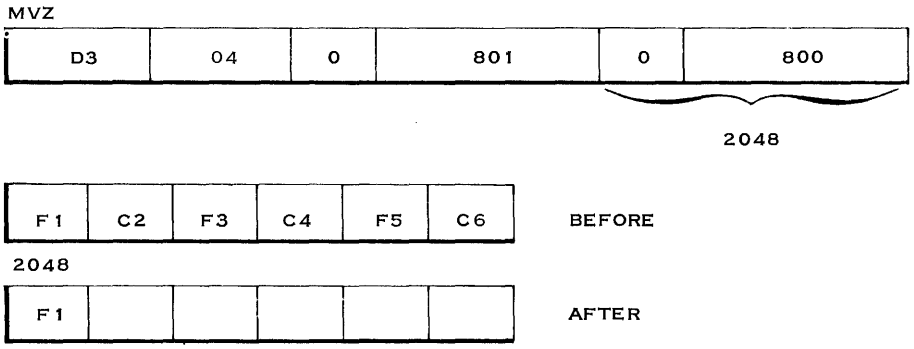
byte

Given the following, show the resulting contents of the 1st operand.





Given the following, show the resulting contents of the 1st operand.



The zone portion of the leftmost byte was moved (propagated) to the right.

Assume that the MVN, MVC or MVZ instruction has a 1st operand address that is one higher than the 2nd operand address. The corresponding numerics, character, or zones of the first byte from the 2nd operand will be propagated to the _____ (left/right).

---

right

You have been studying four "move" instructions. List their mnemonics.

_____ _____ _____ _____

MVI
MVC
MVN
MVZ

Which of the "move" instructions can move only one byte? _____

MVI

The MVC, MVN, MVZ instructions can move information into a maximum of _____ bytes.

256; As specified by their 8-bit long length code.

In the preceding four "move" instructions, the results always change the contents of a main storage area. Assume that the protection key in the PSW does not match the key for the affected storage area. A program interrupt is possible because of the _____ exception.

---

protection

The protection and storage keys do not have to match if: (Circle the most correct statement.)

a. the PSW key is 15.
b. the PSW key is 0.
c. the storage key is 15.
d. the storage key is 0.
e. either key is 15.
f. either key is 0

---

b; The protection key in the PSW being zero will allow the data to be stored and a protection exception will not result.

If a generated storage address in the preceding group of "move" instructions is not available, an _____ exception will occur.

---

addressing

The two programming exceptions possible on the "move" instructions are _____ and _____

---

protection
addressing

The "move" instructions _____ (change/do not change) the condition code.

---

do not change

Let's take a look at a few symbolic programming examples using the instructions you have learned. First let's review the RR, RX, and RS formats.

Format	Symbolic	Example
RR	Mnemonic R1, R2	AR 2, 3
RX	Mnemonic R1, D2 (X2, B2)	AH 2, 1000 (0, 3)
RS	Mnemonic R1, R3, D2 (B2)	BXH 2, 4, 1000 (3)

The SI format will be shown like this:

Mnemonic D1 (B1), I2

Such as:

MVI 2048 (0), 0

The preceding would look like this in actual machine language:

92	00	0	800

IN HEX

MVI 2048 (0), 0

Show the binary bit structure of the byte placed in location 2048 after the execution of the above instruction. 2048 = _____

00000000

MVI 4095 (0), 255

The above instruction would cause a binary bit structure of _____ to be placed in location _____.

11111111
4095

The SS format will be shown like this:

Mnemonic D1 (L, B1), D2 (B2)

Such as:

MVC 2048 (8, 0), 3840 (0)

The preceding instruction would look like this in actual machine language.

D2	07	0	800	0	F00

IN HEX

Notice that the length code in the symbolic example showed the number of bytes while the length code in machine language is one less than the total number of bytes.

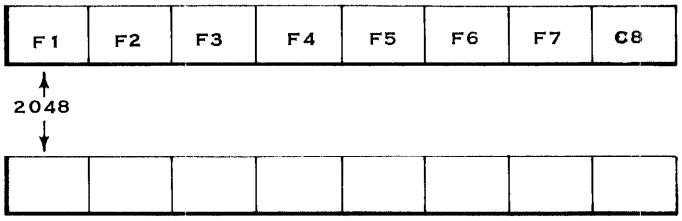
```
MVC     2048 (16, 0), 3840 (0)
```

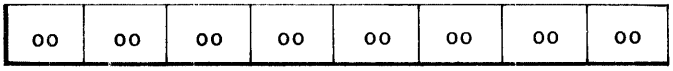
The above instruction would cause the bytes in locations _____ through _____ to be moved to locations _____ through _____.

3840, 3855
2048, 2063

Given the following, show the resulting contents of the storage area.

```
MVI     2048 (0), 0
MVC     2049 (7, 0), 2048 (0)
```



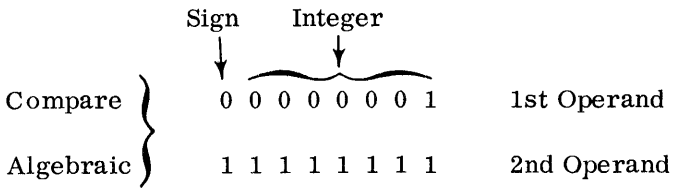


In the preceding problem, the MVI instruction placed a byte of all zero bits in location 2048. Then the MVC instruction propagated this byte into the seven bytes to the right (2049-2055).

You learned three "compare" instructions when you were studying the fixed point instructions. Their mnemonics are:

CR	Compare, RR format
C	Compare, RX format
CH	Compare Halfword, RX format

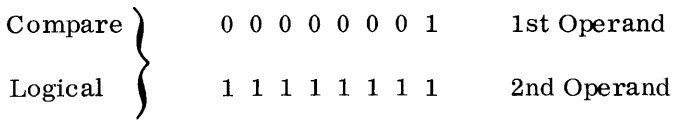
These three "compare" instructions compared on an algebraic basis. In other words, they treated the operands as signed binary integers. The operands were either positive or negative numbers. The "compare logical" instructions you will now learn also treat the operands as binary information. However, they will be considered as unsigned binary fields. For example, consider the comparison of the following binary fields on an algebraic basis.



Because the 1st operand is a positive number (+1) and the 2nd operand is a negative number (-1), the 1st operand is high and the PSW condition code would be set to ____.

---

10 If the same fields were compared on a logical basis, they would be treated as unsigned integers and the absolute values would be compared as follows:



In the above example, the 1st operand would compare low and the PSW condition code would be set to ___. This occurs because an unsigned value of 1 is being compared with an unsigned value of 255.

---

01 The programmer must know what format his data is in before he can compare it. If his data consists of signed binary words or halfwords, he would use his three "algebraic" instructions: CR, C, CH. If his data consists of unsigned binary fields, he would use the "logical" instructions. As a point of interest, the EBCDIC code is so arranged that the special and alphameric characters will collate on a binary basis. That is, the "compare logical" instructions are used to compare EBCDIC characters.

Let's look at the bit sequence of the EBCDIC characters.

Write the EBCDIC code for the character "A." _____

11000001

Write the EBCDIC code for the character "Z." _____

11101001

"A" 11000001
"Z" 11101001

On a compare logical basis, which is low? _____ ("A" or "Z")

"A"

"1" 11110001
"Z" 11101001

On a compare logical basis, which is low? _____ ("1" or "Z")

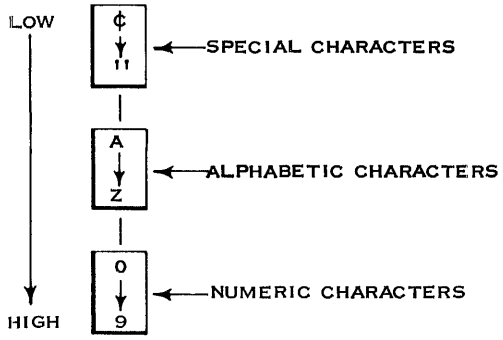
"Z"

"#" 01111011
"A" 11000001

On a compare logical basis, which is low? _____ ("#" or "A")

"#"

The preceding examples should agree with the collating sequence you are familiar with in your past experience with punched card equipment or equipment which used the standard BCD code (BA 8421). This is illustrated as follows:



You now have an idea of the difference between algebraic and logical comparisons. You can also see when they are used. Read the descriptions of the "compare logical" instructions in the Logical Operations section of your Principles of Operation manual.

CL is the mnemonic for _____ _____.

---

compare logical

The CL instruction uses the RX format. CLR is the mnemonic for the "compare logical" instruction that uses the __ __ format.

RR

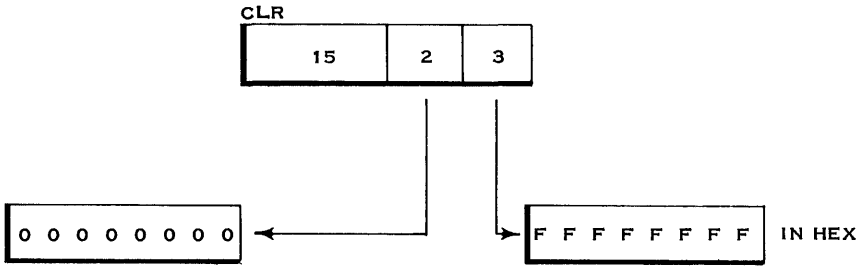
In both the CL and CLR instruction, the 1st operand is the register specified by the ____ field. The instructions cause a _____ (logical/algebraic) comparison. As a result of the comparison, the _____ _____ is set.

R1
logical
condition code

The condition code settings of 00, 01, 10 indicate that the _____ (1st/2nd) operand is equal, low, or high compared to the _____ (1st/2nd) operand. After a compare operation, it is impossible to have a condition code of ____.

1st
2nd
11

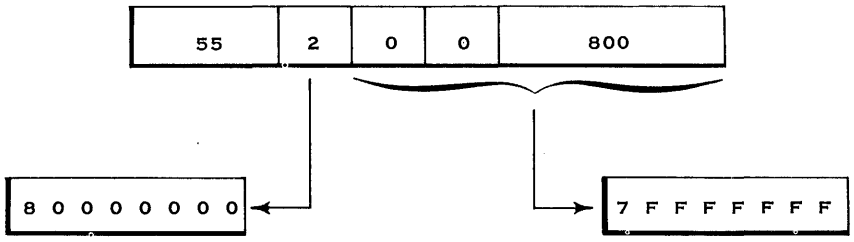
Given the following CLR instruction, indicate the resulting condition code bits in the PSW.



Condition Code = _____

01; If the CR (compare algebraic) instruction had been used, the 1st operand would have been high.

Given the following CL instruction, indicate the resulting condition code.



Condition Code = _____

10; By examining the four high-order bits, you can see that the 1st operand is high.
1st operand - 1000
2nd operand - 0111

Besides the CLR and CL instructions, System/360 can also compare logical using the SI and SS formats. CLI is the mnemonic for the " _____ _____ _____ " instruction.

"compare logical immediate"

The "compare logical immediate" instruction uses the SI format. In this format, the 1st operand is in _____ (main storage/the instruction).

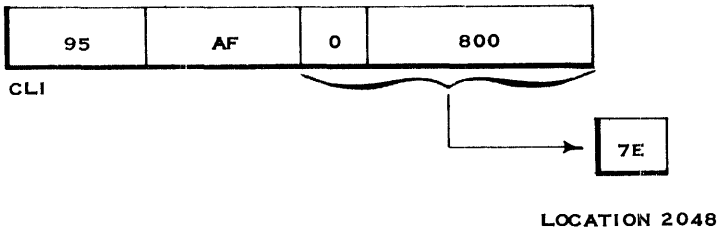
---

main storage

The CLI instruction compares on a(n) _____ (algebraic/logical) basis. The comparison is between one byte in storage and one byte in the _____.

---

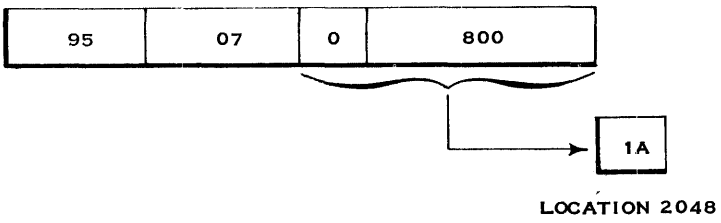
logical instruction



In the above CLI instruction, the 1st operand is _____ (low/high) and the resulting condition code is _____.

---

low
01; In the SI format, the 1st operand is in main storage.



In the above CLI instruction, the 1st operand is _____ (low/high) and the resulting condition code is _____.

---

high
10

The CLR and CL instructions compare one _____ (byte/word/halfword) of data with another.

The CLI instruction compares one _____ (byte/word/halfword) of data with another.

---

word
byte

The compare logical operation can also be done with the __ __ (SS/RS) format.

SS

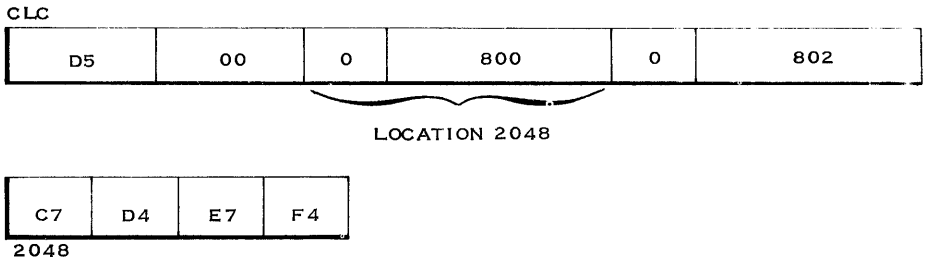
CLC is the mnemonic for the "compare logical" instruction which uses the __ __ format.

SS

CLC means Compare Logical Characters. This instruction has an 8-bit length code and can compare up to _____ characters.

256

The name of the CLC instruction indicates that characters are being compared. Actually, bytes are being compared on an unsigned binary (logical) basis. As was previously pointed out, however, the EBCDIC code assigned to characters is arranged so that they will collate on a binary basis.



In the above CLC instruction, _____ character(s) will be compared and the condition code will be set to ____.

---

two; One from each operand. 01

The coding of the byte at location 2048 above could represent the EBCDIC character "_____".

"G"; Because hex C7 = bits 11000111 which equal the EBCDIC "G."

The coding of the byte at location 2049 could represent the EBCDIC character "_____."

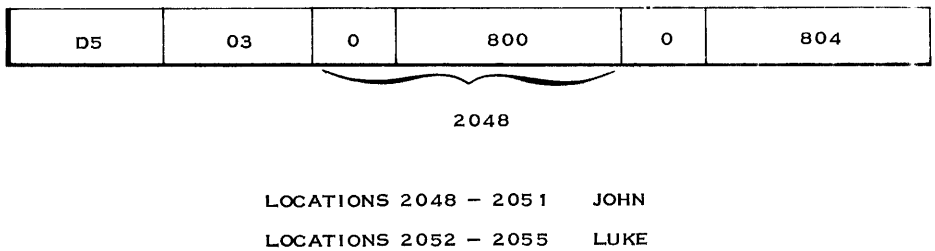
"M"

The coding of the byte at location 2050 could represent the EBCDIC character "____."

"X"

The coding of the byte at location 2051 could represent the character "____."

"4"



Given the above characters and CLC instruction, the condition code will be set to ____.

01; In the preceding problem, JOHN was the 1st operand and LUKE was the 2nd operand. The high-order character (J) of the 1st operand was lower than the high-order character (L) of the 2nd operand.

"J" - 11010001
"L" - 11010011

D5	07	0	800	0	F00

LOCATIONS 2048 – 55 – JOHNSTON

LOCATIONS 3840 – 47 – JOHANSEN

Given the above characters and CLC instruction, the condition code will be set to _____.

10; On the first three high-order characters (JOH), both operands are equal. On the fourth character, the 1st operand will compare high as follows:

1st operand - "N" - 11010101
2nd operand - "A" - 11000001

You should now realize that the comparing is done for all practical purposes from _____ (left to right/right to left).

---

left to right

As a result of comparing the bytes from left to right, it is not necessary to examine the entire field. The compare operation assumes that the fields are equal to begin with. In examining the bytes from left to right, the system can end the compare operation as soon as it finds an u_____ condition.

---

unequal

List the mnemonics and the instruction formats of the four "compare logical" instructions.

Mnemonic	Instruction Format
_____	_____
_____	_____
_____	_____
_____	_____

Mnemonic	Format
CLR	RR
CL	RX
CLI	SI
CLC	SS

List the mnemonics and formats of the three "compare algebraic" instructions.

Mnemonic	Instruction Format
_____	_____
_____	_____
_____	_____

Mnemonic	Format
CR	RR
C	RX
CH	RX

What is the main difference between the CR and CLR instructions?

_____

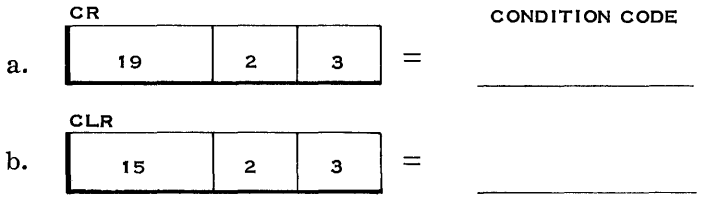
_____

_____

_____

The CR instruction will treat the contents of a particular register as a signed integer (sign and 31 bits). The CLR instruction treats the contents of the same register as an unsigned 32-bit integer. As a result, the condition code setting may vary, depending on the instruction used.

Given the contents of the following two registers, indicate the resulting condition code for the instruction shown.



---

a. 01 - Reg 2 has a negative number.
b. 10 - Reg 2 has a higher value.

Assume that a card record punched in standard hollerith card code has been read in main storage. The record contained alphabetic characters. To compare two fields in this record, which would you do? _____ _____ (compare logical/compare algebraic).

---

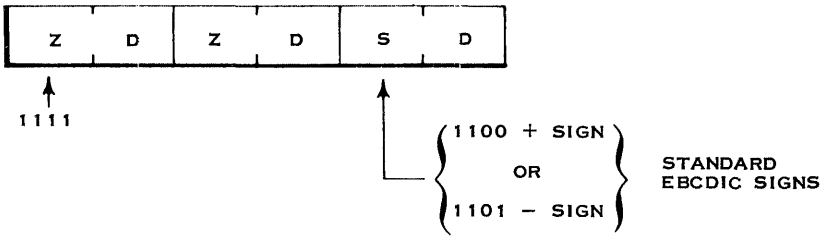
compare logical

Given two fields of zoned decimal data, what would you do? (Circle one of the following.)

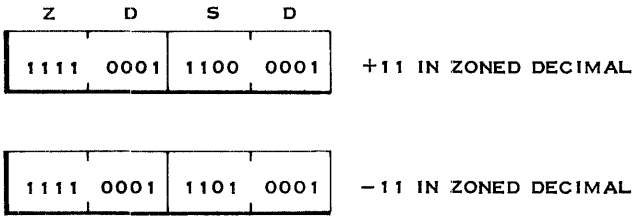
a. Compare Algebraic
b. Compare Logical
c. Neither of the above

---

c; See the following for explanation.



Normally, the "compare logical" instruction is used to compare alphameric information. However, a zoned decimal field presents a special problem. Assume we wish to compare a +11 with a -11. The +11 should be high but look below.



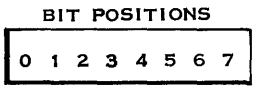
As you can see above, the minus sign bits (due to the 11-hole card punch) would cause the negative number to be high. The plus sign bits (1100) are, of course, due to a 12-hole card punch.

You have now been shown a problem without a solution. One solution, of course, would be to change (via instructions) the data to the binary format and use the "compare algebraic" instruction. Another solution is to place the data (via the "pack" instruction) in the packed format. Then you can compare decimal. We will be covering this instruction later on in this textbook.

Right now, we will continue with more "logical" instructions. The next group of instructions you will study can truly be called "logical" instructions. They allow the programmer to mix data fields together on a basis of AND, OR, as well as Exclusive OR logic.

The "and" instruction is used to mix two operands on a logical AND basis. The definition of an AND condition is this: If both bits are 1, the resulting bit is 1. Otherwise, it is zero.

The following will illustrate the result of ANDing two bytes together.



1st Operand	1 0 1 0 1 0 1 0
2nd Operand	1 0 0 1 1 1 0 0
Result	1 0 0 0 1 0 0 0

Notice that only in bit positions 0 and 4 were both bits set to 1. As a result, only bits 0 and 4 of the result are 1. As in most System/360 operations, the result will replace the 1st Operand.

Given the following two bytes, show the result after they are ANDed together.

1st Operand	0 1 1 1 0 1 1 0
2nd Operand	1 1 0 0 1 1 0 0
Result	

---

0 1 0 0 0 1 0 0; Notice again that the operands are ANDed together on a bit-by-bit basis. There is no connection (carry) from one bit position to another.

Show the result of the following AND operation.

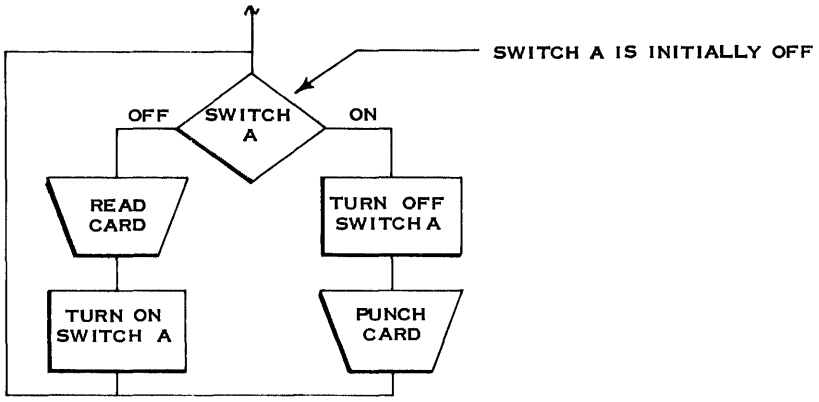
1st Operand	1 1 0 0 0 0 1 1
2nd Operand	1 0 0 0 0 0 0 1
Result	

---

1 0 0 0 0 0 0 1

In a typical user's program, the programmer will occasionally use data as a programmable switch. That is, in a flowchart, a branch decision will occasionally be based on whether a switch is on or off.

For example:



Notice that switch A is used to determine whether to read a card or punch a card. Also, notice that the flowchart assumes that there is a method of turning the switch on and off.

One use of the "and" instruction is to turn off program switches.

---

0 1 0 0 0 0 1 1

Assume that bit position 7 of the byte shown above represents a program switch. In order to turn off this program switch, bit position 7 of the second operand must be ___ (1/0).

0; See below.

1st Operand	0 1 0 0 0 0 1 1
2nd Operand	1 1 1 1 1 1 1 0
Result	0 1 0 0 0 0 1 0

Since a byte contains 8 bits it can hold ___ program switches. A word can contain ___ program switches.

8
32

It is desired to turn off one program switch in a byte without affecting the other switches. Show the 2nd operand necessary to turn off only the switch in bit position 6.

1st Operand	1 0 1 1 0 1 1 0
2nd Operand	_____
Result	1 0 1 1 0 1 0 0

---

1 1 1 1 1 1 0 1; Notice that only bit position 6 was changed. This 2nd operand would work with any 1st operand and still turn off only position 6.

It is desired at the beginning of the program to be sure that all of the program switches are off. Show the necessary 2nd operand.

1st Operand	0 1 1 0 0 0 1 0	8 program switches
2nd Operand	_____	

---

0 0 0 0 0 0 0 0

Now that you can turn off program switches, how about turning them on?

The "and" instructions can be used to turn off program switches. The "or" instructions can be used to turn on program switches. The definition of an OR condition is this: If either bit is 1, the resulting bit is 1. Otherwise, it is zero.

The following will illustrate the result of ORing two bytes together.

1st Operand	1 0 1 0 1 0 1 0
2nd Operand	1 0 0 1 1 1 0 0
Result	1 0 1 1 1 1 1 0

Notice that only in bit positions 1 and 7 neither bit was set to 1. Consequently, only bits 1 and 7 of the result are set to 0. The remaining bits contain a 1.

Given the following operands, show the result if they are ORed together.

1st Operand	0 1 1 1 0 1 1 0
2nd Operand	1 1 0 0 1 1 0 0
Result	

---

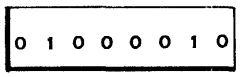
1 1 1 1 1 1 1 0

Show the result of the following OR operation.

1st Operand	1 1 0 0 0 0 1 1
2nd Operand	1 0 0 0 0 0 0 1
Result	

---

1 1 0 0 0 0 1 1



Assume that bit position 7 of the byte shown above is a program switch. In order to turn on this program switch, bit position 7 of the 2nd operand must be ___ (0/1).

1; See below.

1st Operand	0 1 0 0 0 0 1 0
2nd Operand	0 0 0 0 0 0 0 1
Result	0 1 0 0 0 0 1 1

It is desired to turn on one program switch in a byte without affecting the others. Show the necessary 2nd operand to turn on only the switch in bit position 2.

1st Operand	1 1 0 1 0 1 1 0
2nd Operand	_____

---

0 0 1 0 0 0 0 0

Show the result of ORing the previous answer with the 1st operand.

1st Operand	1 1 0 1 0 1 1 0
2nd Operand	0 0 1 0 0 0 0 0
Result	

---

1 1 1 1 0 1 1 0

It is desired at the beginning of a program to be sure that all of the program switches are initially on. Show the necessary 2nd operand.

1st Operand	0 1 1 0 0 0 1 0	8 program switches
2nd Operand	_____	

AND INSTRUCTION - OR INSTRUCTION

---

1 1 1 1 1 1 1 1

Read the description of the "and" and "or" instructions in the Logical Operations section of your Principles of Operation manual.

The mnemonic of the "and" instruction uses the letter ____.

The mnemonic of the "or" instruction uses the letter ____.

N
O

Both the "and" and "or" instructions can use four formats. Use the following mnemonics and indicate the instruction format and whether it is an "and" or "or" instruction.

Mnemonic	Format	Instruction
NR	___ ___	_____
NC	___ ___	_____
OI	___ ___	_____
N	___ ___	_____
OR	___ ___	_____

Mnemonic	Format	Instruction
NR	RR	"and"
NC	SS	"and"
OI	SI	"or"
N	RX	"and"
OR	RR	"or"

After executing an "and" or "or" instruction, the condition code can be set to one of _____ possible settings.

---

two

A condition code of 00 would indicate a result of _____ (all zeroes/all ones).

If the result is not zero, the condition code will be set to ____ (00/01).

---

all zeroes
01

The two possible settings of the condition code after an "and" or "or" instruction are: _____ _____

00
01

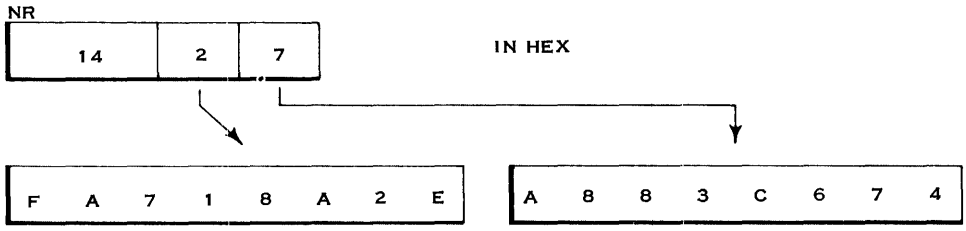
The "branch on condition" instruction can be used after an "and" or "or" instruction to: (Circle one of the following.)

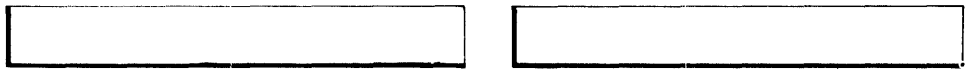
a. Check if all program switches are set to 0.
b. Check if one specific switch is on or off.

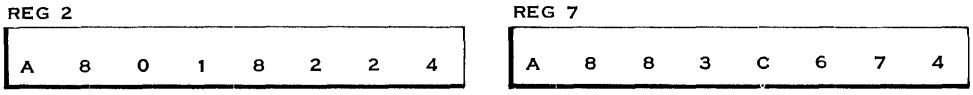
---

a; The condition code reflects the status of the entire result. It can either be zero or non-zero.

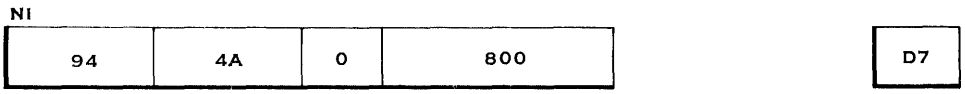
Given the following, show the contents of the registers after the instruction is executed.

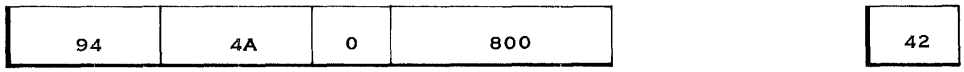




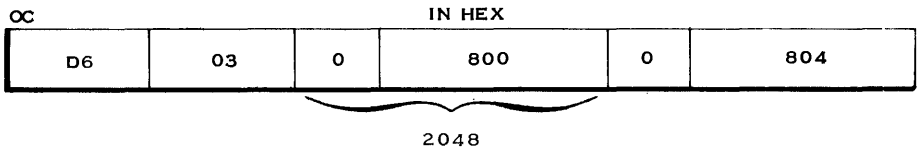


Given the following, show the contents of the instruction and the storage byte after the instruction is executed.





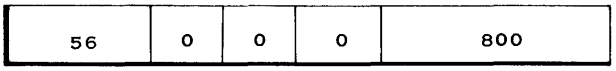
Given the following, show the storage contents after the instruction is executed.

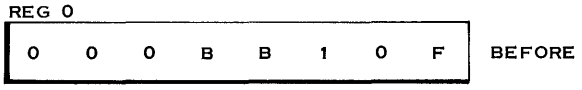


Location	Before	After
2048	DE	_____
2049	A0	_____
2050	7F	_____
2051	8B	_____
2052	9E	_____
2053	01	_____
2054	72	_____
2055	F1	_____

Location	After
2048	DE
2049	A1
2050	7F
2051	FB
2052	9E
2053	01
2054	72
2055	F1

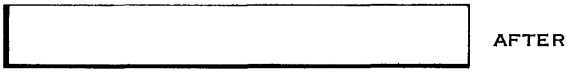
Given the following "or" instruction, show the contents of register 0 and the condition code after the instruction is executed.





IN HEX

LOCATION	CONTENTS
2048	80
2049	7E
2050	01
2051	F0



Condition Code = _____

Reg 0
807FB1FF

Condition Code
01

So far you have had a good look at the "and" and "or" instructions. You have seen how they can be used to turn on and turn off program switches. Another instruction that can be used to alternately turn on and turn off a program switch is the "exclusive or" instruction. The definition of an Exclusive OR condition is this:

> If one and only one of the bits is 1, the result is 1. Otherwise, it is zero.

The following will illustrate the result of Exclusive ORing two bytes.

1st Operand	1 0 1 0 1 0 1 0
2nd Operand	1 0 0 1 1 1 0 0
Result	0 0 1 1 0 1 1 0

Notice that in bit positions 2, 3, 5 and 6 one and only one of the bits was a 1. So only these positions of the result have a 1. In bit position 0, both bits were 1 and the result was 0. In bit position 1, both bits were 0 and the result was 0.

Given the following operands, show the result of Exclusive ORing them.

1st Operand	0 1 1 1 0 1 1 0
2nd Operand	1 1 0 0 1 1 0 0
Result	

---

1 0 1 1 1 0 1 0

Show the result of the following Exclusive OR operation.

1st Operand	1 1 0 0 0 0 1 1
2nd Operand	1 0 0 0 0 0 0 1
Result	

---

0 1 0 0 0 0 1 0

It is desired to change only one program switch in a byte without affecting the others. Show the 2nd operand necessary to change only the switch in bit position 3.

1st Operand	1 1 0 1 0 0 1 1
2nd Operand	_____

---

0 0 0 1 0 0 0 0

It is desired to change all of the program switches in a byte. Show the necessary 2nd operand and the expected result.

1st Operand	1 0 1 1 0 1 1 1
2nd Operand	_____
Result	_____

2nd Operand 1 1 1 1 1 1 1 1
Result 0 1 0 0 1 0 0 0

Assume that bit position 7 of a byte is a program switch. In order to change the setting of this program switch, bit position 7 of the 2nd operand must be ___ (0/1).

1; See below.

1st Operand	0 1 0 0 0 0 1 1
2nd Operand	0 0 0 0 0 0 0 1
Result	0 1 0 0 0 0 1 0

Read the description of the "exclusive or" instructions in the Logical Operations section of your Principles of Operation manual.

The letter X is used for the mnemonic of an " _____ ____ " instruction.

"exclusive or"

Like the "and" and "or" instructions, the "exclusive or" instruction uses ___ (1/2/3/4) formats.

4

List the mnemonics and formats of the four "exclusive or" instructions.

Mnemonic	Format
_____	__ __
_____	__ __
_____	__ __
_____	__ __

Mnemonic	Format
XR	RR
X	RX
XI	SI
XC	SS

Just like the "and" and "or" instructions, the "exclusive or" instruction will cause the condition code to be set to ____ (00/01/10/11) for an all-zero result.

00	For a non-zero result, the "exclusive or" instruction will set bits 34-35 of the PSW to ____.

TEST UNDER MASK INSTRUCTION

01	So far, you can turn off, turn on, or change a program switch by use of the "and, or, exclusive or" instructions, respectively. However, you still can't test them. The "branch on condition" instruction is not sufficient. This instruction will only let you find out if all switches are either on or off. To be able to test a specific switch (or some but not all switches) will require another instruction. This instruction is known as "test under mask." The "test under mask" instruction will let you examine specific bits (program switches) and set the condition code accordingly. Then the "branch on condition" instruction can be used effectively. Read the description of the "test under mask" instruction in the Logical Operations section of your Principles of Operation manual.
	TM is the mnemonic for " _____ _____ _____."
"test under mask"	The TM instruction uses the ___ format. The instruction can be used to test program switches. These switches must be in _____ (main storage/general registers).
SI main storage	The "test under mask" instruction will test one _____ (byte/halfword/word).
byte	A byte can contain 8 program switches. The TM instruction can test _____ (only one/all) of them at one time.
all	The TM instruction can, if desired, test as few as _____ program switch(es) at a time.
one	The bits (program switches) to be tested with the TM instruction are determined by the instruction's ____ field.
I2	The I2 field corresponds bit-by-bit with the main storage byte to be tested. If all bits in the main storage byte are to be tested, the I2 field must contain _____ (in hex).
FF	To test only bit position 0, the I2 field of the TM instruction must contain ____ (in hex).

80

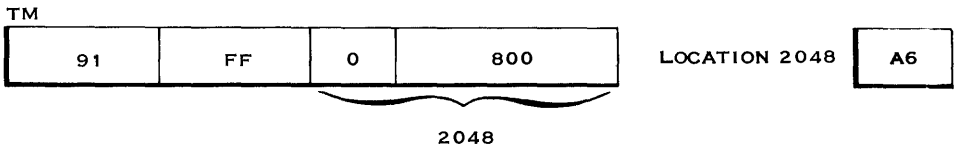
The function of the TM instruction is to set the condition code. A condition code of 00 would indicate that all of the selected bits are zero. Assume that the I2 field of a TM instruction is FF. A condition code of 00 would indicate that the storage byte contains ____ (in hex).

00; An I2 field of FF would test all bits. A condition code of 00 would then indicate that all bits of the byte are zero.

If a TM instruction results in a condition code of 11, it would indicate that all of the selected bits are one. If the I2 field of the instruction used contained FF, it would mean that the storage byte contains _____ (in hex).

FF

A condition code of 01 is also possible after executing a TM instruction. This condition code (01) would indicate that some but not all of the selected bits contain a one. Given the following, what would the resulting condition code be?

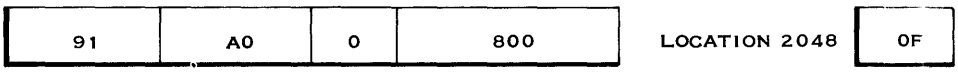


Condition Code = _____

01

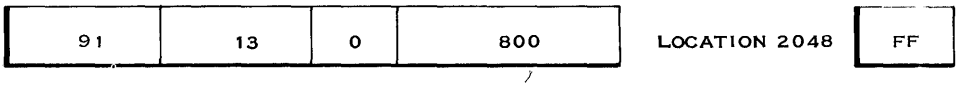
After executing a TM instruction, it is not possible to have a condition code of ____.

10



The above "test under mask" instruction would result in a condition code of ____.

00



The above TM instruction would result in a condition code of ____.

11

91	FA	0	800	LOCATION 2048	6A

The above "test under mask" instruction would result in a condition code of ____.

01

91	4D	0	800	LOCATION 2048	B2

The above TM instruction would result in a condition code of ____.

00

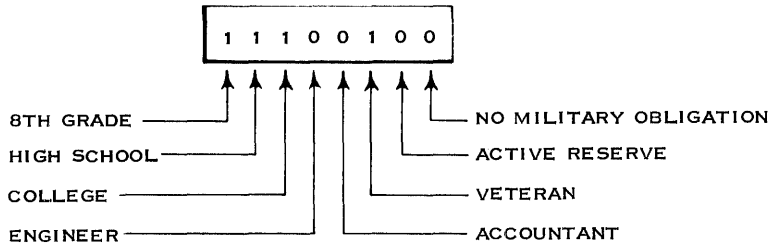
91	10	0	800	LOCATION 2048	21

The above TM instruction would result in a condition code of ____.

00

At this point, you can turn program switches on and off. You can also test them. The instructions that you have been using can be used for purposes other than program switches.

The "and, or exclusive or" instructions can also be used to examine records for specific requirements. Consider the case of a program where it is desired to find all employees whose qualifications fit a particular job description. The following byte will show the minimum requirements for the job.

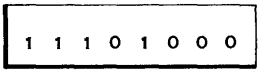


According to the preceding requirements, the employee must have a _____ degree. However, he does not need to be an _____ or an _____.

---

college
engineer
accountant

Assume that a card column in each employee's record is punched to show his qualifications. The following byte shows the qualifications of one such employee.



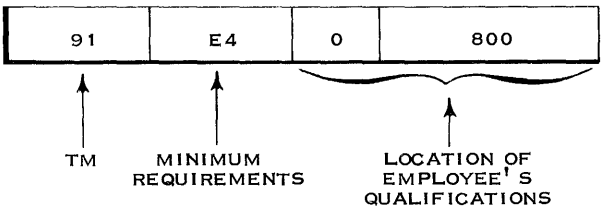
Examine the preceding employee's qualifications and refer back to the requirements for the job. Employee A is not qualified because he is not a _____.

---

veteran

The question now is: How can the "and, or, exclusive or" instructions be used to determine whether an employee meets the requirements?

To determine whether an employee meets the minimum requirements, the "test under mask" instruction could be used. This is shown as follows:



With the TM instruction, the condition code is set according to the status of the selected bits. To meet the minimum requirements, the selected bits in the employee's qualifications must be all _____ (ones/zeroes).

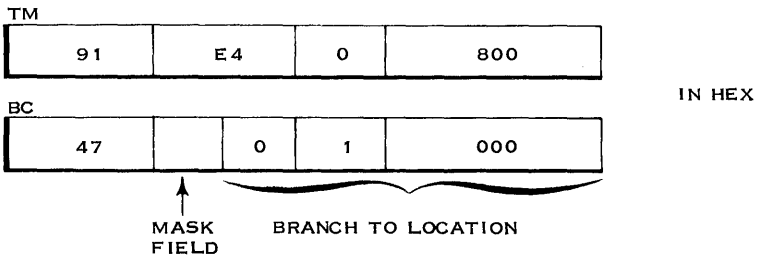
---

ones

If the selected bits are all ones, the TM instruction would cause the condition code to be set to ____ (00/11).

11

The TM instruction could then be followed by a "branch on condition." Write the mask field necessary for a successful branch when the employee meets the minimum requirements. If necessary, review the BC instruction mask field on page 3 of this book or in the Principles of Operations manual.



---

1 ← Hex; 0001 ← Binary
↘ Tests for a condition code of 11

Now suppose that a branch is not desired when the employee meets the minimum requirements. Write the necessary mask field in the BCR instruction.

TM

91	E4	0	800

BCR

17		1

E ← Hex; 1110 ← Binary

Since a condition code of 10 is not possible after a "test under mask" instruction, the following could also be used for the mask field:

C ← Hex; 1100 ← Binary

Suppose that we now change the conditions. We have been looking for an employee who meets the minimum requirements. We did not care if he exceeded those requirements.

Given the following requirements, circle those employees who <u>at least</u> meet the minimum requirements.

Requirements → 1 1 1 0 0 1 0 0

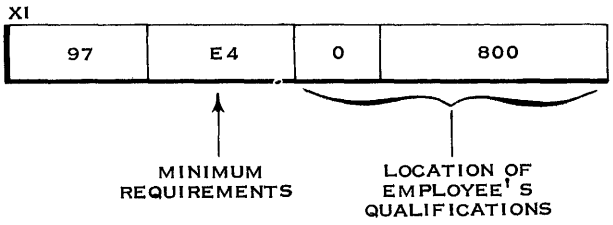
Employee A	1 1 1 0 1 0 0 0
Employee B	1 1 0 0 0 1 1 0
Employee C	1 1 1 0 1 1 0 0
Employee D	1 1 1 1 0 1 0 0
Employee E	1 1 1 0 0 0 0 0

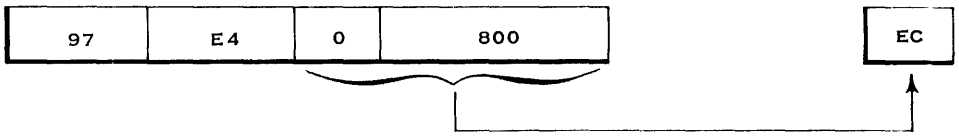
Employee C
Employee D

Referring to the preceding problem, did any of the employees meet the minimum <u>without exceeding</u> the minimum requirements? _____

No

The "exclusive or" instruction can be used to find those employees who meet but do not exceed the minimum requirements. This is shown as follows:





Given the above "exclusive or" instruction and the qualifications of Employee C, the condition code will be set to ____ (00/01).

---

01 ← Indicates a non-zero result. This is shown as follows:

Minimum requirements ⟶ E4 ⟶ 1 1 1 0 0 1 0 0
Employee C Qualifications ⟶ EC ⟶ 1 1 1 0 1 1 0 0
Non-zero Result ⟶ 0 0 0 0 1 0 0 0

If employee C had met without exceeding the minimum requirements, the condition code would have been set to ____ (00/01).

---

00 ← Indicates a zero result.

You have been studying the "and, or, exclusive or and test under mask" instructions. You have seen that their usage is not limited to program switches. Let's continue our study of logical instructions.

INSERT CHARACTER - STORE CHARACTER INSTRUCTIONS

The two instructions that you will study next are the "insert character" and "store character" instructions. Read the description of these instructions in the Logical Operations section of your Principles of Operation manual.

IC is the mnemonic for the "_____ _____" instruction.

STC is the mnemonic for the "_____ _____" instruction.

"insert character"
"store character"

Both the IC and STC instructions use the ___ instruction format. These instructions _____ (change/do not change) the condition code.

RX
do not change

The IC and STC instructions use a storage-to-register concept. These two instructions transfer one _____ (byte/halfword/word) of data.

---

byte

The "insert character" instruction will place the storage operand in bits ____ through ____ of the specified register.

24, 31; The low-order byte.

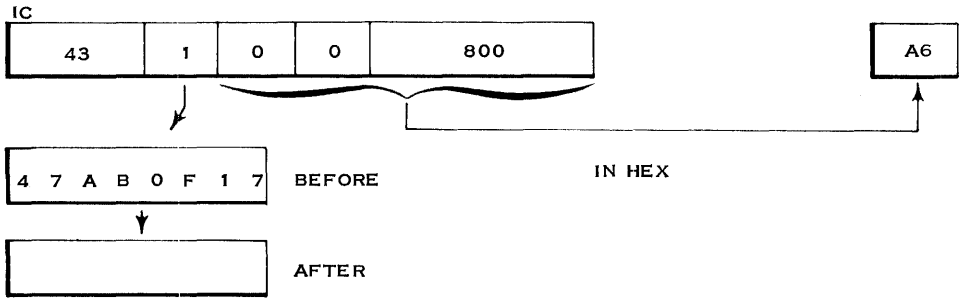
The remaining bits (0-23) of the specified register: (Circle one)

a. Are zeroed out.
b. Remain unchanged.

---

b

Given the following IC instruction, show the resulting contents of the specified register.



---

4 7 A B 0 F A 6

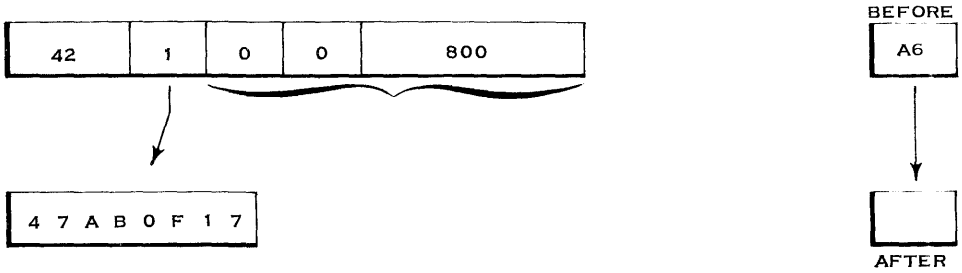
Was the condition code changed by the preceding instruction? _____

No

The "store character" instruction will place in the storage operand the contents of bits ____ through ____ from the specified register.

24, 31

Given the following STC instruction, show the resulting contents of the storage location.



17

If the address of the storage operand is not available on the particular installation, an _____ exception will be recognized.

---

addressing

Any instruction that changes the contents of main storage is subject to the storage protection feature. As a result, the _____ (IC/STC) instruction can cause a protection exception.

STC

A protection or addressing exception will cause a _____ interrupt.

LOAD ADDRESS INSTRUCTION

---

program

Read the description of the "load address" instruction in the Logical Operations section of your Principles of Operation manual.

LA is the mnemonic for the "_____ _____" instruction.

"load address"

The LA instruction has an Op code of 41 (in hex). From bits 0-1 of this Op code you can tell that it uses the __ __ format.

RX

Bits 0-1 of Op Code	Format
00	RR
01	RX
10	RS, SI
11	SS

The "load address" instruction will place in the specified register: (Circle one of the following.)

a. A word from main storage.
b. The generated storage address.

---

b

The generated 24-bit storage address will be placed in bits ___ through ____ of the specified register.

8, 31

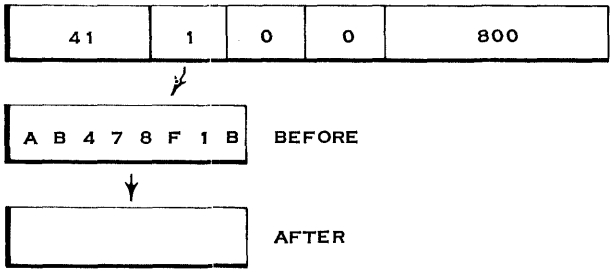
As a result of an LA instruction, bits 0-7 of the specified register: (Circle one of the following.)

a. Remain unchanged.
b. Are zeroed out.

---

b

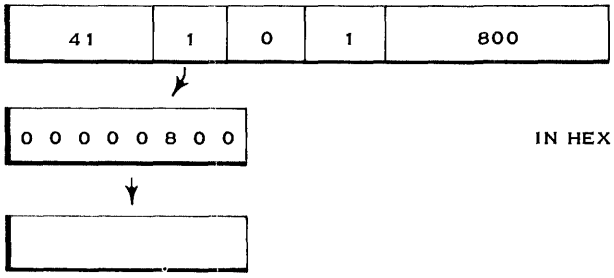
Given the following "load address" instruction, show the resulting contents of the specified register.



---

0 0 0 0 0 8 0 0

Given the following "load address" instruction, show the resulting contents of the register.



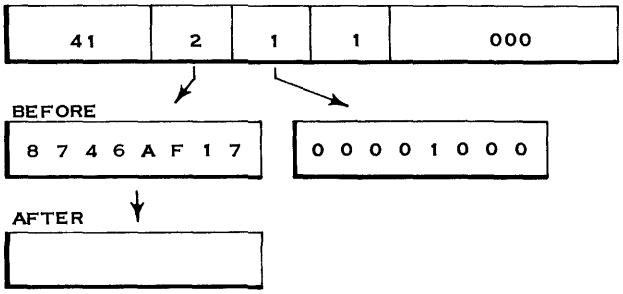
---

0 0 0 0 1 0 0 0; In the preceding problem, the contents of register 1 was used as a base address in generating an effective storage address:

	Base Address	0 0 0 8 0 0	
+	Displacement	8 0 0	Hex Addition
	Effective Address	0 0 1 0 0 0	

The effective address was then placed in register 1.

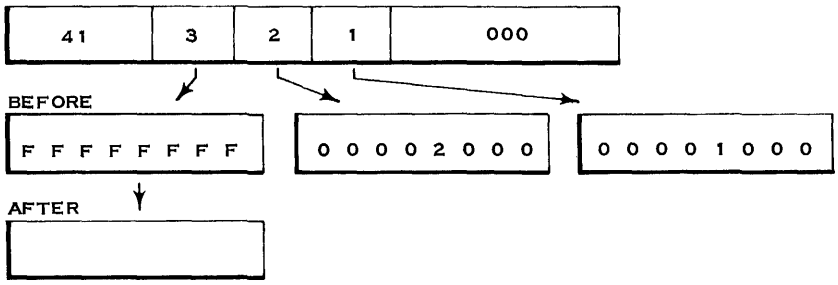
Given the following "load address" instruction, show the resulting contents of the register.



---

0 0 0 0 2 0 0 0

Given the following "load address" instruction, show the resulting contents of the register.



---

0 0 0 0 3 0 0 0

The preceding instructions showed how successive base addresses could be loaded into general registers. Let's take another look at these instructions in a symbolic program.

```
LA    1, 2048 (0, 0)
LA    1, 2048 (0, 1)
LA    2, 0 (1, 1)
LA    3, 0 (2, 1)
```

Examine the preceding program. Then indicate below (decimally) the base address that will be in each register at the completion of the program.

Register 1 _____
Register 2 _____
Register 3 _____

Reg 1 4096
Reg 2 8192
Reg 3 12288

The last two "logical" instructions for you to study are the "translate" instruction and the "translate and test" instruction. If you do not have systems experience and are unfamiliar with the terms translate or table look up, these instructions will be among the most difficult of those you have encountered. Therefore, let's examine the concept of translating before reading the description of these instructions.

First of all, there is data to be translated. This data may be in any code form we wish. The only code you have studied in the System/360 is EBCDIC. There are, of course, other codes in use with computers. For instance, there is an 8-bit paper tape code and the 8-bit ASCII code. The "translate" instruction will allow us to translate data from one code to another, byte by byte.

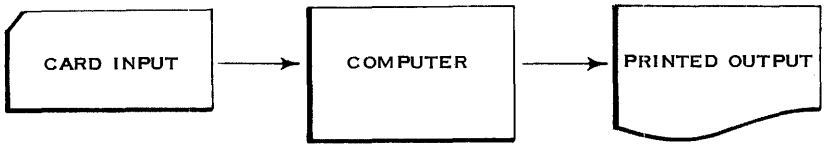
The "translate" instruction will allow us to translate bytes of data: (Circle one of the following.)

a. From one character code to any other character code.
b. Only from EBCDIC to some other character code.
c. Only to EBCDIC from some other character code.
d. Only from EBCDIC to ASCII.

---

a; The bytes to be translated can be in <u>any</u> character code. These bytes can be translated to <u>any</u> other desired code.

Let's look at this concept of translating from a programmer's viewpoint and see how he would handle a <u>simplified</u> translating problem.



The basic job that is to be accomplished is the printing of a report. Input to the system is in the form of _____.

IBM cards

Assume that the cards were punched on a card punch that <u>did not</u> have special character keys. The machine could only punch numeric and alphabetic characters.

The operator who punched the cards used alphabetic symbols to represent the special characters. For example, the character P was used to represent a + sign.

Special Characters	Alphabetic Symbol
+	P
-	M
#	N
$	D
¢	C
&	A

The chart shows how each of the special characters was represented by an alphabetic character.

Given the following listings on a source document, indicate the characters that the operator actually punched in the cards.

Source Document	IBM Card
-79¢	_____
$120+	_____
E & F & # 3	_____

M79C
D120P
EAFAN3

The input cards that are used in our simplified application _____ (do/do not) contain special character punching.

---

do not

The output of this simplified application is to be in _____ form. It is desired to have the listings on the printed report contain the special characters rather than the alphabetic symbols. Therefore, the computer must convert or t_____ the input data before sending it to the printer.

---

printed
translate

Now, let's see how the programmer can use the "translate" instruction to solve the problem just discussed.

First, two tables must be established. They are the function table and the argument table.

The f_____ table consists of the desired characters. In our application, the function table will consist of the _____ (special/alphabetic) characters.

---

function
special

The a_____ table consists of all the data that may have to be converted. In our application, the argument table will consist of the _____ symbols.

---

argument
alphabetic

In the next step, the programmer writes down all the possible data to be converted. Then he arranges it in binary bit sequence and forms the a_____ table.

Argument Table		
A		1100 0001
	US (Unused Symbol)	1100 0010
C		1100 0011
D		1100 0100
	US	1100 0101
	US	1100 0110
	US	1100 0111
	US	1100 1000
	US	1100 1001
	US	1100 1010
	US	1100 1011
	US	1100 1100
M		1100 1101
N		1100 1110
	US	1100 1111
P		1101 0000

---

argument

Now the programmer can make up the f_____ table. The table will indicate where the s_____ characters should be stored so that they can be easily located and used in place of the a_____ symbols (argument table).

---

function
special
alphabetic

Argument Table			Function Table	(Table Address
Argument Bytes			Function Bytes	Storage Locations	is 6807)
A		1100 0001	&	7000	
	US	1100 0010		7001	
C		1100 0011	¢	7002	
D		1100 0100	$	7003	
	US	1100 0101		7004	
	US	1100 0110		7005	
	US	1100 0111		7006	
	US	1100 1000		7007	
	US	1100 1001		7008	
	US	1100 1010		7009	
	US	1100 1011		7010	
	US	1100 1100		7011	
M		1100 1101	-	7012	
N		1100 1110	#	7013	
	US	1100 1111		7014	
P		1101 0000	+	7015	

The function table is actually located in s_____.

The argument table is made up on paper by the programmer. Its only use is to create the f_____ table in _____.

---

storage
function
storage

Go to the Principles of Operation manual and read the description of the "translate" instruction in the Logical Operations section. Do not read the description of the "translate and test" instruction at this time.

The byte or bytes in the first operand are the characters that are to be converted or t_____. They are called a_____ bytes. In the simplified application that you just studied, a first operand argument byte could be a ____ (D/$).

---

translated
argument
D

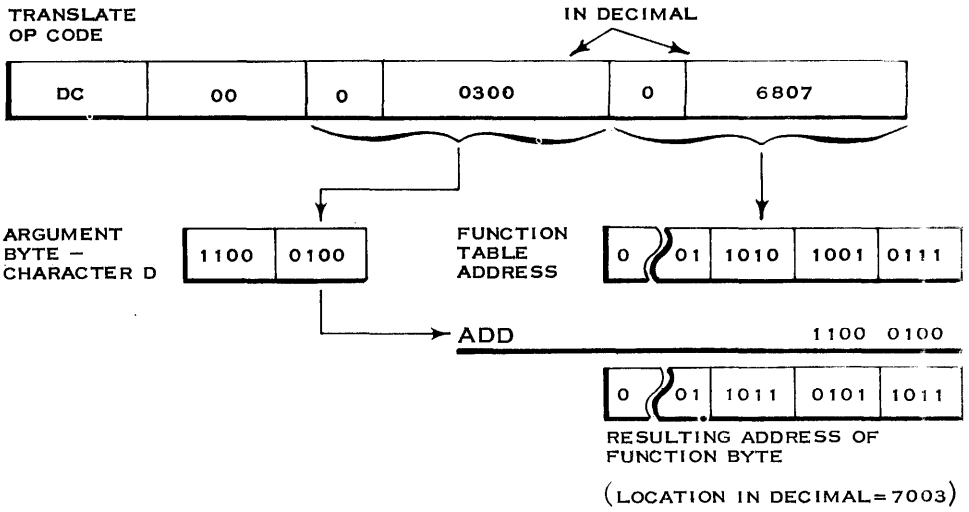
The second operand is the f_____ table. In the simplified application, the address of the second operand would be _____.

---

function
6807

The "translate" instruction does the following:

1. Takes the binary bit value of an argument byte and adds it to the second operand's address.
2. The resulting address is used to locate a function byte.
3. The function byte replaces the argument byte (1st operand).



The above instruction will translate ___ byte(s). The character to be translated is a ____. The instruction goes to location 7003 in the function table (refer to the preceding simplified function table) and finds a ____ character. It takes this $ character and puts it in location _____ where it replaces the D.

1
D
$
0300

Now, let's go back and review the entire concept of translating and use a more typical application.

To translate, a table of the desired code must be available. For instance, assume that we wished to translate from EBCDIC to the 8-bit ASCII code. For simplicity we will only deal with the characters A-H. As a result our table will be only 8 bytes long.

FUNCTION BYTES

1 0 1 0 0 0 0 1	– A	TABLE OF 8 BIT ASCII (A TO H)
1 0 1 0 0 0 1 0	– B	
1 0 1 0 0 0 1 1	– C	
1 0 1 0 0 1 0 0	– D	
1 0 1 0 0 1 0 1	– E	
1 0 1 0 0 1 1 0	– F	
1 0 1 0 0 1 1 1	– G	
1 0 1 0 1 0 0 0	– H	

The above table is located in main storage in 8 successive byte locations. As you can see, the bytes in the table are called _____ bytes.

---

function

The function bytes represent: (Circle one of the following.)

a. The bytes to be translated.
b. The desired character code.

---

b; The desired character code.

Besides the table of function bytes, which represent the desired code, there must also be data bytes which need translation. The following is a five-character record which needs translating:

ARGUMENT BYTES

1 1 0 0 0 1 1 0	– F	DATA TO BE TRANSLATED
1 1 0 0 0 0 0 1	– A	
1 1 0 0 0 1 0 0	– D	
1 1 0 0 0 1 0 1	– E	
1 1 0 0 0 1 0 0	– D	

The bytes to be translated are called _____ bytes.

---

argument

The above record of five EBCDIC characters is to be translated by using a table of f_____ bytes.

---

function

The "translate" instruction consists of replacing the characters to be translated with the characters of the desired code. In other words, the _____ bytes are replaced with the correct _____ bytes.

---

argument
function

The "translate" instruction will replace all of the argument bytes with the desired characters from the function table.

Given the following function table and argument bytes, show the resulting contents of the argument field.

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

FUNCTION TABLE OF ASCII BYTES

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

ARGUMENT FIELD OF FIVE EBCDIC CHARACTERS

---

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

You should now know what is meant by a function byte or an argument byte. You should also realize that the argument bytes are to be replaced by the desired function bytes. We can review the translating concept by asking ourselves "How does the machine know which function bytes to select?" The answer lies in the organization of the function table. This table must be arranged so that the desired characters match the binary sequence of the argument table. This is shown as follows:

EBCDIC	ASCII
00000001	10100001
00000010	10100010
00000011	10100011
11110111	01010111
11111000	01011000
11111001	01011001

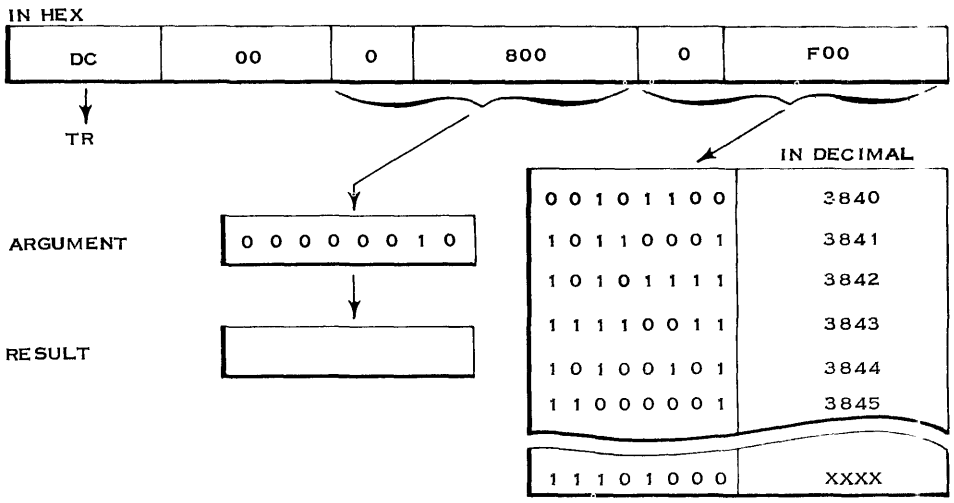
Table of all possible argument bytes is arranged on paper, in binary bit sequence. The table is used to develop the correct sequence for the function table.

Table of function bytes is arranged to match the respective argument bytes.

	The table of FUNCTION bytes in storage is arranged so that: (Circle one of the following.) a. The function bytes are in binary sequence. b. The binary sequence of the argument bytes determines the sequence of FUNCTION bytes.
b	Go back to the Principles of Operation manual and reread the description of the "translate" instruction.
	TR is the mnemonic for the " _____ " instruction.
"translate"	The "translate" instruction has a hexadecimal Op code of DC. From bits 0-1 of this Op code you can tell that the TR instruction uses the ___ ___ format.
SS	The 1st operand of the TR instruction represents: (Circle one of the following.) a. The bytes to be translated. b. The desired coded bytes.
a	The 2nd operand of the TR instruction represents: (Circle one of the following.) a. The function bytes. b. The argument bytes.
a	The function table must be long enough to take care of all expected bit combinations of the argument bytes.
	The length code refers to: (Circle one of the following.) a. The argument bytes. b. The function bytes. c. Both argument and function bytes.

a To find the desired character in the function table, the argument byte is added to the beginning of the table.

Given the following argument byte, what bit combination will replace it?



10101111; The 1st operand's binary bit value (decimal 2) was added to the 2nd operand's address (decimal 3840). The byte at location 3842 replaced the 1st operand.

You should now understand why the function table must be arranged according to the binary sequence of the argument bytes. This is because the argument byte is added to the initial table address. The coded character at that location then replaces the argument byte.

TR

DC	17	0	800	0	F00

Given the above "translate" instruction. How many argument bytes will be translated? _____ (Remember that the instruction is shown in hex.)

24; 17 in hex equals 00010111 in Binary which equals 23 in Decimal,and a length code always is one less than the total number of bytes.

Given the following "translate" instruction, show with six hex digits the starting address of the data bytes. The data bytes are the argument bytes.

TR

DC	17	0	800	0	F00

Starting Address of Data Bytes = _____

000800

TR

DC	17	0	800	0	F00

Given the above "translate" instruction, show with six hex digits the starting address of the function table. _____

000F00

DC	17	0	800	0	F00

TR

Given the above "translate" instruction, how many bytes are in the function table? _____ This question can be tricky, so answer carefully.

Unknown; The proper function byte is selected from the table by adding the argument byte to the starting address of the table. As a result, the table might contain a maximum of 256 bytes. This would depend on the total number of characters in the codes involved.

ARGUMENT BYTE – 00110001 CHARACTER ADDRESS (IN HEX) = _____

TR

DC	17	0	800	0	F00

Given the above "translate" instruction, show with six hex digits the address of the character that will replace the argument byte.

000F31; As shown below.

The following byte is added to the starting address of the function table:

Shown in Hex

000F00 – Table Address
31 – Argument Byte
000F31 – Address of function byte selected to replace the argument byte.

ARGUMENT BYTE – 11001001 CHARACTER ADDRESS (IN HEX) = _____

TR

DC	17	0	800	0	F00

Given the above "translate" instruction and one of the argument bytes, show with six hex digits the address of the function byte that will be selected.

000FC9

ARGUMENT BYTE – 11110111 CHARACTER ADDRESS (IN HEX) = _____

TR

DC	17	0	800	0	F00

Given the above, show the six hex digits of the address of the selected function byte.

000FF7

Given the following data, show the contents of the argument field after the "translate" instruction is executed.

TR IN HEX

DC	07	0	800	0	F00

LOCATIONS	Data Before	Data After
2048	F7	____
2049	F2	____
2050	61	____
2051	F2	____
2052	F5	____
2053	61	____
2054	F3	____
2055	F2	____
IN DECIMAL	IN HEX	

Function Table

LOCATIONS		
3840	0A	
3937	11	
3938	00	
4080	0A	
4081	01	
4082	02	
4083	03	
4084	04	
4085	05	
4086	06	
4087	07	
4088	08	
4089	09	
IN DECIMAL	IN HEX	

Location	After
2048	07
2049	02
2050	11
2051	02
2052	05
2053	11
2054	03
2055	02

The "translate" instruction can be summarized as follows:

1. The translation will be done by replacing an argument byte with a function byte from a table.

2. The length code (L field) gives the number of the argument bytes less one. A length code of 7 would indicate 8 argument bytes.

3. The address of the 1st operand is the address of the argument bytes (those to be translated).

4. The address of the 2nd operand is the address of the function table (those bytes which will be used to replace or translate the argument bytes).

5. In order to obtain the proper function bytes, the table must be arranged according to the binary bit sequence of the argument bytes.

6. The argument byte is added to the function table address. The resulting address is used to select a byte from the function table and replace the argument byte with it.

7. The "translate" instruction continues until all the argument bytes (determined by the length code) have been translated.

TRANSLATE AND TEST INSTRUCTION

Read the description of the "translate and test" instruction in the Logical Operations section of your Principles of Operation manual.

In the "translate" instruction, the argument bytes are replaced with function bytes. In the "translate and test" instruction, the argument bytes _____ (are replaced with function bytes/ remain unchanged).

---

remain unchanged

Is any translation actually done by the "translate and test" instruction? _____

No

The "translate and test" instruction tests the argument bytes by selecting the corresponding function bytes. The test results are recorded by changing the _____ _____.

---

condition code

How does the machine know which function bytes are to be tested? _____

It adds the argument byte to the starting address of the function table. The function byte at the resulting address is then tested.

The selected function byte is tested to see if it is _____.

---

zero

What happens if the function byte is zero? _____

The operation continues with the next argument byte being added to the table address and another function byte being selected.

If all of the function bytes selected by the argument bytes are zero, the operation is completed by setting the condition code to _____.

---

00 (Binary) which is 0 in hex.

After a "translate and test" instruction, a condition code of 00 would indicate: (Circle one.)

a. That one of the selected function bytes was zero.
b. That all of the selected function bytes were zero.
c. That none of the selected function bytes were zero.
d. That all of the argument bytes were zero.

---

b; A condition code of 00 would indicate that all of the argument bytes had been used in selecting function bytes. It would also mean that all of the selected function bytes were zero. It does not mean that all of the function bytes in the table are zero. It means that the selected ones were zero.

The "translate and test" instruction is used to examine a data field (the argument bytes) for characters with special meaning. The function table would again be arranged (as in the "translate" instruction) according to the binary sequence of the data code.

For all characters that do not have a special meaning (non-significant characters), the function byte location would contain zero.

For all characters that do have a special meaning (significant characters), the function byte location would contain some non-zero bit configuration.

A resulting condition code of 00 would then indicate that the entire data field had been examined and that no significant characters were found. By significant characters, we mean those with special meaning in a data field.

If a character with special meaning (significant character) is found, the instruction is terminated. A significant character would be indicated by selecting a function byte that was _____ (zero/non-zero).

---

non-zero

If a significant character is found before the entire data field is examined, the resulting condition code is 01 and the operation _____ (continues/is terminated).

---

is terminated

After a TRT instruction, a condition code of 01 would mean: (Circle one of the following.)

a. No significant character was found.
b. All the argument bytes were used and a significant character was found.
c. A significant character was found.
d. One or more significant characters were found.

---

c; As soon as a significant character is found, the operation is terminated without testing any more bytes.

A condition code of 01 then means that a significant character was found and some argument bytes haven't been tested. If the last argument byte is significant, the condition code is set to 10.

After a TRT instruction, a condition code of 10 would mean: (Circle one of the following.)

a. All argument bytes were used and none located a non-zero function byte.
b. All of the argument bytes were not used. One of them was significant.
c. The last argument byte located a non-zero function byte.
d. All the argument bytes were used. One or more were significant.

---

c

After a "translate and test" instruction, which of the following condition codes would indicate that the entire field of argument bytes hasn't been examined? (Circle one of the following.)

00
01
10

01

Which of the following condition codes would indicate that none of the argument bytes had special meaning? (Circle one of the following.)

00
01
10

00

Either a condition code of 01 or 10 will indicate that a significant character was found. Why then does the programmer need both settings?

If the code were 01, the programmer would have to execute the TRT instruction again to see if the remaining argument bytes contained any characters with special meaning.

DECIMAL DATA FIELD IN STORAGE (COMPOSED OF ARGUMENT BYTES)

A	1	B	,	3	,	6	X

The purpose of the TRT instruction is to find significant characters in a data field. In the example above, the instruction could be used to find the location of commas in a decimal field. It would not make sense to know that there is a significant character without knowing where it is located. As a result, the TRT does more than just set the condition code. The address of the significant argument byte is placed in bits 8-31 of register 1.

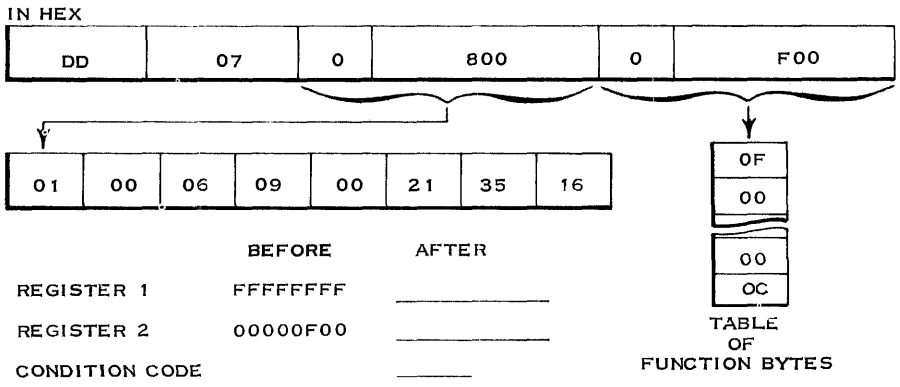
When a TRT instruction results in a condition code of 01 or 10, register 1 will contain: (Circle one of the following.)

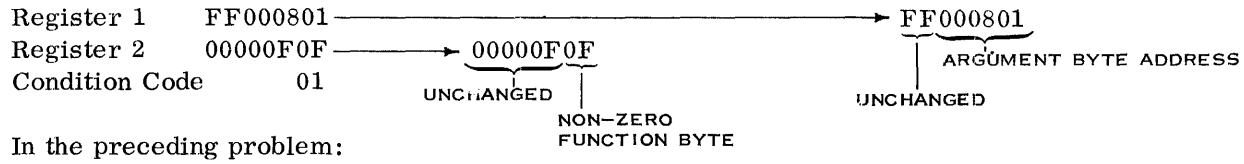
a. The address of a function byte.
b. A function byte.
c. The address of an argument byte.
d. An argument.

c; When an argument byte which contains a significant character is found, its address is placed in register 1. If the condition code is 00, register 1 is unchanged.

Besides placing the address of the significant argument byte in register 1, the TRT instruction will also place the non-zero function byte in bits 24-31 of register 2. The rest of register 2 remains unchanged.

Given the following TRT instruction, show the resulting condition code and the contents of register 1 and 2.





In the preceding problem:

The 1st argument byte pointed to a zero function byte (the second byte in the function table).

The second argument byte pointed to a non-zero function byte (the first byte in the table).

The non-zero function byte is placed in the low order 8 bits of register 2.

The address of the argument byte is placed in the low order 24 bits of register 1. The rest of registers 1 and 2 remains unchanged.

The length code indicates a total of 8 bytes. Since a significant character was detected prior to using all argument bytes, the condition code is 01.

The "translate and test" instruction can be summarized as follows:

1. The TRT instruction uses the SS format in which the length code gives the number of argument bytes less one.

2. The 1st operand consists of the argument bytes (the field that is to be searched for characters that have special meaning).

3. The 2nd operand consists of function bytes. These function bytes are pre-arranged according to the binary sequence of the argument bytes. The locations in this table that match the special meaning argument bytes have non-zero bit configurations.

4. An argument byte is added to the starting address of the function bytes. The function byte at the resulting address is tested for a non-zero bit configuration. If it is non-zero, the operation is terminated. The address of the argument byte is put into register 1 and the corresponding non-zero function byte is placed in register 2. The condition code is set to 01 or 10, depending on whether or not the last argument byte has been translated.

5. If all tested function bytes are zero, the operation is terminated by setting the condition code to 00. Registers 1 and 2 remain unchanged.

SECTION III LEARNING OBJECTIVES

At the end of this section, you should be able to do the following when given the mnemonic of any decimal feature 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.

Decimal Operations

This section of your self-study text covers the eight instructions which make up the decimal (sometimes called commercial) feature of System/360. This feature is optional on models 30, 40 and standard on models 50-70. The instructions are as follows:

Mnemonic	Title
AP	Add Decimal
SP	Subtract Decimal
ZAP	Zero and Add
CP	Compare Decimal
MP	Multiply Decimal
DP	Divide Decimal
ED	Edit
EDMK	Edit and Mark

All of the above instructions use the SS format. As indicated by their mnemonics, both operands of the first six instructions must be in the _____ (packed/zoned) decimal format.

---

packed

The "edit" and "edit and mark" instructions are used to change packed data to the zoned format and insert the punctuation necessary for a printed report.

Besides the eight decimal instructions, one other instruction will also be covered in this section. It is "move with offset." This instruction is part of the System/360's standard instruction set. However, you will find it easier to understand if it is covered here.

Before studying the decimal instructions, read the following introductory material in the Decimal Arithmetic section of your Principles of Operation manual.

Decimal Arithmetic
- Data Format
- Number Representation
- Condition Code
- Instruction Format
- Instructions (do not read the description of the individual instructions)

A numeric field punched in the standard card code will be brought into main storage in the _____ (zoned/packed) format.

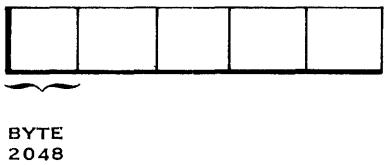
---

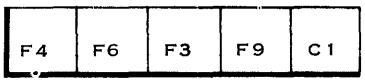
zoned

Assume that columns 21-25 of an IBM card contain the following punches:

Col. 21	4 hole punch
Col. 22	6 hole punch
Col. 23	3 hole punch
Col. 24	9 hole punch
Col. 25	12, 1 hole punches

Show in hex how the above data field would appear after being read into main storage, starting at location 2048.



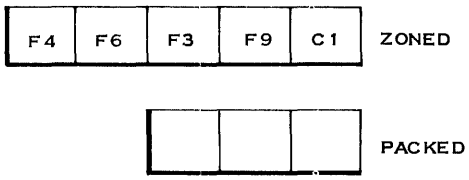


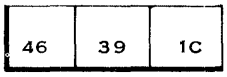
A zoned decimal data field can be changed to the packed format by means of _____.

---

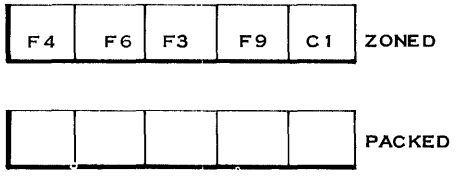
a "pack" instruction; You previously used this instruction when you studied the binary operations. At that time, you used it to change zoned data to packed data, which was then converted to binary with another instruction.

Show how the following data field would look if it were packed into 3 bytes.





Show how the following data field would look if it were packed into 5 bytes.



00	00	46	39	1C

Notice that the resulting field is extended with high-order zeroes.

At this point you may want to review the "pack" as well as the "unpack" instructions. If so, you will find their descriptions in the Decimal Arithmetic section of your Principles of Operation manual.

System/360 carries its negative binary numbers in complement form. On the other hand, its decimal numbers (whether positive or negative) are always carried in _____ (true/complement) form.

---

true

In the packed decimal format, the sign is represented by bits 4-7 of the low-order bytes. Valid sign bit combinations are in the range of _____ to _____.

1010, 1111

The standard minus sign bit combinations are 1101 if operating in EBCDIC Mode or 1011 if in ASCII Mode. This is determined by bit 12 of the PSW.

The standard plus sign bit combinations are 1100 for EBCDIC mode and 1010 for ASCII mode.

PSW bit 12 is 0 - EBCDIC
PSW bit 12 is 1 - ASCII

For the remainder of this section, we will assume that the system is always in EBCDIC mode. That is, PSW bit 12 is zero.

The standard EBCDIC plus and minus sign bit combinations are such that a 12-hole card punch or an 11-hole card punch over the units position of a data field will represent a plus or minus sign respectively. A numeric data field that is punched into a card without the appropriate zone punch over the low-order digit will have a sign bit combination of 1111. That is normal for numbers in the EBCDIC code. This sign bit combination is considered plus. To summarize, these are the sign codes you can expect with the EBCDIC code:

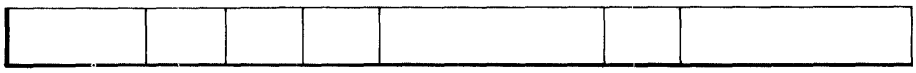
Plus = 1100 (12-hole punch over a digit)
       1111 (no zone punch)

Minus = 1101 (11-hole punch over a digit)

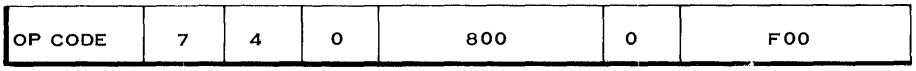
Although the remaining combinations between 1010 and 1111 are valid sign codes, the preceding three combinations are the ones you will more likely encounter.

The instructions of the decimal feature use the SS format. In this format both operands are in main storage. With the exception of the two "edit" instructions, each operand will have its own four-bit length code. That is, the packed decimal operands can be from 1-16 bytes long.

Label the fields of the following SS format.



Given the following SS format.



The 1st operand starts at byte location 2048 and is ___ bytes long.
The 2nd operand starts at byte location 3840 and is ___ bytes long.

ADD DECIMAL INSTRUCTION

8
5

Let's begin our study of the decimal instructions. The first instruction to learn is "add decimal." Read its description in the Decimal Arithmetic section of your Principles of Operation manual.

AP is the menmonic for the " _____ _____ " instruction.

"add decimal"; The ending letter of P tells us that both operands must be in the packed decimal format.

In the "add decimal" instruction, the sum of the 1st and 2nd operands replaces the _____ operand.

1st

Since all packed decimal numbers are in true form, the signs of the field must be analyzed prior to any addition. If the signs are different (one plus and one minus), the 2nd operand is _____ (added to/ subtracted from) the first operand.

---

subtracted from

Subtraction on a computer is done by means of _____ addition.

---

complement

Given the following AP instruction, show the resulting contents of the 1st operand.

AP

FA	3	3	0	800	0	F00

BEFORE	07	42	56	7C	04	31	72	1C
AFTER								

11	74	28	8C

Notice that the sign bits weren't added.

The 2nd operand _____ (was/was not) changed by the preceding problem.

---

was not

In the preceding problem, two positive numbers were added together. The resulting positive sum would set the condition code to ____.

10

Given the following AP instruction, show the resulting 1st operand field and the condition code.

FA	3	2	0	800	0	804

BEFORE	47	44	96	7D		01	47	8D
AFTER					COND. CODE _____			

47	46	44	5D

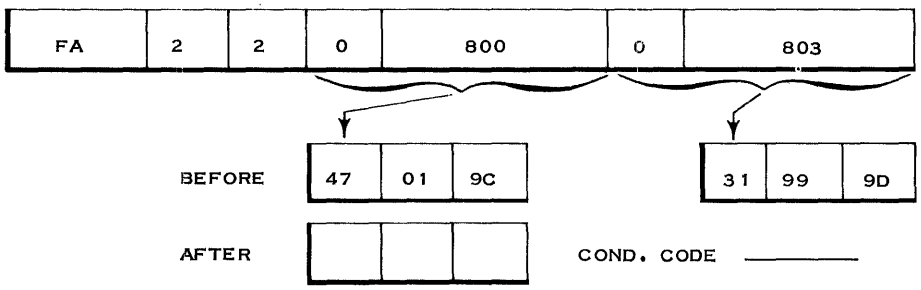
COND. CODE 01

In the preceding problem, two _____ (positive/negative) numbers were added together and the sum was _____ (positive/negative).

---

negative
negative

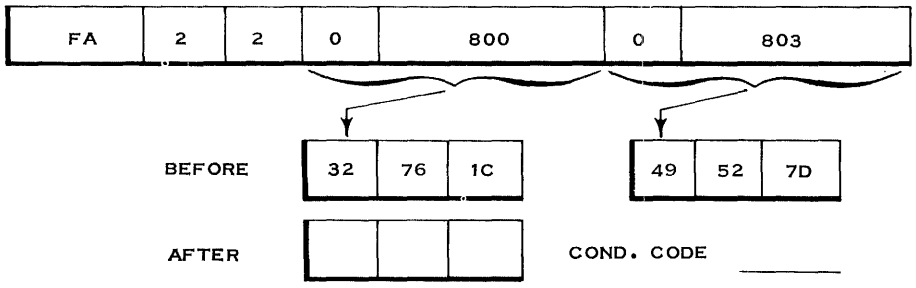
If the signs of the two operands are different, the 2nd operand is effectively subtracted from the 1st operand. Given the following AP instruction, show the resulting 1st operand and condition code.



15	02	0C

COND. CODE 10

Given the following AP instruction, show the resulting 1st operand and condition code.



16	76	6D

COND. CODE 01

As you previously learned, subtraction in a computer is usually done by means of complement addition.

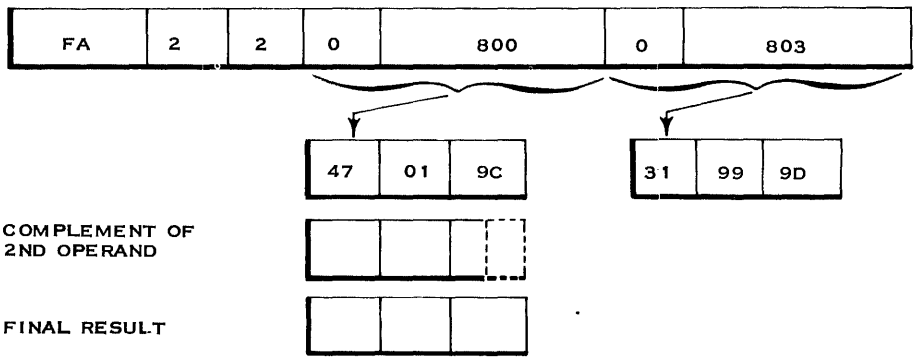
Complement addition consists of adding the two operands after: (Circle one of the following.)

a. Complementing one of the operands.
b. Complementing both of the operands.

---

a

Given the following "add decimal" instruction with operands that have different signs, subtract the operands by complementing the 2nd operand and then adding.



Complement of 2nd operand = 68001

Final Result =

15	02	0C

The result of complement addition could be in either true form or in complement form. How would the system know that the preceding answer was in true form? _____

_____.

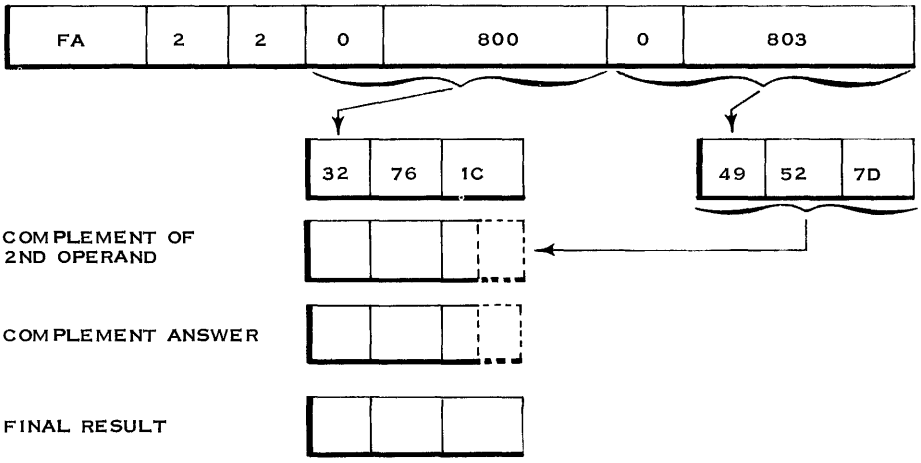
There was a carry out of the high-order digit during complement addition.

If there were no carry out of the high-order digit, the system would know that the answer was in complement form. What must the system do before the instruction is completed? _____

_____.

It must re-complement the answer and change the sign of the result field (1st operand).

Given the following "add decimal" instruction with operands that have different signs, subtract the 2nd operand from the 1st operand by means of complement addition.



50	47	3

83	23	4

16	76	6D

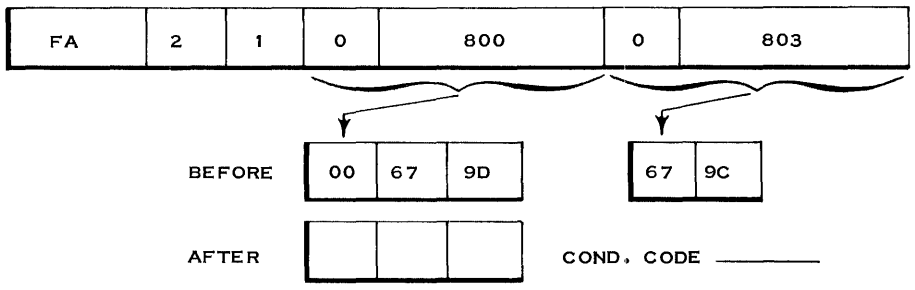
Since decimal data is always carried in true form, the signs must be analyzed. Given the following signs, indicate whether the fields are true or complement added. The instruction is "add decimal." (Circle the correct answer for each set of signs.)

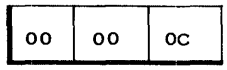
	1st Operand	2nd Operand	
a.	+	+	True add/Complement add
b.	+	-	True add/Complement add
c.	-	-	True add/Complement add
d.	-	+	True add/Complement add

---

a. True add
b. Complement add
c. True add
d. Complement add

Given the following AP instruction, show the resulting contents of the 1st operand and the condition code.





COND. CODE 00

In the previous problem, there are two equal values with different signs. In actual operation, the second operand would have its high order propagated with zeros so that it matches the first operand. These high order zeros would be complemented along with the rest of the second operand.

Since one quantity would be subtracted from the other, the result would be zero as indicated by the condition code setting. A zero result is always plus. That is the reason for changing the sign of the 1st operand from minus to plus.

The original length of the 1st operand will never be exceeded regardless of the result. Carries beyond the 1st operand's high order are lost. When there is a high-order carry, the condition code is set to _____.

11

A carry out of the high order during an AP instruction is called a _____ _____.

---

decimal overflow

A decimal overflow _____ (can/cannot) cause a program interrupt.

---

can

When will a decimal overflow not cause a program interrupt? _____

When the appropriate mask bit in the PSW is set to zero.

Is the "set program mask" instruction a privileged one? _____

No; The problem programmer can change the program mask (PSW bits 36-39) whenever he wishes.

What else can cause a decimal overflow besides a high-order carry?

_____

_____.

The number of significant digits in the 2nd operand exceeding the length of the 1st operand.

Which of the following can cause a decimal overflow on an AP instruction? (Circle one of the following.)

	1st Operand		2nd Operand		
a.	47	9C	52	0C	
b.	98	1C	22	7D	
c.	47	2C	00	37	6C
d.	None of the above				

---

d

Which of the following can cause a decimal overflow on an AP instruction? (Circle one of the following.)

	1st Operand		2nd Operand		
a.	22	7C	00	90	7C
b.	50	0D	50	0D	
c.	04	7C	01	00	1C
d.	All of the above				

---

d

Besides a decimal overflow, there are other programming exceptions that can occur on an "add decimal" instruction. They are:

1. Operation - If the decimal feature is not installed on a system, any of the eight decimal instructions are considered illegal.

2. Protection - Since the results of the instruction replace the contents of main storage, this instruction is subject to a storage protection violation. The protection exception occurs if the storage key does not match the protection key in the PSW.

3. Addressing - Any instruction which addresses main storage for an operand is subject to an addressing exception. This exception occurs when the address is not available on a particular system (such as an address 16000 on an 8K system).

4. Data - All packed decimal operands are checked for valid digits and sign. All of the digit positions must be coded from 0000-1001. The sign position must be coded from 1010-1111.

What would happen if the "add decimal" instruction were used to add two zoned decimal fields? _____

A data exception would be recognized and a program interrupt would occur.

Would a protection exception be recognized on an "add decimal" instruction if the storage key were 4 and the PSW's protection key were zero? ____

No; The two keys do not need to match if the protection key is zero.

The decimal feature is optional on models 30 and 40 of System/360. What would happen if an "add decimal" instruction was fetched on a model 30 which doesn't have the decimal feature installed? _____

SUBTRACT DECIMAL INSTRUCTION

An operation exception would be recognized and a program interrupt would occur.

The next instruction to be covered is the "subtract decimal" instruction. Read the description of the SP instruction in the Decimal Arithmetic section of your Principles of Operation manual.

The mnemonic for the "subtract decimal" instruction is _____.

SP

The operation of "subtract decimal" instruction is similar in all respects to the "add decimal" instruction. The only difference is that the AP instruction adds and the SP instruction subtracts.

Given the following signs, indicate whether the fields will be true or complement added on an AP instruction. (Circle the answers.)

	1st Operand	2nd Operand	
a.	+	-	True add/Complement add
b.	+	+	True add/Complement add
c.	-	-	True add/Complement add
d.	-	+	True add/Complement add

---

a. Complement add
b. True add
c. True add
d. Complement add

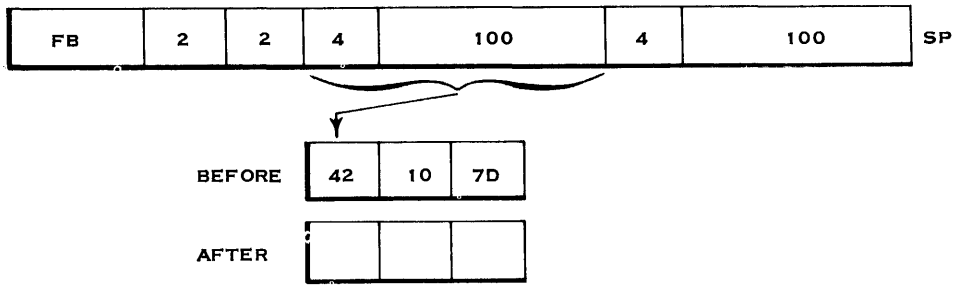
Given the following signs, indicate whether the operands will be true or complement added on an SP instruction. (Circle the answers.)

	1st Operand	2nd Operand	
a.	+	+	True add/Complement add
b.	+	-	True add/Complement add
c.	-	+	True add/Complement add
d.	-	-	True add/Complement add

---

a. Complement
b. True
c. True
d. Complement

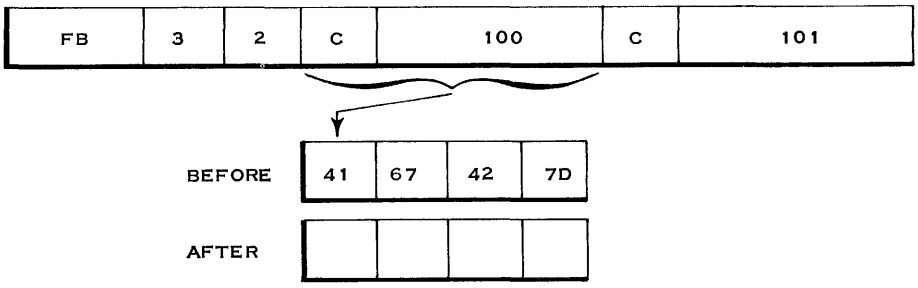
One use of the "subtract decimal" instruction is to zero out a packed decimal field. Show the resulting contents of the 1st operand for the following SP instruction.



00	00	0C

Notice that a zero difference results in a plus sign.

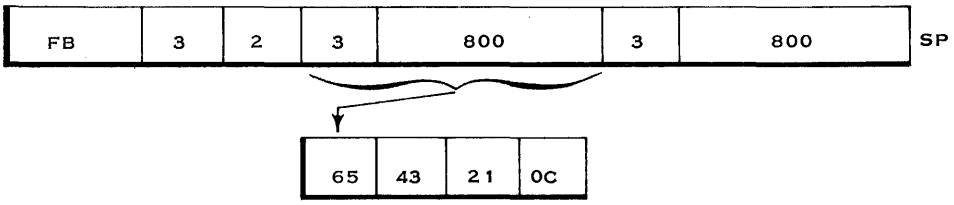
The SP instruction can also be used to zero out the low order of a field. Show the result of the following instruction.



41	00	00	0D

Notice that since only part of the field was zeroed out, the sign remained minus.

What would happen on the following instruction?



Your response: _____

A data exception would be recognized and a program interrupt would occur. This occurs because the 2nd operand's low-order byte contains 21. Bits 4-7 of this byte would be recognized as an invalid sign code.

1ST OPERAND

65	43	21	0C

2ND OPERAND

ZERO AND ADD INSTRUCTION

The next decimal instruction is "zero and add." Its mnemonic is ZAP. Read the description of this instruction in the Decimal Arithmetic section of your Principles of Operation manual.

The ZAP instruction will replace the _____ (1st/2nd) operand with the _____ (1st/2nd) operand.

1st
2nd

Is the 1st operand (data) used on a ZAP instruction? _____

No; It is ignored.

Does the 2nd operand need to be in the packed decimal format or can any type of data be moved by the ZAP instruction? _____
_____.

The 2nd operand must be valid packed decimal data or a data exception will be recognized and cause a program interrupt.

Do both operands on a ZAP have to be of equal length? _____

No

What happens to the extra bytes of the 1st operand when the 1st operand is longer than the 2nd operand? _____.

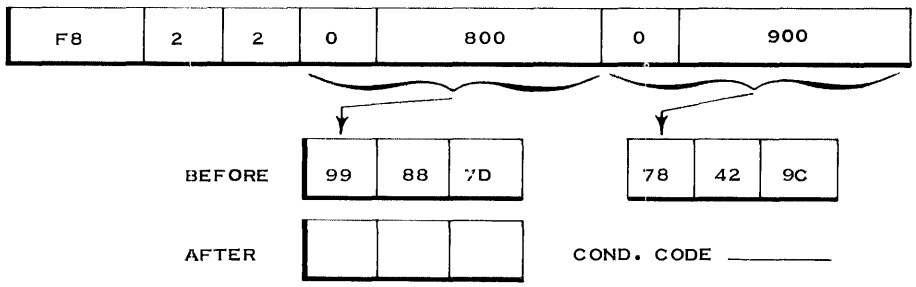
They are zeroed out.

If the 1st operand is too short to contain all of the significant digits from the 2nd operand, a _____ _____ will be recognized.

---

decimal overflow

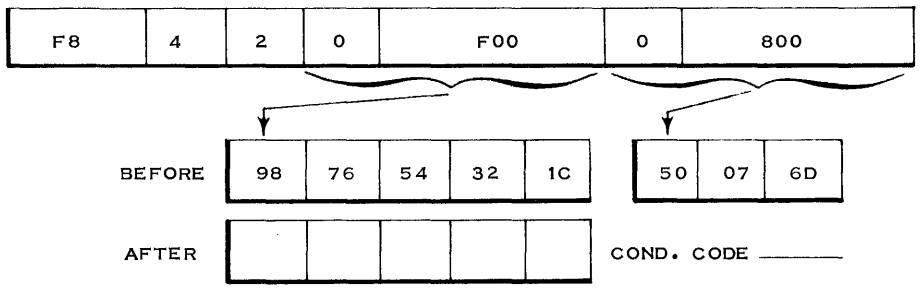
Given the following ZAP instruction, show the resulting contents of the 1st operand and the condition code.



78	42	9C

COND. CODE 10

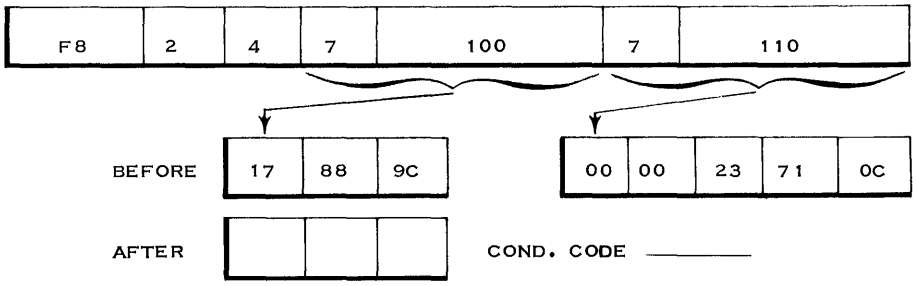
Given the following ZAP instruction, show the resulting contents of the 1st operand and the condition code.



00	00	50	07	6D

COND. CODE 01

Given the following ZAP instruction, show the resulting contents of the 1st operand and the condition code.



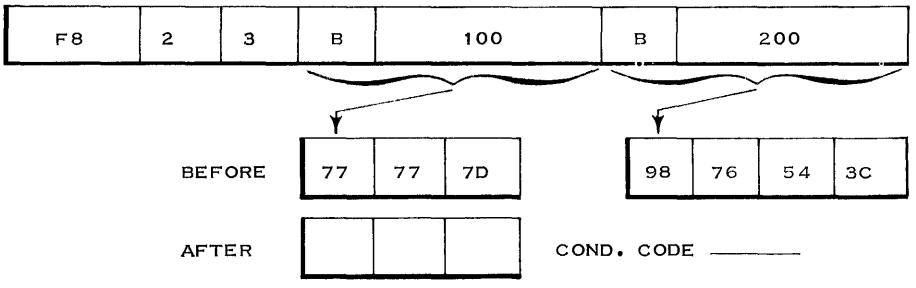
23	71	0C

COND. CODE 10

In the previous problem, the 2nd operand was longer than the 1st operand. Why wasn't a decimal overflow indicated in the condition code? _____

All significant digits from the 2nd operand were able to fit in the 1st operand.

Given the following ZAP instruction, show the resulting contents of the 1st operand and the condition code.



76	54	3C

COND. CODE 11

Will a program interrupt occur after the preceding instruction is executed? _____

Yes; This is assuming that the decimal overflow mask bit in the PSW's Program Mask is set to 1. If the mask bit is set to 0, the program interrupt does not occur. However, since the condition code indicates a decimal overflow, the next instruction could be a "branch on condition."

Besides the data and decimal overflow exceptions, the ZAP instruction is subject to other exceptions. They are: _____, _____ and _____.

COMPARE DECIMAL INSTRUCTION

Operation (if the decimal feature is not installed)
Addressing
Protection

The next instruction you will study is the "compare decimal" instruction. This instruction makes an algebraic comparison of two packed decimal fields. It does not compare alphameric information. The "compare logical" instruction which you previously studied is used for that purpose. Read the description of the "compare decimal" instruction in the Decimal Arithmetic section of your Principles of Operation manual.

If the fields addressed by a CP instruction are not in the packed decimal format, a _____ exception will be recognized.

data

The result of the comparison is recorded in the _____ _____.

condition code

A condition code of 00 would indicate that the operands were _____.

equal

A condition code of 01 would indicate that the _____ (1st/2nd) operand was high.

2nd

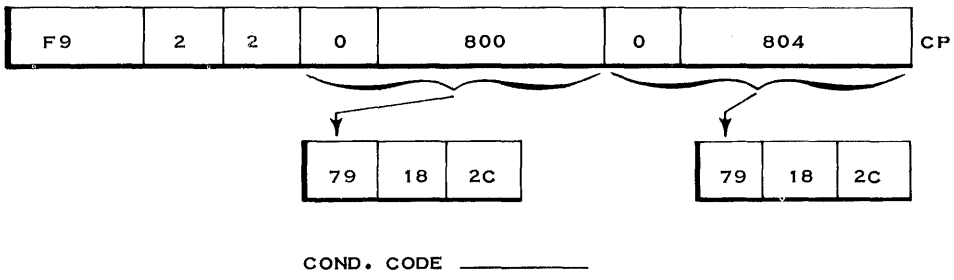
A condition code of 10 would indicate that the _____ (1st/2nd) operand was high.

1st

The "compare decimal" instruction _____ (does/does not) change the operands.

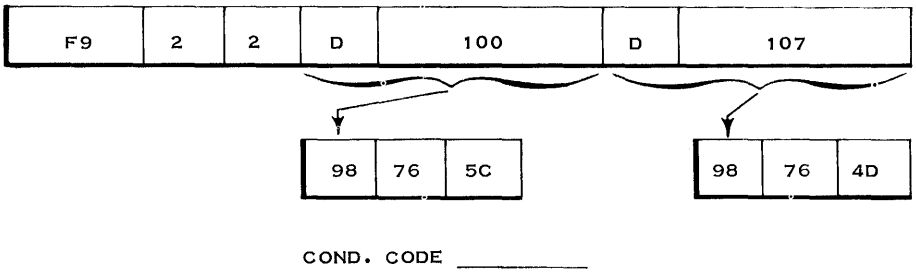
does not

Show the resulting condition code for the following "compare decimal" instruction.



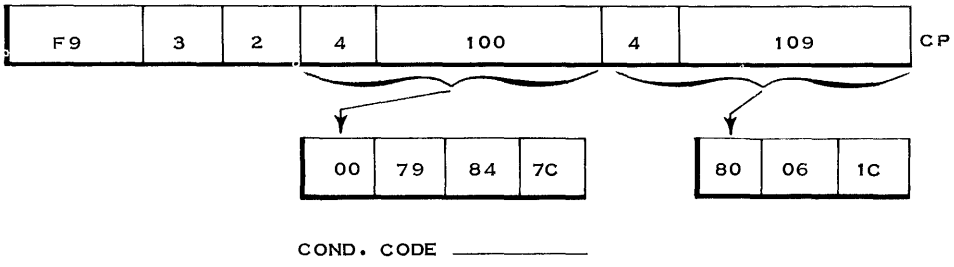
00; Both operands were equal.

Show the resulting condition code for the following CP instruction.



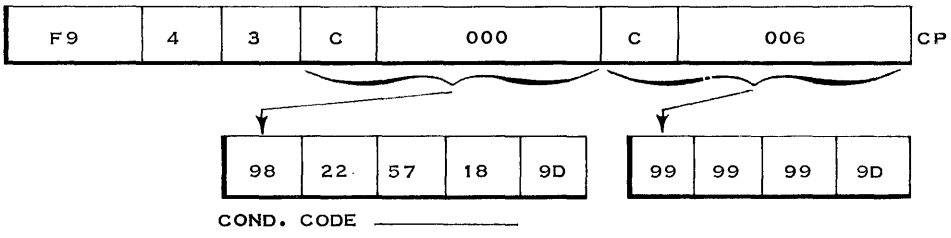
10; Since the 1st operand is positive, it is high.

Show the resulting condition code for the following "compare decimal" instruction.

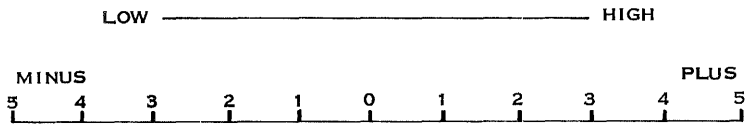


01; Even though the 1st operand is longer, its algebraic value is less than that of the 2nd operand.

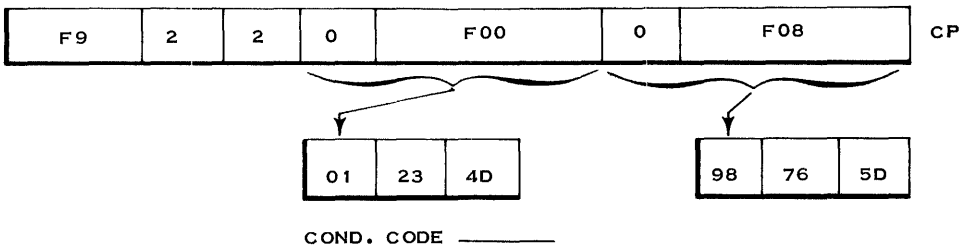
Show the resulting condition code for the following "compare decimal" instruction.



01; The numeric value of the 1st operand is greater; however, both operands are negative. Algebraically, a small negative number is greater than a large negative number as shown below.



Show the resulting condition code for the following "compare decimal" instruction.



MULTIPLY DECIMAL INSTRUCTION

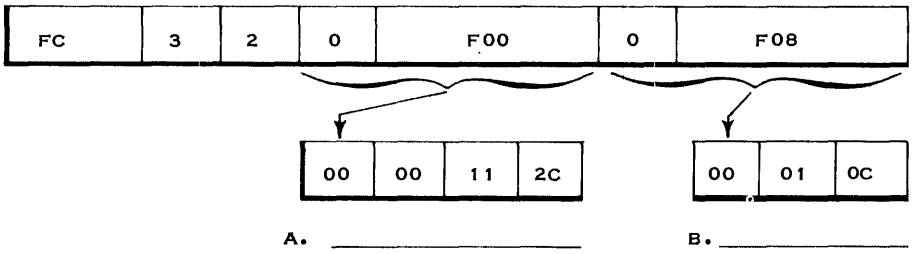
10; The 1st operand is less negative and therefore algebraically greater.

You are ready to study the "multiply decimal" instruction. Its mnemonic is MP and like the other decimal instructions, it operates with packed decimal data. Before reading the description of this instruction, let's take a few simple examples.

In the MP instruction, the 1st operand is the multiplicand. The 2nd operand is the multiplier. As with most instructions, the product will replace the _____ (multiplicand/multiplier).

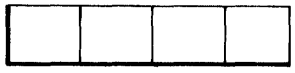
multiplicand; It is the 1st operand.

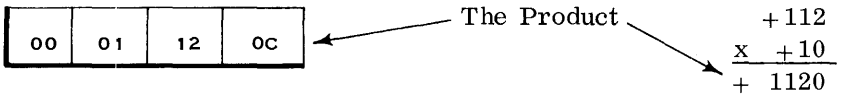
Given the following MP instruction, identify the multiplier and multiplicand.



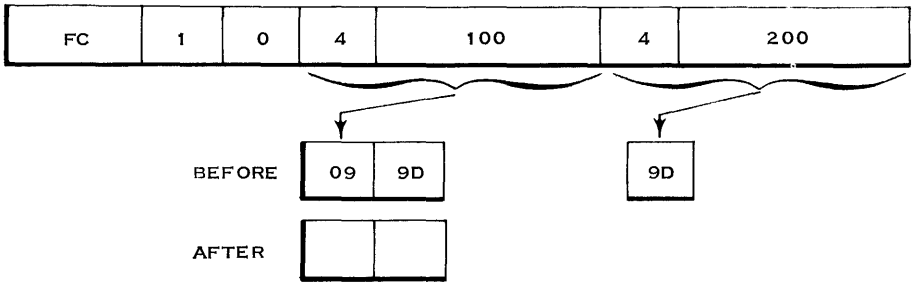
a. Multiplicand
b. Multiplier

For the preceding MP instruction, show the resulting contents of the multiplicand.



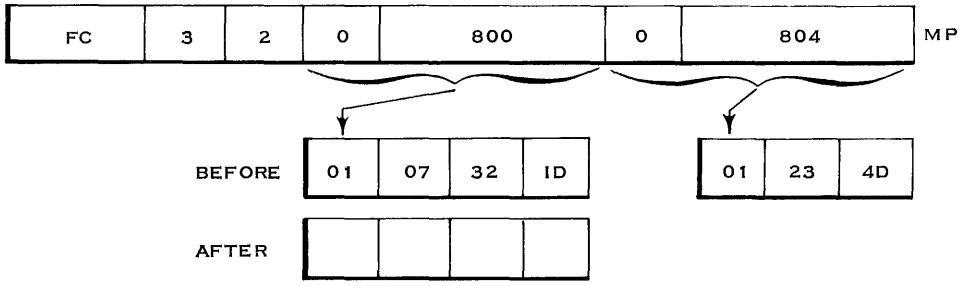


Show the resulting product for the following MP instruction.



89 1C

Like signs give a positive product.



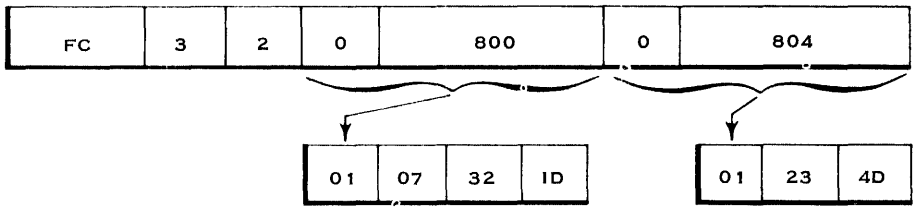
Can the resulting product for the above MP instruction fit into the multiplicand field? _____

No; The rule of thumb is that the number of digits in the product is equal to the sum of the number of significant digits in both operands.

To prevent the product from overflowing the multiplicand field on a "multiply decimal," the System/360 has the following restriction on the multiplicand.

> The number of high-order zeroes in the multiplicand must be at least equal to the number of digits in the multiplier. This includes high-order zeroes in the multiplier. For example:
>
> If the multiplier is 01 23 4D , there must be 5 high-order zeroes in the multiplicand such as:
>
> 00 00 01 07 32 1D

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



The above MP instruction will result in a _____ exception and cause a program interrupt.

---

data; Because the number of high-order zeroes in the multiplicand is less than the size of the multiplier.

What must be done to prevent a specification exception on an MP instruction? _____

The multiplier must be shorter than the multiplicand and cannot have a length code greater than 7 (15 digits and a sign).

One problem that is often encountered after a multiply operation is the placement of the decimal point. For instance, 0001120C multiplied by 00010C equals 0011200C. However, suppose these numbers represented dollars and cents, such as:

$$
\begin{array}{r}
\$\ 11.20 \\
\underline{\$\quad .10} \\
\quad 0000 \\
\quad 1120 \\
\hline
\$1.1200
\end{array}
$$

As you can see, the decimal point was shifted by the multiplication.

What is usually necessary is shifting the product to the right in order to reestablish the proper place for the decimal point. There are no "shift" instructions for the storage-to-storage operations. However, the "move" instructions (which were covered under Logical Operations) can be used to effectively shift storage data.

In our previous example, the product had to be shifted two places to the right in order to maintain the decimal point. For instance:

```
 00011.20C
x    0.10C
-----------
 001.1200C
```

The above product really should be like this: 00001.12C. This can be accomplished by use of the "move numerics" (MVN) instruction followed by a "zero and add" (ZAP). See the example below.

PROGRAM TO MULTIPLY AND TO CORRECT THE DECIMAL POINT

MP

FC	3	1	0	800	0	804

MVN

D2	00	0	802	0	803

ZAP

F8	3	2	0	800	0	800

STORAGE CONTENTS BEFORE EXECUTION OF THE ABOVE INSTRUCTIONS

00	01	12	0C

2048

01	0C

2052

STORAGE CONTENTS

AFTER MP

00	11	20	0C

2048

AFTER MVN

00	11	2C	0C

2048

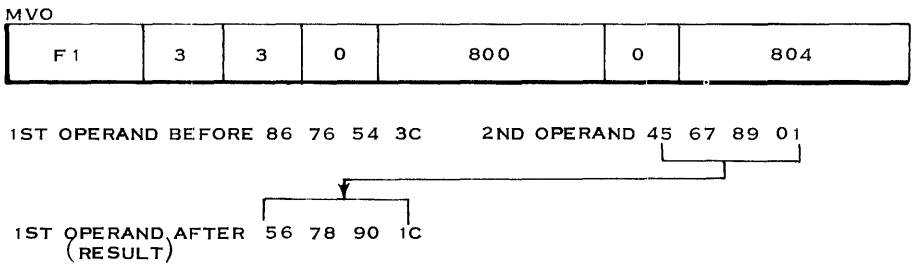
AFTER ZAP

00	00	11	2C

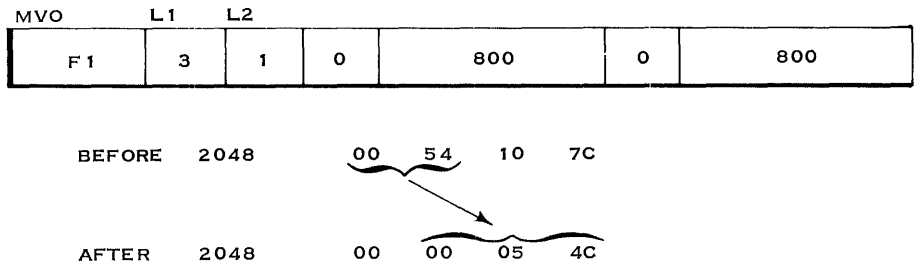
2048

Any time a packed decimal field is to be shifted an even number of places to the right, the MVN instruction can be used to place the sign next to the new low-order digit. As shown previously, the packed decimal field can then be shifted to the right by use of the ZAP instruction.

To right shift a packed decimal field an odd number of places, the "move with offset" instruction can be used. You haven't studied this instruction yet. The following is an example of how the "move with offset" (MVO) instruction works.



Note that the 2nd operand replaced the 1st operand. However the sign of the 1st operand (rightmost four bits) was left undisturbed. Now let's see how the MVO instruction can be used to effectively right shift a packed decimal data field an odd number of digit places.

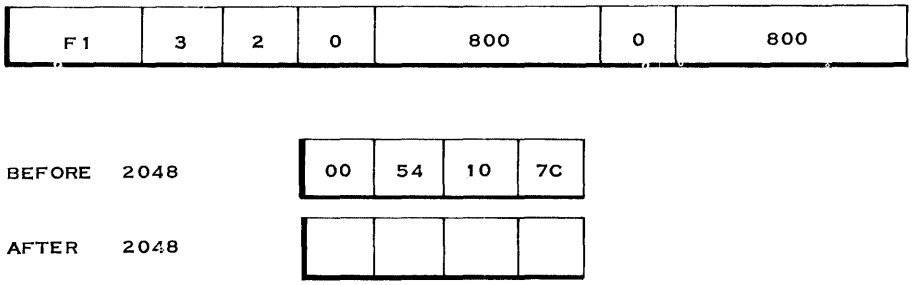


Notice that this instruction does not disturb the sign of the 1st operand. By offsetting each group of four bits in their bytes, an effective shift of an odd number of places is accomplished as the 2nd operand is moved into the 1st operand.

MOVE WITH OFFSET INSTRUCTION

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

Given the following MVO instruction, show the resulting contents of the 1st operand.



00	05	41	0C

Is the data moved by the MVO instruction checked to see if it is valid packed decimal data? _____

No; You have seen it being used to right shift packed decimal data an odd number of places. However, any type of data can be moved by this instruction.

Is the MVO instruction part of the System/360 decimal feature? ____

No; The MVO instruction is part of the standard instruction set.

Go to the Appendix in the Principles of Operation manual. Find the List of Instructions by Set and Feature. You will see the instructions that make up the standard instruction set and also the ones that make up the decimal feature.

So far you have seen how to right shift packed decimal data an even number of decimal places by using an MVN instruction followed by a ZAP. You used an MVO instruction to right shift an odd number of decimal places. How about left shifting packed decimal data?

Left shifting of decimal data is a little more complex. For instance, suppose the following data field is to be shifted two places to the left.

00	47	12	7C

After being shifted it should look like this:

47	12	70	0C

The preceding shift can be done by the following three instructions:

D2	02	0	800	0	801	MVC
F1	00	0	803	0	F00	MVO
94	F0	0	802			NI

LOCATION 2048	00	47	12	7C
LOCATION 3840	00			

After executing the MVC (Move Characters) instructions, locations 2048-2051 would look like this:

2048 – 2051	47	12	7C	7C

In the preceding MVC instruction, locations 2049-2051 were moved a byte at a time into locations 2048-2050. The fields were processed in a left to right direction.

The 2nd instruction (MVO) would take the constant of zero from location 3840 and move it into the high order of location 2051. Since the length codes are zero, only location 2051 is changed. The result would look like this:

2048 – 2051	47	12	7C	0C

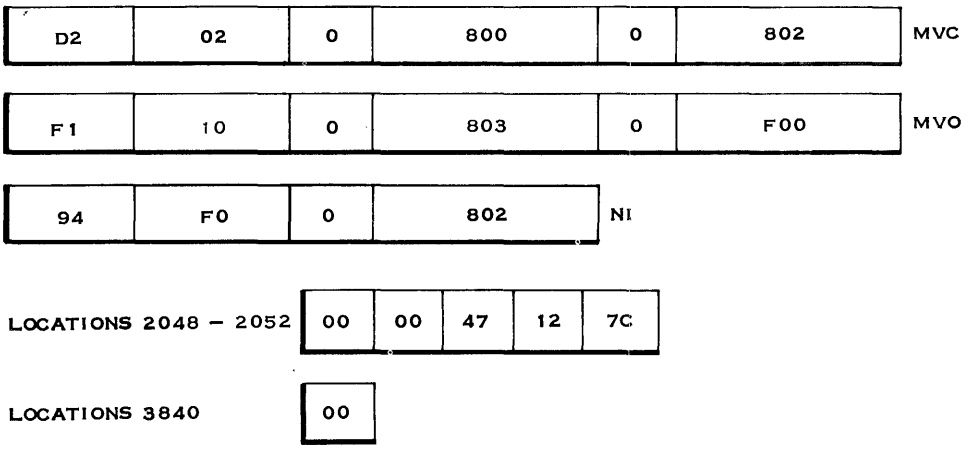
The third instruction (NI) is an "and immediate" instruction. The immediate operand (F0) is ANDed with location 2050 as shown below:

Immediate Operand	1111 0000
Location 2050	0111 1100
Result	0111 0000

Locations 2048-2051 now look like the desired result:

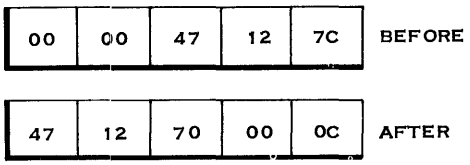
2048 – 2051	47	12	70	0C

The preceding series of instructions is only one way of left shifting an even number of places.

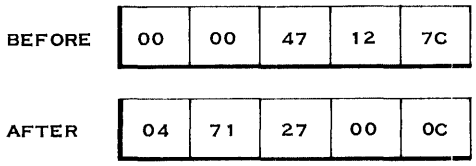


Given the above sequence of instructions, how many places was the data field shifted? _____

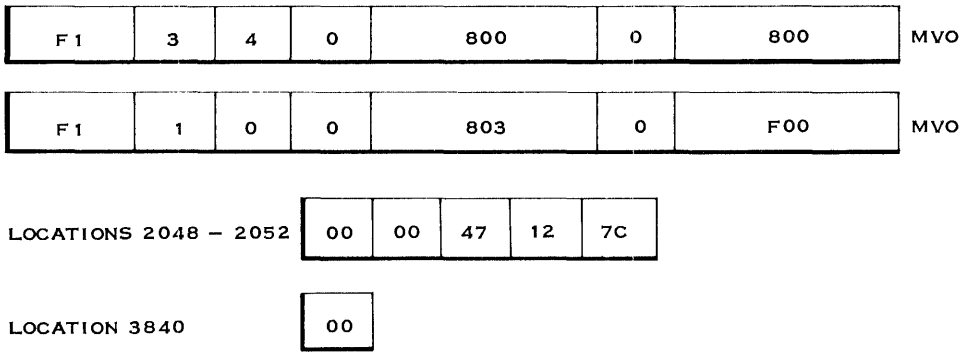
4; As shown below:



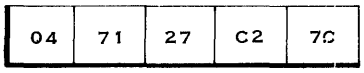
Now let's shift an odd number of places:



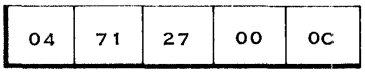
To achieve the above result requires an effective shift of three places. This can be accomplished as follows:



After executing the 1st MVO instruction, locations 2048-2052 would look like this:

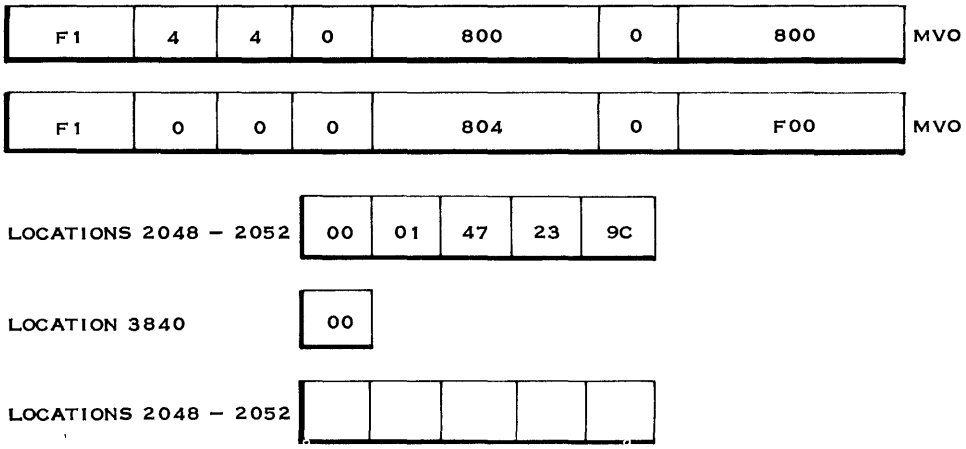


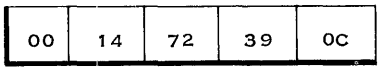
After executing the 2nd MVO instruction, locations 2048-2052 would look like this:



The preceding is the desired result.

Show the resulting contents of locations 2048-2052 for the following sequence of instructions.

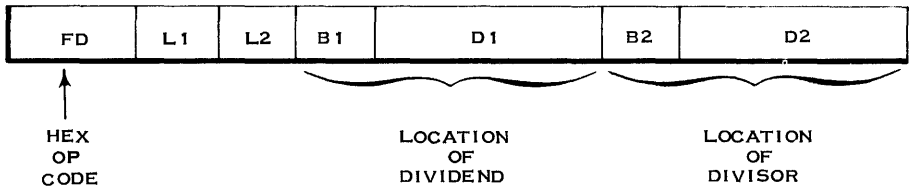




You have seen some ways of shifting packed decimal data. By proper use of the "move" instructions, you should be able to come up with other methods. For now, let's continue on to the "divide decimal" instruction.

DIVIDE DECIMAL INSTRUCTION

You are now ready to study the "divide decimal" instruction. The following is a "divide decimal" instruction:



As you can see above, the dividend is the _____ (1st/2nd) operand and the divisor is the _____ operand.

1st
2nd

As in the other instructions you have studied, the generated effective storage addresses refer to the _____ (high/low) order byte of the data fields.

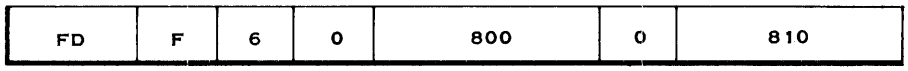
---

high

The mnemonic for the "divide decimal" instruction is DP. This indicates that the divide instruction operates on _____ (packed/zoned) decimal data.

---

packed

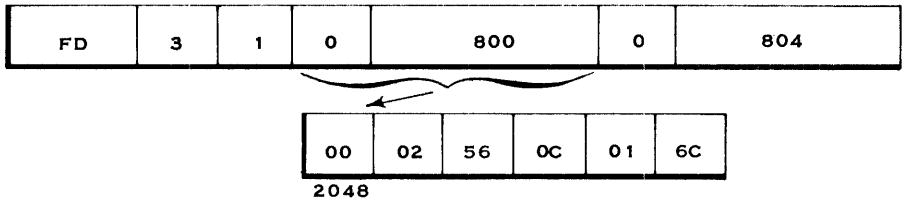


The above DP instruction has a dividend that is ____ bytes in length.

16

Sixteen bytes of packed decimal data can contain ____ digits and a sign.

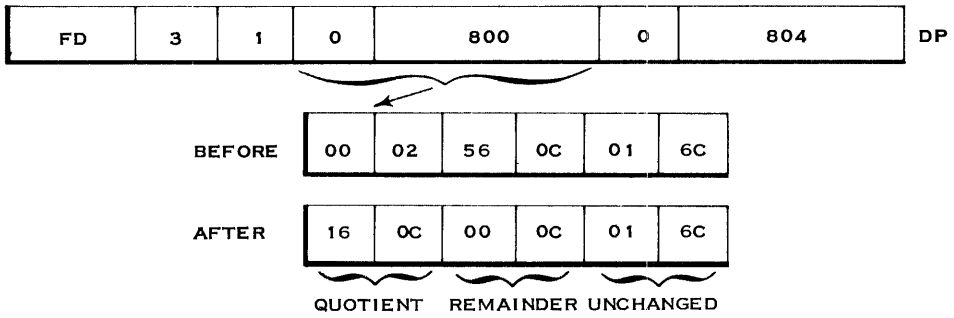
31



Given the above DP instruction, a value of _____ will be divided by a value of _____.

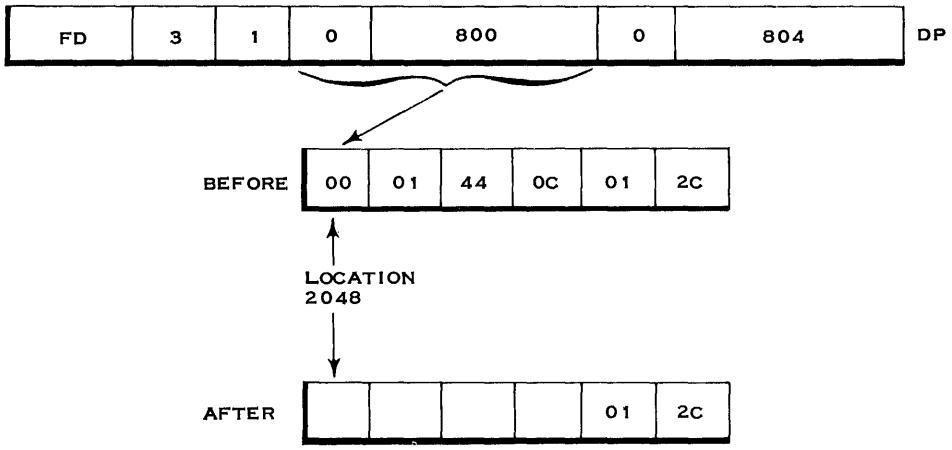
+2560
+16

The DP instruction will have as a result both a quotient and a remainder. These two results will be in the packed decimal format and will replace the dividend. The quotient will replace the high order and the remainder will replace the low order of the dividend. The following example will illustrate this.



NOTE: The remainder is always the same size as the divisor!

Given the following "divide decimal" instruction, show the resulting contents of the dividend field.



	12 0C 00 0C
	The remainder is placed in the low order of the dividend field and always contains the same number of bytes as the _____.
divisor or 2nd operand	The quotient is placed in the dividend field just to the left of the _____.
remainder	Read the description of the "divide decimal" instruction in the Decimal Arithmetic section of your Principles of Operation manual.
	The address of the quotient of a DP instruction will be the same as the original _____.
dividend or 1st operand	The size of the quotient will be equal to the dividend size minus the _____ size.
divisor	If the quotient cannot be fitted into its area, a _____ _____ exception will be recognized.
decimal divide	When a decimal divide exception is recognized, the dividend field will _____ (remain unchanged/contain part of the quotient).

remain unchanged

A Decimal Divide exception can be determined by aligning the leftmost digit of the divisor (2nd operand) with the next to leftmost digit of the dividend (1st operand). The divisor should be greater than that part of the dividend with which it is aligned.

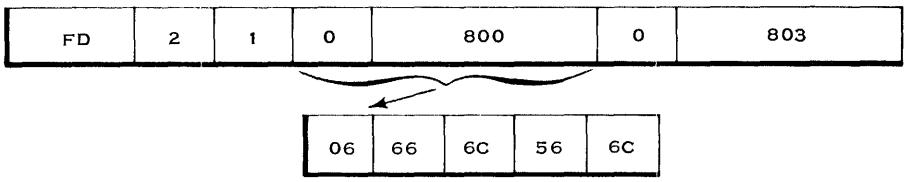
For example:

The following operands would result in a decimal divide exception because the divisor is not greater than the aligned section of the dividend.

1st operand (dividend)	00 16 0C
2nd operand (divisor)	0 16 C

The following operands would NOT result in a decimal divide exception because the divisor is greater than the aligned section of the dividend.

1st operand (dividend)	00 15 9C
2nd operand (divisor)	0 16 C

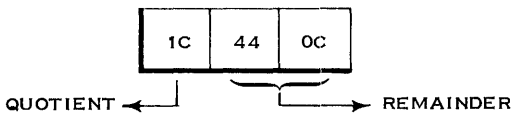


The above instruction _____ (would/would not) result in a decimal divide exception.

Would; This is because the high-order divisor digit (5) is not greater than the two high-order digits of the dividend (06).

How can you be sure that the quotient can't be fitted into its area unless the preceding rule is met? Let's work out the problem.

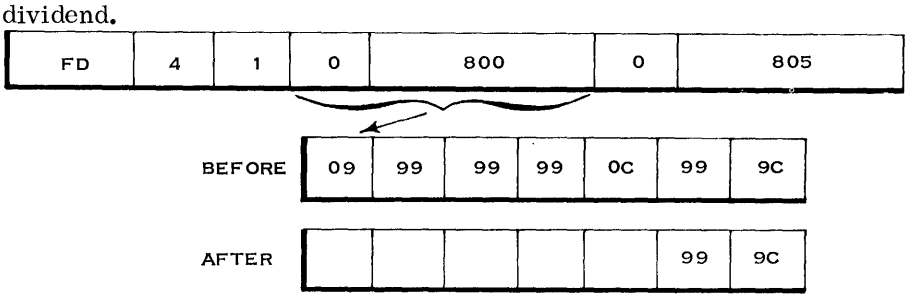
```
        11
    ------
566 ) 06666
      566
      ----
      1006
       566
      ----
       440
```



The original dividend was three bytes in length. The remainder of +440 will take up two bytes. The quotient of +11 cannot be fitted into the remaining byte.

Of course, no division takes place when the decimal divide exception is recognized. The System/360 will check to be sure that the divisor is greater than the aligned section of the dividend. If not, division does not take place and a program interrupt occurs.

Given the following DP instruction, show the resulting contents of the dividend.



09	99	99	99	0C

In the preceding problem, the divisor was not greater than the aligned section of the dividend. A decimal divide exception was recognized, the dividend was left unchanged, and a program interrupt was taken.

Another programming rule that applies to the "divide decimal" instruction is this:

The divisor must be shorter than the dividend and cannot exceed eight bytes. That is, L2 < L1 and L2 < 8.

FD	9	8	0	800	0	810

The above DP instruction will result in a specification error because the divisor's length code is greater than ____.

7

FD	3	3	0	800	0	804

The above DP instruction will result in a specification error because _____.

The divisor is not shorter than the dividend.

Let's summarize the "divide decimal's" data fields.

1. The Dividend
   a. The 1st operand is the dividend.
   b. The dividend has a maximum size of 31 digits and a sign.
   c. The dividend will be replaced by the quotient and remainder.
   d. The dividend must have at least one high-order zero digit.
2. The Divisor
   a. The 2nd operand is the divisor
   b. The divisor has a maximum size of 15 digits and a sign.
   c. In all cases, the divisor must be shorter than the dividend.

(Frame continued on next page.)

3. The Remainder
   a. The remainder replaces the low order of the dividend field.
   b. The remainder has the same length as the divisor.
   c. The sign of the remainder is the same as the sign of the original dividend.

4. The Quotient
   a. The quotient replaces the high order of the dividend field.
   b. The size of the quotient is equal to the dividend size minus divisor size (L1 - L2).
   c. Since the quotient is placed in the high order of the divide field, its address will be the same as the dividend's.
   d. The sign of the quotient follows the rules of algebra.
      (1) Like signs = +
      (2) Unlike signs = -

5. Decimal Divide Exception
   a. This exception indicates that the quotient would be too large to be fitted into its allotted field.
   b. This exception is recognized whenever the divisor is not greater than the aligned section of the dividend. It is for this reason that the dividend must have at least one high-order zero digit.
   c. The decimal divide exception is recognized prior to any division. The dividend field is left unchanged and a program interrupt is taken.

6. Specification Exception
   This exception is recognized on a "divide decimal" instruction whenever:
   a. The divisor is longer than eight bytes.
   b. The dividend is not longer than the divisor.

EDIT INSTRUCTION

The two remaining instructions that make up the decimal feature are the "edit" instruction and the "edit and mark" instruction. The purpose of these edit operations is to produce easy-to-read documents by inserting the proper punctuation into a data record. The data to be edited is called the source field and must be in the packed decimal format. Consider the following source field:

00	12	49	07	10	7C

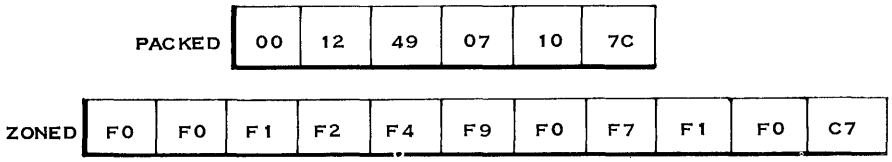
SOURCE FIELD

In its present format, the preceding field cannot be printed. The data must be in EBCDIC before being printed.

One of the functions of the edit operation is to change a source field from the ____ format to the _____ decimal format.

packed
zoned

If changing from the packed to the zoned format were all that was necessary to produce a legible report, the "edit" instruction wouldn't be necessary. The "unpack" instruction, which you previously studied, would be sufficient. For instance, if the previous packed decimal operand were changed to the zoned format, it would look like this:



If the above zoned decimal field were printed, it would look like this:

0 0 1 2 4 9 0 7 1 0 G

By examining the printed document, you could tell by looking at the low-order character (G) that it was a positive number with a low-order digit of 7. However, the printed document is still not too legible. Perhaps the number represents money. It would be better if it could look like this:

$1, 249, 071. 07

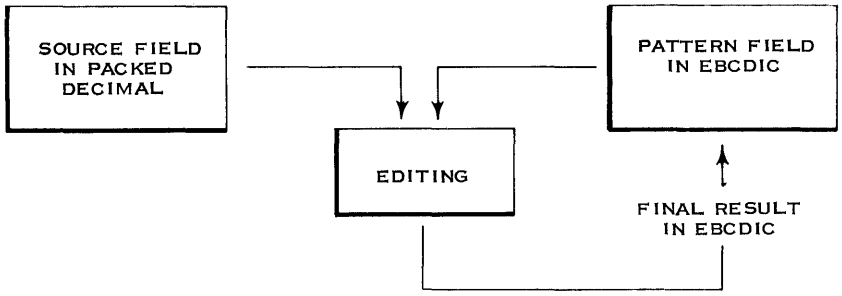
As well as other editing, this would require inserting the commas and decimal points in the right places. This is another function of the edit operations.

The edit operation will change a packed decimal field which is called the _____ field, into the zoned format and insert the necessary punctuation characters.

---

source

The edit operation consists of moving the source field into a pattern field. The pattern field will be made up of EBCDIC characters that will control the editing. The final edited result will replace the PATTERN field.



During an edit operation, the _____ field is edited under control of the _____ field.

---

source pattern — The source field contains packed decimal data while the pattern field contains _____ characters.

EBCDIC — The edited result will replace the _____ field.

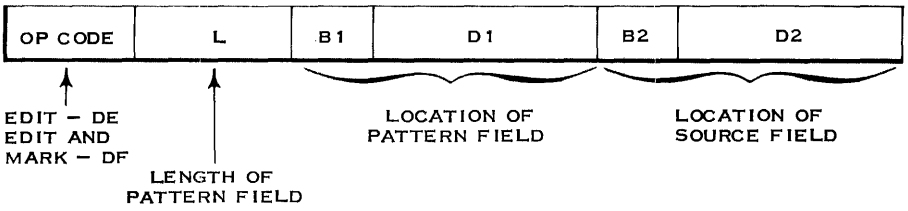
---

pattern — The pattern to be used is normally kept in storage as a constant. It will be moved into a storage work area prior to the edit operation. The "move characters" instruction can be used for this purpose.

Although the pattern field is replaced (destroyed) by the edited result, the original pattern is retained as a constant in another location of _____.

---

storage — The "edit" instructions use the SS format as shown below:



As you can see above, the source field is the _____ (1st/2nd) operand.

2nd — Like most instructions, the results of the edit operation replace the 1st operand which is the _____ field.

---

pattern — The length code refers to the pattern field which can be a maximum of _____ bytes in length.

---

256 — The characters in the pattern field determine the editing that will take place. The high-order (leftmost) character in the pattern field is known as the fill character. For many edit operations, the fill character would be an EBCDIC blank (01000000).

The fill character is used in the edit operation to replace certain characters in the pattern field. You will see this more clearly later.

The fill character is the _____ byte in the _____ field.

---

leftmost pattern — Any of the 256 possible EBCDIC combinations can be used as the fill character. However, in many edit operations the fill character will consist of an EBCDIC _____.

---

blank; For explanatory purposes, a blank will be represented by a small b in a field such as JOHNbSMITH. Of course, blanks won't be printed out.

Besides the fill character, there are three more characters in the pattern field that have special meaning. They are:

1. The Digit Select Character.
2. The Significance Start Character.
3. The Field Separator Character.

These three characters can appear anywhere in the pattern field.

The digit select character has the following bit structure: 00100000. There is no character symbol for this combination. It is normally represented by a small "d" just as blanks are represented by a small "b."

When a digit select character is encountered in a pattern field, it is usually replaced by a digit from the source field. If the source digit is a high-order zero, the fill character is used instead to replace the digit select character. By using a blank as the fill character, high-order zeroes can be blanked out.

If an asterisk is used as the fill character, asterisk protection for paychecks can be achieved.

What characters can replace a digit select character in the pattern field?

1. _____
2. _____

---

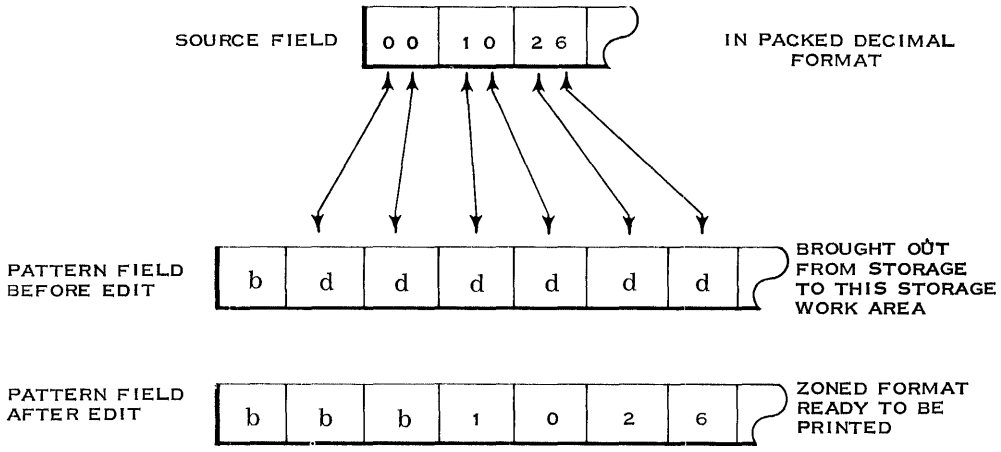
1. A digit from the source field.
2. A fill character.

What symbol is used to denote a digit select character? _____

---

d; Actually, the binary bit of a digit select character is 00100000 or a hex 20. There is no character for this combination on any of the System/360 printers. The "d" is used to represent this combination in your textbooks.

Let's look at an example of the use of the fill character and the digit select character.



Prior to the "edit" instruction, the pattern was brought out of the constant storage area and put in the p_____ f_____.

pattern field

The leftmost position of the pattern field contains the fill character which in this example is a _____. The remaining positions of the pattern field contain _____ _____ characters.

blank digit select

The source field contains high-order zeroes which must be edited out. This field is in the _____ (packed/zoned) format.

packed

The "edit" instruction looks at the second position of the pattern field and finds a digit select character. It then looks at the first character in the source field and finds a nonsignificant (zero) digit.

The digit select character is replaced by the f_____ character (blank).

fill

Next, the "edit" instruction looks at the third position of the pattern field. The previous operation is repeated. The third position of the pattern field ends up with a _____ (blank/digit).

blank

The "edit" instruction looks at the pattern field's fourth position, finds another _____ _____ character which tells it to look at the source field. A significant digit is found and moved into the pattern field.

digit select

The pattern field now contains blank-blank-blank-1. The fact that a significant digit was found is remembered so that the rest of the source field (including zeroes) can be put into the pattern field.

Since a digit select character is replaced by a source digit or the fill character, the system needs some way of knowing which of the two to choose. This is determined by a remembering device called the S trigger in the system's circuitry.

The S trigger can be set to one of two states: 0 state or 1 state.

When set to 1, the S trigger indicates that the digits from the source field are significant. As a result, the digit select characters in the pattern field are replaced with the digits from the source field.

At the beginning of the edit operation, the S trigger is set to 0. As long as the S trigger is 0, the digit select characters in the pattern field are replaced with the fill character.

What determines whether a digit select character is replaced with a source digit or with the fill character? _____

The S trigger

At the beginning of the edit operation, the S trigger is set to ___ (1/0).

0

When the S trigger is set to 0, a digit select character in the pattern field is replaced with _____.

---

the fill character

When the S trigger is set to 1, a digit select character in the pattern field is replaced with _____.

---

the digit from the source field

Both the source field and the pattern field are processed left to right, a character or digit at a time. Each time the digit from the source field replaces a digit select character, the 4-bit digit has the proper EBCDIC or ASCII zone bits inserted. PSW bit 12 determined whether the EBCDIC or ASCII zone bits are inserted. For the purposes of this text, we will assume that the system is in EBCDIC mode.

The S trigger is set to 0 at the beginning of the edit operation. It is set to 1 by one of two methods:

1. A significant (non-zero) digit from the source field.

2. A significance start character in the pattern field.

The significance start character has a bit pattern of 00100001 (hex 21). This bit pattern has no character symbol. The symbol for the left parenthesis is used to represent a significance start character such as "(".

Which symbol, "d," "b," ")," or "(," is used to represent each of the following?

Blank = _____
Significance Start Character = _____
Digit Select Character = _____

Blank = "b"
Significance Start Character = "("
Digit Select Character = "d"

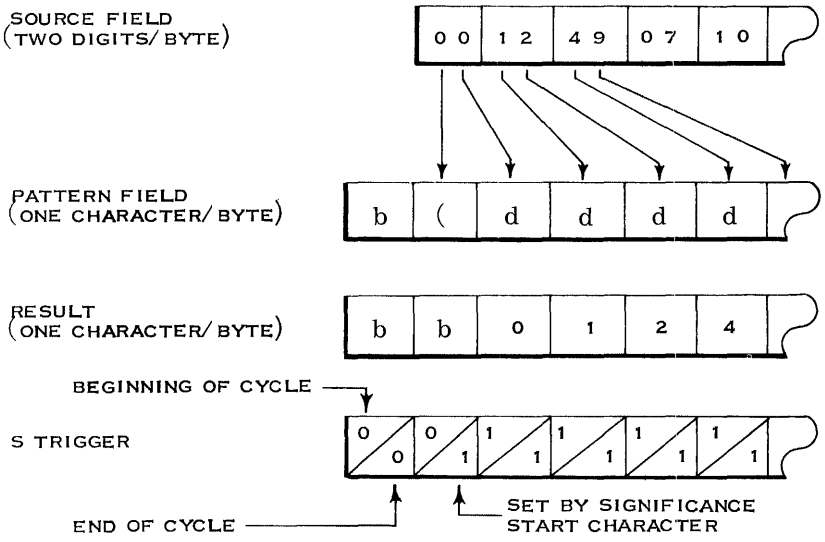
What two characters can set the S trigger to 1?

1. _____.
2. _____.

---

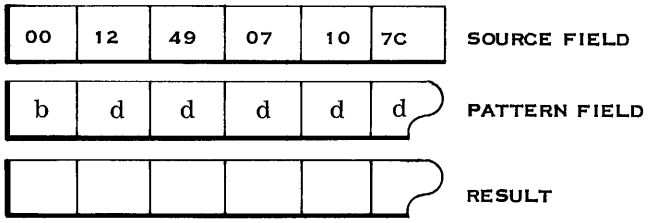
1. A non-zero digit from the source field.
2. A significance start character in the pattern field.

A significance start character is replaced (as was the digit select character) by either a digit from the source field or by the fill character. For example:



The edit operation begins by examining the fill character. If it is not a digit select or a significance start character, it is left in place in the pattern field. Then the next pattern character is examined. In the previous example, this was a significance start character. The high-order source digit is then examined. Because the source digit is zero and the S trigger is 0 (at this time), the significance start character is replaced with the fill character. However, the significance start character does set the S trigger to 1 so that all subsequent source digits are significant. The remaining pattern characters in our example are digit select characters which are replaced with source digits.

Given the 1st few characters of a source and pattern field below, show the resulting contents of the pattern field after editing.



---

b b b 1 2 4

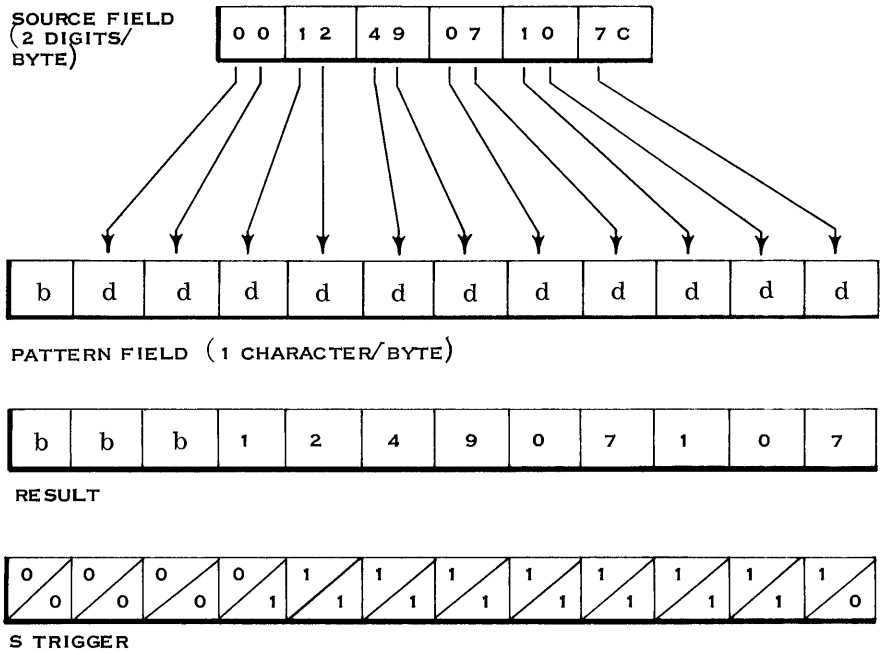
In the previous problem, there was no significance start character. As a result, the two high-order zeroes from the source field did not go into the pattern field. The fill character was used instead. Once significance was started, the remaining pattern characters were replaced by source digits.

Once significance is started, the S trigger will remain on until one of two things happens:

1. The sign of the source field is examined and is plus.
2. A field separator character (00100010) is recognized.

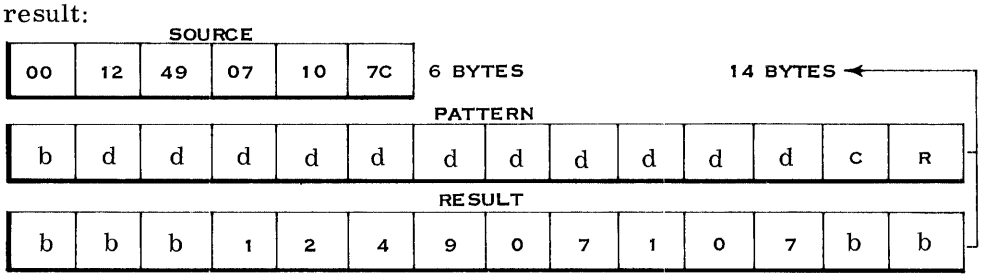
The field separator character is used when two or more packed decimal source fields are to be edited into a pattern with one instruction field. We'll examine this later. For now, let's discuss the handling of the sign.

Since the sign is in the right half of the source field's low-order byte, it can be examined at the same time as the low-order digit is examined. The sign itself is skipped over but, if plus, the S trigger is set to zero. The following will illustrate this.



When a pattern character is examined and is not one of the three special control characters, it is left in place if the S trigger is 1. Otherwise, it is replaced by the fill character.

The source field is not examined. The usual method of indicating a negative quantity in a printed report is with the letters "CR." If we take another look at the previous example and add the CR symbol, this would be the result:



Because the plus sign set the S trigger to 0, the remaining pattern characters (CR) were replaced by the fill character. If the sign of the source field had been minus, the "CR" would have been left in the pattern field.

Let's take the following source field and produce the edited result.

Source | 00 | 12 | 49 | 07 | 10 | 7D |

Edited Result b b b 1 , 2 4 9 , 0 7 1 . 0 7 b C R b

The original pattern would look like this:

Pattern b d d d , d d d , d d d . d d b C R b

Notice that the commas, decimal point, and "CR" were left in place. This occured because the S trigger was set to 1 and remained there.

Given the following, show the edited result.

Source | 00 | 14 | 71 | 3C |

Pattern b d d , d d d . d d b C R b

Result _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

---

b b b b 1 4 7 . 1 3 b b b b

Of course, the blanks in the previous answer won't print in the final printed report which would look like this:

1 4 7 . 1 3

Read the description of the "edit" instruction in the Logical Operations section of your Principles of Operation manual.

Which of the following can set the S trigger to 0? (Circle one or more.)

a. Digit Select Character
b. Significance Start Character
c. Field Separator Character
d. Beginning of edit operation
e. Plus sign in source field

---

c, d, e

Which of the following can set the S trigger to 1? (Circle one or more.)

a. Digit Select Character
b. Significance Start Character
c. Minus sign in source field
d. 1st non-zero character in source field
e. Field Separator Character

---

b, d

A digit select character in the pattern is replaced by the _____ _____ if the S trigger is 0 or by a _____ _____ if the S trigger is 1.

---

fill character source digit

A significance start character in the pattern is replaced by the _____ _____ if the S trigger is 0 or by a _____ _____ if the S trigger is 1.

---

fill character source digit

A field separator character always sets the S trigger to 0 and is replaced by the _____ _____.

---

fill character

Characters in the pattern other than the control characters are either replaced by the _____ _____ if the S trigger is 0 or are _____.

---

fill character left in place

Show the results of the following "edit" instruction.

ED

DE	0C	0	800	0	900

1st Operand	b d d , d d (. d d b C R	(characters)
2nd Operand	0 7 9 4 7 6 9 C	(digits and sign)
Result	_____	

---

b b 7 , 9 4 7 . 6 9 b b b

Show the result of the following "edit" instruction.

ED

DE	0C	0	800	0	900

1st Operand	b d d , d d (. d d b C R	(characters)
2nd Operand	0 0 0 0 0 6 9 D	(digits and sign)
Result	_____	

---

b b b b b b b . 6 9 b C R

Referring to the previous problem, show the result for the following pattern.

Pattern	b d d , d d d . d d b C R
Result	_____

---

b b b b b b b b 6 9 b C R; Note that a significance start character should be in the pattern to protect the decimal point in case the amount is less than a dollar.

Show the result of the following "edit" instruction.

ED

DE	0C	0	800	0	900

1st Operand	* d d , d d (. d d b C R	
2nd Operand	0 0 0 0 0 6 9 C	
Result	_____	

---

* * * * * * * . 6 9 * * *; Note that the use of an asterisk as the fill character will provide asterisk check protection.

The following "edit" instruction will edit multiple adjacent source fields. Show the result.

ED			IN HEX		
DE	13	0	800	0	900

1st Operand b d d (. d d b C R) d d d . d d b C R
2nd Operand 0 1 7 7 6 C 0 0 0 0 0 D
Result _____

---

b b 1 7 . 7 6 b b b b b b b b b b b b b; Note that the field separator character set the S trigger to zero. No significant digits were found in the 2nd source field. As a result, the pattern characters were replaced by the fill character (blanks).

Up to this point you have seen that the "edit" instruction can be used to:

1. Eliminate high-order zeroes
2. Provide asterisk protection
3. Handle sign control (CR)
4. Provide punctuation
5. Blank out an all-zero field
6. Edit multiple adjacent fields via the field separator character
7. Protect the decimal point by use of the significance start character. This character can also be used to retain high-order zeroes when desired.

EDIT AND MARK INSTRUCTION

The "edit" instruction makes no provision for using a floating currency symbol (such as $) as part of the editing process. For this edit feature, the "edit and mark" instruction must be used. Read the description of the "edit and mark" instruction in the Logical Operations section of your Principles of Operation manual.

ED is the mnemonic for the "edit" instruction while EDMK is the mnemonic for the " _____ _____ _____ " instruction.

"edit and mark"

Is there anything that the ED instruction can do that the EDMK instruction can't do? _____ (Yes/No)

No

What, then, is the difference between the ED and EDMK instructions?

_____

_____

The EDMK instruction causes the address of the 1st significant digit of the result to be placed in general register 1.

Can the EDMK instruction be used to edit multiple fields? _____

Yes; However, the address in register 1 will only pertain to the last source field edited.

What happens on an EDMK instruction when significance is started by a significance start character? _____

No address is placed in register 1.

Does the EDMK instruction insert the floating currency symbol? _____

No; The symbol must be inserted by subsequent instructions.

The address placed in register 1 is: (Circle one of the following.)

a. The location where the currency symbol (such as $) should be inserted.

b. The location + 1 where the currency symbol should be inserted.

---

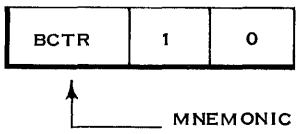
b; Register 1 has the address of the 1st significant digit. The currency symbol (such as $) should be placed just to the left of this digit.

What instruction can be used to reduce the address in register 1 by one?

_____

Branch on Count; without a branch.

By using the RR format and an R2 field of zero, register 1 can be reduced For example:



After the BCTR instruction, the "move character" instruction (MVC) can use register 1 as a base register and move a dollar sign (currency symbol) into the desired location. A "move immediate" (MVI) instruction with the currency symbol as the immediate operand could also be used.

You have now completed your study of the System/360 standard instruction set with the decimal feature. You have not yet studied I/O operations. These will be covered in your next self-study book. For now, let's take a look at some programming examples.

SECTION IV LEARNING OBJECTIVES

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

Write programs, using stored data in the form of 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 Decimal Feature Programs

Notice: This section of the decimal 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 branching, logical, or decimal instruction.

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

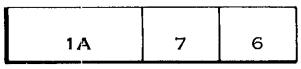
To make the following programs easier to read, we are using symbolic instructions. The symbolic instruction format similar to, but not necessarily identical to, the source language format required by the System/360 Assembler Program.

The RR format is shown this way:

Mnemonic R1, R2

Write the machine language instruction generated by the following symbolic statement: (The hex Op code for AR is 1A.)

AR 7, 6 



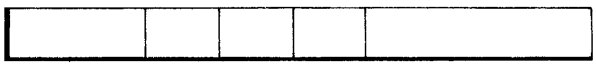
The RX format is shown this way:

Mnemonic R1, D2 (X2, B2)

Note - Decimal notation is used instead of hexadecimal in the symbolic format.

Write (in hex) the machine language instruction that corresponds to the following: (45 is the hex Op code for BAL.)

BAL 15, 2048 (0, 0)



45	F	0	0	800

The RS format is shown this way:

Mnemonic R1, R3, D2 (B2)

Write the machine language instruction that corresponds to the following: (86 is the hex Op code)

BXH 2, 4, 3840 (3)

86	2	4	3	F00

The SI format is shown this way:

Mnemonic D1 (B1), I2

Write the machine language instruction for the following: (94 is the Op code)

NI 2048 (0), 240

94	F0	0	800

F0 is the hexadecimal equivalent of a decimal 240.

The SS format will be shown this way for an 8-bit length code:

Mnemonic D1 (L, B1), D2 (B2)

The SS format will be shown this way for instructions with two 4-bit length codes.

Mnemonic D1 (L1, B1), D2 (L2, B2)

Show the machine language instruction for the following:
Note: 1. Op code is D2.
2. Symbolic length code is = total # of bytes.

MVC 2048 (256, 0), 3840 (0)

D2	FF	0	800	0	F00

Show the machine language instruction for the following: (Op code is FD)

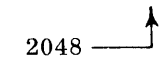
DP 2048 (15, 0), 3840 (8, 0)

FD	E	7	0	800	0	F00

PROGRAM #1

```
BALR    4, 0
LA      6, 2048 (0, 0)
ZAP     4 (2, 6), 2 (2, 6)
AP      4 (2, 6), 0 (2, 6)
```

Storage contains 017C 402C 0000



What would be the contents of locations 2048 through 2053 after Program #1 is executed? _____

017C 402C 419C

If you had the correct answer, proceed to Program #2. Otherwise, continue on with the step-by-step analysis of Program #1.

The 1st instruction is a "branch and link" instruction using the ____ format.

RR

Because the R2 field of the BALR instruction is zero, a branch _____ (does/does not) occur.

---

does not

What is placed in general register 4 as a result of the BALR instruction?

_____

The address of the next instruction's (LA) Op code.

NOTE: General register 4 is not used in Program #1. However, it does provide us with a means of branching back to the program if we had wanted to.

The second instruction is "load address." The generated effective address will be loaded into register ___.

6

Show (in hex) the contents of bits 8-31 of register 6 after the LA instruction is executed. _____

000800

The 3rd instruction will cause byte locations _____ through _____ to be placed in locations _____ through _____.

2050, 2051
2052, 2053

Note that in these symbol instructions, the length code is equal to the total number of bytes. Of course, in actual machine language instructions, the length code is one less than the total number of bytes.

What will be the contents of locations 2048-2053 after the ZAP instruction is executed? _____

017C 402C 402C

The last instruction of Program #1 will cause byte locations _____ through _____ to be added to locations _____ through _____.

2048, 2049
2052, 2053

What will be the contents of locations 2048-2053 after Program #1 is executed? _____

PROGRAM #2

017C 402C 419C

```
LA      1, 16 (0, 0)
LA      2, 2048 (0, 0)
BALR    3, 0
ZAP     2 (4, 2), 0 (2, 2)
MP      2 (4, 2), 0 (2, 2)
DP      2 (4, 2), 0 (2, 2)
BCT     1, 6 (0, 3)
```

Locations 2048-2053 contain 016C00000000

What will be the contents of locations 2048-2053 after Program #2 is executed for the first time? _____

016C 016C 000C

Quotient — Remainder

There will be a program interrupt during the 2nd execution of the program. What exception will cause this program interrupt? _____

Data; The second time the "multiply" instruction is executed, the sign of the quotient will be recognized as an invalid packed decimal digit. To avoid this exception, the DP instruction should be followed by a ZAP instruction as shown below.

```
ZAP     2 (4, 2), 2 (2, 2)
```

For your convenience, this is a repeat of Program #2.

```
LA      1, 16 (0, 0)
LA      2, 2048 (0, 0)
BALR    3, 0
ZAP     2 (4, 2), 0 (2, 2)
MP      2 (4, 2), 0 (2, 2)
DP      2 (4, 2), 0 (2, 2)
BCT     1, 6 (0, 3)
```

Locations 2048-2053 contain 016C00000000

If you were able to answer the preceding questions correctly, proceed to Program #3. Otherwise, continue with the following step-by-step analysis of Program #2.

Show (in hex) the contents of general register 1 after executing the 1st instruction of Program #2. _____

00000010; A value of 16 was loaded into register 1.

Show (in hex) the contents of register 2 after the 2nd instruction is executed. _____

00000800

The 3rd instruction is a "branch and link."

a. What address is placed in Register 3? _____

b. Does a branch occur? _____

---

a. The address of the Op code (ZAP) of the following instruction.
b. No; Because the R2 field is zero.

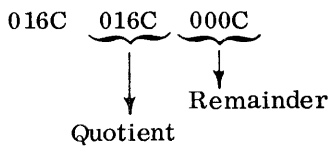
Show the contents of locations 2048-2053 after the ZAP instruction is executed. _____

016C0000016C

Show the contents of locations 2048-2053 after the "multiply decimal" instruction is executed for the first time. _____

016C0000256C

Show the contents of locations 2048-2053 after the "divide decimal" instruction is executed for the first time. _____



After a "divide decimal" instruction, the remainder is placed in the low order of the dividend field. All remainders (even zero) are the same length as the original _____.

---

divisor

The final instruction of Program #2 is a "branch and count." Show (in hex) the contents of register 1 after this instruction is executed for the 1st time. _____

0000000F; The register was reduced by one from 16 to 15.

Since the contents of register 1 were not reduced to zero, a branch _____ (will/will not) occur.

---

will

On the System/360, a branch is taken by replacing _____ _____ with the "branch to" location.

---

the instruction address portion of the PSW (bits 40-63)

What will be placed in PSW bits 40-63 after the BCT instruction is executed? _____

The contents of register 3 plus a displacement value of 6; This effective address is the location of the "multiply decimal's" Op code.

What will happen the 2nd time the "multiply decimal" instruction is executed? _____

A data exception will be recognized and cause a program interrupt. This occurs because the multiplicand (2050-2053) contains an invalid digit code.

```
0 1 6 C 0 0 0 C
      ↑
      └── Invalid
```

Continue on to Program #3.

PROGRAM #3

The following 80 character EBCDIC card record is in locations 3840-3919 of main storage:

Columns 1- 6	Man Number		Field A
Columns 10-12	Hours Worked	(XX.X)	Field B
Columns 15-17	Pay Rate	(X.XX)	Field C
Columns 20-24	Deductions	(XXX.XX)	Field D
Columns 30-34	Gross Pay	(XXX.XX)	Field E
Columns 40-44	Net Pay	(XXX.XX)	Field F

Given the above information, write a program that will calculate gross and net pay using the instructions of the decimal feature. Remember now, that the data you are working with is in EBCDIC (zoned decimal). The data in fields C, D, E, and F will eventually be sent to an output device and must end up in the proper data format.

Which of the following equations will solve for net pay? The fields are lettered from A to F as shown above.

a. $F = \frac{B \times C}{D}$ b. $F = B \times C - E$ c. $F = E - D$

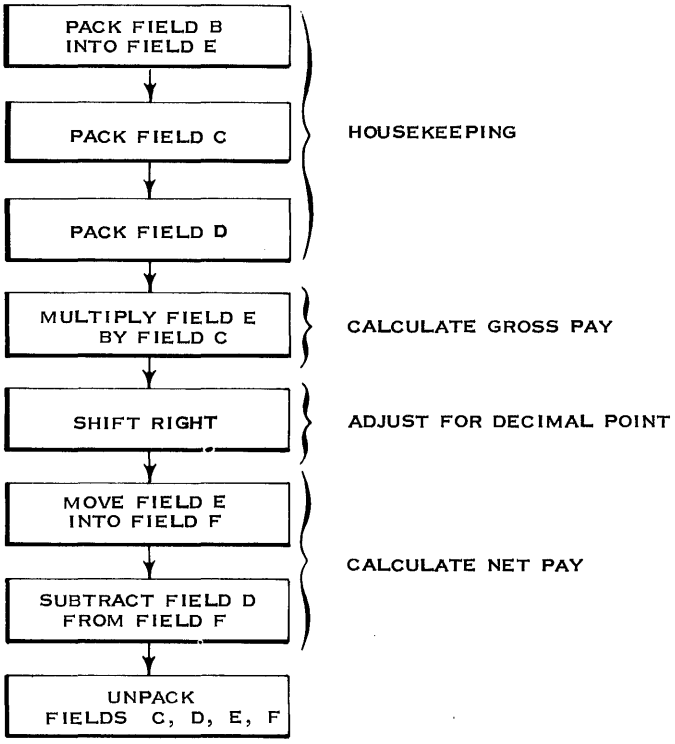
---

c; Gross pay - Deductions = Net Pay

Write the equation that will solve for gross pay. _____

E = B x C

Now that we know the formula to use in solving our problem, the next step is to flowchart our program. Draw the flowchart you intend to use.
NOTE: The data you are using is in EBCDIC.



Your flowchart should be similar to this one. If it is different, resolve the differences before continuing.

Using your flowchart, write the necessary symbolic instructions to solve the program. Assume your program will start at location 2048 and that register 1 contains 2048 as a base address. Remember to adjust the decimal point after a multiplication.

NOTE: Card record in locations 3840-3919

```
PACK    1821 (5, 1), 1801 (3, 1)
PACK    1806 (3, 1), 1806 (3, 1)
PACK    1811 (5, 1), 1811 (5, 1)
MP      1821 (5, 1), 1806 (3, 1)
MVO     1821 (5, 1), 1821 (4, 1)
MVC     1831 (5, 1), 1821 (1)
SP      1831 (5, 1), 1811 (5, 1)
UNPK    1806 (3, 1), 1806 (3, 1)
UNPK    1811 (5, 1), 1811 (5, 1)
UNPK    1821 (5, 1), 1821 (5, 1)
UNPK    1831 (5, 1), 1831 (5, 1)
```

Your answer should agree to some extent with the above answer. If it does not, try to correct any differences before continuing on to Program #4.

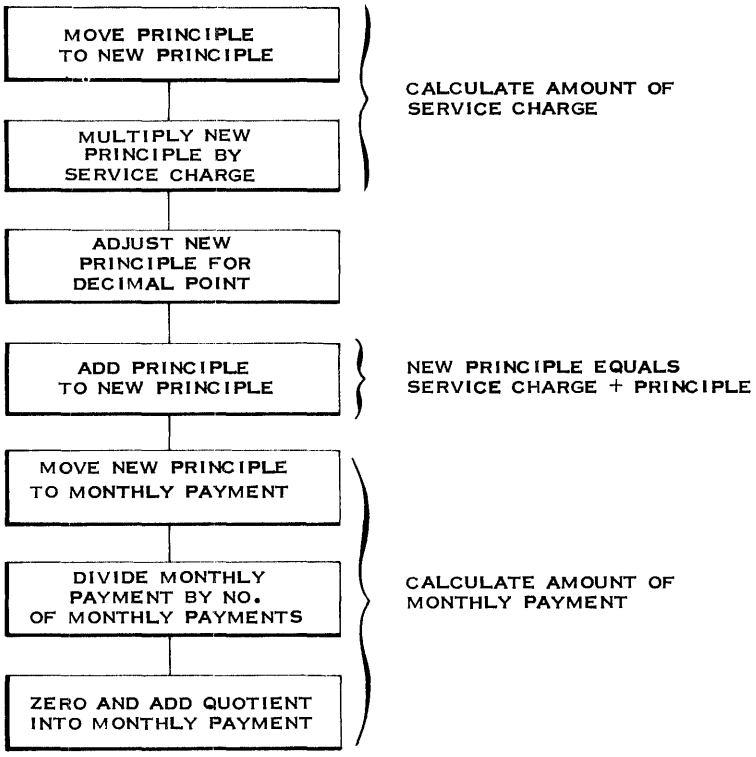
PROGRAM #4

A man borrows $1,000 from a bank. He agrees to pay off the debt in one year. As a result, a 6% service charge is added on to the principle.

Draw a flowchart for a program that will do the following:

a. Calculate the man's monthly payment.

Principle	$1,000.00	In Packed Decimal
Service Charge	.06	
No. of Monthly Payments	12	
New Principles	XXXX.XX	
Monthly Payment	XX.XX	



Your flowchart should be similar to the preceding answer. Now write the necessary instructions that will make up the program. Your program may start anywhere in main storage, so load register 6 with the base address using the "branch and link" instruction. Assume the following data will be located 2050 bytes from the beginning of the program.

7 Bytes	+100000	In Packed Decimal
1 Byte	+6	
2 Bytes	+12	
7 Bytes	New Principle	
7 Bytes	Monthly Payment	

```
BALR    6, 0
MVC     2058 (7, 6), 2048 (6)
MP      2058 (7, 6), 2055 (1, 6)
MVN     2063 (1, 6), 2064 (6)
ZAP     2058 (7, 6), 2058 (6, 6)
AP      2058 (7, 6), 2048 (7, 6)
MVC     2065 (7, 6), 2058 (6)
DP      2065 (7, 6), 2056 (2, 6)
ZAP     2065 (7, 6), 2065 (5, 6)
```

Your solution should be similar to the preceding answer to corrent any differences before continuing to Program #5.

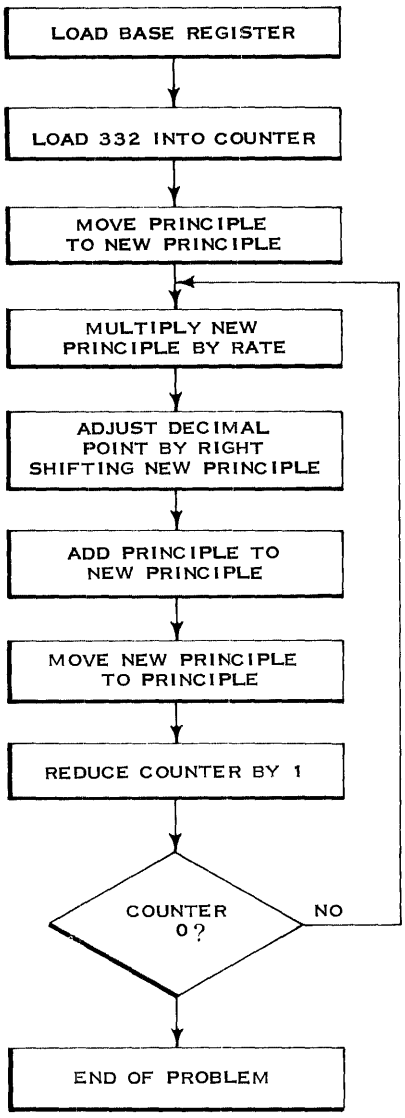
PROGRAM #5

The Indians sold the island of Manhattan to a foreign real estate firm 332 years ago for $24.00 and a pair of wooden shoes. They converted the shoes into toy boats and deposited the money in a savings account where it has been drawing interest all these years at 3% compounded annually.

Draw a flowchart that will solve for the present value of the Indians' principle. You will be given only the following data which is displaced 500 bytes from the beginning of your program. Use register 1 for the base register and register 2 as a counter (use the "load address" instruction to initially set the counter).

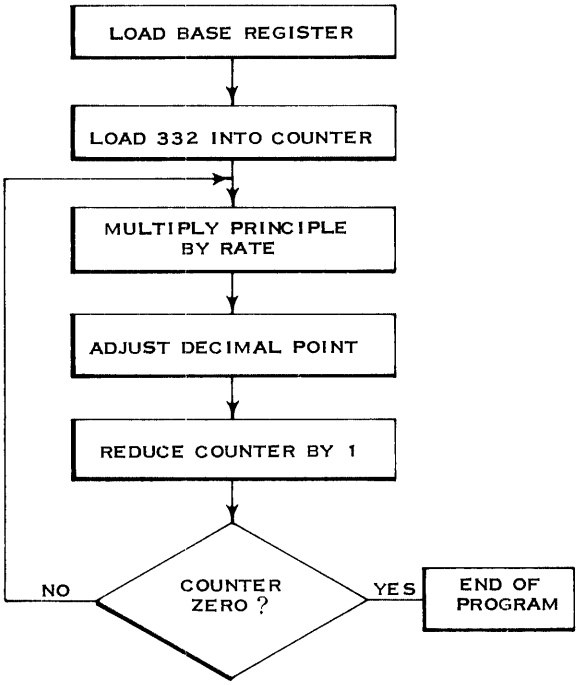
*5 bytes	+2400	In Packed Decimal
1 byte	+3	
5 bytes	New Principle	

*The New Principle will not exceed 10 digits.



The preceding flowchart was more complicated than it need be. This was because you're given a constant of +3 to use as the Rate. As a result, you had to figure out the interest and then add the previous principle to it. If you had been given a constant of +103 to use, the flowchart could have been simpler.

Draw a flowchart to solve the previous problem. All information remains the same except this time you are given a two-byte constant of +103 rather than one byte of +3.



Using the preceding flowchart, write the necessary symbolic instructions that will do the job. The following packed decimal data is displaced 500 bytes from the beginning of the program.

	Data	
Beginning address of program +500 ⟶	+2400	(5 bytes)
Beginning address of program +505 ⟶	+103	(2 bytes)

```
BALR   1, 0    Load Base Register
LA     2, 332 (0, 0) - Load Counter
MP     498 (5, 1), 503 (2, 1)
MVN    501 (1, 1), 502 (1)
ZAP    498 (5, 1), 498 (4, 1)
BCT    2, 4 (0, 1)
```

Your program should look similar to the one above. Notice that there is no "halt" instruction to end the program. Since the System/360 will usually operate under control of a supervisor program, a "supervisor call" instruction could be used to indicate the end of the program.

Do you need a review? If you think that you may require a review of certain 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 objectives indicate.

Starting on the next page is a self-evaluation quiz. It will allow you to check your understanding of decimal operations.

REVIEW QUESTIONS FOR BRANCHING, LOGICAL, AND DECIMAL OPERATIONS

- Use only 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 141, 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. Branching is accomplished by:

   a. Storing the PSW and fetching a new PSW.
   b. Replacing the entire PSW with the "branch to address."
   c. Replacing bits 40 - 63 of the PSW with the "branch to address."
   d. Adding bits 40 - 63 of the PSW to the effective generated address.
   e. None of the above.

2. Which of the following M1 (Mask) fields would be used by a "branch on condition" instruction to check only for a condition code of 11?

   a. 0011
   b. 1100
   c. 1000
   d. 0001
   e. 1111

3. Which of the following "branch" instructions will always retain the previous contents of the PSW's instruction address?

   a. Branch on Condition
   b. Branch on Count
   c. Branch and Link
   d. Branch on Index High
   e. Branch on Index Low or Equal

4. The "branch on count" instruction will:

   a. Add a value of 1 - 16 to the first operand and unconditionally branch.
   b. Add a value of 1 to the first operand and branch if there is a high order carry.
   c. Subtract a value of 1 from the first operand and branch if the result is zero.
   d. Branch if the count in the first operand has been reduced to zero by a previous instruction.
   e. Subtract a value of 1 from the first operand and branch if the result is not zero.

5. The "branch on index high" instruction will branch after:

   a. Reducing the first operand by 1 and comparing the result with a second operand.
   b. Adding the second operand to the first operand and comparing the sum with a third operand.
   c. Comparing the second operand to the first operand.
   d. Using an index register to generate the "branch to address."
   e. Determining that the value in the index register is greater than that in the base register.

6. Examine the following symbolic program. The starting address is decimal 2000.

```
BALR  1, 0
LA    1, 0 (0, 1)
LA    2, 8 (0, 1)
LPR   2, 2
BCTR  1, 2
```

How many times will the "load positive" instruction be executed?

a. 2000
b. 2002
c. 2004
d. 2006
e. None of the above.

7. Which of the following symbolic instructions would zero out register 2.

a. SR 2, 2
b. SLL 2, 0032 (0)
c. XR 2, 2
d. All of the above.
e. None of the above.

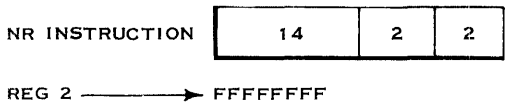
8. The "move zones" instruction

a. Moves bits 0 - 3 of the second operand bytes into bits 0 - 3 of the first operand.
b. Zeros out the zones of the first operand.
c. Processes the bytes in a right to left direction.
d. Can only move the bits from a maximum of sixteen bytes.
e. None of the above.

9. The "move with offset" instruction:

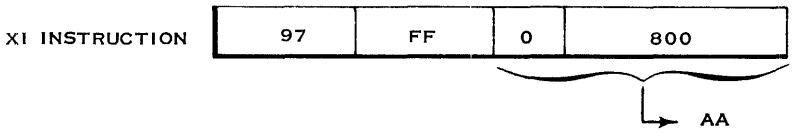
a. Does not change the right-most four bits in each byte of the first operand.
b. Moves a field from the second operand into the first operand, displacing it four bits to the left.
c. Processes the bytes in a left to right direction.
d. All of the above.
e. None of the above.

10. What is the result of the following "and" instruction?



a. 0 0 0 0 0 0 0 0
b. 0 0 0 0 0 0 0 1
c. F F F F F F F E
d. F F F F F F F F
e. None of the above.

11. What is the result of the following "exclusive or" instruction (SI format)?



a. 00
b. 55
c. 66
d. FF
e. None of the above.

12. The "compare logical" instruction can be used to compare:

a. Alphanumeric information
b. Zoned decimal operands
c. Packed decimal operands
d. All of the above.
e. None of the above.

13. Examine the following symbolic program.

```
LA   5, 2048 (0, 0)
LA   5, 0 (5, 5)
LA   6, 0 (5, 5)
LA   7, 0 (6, 5)
```

Indicate the base address (decimally) that will be in register 6 at the end of the program.

a. 2048
b. 4096
c. 6144
d. 8192
e. 10240

14. The "translate" instruction can be used to translate bytes of data:

    a. From one character code to any other character code.
    b. Only from EBCDIC to some other character code.
    c. Only to EBCDIC from some other character code.
    d. Only from ASCII to EBCDIC.
    e. None of the above.

15. Given the following "translate" instruction and an argument byte (in hex), choose the correct statement.

TR INSTRUCTION

DC	00	0	800	0	F00

ARGUMENT BYTE → F7

    a. The argument byte will be replaced by the function byte from hex location 0008 F7.
    b. The argument byte will be replaced by the function byte from hex location 000FF7.
    c. The argument byte will replace the function byte at hex location 0008F7.
    d. The argument byte will replace the function byte at hex location 000FF7.
    e. None of the above.

16. After a "translate and test" instruction, a condition code of 01 would mean:

    a. No significant character was located.
    b. All the argument bytes were used and a significant character was located.
    c. One or more significant characters were located.
    d. One significant character was located and replaced with a function byte.
    e. One significant character was located and its address is in register 1.

17. Which of the following can cause a decimal overflow on an AP instruction?

	1st operand	2nd operand
a.	47 9C	52 0C
b.	47 2C	00 37 6C
c.	04 7C	01 00 0C

    d. All of the above.
    e. None of the above.

18. Show the storage contents after executing the following "multiply decimal" instruction.

MP INSTRUCTION

FC	1	0	0	100	0	102

HEX LOCATIONS 100 – 102 → 097D6C

a. 58 2D 6C
b. 00 58 2D
c. 58 2C 6C
d. 00 58 2C
e. 09 7D 6C

19. Which of the following pattern fields would be necessary in order to produce the following edit result?

Source Fields 0007 6 D in packed decimal digits
Edited Fields **** . 76 b CR b

a. * d d d . d d b c R b
b. * d d d . d d b b b b
c. b d d) . d d * * * *
d. * * *) . d d b CR b
e. * d d) . d d b CR b

20. The "edit and mark" instruction will:

a. Edit the field and put the dollar sign next to the most significant digit.
b. Edit the field and put the address of the most significant digit minus 1 in register 1.
c. Edit the field and put the address of the most significant character in register 1.
d. Not edit the field.
e. Not edit the field but will set the condition code to 00 if no significant characters are located.

ANSWERS TO REVIEW QUESTIONS

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

You have now completed your study of the System/360's standard instruction set and the decimal feature with the exception of input-output operations. These I/O operations will be covered in your next self-study book.

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 this book).

Alphabetical Index

Book 4 System/360 Branching, Logical and Decimal 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:	Branching Operations	Yes ☐ No ☐
Section II:	Logical Operations	Yes ☐ No ☐
Section III:	Decimal Operations	Yes ☐ No ☐
Section IV:	Analyzing Decimal Feature 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 R23-2958-1

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.

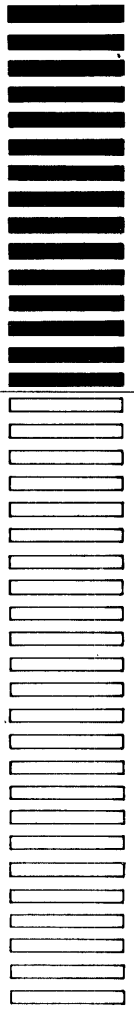
- Were you given a copy of this text to write in and keep? . . . . . Yes ☐ No ☐
- How many hours per day were scheduled for this course? _____
- Were you interrupted during this time? . . . . . Yes ☐ No ☐
- How many hours were needed to complete this course? _____
- Did you require assistance during 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

FOLD

FOLD

TEAR HERE