Classification of Instruction based on Computation

→ According to the Operation carried by the computer, the instructions are classified into 3 categories.

* Data transfer Instruction
* Data Manipulation Instruction : These are divided into 3 types.
  - (i) Arithmatic Instruction
  - (ii) Logical "
  - (iii) Shift "
* Program control Instruction.

Data transfer Instruction :-

→ The instruction which are transferred from one location to another location. The locations are memory, registers, I/O devices

Name	Mnemonics
Load	LD
Store	ST
MOVE	MOV
Exchange	XCH
Input	IN
Output	OUT

Data manipulation Instruction :-

Arithmatic Instruction :-

Name	Mnemonics
Increment	INC
Decrement	DEC
ADD	ADD
SUB	SUB
Multiplication	MUL
Add with Carry	ADDC
Sub with Borrow	SUBB
Negate	NEG

Q IMP

Logical Instruction

Name	Mnemonics
Clear	CLR
Complement	COM
And	AND
OR	OR
Exclusive OR	XOR
Clear carry	CLRC
Set Carry	SETC
Enable Interrupt	EI
Disable Interrupt	DI

Shift Instruction:-

Name	Mnemonics
Logical shift right	SHR / L Shift R
Logical shift left	SHL / L Shift L
Arithmatic Shift right	ASHR / A shift R
Arithmadic shift left	ASHL / A Shift L
Rotate Right	ROR
Rotate left	ROL
Rotate Right through carry	RORC
Rotate left through carry	ROLC

Program Control Instruction:-

Name	Mnemonics
Branch	BR → conditional Branch
Jump	JMP → un " "
Subroutine call	CALL
Subroutine Return	RET
Compare	CMP
Test	TST

Logical Instructions:-

→ Logic operations are AND, OR, and NOT.

Ex :- NOt dst : Complements all bits contained in the destination operand.

NOt dst

(or) Negate dst

```
Not  R0
Add  #1, R0
```

I/P :- R0 | 3 |

2's complement of a number

0000 0011

1111 1100

+1

1111 1101

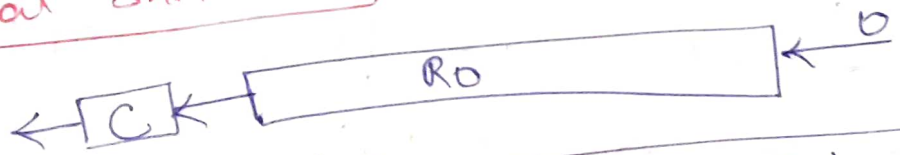
-4+1 = -3

Ex :-

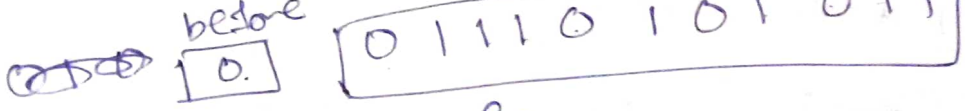
```
And     #$FF000000, R0
Compare #$5A000000, R0
Branch=0  YES
```

SHIFT and ROTATE INSTRUCTIONS :-

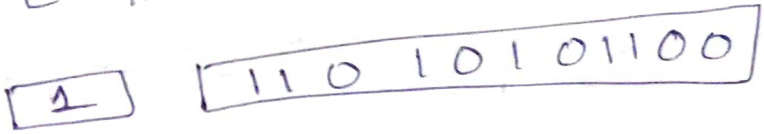
(i) Logical Shift left :-



Ex :-

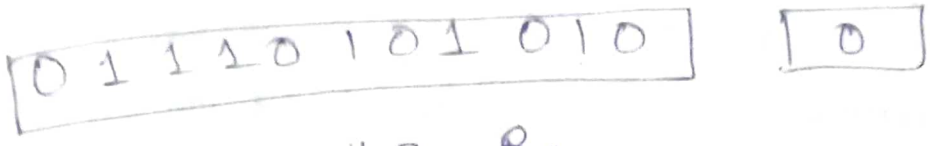


LShift L #2, R0

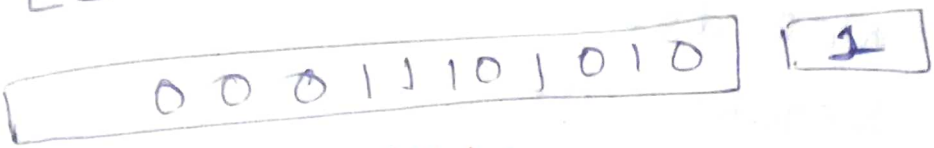


(2) Logical Shift Right

Ex

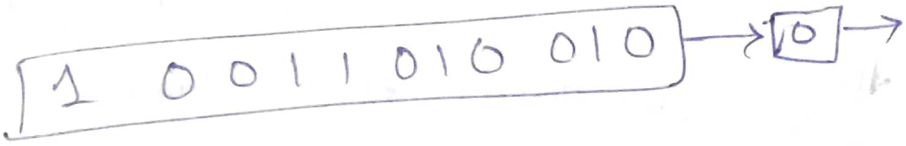


L Shift R #2, R0

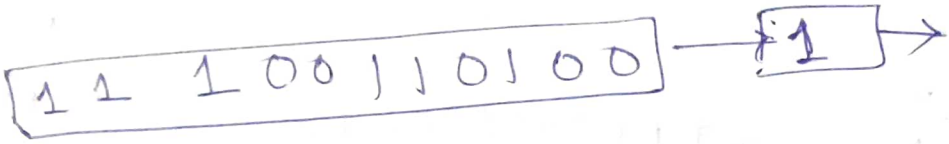


(3) Arithmatic Shift right :-

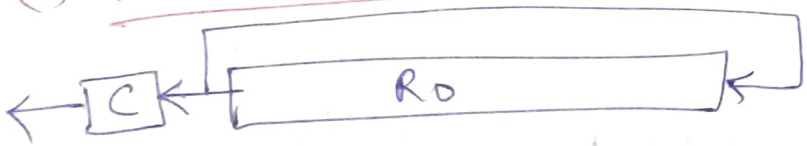
Ex :-



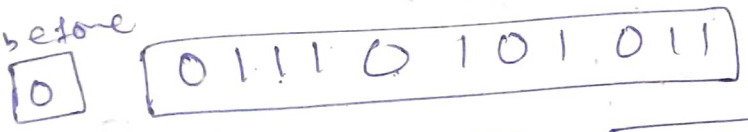
A Shift R #2, R0



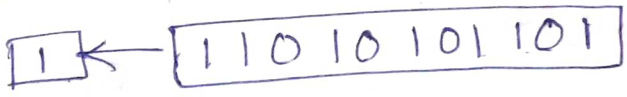
(4) Rotate left without carry :-



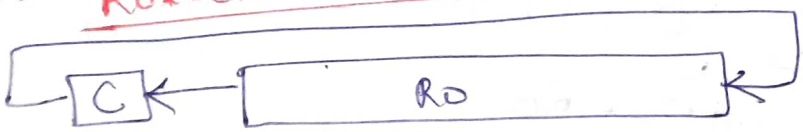
Ex :- before



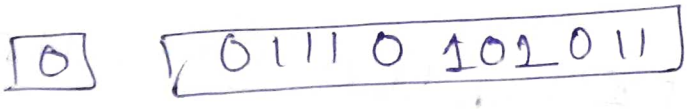
Rotate L #2, R0



(5) Rotate left with carry :-



Ex :-



Rotate LC #2, R0