S. No.	Date	Title	Page No.	Teacher's Sign / Remarks

System Programming

System Programming are programs which are used to execute user Application Programs.

- Components :

1) (Assembler) → Assembly language → Assembler → Machine Language.

2) (Macroprocessor) → It allows abbreviations instead of some repeated group of identical statements.

3) (Loader) → It prepares object program for
   - Execution
   - Start the execution.

4) (File & OS) → Allow Flexible storing and retrieval of information.

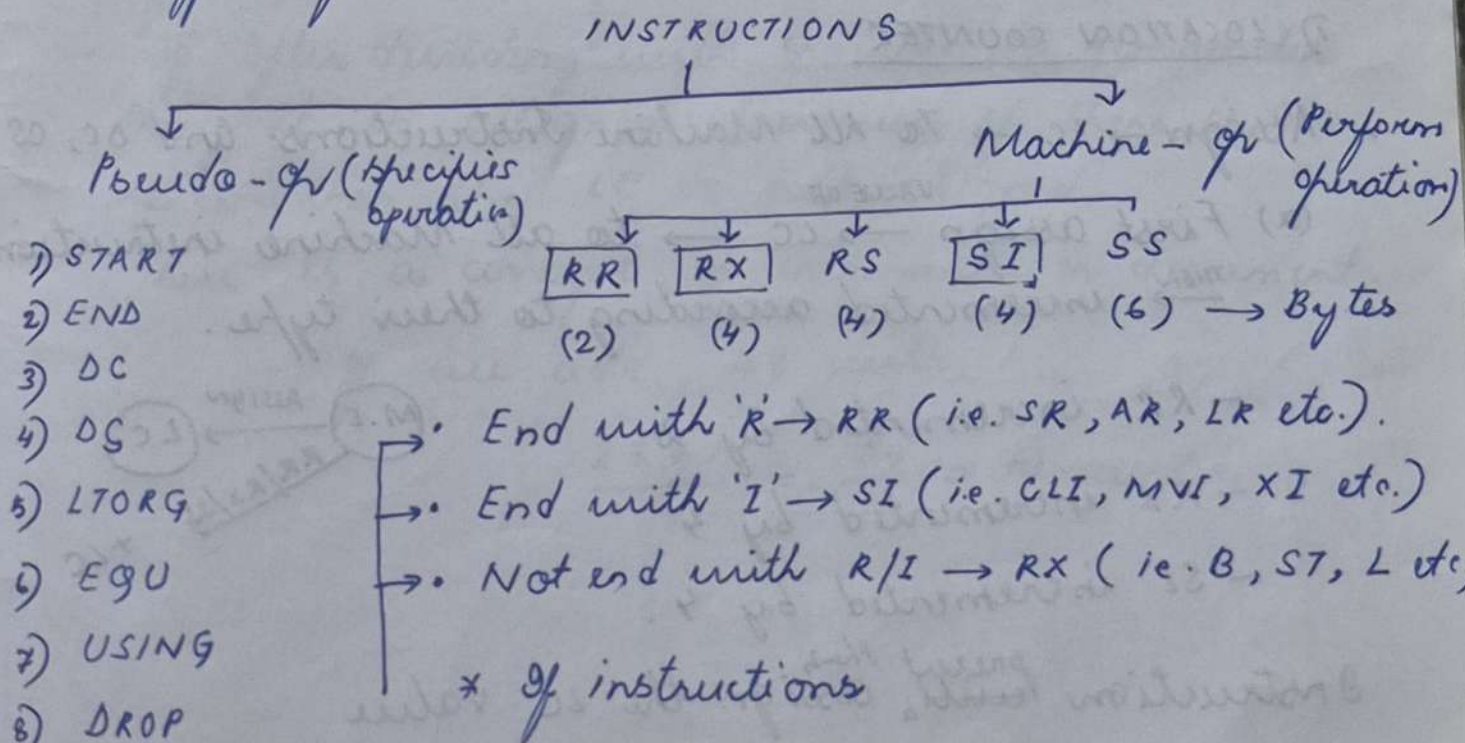
5) (Compilers) → High language program → Compiler → Machine lang Program

ASSEMBLERS

- Suppose we need to add something, we don't need to write 01101... directly we can write "ADD"
- Assembler converts Assembly Language into Machine equivalent language.
- ADD, MUL etc. are the Mnemonics.
- Advantages is that the programmer who writes Assembly Language is more efficient & productive. because it is human understandable.
- Program → Set of instructions
- General instruction format:

LABEL	OPCODE	OPERAND
(Optional)	(Instruction)	(0/1/2) ← Number of operands

- Types of Instructions:



1) START
2) END
3) DC
4) DS
5) LTORG
6) EQU
7) USING
8) DROP

- End with 'R' → RR (i.e. SR, AR, LR etc.).
- End with 'I' → SI (i.e. CLI, MVI, XI etc.)
- Not end with R/I → RX (ie. B, ST, L etc)

* Of instructions

(3)

→ Memory Unit:

INFORMATION	BYTE
Character (C)	1
Halfword (H)	2
Fullword (F)	4
Doubleword (D)	8

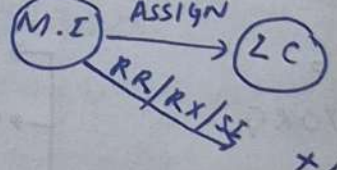
→ DESIGN PROCEDURE OF ASSEMBLER.

We need to cover first four procedure of the 5.

1) LOCATION COUNTER (LC)
2) SYMBOL TABLE (ST)
3) LITERAL TABLE (LT) — Optional
4) BASE TABLE (BT)
5) MACHINE CODE.

1) LOCATION COUNTER

- Assign → LC → To All Machine Instructions and DC, as
  (a) First assign → VALUE OF LC → to all Machine instruction THEN → incremented according to their type.
  - RR incremented by 2
  - RX incremented by 4
  - SI incremented by 4.



Instruction present then assign the LC value
After assigning it will check the type of LC.

There are 3 types of instructions (RR, RX, SI).

(b) Assign $\xrightarrow{\text{value of}}$ LC $\longrightarrow$ To DC and DS First Alignment should be done.

- C multiple of 1
- H multiple of 2
- F multiple of 34
- D multiple of 8

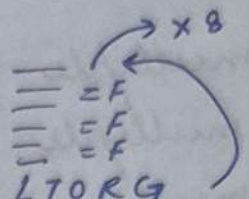
Before assigning, we should Align and the meaning of alignment means Multiple.

- For eg. in DC and DS, an F has 9 come. And the current value has come to be 23. This is the location counter. First we do alignment.
- It should be multiple of 4 as F has multiple of 4.
- If after deviding with 4 there is remainder then it is alligned otherwise if there is no '0' then the LC is not aligned.
- There is a concept of increment or decrement.

If we add 23 with 1

$$\frac{23+1}{4} = \frac{24}{4} = \text{remainder } 0.$$

→ (c) If LTORG encounters then first Assign the value of LC Literal (=F) First and First Literal always stored in Next Doubleword Location (i.e. Multiple of 8).



- If we encounter with LTORG; then we will wait and the literals present before ~~Litero~~ LTORG and we will assign the value of LC literals (i.e. =F).
- First literal should be multiple of 8.

2) SYMBOL TABLE

Symbol	Value	R/A
—	—	—
—	—	—
—	—	—
—	—	—

R → Relocation
A → Allocation

- Value is written from LC.

3) LITERAL TABLE

Literal	Value	R

4) BASE TABLE

CONTENT	REGISTER NO

Base table is prepared by USING instructions.

→ Design procedure of Assembler :

*. We have studied the first 4 instructions.

5) MACHINE CODE :

To generate machine code we need to know the machine instructions.

(a) Pseudo-op, (b) Machine-op.

(b) Machine-op - There are 5 operands.

(i) RR - We will generate machine code.

Format: op1, op2 ⇒ RR val1 val2

Value 1 will be calculated from operand 1
Value 2 " " " " operand 2.

* When we generate machine code of RR instructions, operand will allow only Numeric value.

(ii) RX Format: op1 op2 ⇒ RX val1 offset (X, B)

(X, B)
↓ ↓
INDEX BASE.

(iii) SI Format:

op1, op2 ⇒ SI val2, offset (B).

(iv) If we encounter with BNE, with only 1 operand. int.

BNE op ⇒ BC 7, offset (X, B). (# convert BNE → BC 7)

(v) BR 14 ⇒ BCR 15, 14 (# convert BR 14 → BCR 15,14)

(vi) BC ⇒ corresponding const.

(vii) DS ⇒ blank.

Formula for calculating offset:

offset = value (Symbol / Literal) − [B] → content of base register

* Pseudo-op - START, END, DC, DS, LTORG, EQU, USING, DROP

* Machine-op - RR, RX, SI.
2 4 4 bytes.

EXAMPLE

	INST^n		LC	Pseudo-op
MYPRO	START	0		0
BEGIN	BALR	15, 0	0 + 2 byte	
	USING	BEGIN + 2, 15		2
RR int →	SR	4, 4	2 + 2	
RX int →	L	3, = F '10'	4 + 4	
LOOP	L	2, DATA (4)	8 + 4	
RX →	A	2, = F '25'	12 + 4	
RX →	ST	2, DATA (4)	16 + 4	
RX →	A	4, = F '4'	20 + 4	
RX →	BCT	3, LOOP	24 + 4	
RR →	BR	14	28 + 2	
	LTORG			30
DATA	DC	f '1, 3, 3, 3, 3, 4, 5, 9, 0'	44 ←	44 + 36
	END			80

- First we will check what is the instruction, whether it is Pseudo op / Machine op.
- If we get a pseudo-op, then we will not assign value in 'LC'. For our identification we will write '0' against pseudo-op. Eg. START | LC | 0
- But if we get DC/DS, then we can assign value.
- Next we will see BALR. It is not a pseudo-op.

- When LTORG is encountered we will stop the process, and the literals present before LTORG, we will assign value of Location counter and the first literal should be multiple of 8.
- The ones which have = in front are the literals.
- = F '10'
  = F '25'
  = F '4'
- Now what is the current value? It is (30) First literal should be multiple of 8.

$$8\overline{)30}\,(3 \quad \frac{24}{6}$$

The remainder is not 0. Now we will increment 30, so that it becomes multiple of 8.

$$8\overline{)31}\,(3 \quad \frac{24}{7} \qquad , \qquad 8\overline{)32}\,(4 \quad \frac{32}{0}$$

F → 4 bytes

LDC = 4

= F '10'	32 + 4
= F '25'	36 + 4
= F '4'	40 + 4 = 44

So, we will write 44 in the next tab.

Now, ~~if~~ F means '4', we will check whether 44 is multiple of 4, if yes, we will write in 'LC'

. Now we will check how many constants are there in 'DC'. There are 9 constants ('1, 3, 3...0')

f is 4 bytes

$\therefore 4 \times 9 = 36$

ie $44 + 36 = 80$

Now, we shall prepare the

(a) SYMBOL TABLE

SYMBOL	VALUE	RELOCATION
MYPRO	0	R
BEGIN	0	R
LOOP	8	R
DATA	44	R

* We shall write all the labels in the 'SYMBOL'
* For writing the 'value', we will always prefer the 'LC' but if not present we will search the 'Pseudo-op' column.

*. When there is the value of LC and pseudo-op we will write the LC value.

(b) LITERAL TABLE

LITERAL	VALUE	RELOCATION
=F'10'	32	R
=F'25'	36	R
=F'4'	40	R

(c) BASE TABLE

CONTENT	REGISTER
2	15

Now Finally we will generate Machine Code:

(d) MACHINE CODE

RELATIVE LOCATION	CONTENT
0	BALR 15,0
2	SR 4,4
4	L 3, 30(0, 15)
8	L 2, 42(4, 15)
12	A 2, 34(0, 15)
16	ST 2, 42(4, 15)
20	A 4, 38(0, 15)
24	BCT 3, 6(0, 15)
28	BCR 15, 14
32	10
36	25
40	4
44	1

literal

INC 4 bytes.

BEGIN+2 = 0+2 = 2

* For base table we will always be inserting the 'USING' instructions.

[USING BEGIN+2, 15]
CONTENT REGISTER

* BEGIN we will write from 'SYMBOL' table.

* Relative location contain the value of 'LC'.

OFFSET.

L 3, =F'10'

3, (value - content)

3, (32 - 2)

3, 30(0, 15)

INDEX REG

* IF THERE IS NO BRACKET WE WILL USE (0)

44-2

48	3
52	3

```
    L   3, =F'10'
*   L   3, 32-2(0,15)
    L   3, 30(0,15).

*   L   2, DATA(4)
    L   2, 44-2(4,15)
    L   2, 42(4,15).

*   A   2, =F'25'
    A   2, 36-2(0,15)
    A   2, 34(0,15)

*   ST  2, DATA(4)
    ST  2, 44-2(4,15)
    ST  2, 42(4,15)
*   A   4, =F'4'
    A   4, 40-2(0,15).
    A   4, 38(0,15)

*   BCT 3, LOOP
    BCT 3, (8-2)(0,15)
    BCT 3, 6(0,15)
```

SINCE NO BRACKET WAS PRESENT AFTER (F'10')

REGISTER

* (-2) & 15 is written from base table whenever offset is encountered.

Then we will increment by 4 bytes & put the value of DC

48 3

52 3

56 3

* BR 14 converted to BCR 15,14.

* LITERAL values are written

31 → 10

40 → 25

44 → 4

cont.