Answers

Q.1.

1. Features of macros.

Ans. • Definition of Macro :-
Macro is defined to be ~~as a~~ single line abbreviation for group of instructions.

~~Features of Macros :-~~

• Macroprocessor :-
It is a program which is responsible for processing of the macro.

• Format :-

```
MACRO   ←  Start of definition
 ___    ←  Macro name
 ___  }
 ___  } ←  Macro body
 ___  }
MEND    ←  End of definition
```

• Features of Macros :-

i. Parameters -

a. FORMAL Parameters :-
These are the parameters that appear in the macro definition.

b. Actual Parameters :-
These are the parameter that appear in the macro call.

E.g. —

```
        MACRO
        INCR1 &ARG    }
        A1, &ARG      }  Formal
        A2, &ARG      }
        A3, &ARG      }
        MEND
          :
                               A1, DATA 1
       { INCR1  DATA 1  <----  A2, DATA2
       {   :                   A3, DATA 3
Actual {   :
       {   :                   A 1, DATA 2
       { INCR 2. DATA 2 <----  A2, DATA 2
           |                   A3, DATA 2
           |
       END
```

Parameter Passing Techniques :

a. Positional Parameters (PP)
In this technique the parameters correspond to their position.

b. Keyword Parameters (KP)
In this technique the parameters correspond to their keywords.

E.g of Positional Parameter: -

```
MACRO
INCR 2, &ARG1, &AR2, &ARG3
A1, &ARG1
A2, &ARG2
A3, &ARG 3
MEND
  :
```

INCR 2 DATA 1, DATA 2, DATA 3 — A1, DATA 1
— A2, DATA 2
— A3, DATA 3

⋮

INCR 2 DATA 2, DATA 3, DATA 1 — A1, DATA 2
— A2, DATA 3
— A3, DATA 1

⋮

END

E.g. of Keyword Parameter :—

MACRO
INCR 3 &ARG1 =, &ARG 2 =, &ARG 3 =
A1, &ARG 1
A2, &ARG 2
A3, & ARG 3
MEND

⋮

INCR 3 &ARG 1 = DATA 1, &ARG 2 = DATA 2,
&ARG 3 = DATA 3 ~~[illegible]~~

~~INCR~~

INCR 3 &ARG2 = DATA 2, ARG1 = DATA 1, &ARG3 = DATA 3

⋮

INCR 3 DATA 1, DATA ~~3~~2, DATA 3

⋮

END

A1, DATA 1
A2, DATA 2
A3, DATA 3

- Functions of Macroprocessor

→ Recognize the macro definition.] Pass 1
→ Store the macro definition.
→ Recognize the macro call.] Pass 2
→ Perform macro expansion.

2. Assembler - Forward Reference Problem Theory

Ans. • Assembler Definition:-
Assembler is a language translator that takes as input assembly language program and generates its machine level language equivalent along with the information required by the loader.

• Forward Reference Problem:-
i. The rules of assembly language program state that the symbol can be defined anywhere in the program.
Hence, there may be some cases in which the reference is made to the symbol prior to its definition and such a reference is called forward reference

ii. Due to forward reference assembler cannot assemble the instruction and such a problem is called forward reference problem.

iii To solve the problem assembler will make two passes over the input program.

iv. The purpose of Pass 1 is to define the symbols and literals encountered in the program.
The purpose of Pass 2 is to assemble the instruction and assemble the data.

[illegible]

3. Difference between macro and sub-routine.

Ans.

	Macro	Subroutine
1.	Macro increases the size of the program.	Subroutine does not increase the size of program.
2.	Does not alter the flow of execution.	Alters the flow of execution.
3.	Programs using macro would get executed faster.	Programs using subroutine would get executed comparitively slower.
4.	Macro does not require any return address.	Subroutines require return address.
5	Macro cannot return a value.	Subroutine can return a value.
6.	Macro requires an overhead of macroprocessor.	Subroutines require no such overhead.
7.	Macro are calls processed at translation time.	Subroutine calls are processed at execution time.
8.	If LOC (Lines of Code) is in the range of 3-5 then, macro usage is recommended.	If LOC is beyond 5 then subroutine usage is recommended.

4. Absolute Loader.

Ans. Loader Definition :-
Loader is a system program which is responsible for preparing the object program for execution and start the execution.

- Absolute loader :-

☆ Allocation is done by programmer.
An absolute loader is a loader that places absolute code into main memory beginning with the initial address (absolute address) assigned by the assembler. No address manipulation is performed.

- Design of absolute loader :-

Absolute loader requires the following types of cards from the assembler :-

i. TXT cards :-

Card Type	Count	Address	Contents
0	2	4050	OP | R_1 | R_2

→ Card Type : It is always 0 for TXT cards.
→ Count : It indicates in bytes the amount of binary information ~~which is to be loaded~~ available in the card
→ Address : It indicates the location to which the binary information should be loaded.
→ Contents : It contains the binary information which is to be loaded (binary information mean

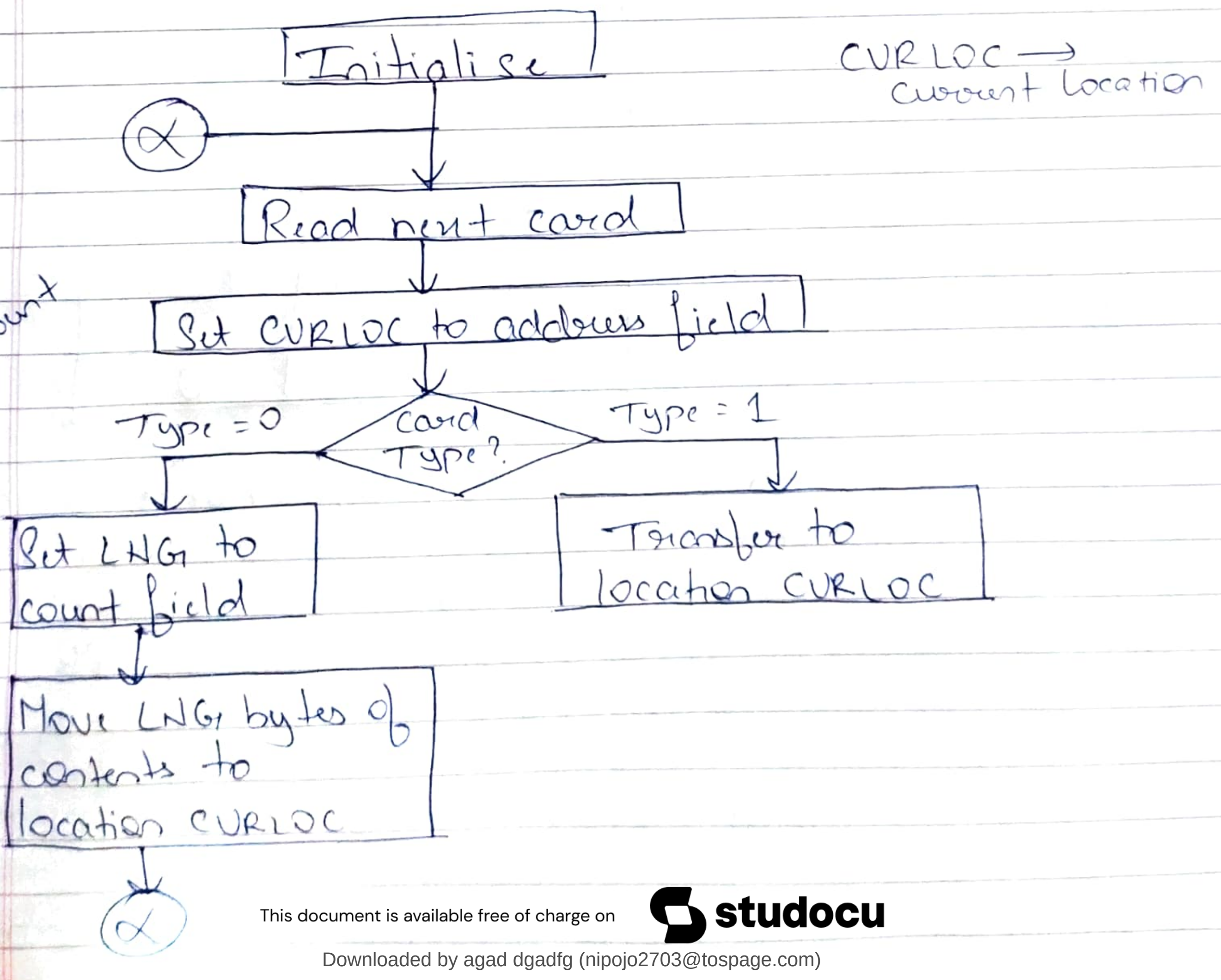
assembled instructions and data).

ii. Transfer Card :-

Card Type	Count	Address	Content
1	0	$\boxed{}$	—

→ Card Type : It is always 1 for transfer card.
→ Count : It is always 0 for transfer card.
→ Address : It indicates the location from where the execution of object program should begin.
→ Contents : It is always blank for transfer cards.

Detailed a Flowchart of Absolute Loader



Initialise
α
Read next card
Set CURLOC to address field
LNG = count
Card Type?
Type = 0
Type = 1
length
Set LNG to count field
Transfer to location CURLOC
Move LNG bytes of contents to location CURLOC
α
CURLOC → current location

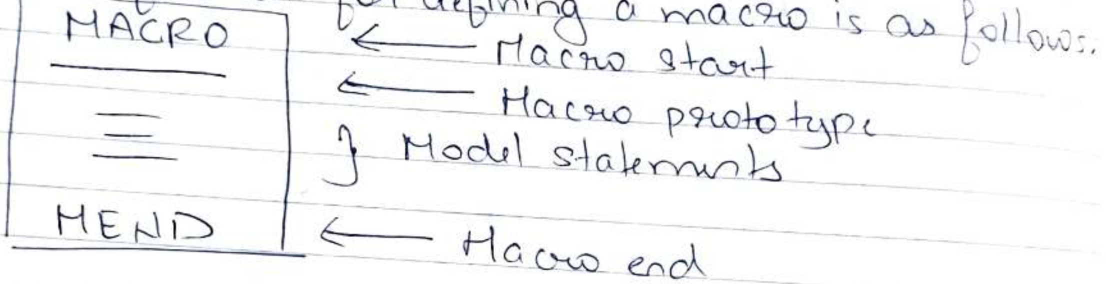
21 5

Q2.

1. Working of single pass macro with flowchart.

Ans:→ ^Macro definition^ This design is based on a rule which states that all the ~~main~~ macro definitions should appear back to back at the beginning of the program.

→ The format for defining a macro is as follows.



- Format of Database:—

i. Macro Name Table (MNT):

Macro name	MDTP	#PP	#KP	KPTP	# EVS
EVAL		1	2		0

→ MNT is used for storing the macro name along with MDTP which indicates the location in MDT where the corresponding definition is store.

→ MNT keeps a track on the number of PP's and the number of KP's for error handling purpose.

→ MNT also maintains a pointer to KPT called KPTP

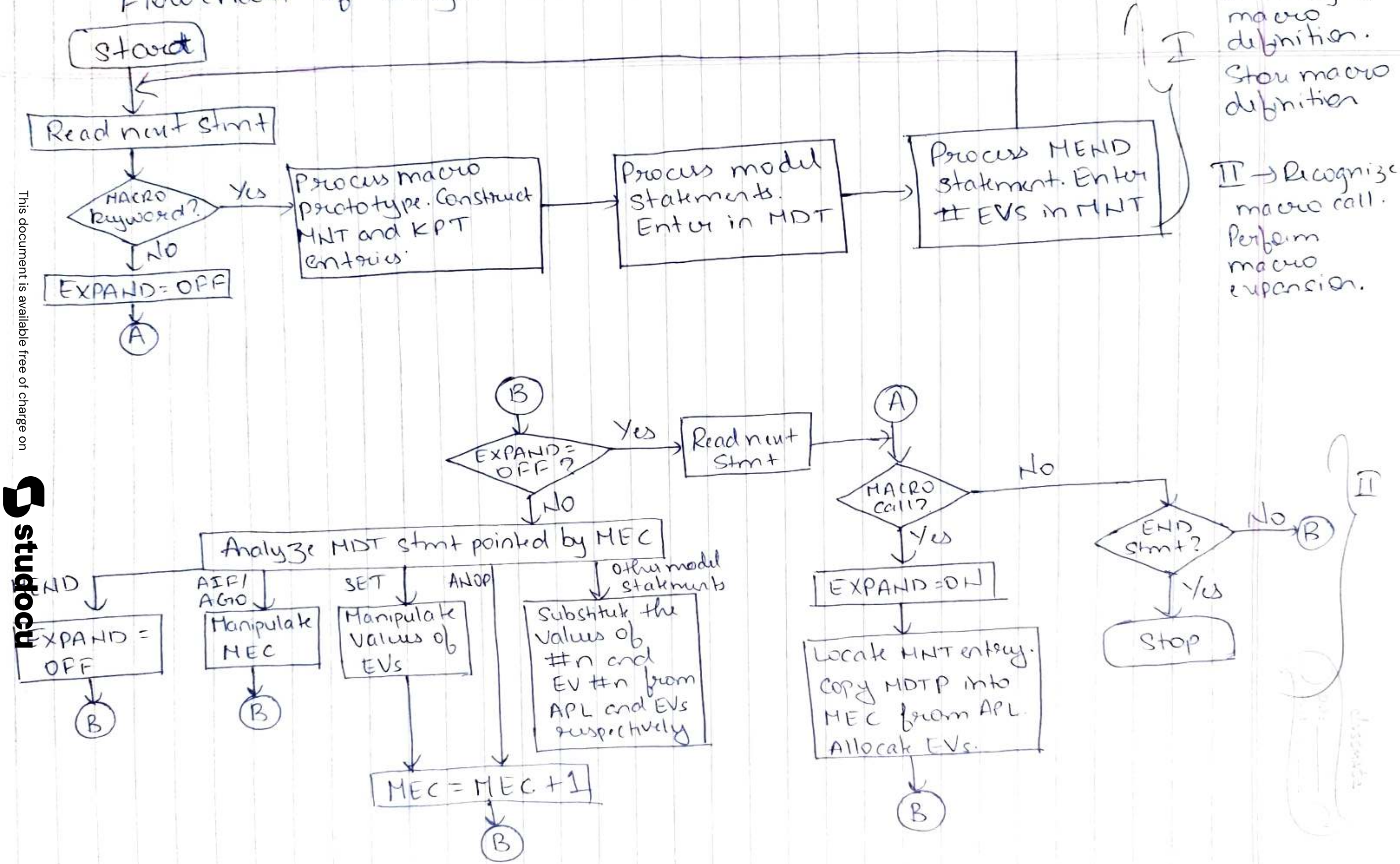
→ MNT also keeps a track on the number of expansion variables required by the macro

I → Recognize macro definition

Flowchart of Single Pass Macro Processor

start

Flowchart of Single Pass Macro Processor


Start
Read next stmt
MACRO keyword?
Yes
No
EXPAND=OFF
A
Process macro prototype. Construct MNT and KPT entries.
Process model statements. Enter in MDT
Process MEND statement. Enter # EVs in MNT
I
I → Recognize macro definition. Store macro definition
II → Recognize macro call. Perform macro expansion.
B
EXPAND= OFF?
Yes
Read next Stmt
No
Analyze MDT stmt pointed by MEC
MEND
AIF/ AGO
SET
ANOP
Other model statements
EXPAND = OFF
Manipulate MEC
Manipulate values of EVs
Substitute the values of #n and EV #n from APL and EVs respectively
MEC = MEC + 1
A
MACRO call?
Yes
No
EXPAND=ON
Locate MNT entry. Copy MDTP into MEC from APL. Allocate EVs.
END stmt?
No
Yes
Stop
II

ii. Macro Definition Table (MDT):

```
Macro Definition
  load #1
  #3 #2
  Store #1
  MEND
```

Index	Macro definition

→ MDT is used for storing the macro definition along with MEND statement.
→ While storing the definition, the parameters and expansion variables would be replaced by #n, and EV #n respectively.
→ While performing the expansion, #n and EV #n would be replaced by their values from APL and EVS respectively.

iii. Keyword Parameter Table (KPT):

Keyword	Default
&Y	
&OP	ADD

→ KPT is used for storing the default values of keyword parameters.

iv. Expand:
→ The value of expand indicates whether the macro expansion is ON/OFF.

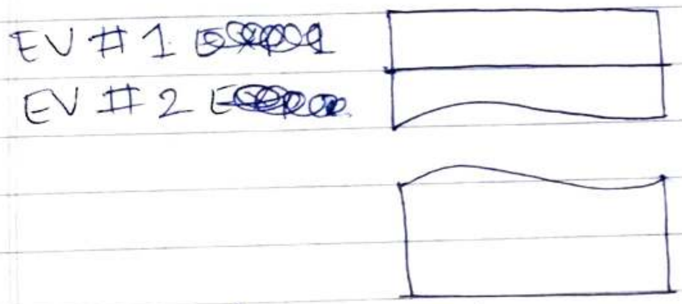
V. Macro Expansion Counter (MEC):
→ MEC points to the MDT statement which is under expansion.

vi. Actual Parameter List (APL):

#1	P
#2	Q
#3	SUB

→ APL is used for parameter replacement procedure.

vii. Expansion Variable Storage (EVS):



→ At the time of macro expansion, EVS would be allocated depending on the number of expansion variables required by the macro.

2. Direct Linking Loader (DLL)

Ans. DLL is a general relocatable loader.
DLL requires the following types of cards from the assembler
i. External Symbol Directory Cards (ESD).
ii. Text Cards (TXT).
iii. Relocation and Linkage Directory Cards (RLD)
iv. End card (END)
v. End of File (EOF) or Loader Terminator (LDT).

Pass 1 Database:
i. Object Cards –
These cards contain the object program in a

format required by the loader.
(ESD, TXT, RLD, END, EOF)

ii. Initial Program Load Address - (IPLA).
IPLA is obtained by the loader from os

iii Program Load Address (PLA) -
PLA is used for assigning the addresses to the segments and local definitions.

iv. Global External Symbol Table (GEST) -
GEST is used for keeping a track on the address that are assigned to the symbols

v. Copy File (CF) -
It contains the copy of the object cards to be used by Pass 2.

vi. Load Map -
It is a printed ~~to~~ listing of GEST for program reference.

Pass 2 Database -
i. Copy File (CF)
ii. IPLA
iii. PLA
iv. GEST
v. LESA (Local External Symbol Array) -
It is used for performing relocation and linking.
vi. Execution Address -
It indicates the location from where the execution of object program should begin.

Program :-

```
0    PG1     START
                ENTRY  A, B
                EXTRN  PG2, C

20   A

30   B
40              A (A)
44              A (B+25)
48              A (C-PG2)
52              END

0    PG2        START
                ENTRY C
                EXTRN A, B

16   C
24              A (A)
28              B (B+15)
32              END
```

Assembler :-

ESD (PG1) -

Symbol	Type	ID	Rel Addr.	Length
PG1	SD	1	0	52
A	LD		20	
B	LD		30	
PG2	ER	2		
C	ER	3		

ESD (PG2) -

~~Symbol Type Rel Addr. ID~~

Symbol	Type	ID	Rel Addr.	Length
PG2	SD	1	0	32
C	LD		16	
A	ER	2		
B	ER	3		

TXT (PG1) -

Card Type	Count	Rel Address	Contents	Comments
0		⋮		
		40	20	
		44	55	
		48	08	← Unknown due to ER

TXT (PG 2) -

Card Type	Count	Rel Address	Contents	Comments
0		⋮		
		24	0	← unknown due to ER
		28	15	

Lng ID of SD

RLD (PG1) -

ESD ID	Length	Flag	Rel Address
→ 1	4	+	40
1	4	+	44
3	4	+	48
2	4	−	48

RLD (PG2) -

ESD ID	Length	Flag	Rel Address
2	4	+	24
3	4	+	28

END (PG1) -

Address
52

END (PG2) -

Address
32

Pass 1 :- Define Segment and Local definition
Pass 2 :- Transfer TXT card and perform
Relocation and Linking.

Q.3.

1. Difference between system software and application software.

Ans System Software	Application Software
1. Maintain the system resources and give the path for application software to run.	Is built for specific task.

2.