8871
Upmanyu Jha
Te-comps A

Assignment 2

SPCC

S1 Solution:- Pass I Output

- Macro name table (MNT)

Index	Macro Name	MDT Index / MDT Value
1	INCR	1

- Macro Definition Table (MDT)

Index	code
1	INCR
2	m 1 =f'2'
3	st 1, temp
4	sr 2, 2
5	MEND

- Argument List Array empty

Intermediate code:-

```
PG12    START   0
        USING   *,0
        l   1, two
        ar  1, 2
        INCR
        a   1, three
        m   1, =f'2'
        st  1, temp
        sr  2, 2
        INCR
        s   5, =f'5'
        d   5, three
b       eqe 4
two     dc  h'2'
three   dc  f'2'
temp    ds  1f
        end.
```

Pass 2 Output

Expand Source code.

```
PG2     START   0
        USING   *, b
        l   1, two
        ar  1, 2
        m   1, =f'2'
        st  1, temp
        sr  2, 2
        a   1, three
        m   1, =f'2'
        st  1, temp
        sr  2, 2
        m   1, =f'2'
        st  1, temp
        sr  2, 2
        s   5, =f'5'
        d   5, three
b       equ 4
two     dc  f'2'
three   dc  f'3'
temp    ds  1f
        end
```

Q2

① External Symbol Dictionary contains information about all symbols that defined in the program but referenced somewhere.

② Relocation and linkage Directory contains information about the locations in the program where the content depends on the address at which program is placed.

③ Thus, ESD & RLD table will be as follows:

ESD

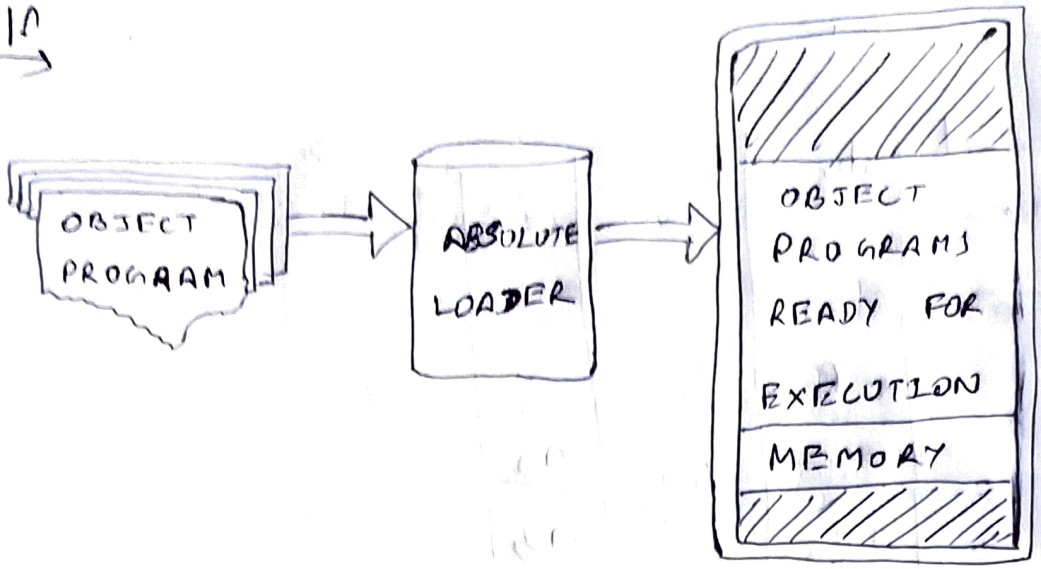
Symbol	Type	ID	Relative Addr	Length
PG1	SD	01	0	34
A	LD		22	
Solution	ER	2		
Delta	ER	3		

RLD

ESD ID	Length	Flag	Rel. Addr
1	4	+	14
2	4	+	22
3	4	+	30

Q3

Sol^n



Absolute Loader scheme

- The loader simply accepts machine language code and places it into main memory specified by the assembler.
- It is similar to the "compile and Go" loaders.
- The loader accepts the machine language instructions and places it into the core, then initiates execution by transferring control to the starting of the program in this the data is store on cards instead of being placed in main memory for execution.

- The object code is loaded at the specified locations in the memory. At the end the loader jumps to the specified location to begin the execution of the loaded program. In absolute loader no linking or relocation is done.

Absolute loader requires single pass operation, in that it checks.

- Single pass operation of Absolute Loader:
  - Check H (Header Record) record to verify that correct program has been presented for loading.
  - Read each T (Text Record), record, and move object code into the indicated address in memory.
  - At E (End Record) record, jump to the specified address to begin execution of the loaded program.

P.T.O →

Algorithm for Absolute Loader.

Start

read Header record
Verify Program length and name
read first text record
While record type != E do
begin
if object code is in character form
convert it into internal
representation move object-
code to specified location in
memory
read next object program record
end
Jump to address specified in End record

Stop.

- Advantages:

  - simple and efficient to implement.

- Disadvantages:
  - The programmer must specify to the assembler the address in the core where the program is to be loaded.

Sq
Soln →

Dynamic loading	Static loading
1. It refers to mapping (or less often copying) an executable or library into a process's memory after it has started.	Dynamic linking refers to resolving symbols - associating their names with addresses or offsets - after compile time.
2. It is similar to plugins that is an exe can actually execute before the dynamic loading happens	The linker while creating the exe does minimal work. For the dynamic linker to work it actually has to load the libraries.

3. Dynamic loading is concerned with loading the subroutines of a program as ~~and~~ and when required during runtime, instead of loading all the subroutines at once before the program execution starts.

Dynamic linking is concerned with linking library routines at runtime instead of combining them with executable program code before the program execution starts i.e Static linking. To achieve this, a small code call 'stub' is inserted in the program code wherever library routine is required. The stub contains information about where

to find the routine if it is not already in the main memory.

4. Efficient use of memory

Efficient use of memory.

5. It does not require special support from operating system; it is the responsibility of the programmer to check whether the routine that is to be loaded does not exist in main memory

Dynamic linking requires special support from operating system, the routine loaded through dynamic linking can be shared across various processes.