Name - SANDIPAN SAHA Phn - 8017885131
Roll - 001810501101 Email - sandipans814@gmail.com

Part B.

CO1

1a) **Utility of System software**

System software is a type of computer program that is designed to run a computers hardware and application programs. It is the interface between hardware and user applications. It generate basic services which are machine or system dependent and should be abstracted from the user. System software consists of a variety of programs that support the operation of a computer while allowing users to work without needing to know the details of how the machine works internally. For example

- Text Editor - to create and edit the program
- Compiler & assembler - translate into machine language
- Loader & Linker - to load program into main memory for execution.
- Debugger - to help detect errors in the program

b) **Compiler** - The language processor that reads the complete source program written in high level language in one go as a whole and converts it to equivalent machine language program

Assembler - The language processor converts assembly language into machine code. The output generated by

assembler is called object code.

Interpreter: Converts single statement of source Program into machine code at a time. Terminate and displays error at first line of error.

Linker - Combine the object file generated by compiler or assembler and shared objects to generate an executable file.

Loader - Loads the executable image into memory for execution.

Microprocessor - It is a multipurpose, clock-driven, register-based digital integrated circuit that accepts binary as input, performs some system dependent operations using machine code.

Cross Assembler: It is an assembler that runs on a Computer with a type of processor but generates machine code for a different type of processor allowing development and execution on a different platform.

Hardware dependent machine.

0018105016 ~~8078851~~

1)

LABEL	Mnemonics	opcode	LOCCTR	LOCCTR Roughwork
MAIN	START			0
	BALR	10, 0	0	
	USING	*, 10		2
	L	5, = F'4'	2	
	SER	6, 6	6	
	L	4, 0(0,1)	8	
	L	6, 4 (0, 1)	12	
CYCLE	LE	2, 0 (0, 4)	16	
	LE	4, 0 (0,6)	20	
	MER	2, 4	24	
	AER	6, 2	26	
	A	6, F'4'	28	
	A	4, = F'4'	30	
	BCT	5, CYCLE	32	
	L	4, 8 (0,1)	36	
	STE	6, 0 (0, 4)	40.	
	END	MAIN		

i) Symbol Table

44.

SYMBOL	declaration	value	R/A	segment
MAIN	Yes	0	R	CS
CYCLE	Yes	16	R	CS.

ii) Literal Table.

Literal	LC value	Duplicate status	R/A
= F'4'	48	no	R

Sandeepan Sahoo

001810501 00
~~801788513~~

iii) Base Register table

Reg No	Content
10	2

iv) LV in a single table.

LOCATION	General machine Instruction	Generated hexadecimal
0	BALR. 10, 0	06A0
2	L 5, =F'4'	5850A02E
	=> L 5, 46(0,10)	
6	SER 6,6	3B66.
8	L 4, 0(0,1)	58401000
12	L 6, 4(0,1)	5860 1004
16	LE 2, 0(0,9)	7820 4000
20	EE 4, 0(0,6)	7846 6000
24	MER 2,4	3C24.
26	AER 6,2	3A62.
28	A 6, F'4'	5A64
30	A 4, =F'4' => A 4,46(0,10)	5A4A
32	BCT 5, 14(0,10)	4650A00E
36	L 4, 8 (0, 1)	58401008
40	STE 6,0 (0, 4)	70604000.

Sreedipan Saha

CO2

2A:- 00) COPY S1, S2.

03) COPY S3, S4

06) READ S5 S5 ←

08) WRITE S4 → S4

L2: 10) LOAD S2 A = S2

12) AND S4 A = A + S4

14) STORE S6 S6 = A

16) SUB S5 A = A - S5

18) BAPOS L1 If, A > 0, to L1

20) WRITE S6 → S6

22) COPY S4, S2 S2 = S4

24) COPY S6, S4 S4 = S6

28) BR L2 to L2

L1: 30) WRITE S5 → S5

32) STOP

33) S1 (Literal) "00"

34) S2 (Literal) "01"

Symbol Table

S.No.	Symbol Name	Address
01	S1	33
02	S2	35
03	S3	
04	S4	34
05	S5	36
06	S6	38
07	L1	37
08	L2	30
		10

0818105101001

Q4.

5) Given string : "unlimited file size and line length editing with little degradation in performance for larger files".

Initially original file contains 99 characters. This is read only.

Initially the 'Add file' (append only) is empty

The piece table is:

file	start	length
original	0	99

The sequence is same as original string.

a) Delete the word 'editing'

The add file is still empty.

The piece table is:

file	start	length
original	0	36
original	44	55

The sequence becomes : "unlimited file size and line length with little degradation in performance for larger files."

b) Insert the words "(or not)" in between little & degradation

The add file becomes: or not

Piece table:

File	start	length
original	0	36
original	44	12
Add	0	7
original	55	43.

The sequence becomes: "unlimited file size and line length with little or not degradation in performance for larger files"

Sandipan saha.

00181050[00]

5c) Delete the word 'performance'

The 'Add file' remains the same : or not

The piece table becomes

File	Start	Length
original	0	36
original	44	12
Add	0	7
original	55	15
original	82	17

The sequence becomes : "unlimited file size and line length with little or no degradation in for larger files.

5d) Insert the word accomplishment where performance was earlier

The 'Add file' becomes : or not accomplished.

The piece table becomes

File	Start	Length
original	0	36
original	44	12
Add	0	7
original	55	15
Add	7	13
original	82	17

The sequence becomes : "unlimited file size and line length with little or no degradation in accomplishment for larger file"

Sandipan Saha.

001810501001

5 e) Change the word "longer" to "bigger"

First we will delete "longer" from the sequence and then insert "bigger" in the exact same position.

The 'Add file' becomes: original not accomplished bigger

The piece table becomes

File	start	length
original	0	36
originall	44	12
Add	0	7
originl	55	15
Add	7	15
original	82	4
Add	22	7
original	94	5

The sequence becomes: "unlimited file size and line length with little or not degradation in accomplishment for bigger files".

Sandip on Sahe

0018105 01 [illegible]
[illegible]

CO2

3 a) In case of forward referencing in one-pass Assembler Assembler doesnot consider undefined symbol to as error.

In case of forward reference in one pass assembler for a symbol which has not been defined yet we :-

1) omit the address translation
2) insert the symbol into SYMTAB and mark it undefined
3) the address that refers to the undefined symbol is added to a list of forward references associated with the symbol table entry.
4) when the definition for a symbol is encountered, the proper address for the symbol is then inserted into any instruction previously generated according to forward reference list.

Hence this not treated as an error.

3 b) Compare one pass, 2 pass, Multi pass Assembler.

One Pass Assembler	Two Pass Assembler	Multipass Assembler.
• Passes through the source code exactly once	• Requires 2 pass First for label definition and introducing them to symbol table Second translates instruction into assembly & execution	• Passes through the source code more than 2 times.
• Generally i) deals with syntax ii) constructs symbol table iii) creates label list	• Generally. • defines label in First Pass • translates instruction into assembly & execution in Second Pass.	• Along Pass 1 & 2 • Generates actual opcode • Compute actual address of every label. • Translates operand name into appropriate register / memory address

[illegible]

• cannot resolve forward data reference	• Can reference forward data reference	• can reference forward data reference.
• No object program is written, No loader required.	• Loader is required	• Loader is required
• Tends to be faster than 2 pass and multipass assembler.	• Tends to be slower than one pass but faster than multipass	• Slowest among all 3.

3(a) Load and Go assemblers are typically a type of one pass assembler that generates their object code in memory for immediate execution. No object program ~~is needed~~ written out, thats why no loader is needed. It is useful in a system oriented toward program development and testing.

How it handle literals

1) Omits the literal address, if the literal has not been defined yet.

2) Enters the undefined literal into symbol table & marks it undefined.

3) Adds the address of this address to a list of forward references associated with literal pool entry.

4) Scans the references list and inserts the address when the definition for the literal is encountered.

5) Reports error ~~notes~~ if there are still undefined literals.

Sandipan Saha

0018105010(0)
~~8017805531~~

Example

```
1000  COPY   START  1000
100D  ABC    ALLD   C'ABC'   ← Reference will be
                               created at literal pool
      LAST   STL    RATBCD
        :
        :
1050  THREE  WORD   3
      EOF    BYTE   C'EOF'   ← Referenced letter will
                               follow standard
                               forward reference
                               resolving algorithm.
```

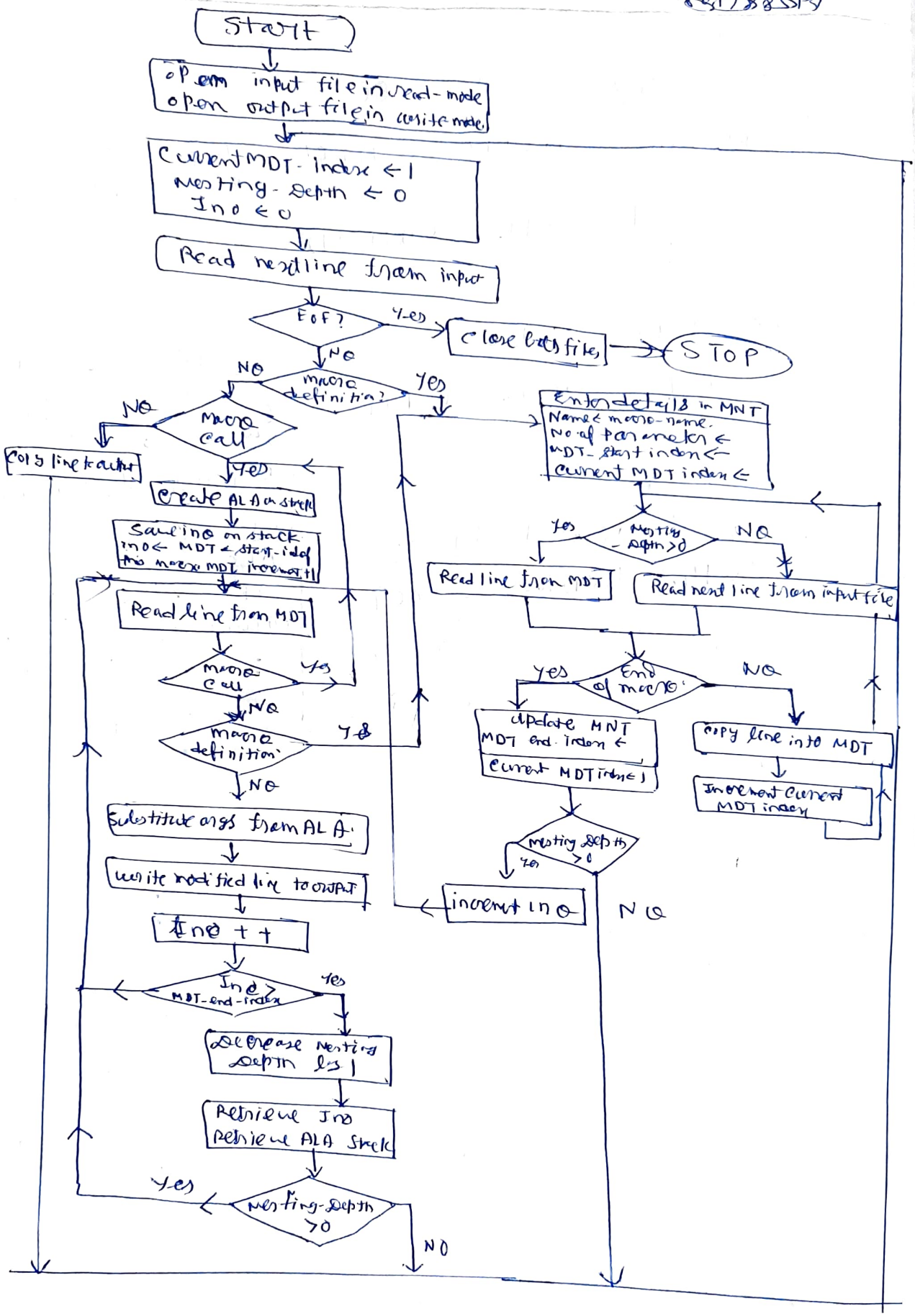
Generated obj code

100948 BCD1005 - - 1009 -

Symbol	VALUE
ABC	1000
EOF	• → 1050 | ∅
RATBCD	1005
:	:

Sandipan Sarkar

6)



Flowchart of a macro processor with nested definition & calls.

7.b) (i) CO6 0018105010

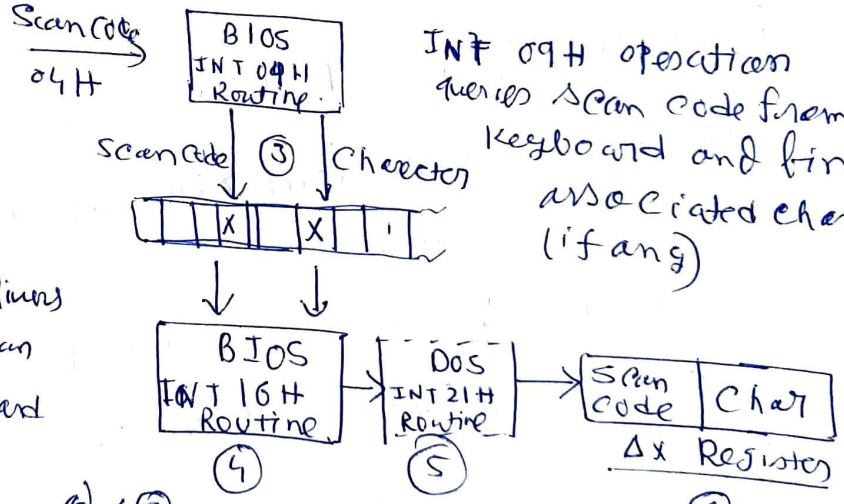
Bootstrap loader is an absolute loader program which is permanently resident in a read only memory (Rom)

The Program is executed directly in the RAM.

The Program is copied from ROM to main memory & executed there.

Unlike other traditional loader, the bootstrap loader loads the OS itself upon startup from an absolute address and prepares it for execution. It is required at the very beginning of all object Programs that are to be loaded into an empty & idle system.

(ii) Keyboard Generate 09H (1)



INT 09H operation queries scan code from keyboard and finds associated characters (if any)

INT 09H also delivers character and scan code to the keyboard buffer

④ & ⑤ → Program requests INT 16H either directly or via INT 21H.

INT 16H queries buffer & delivers character to AL & scan code to AH

Sandipan Saha

Q3

0018105010001

4a) **Good Debugger :**

A debugger is a computer program used to test and debug other programs. The main use of debugger is to run target program under controlled conditions. Typical debugging facilities include the ability to run or halt the target program at specific points, display the memory contents, CPU registers or storage devices and modify their content. A debugger encompassing all this features is known as Good Debugger.

Essential steps for good debugger design

Unit test : A breakpoint debugger must have a set of unit test functions.

Execution sequencing :- It is possible to control the flow of program execution.

Unconditional breakpoint :- The programmer can set and define the breakpoint for suspension of execution of the program at a particular statement.

Trace It is possible to trace the flow of execution logic.

on 18/05/2021

Single step execution Execute one instruction at a time & check the present state.

Reverse execution The process of execution in a debugger helps to roll back to previous state

Conditional breakpoint

Watch points

Check point.

(7.8)

Inspite of the advantages noted, there are still relatively few general purpose macro processors.

- In a typical Programming environment there are several situations in which normal macro parameter substitution should occur
- Another difference between programming languages is related to their facilities for grouping together terms.
- A more general problem involves the tokens of the programming languages.
- Another potential problem with general purpose macro processes involves the syntax used for macro definition and macro invocation statement with most special-purpose macro processors macro invocations are very similar in form to statements in the source programming language.

Sandeep