FINAL SCHEME

1100CST305122201

Total Pages: 11

Scheme of Valuation/Answer Key

(Scheme of evaluation (marks in brackets) and answers of problems/key)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

FIFTH SEMESTER B.TECH DEGREE(R/S) EXAMINATIONS, DECEMBER **2022**

(2019 Scheme)

Course Code: CST 305

Course Name:SYSTEM SOFTWARE

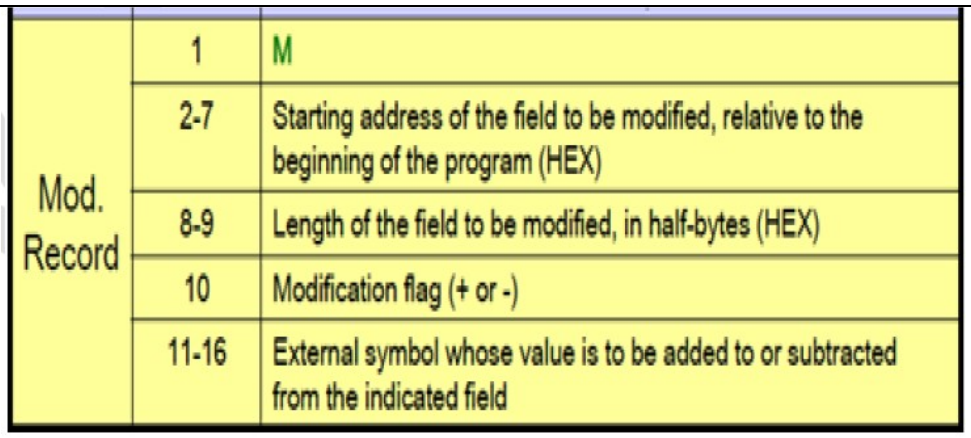
Max. Marks: 100 | Duration: 3 Hours

PART A

(Answer all questions; each question carries 3 marks)

			Marks
1		Compiler-1.5 marks Interpreter-1.5 marks	3
2		<table><tr><th>Mnemonic</th><th>Number</th><th>Special use</th></tr><tr><td>A</td><td>0</td><td>Accumulator; used for arithmetic operations</td></tr><tr><td>X</td><td>1</td><td>Index register; used for addressing</td></tr><tr><td>L</td><td>2</td><td>Linkage register; the Jump to Subroutine (JSUB) instruction stores the return address in this register</td></tr><tr><td>PC</td><td>8</td><td>Program counter; contains the address of the next instruction to be fetched for execution</td></tr><tr><td>SW</td><td>9</td><td>Status word; contains a variety of information, including a Condition Code (CC)</td></tr></table> Any three registers with use- 1mark each	3
3		• **Basic Functions of an Assembler** 1. Convert mnemonic operation codes to their machine language equivalents ▪ Eg: Translate STL to 14 2. Convert symbolic operands to their equivalent machine addresses ▪ Eg: Translate the operand RETADR to 1033(address of RETADR) 3. Convert the data constants to internal machine representations ▪ Eg: Translate EOF to 454F46 4. Build the machine instructions in the proper format 5. Write the object program and the assembly listing Any 3 functions-1 mark each	3

4		**SIC** PGM1 START 1000 LDA ALPHA STA GAMMA LDA BETA STA ALPHA LDA GAMMA STA BETA ALPHA WORD 10 BETA WORD 20 GAMMA RESW 1 END 1000	3
5		**1. EQU Statement** EQU is an assembler directive that allows the programmer to define symbols and specify their values Syntax: **Symbol EQU value** **2. ORG Statement** ORG is an Assembler directive that allows the assembler to reset the PC to values Syntax: **ORG value** 1.5 marks each	3
6		Program blocks allow the generated machine instructions and data to appear in a different order while they are loading in memory-1 mark Assembler directive: **USE**-1 mark Syntax: USE [blockname] -1 mark	3
7		• Allocation • Linking • Relocation • Loading ------0.75 marks each	3
8		A Modification record is used to describe each part of the object code that must be changed when the program is relocated.-1.5 marks	3

		 -1.5 marks	
9		Assembler Directives- MACRO and MEND—1.5 marks Example---1.5 marks	3
10		Device specific—1.5 marks Operating System specific—1.5 marks	3
PART B			
(Answer one complete question from each module)			
Module -1			
11	a)	7 features-memory,registers,instruction format,data format,addressing mode,Instruction set, I/O	8
	b)	Definition-2 marks Any 4 assembler directives—1 mark each START RESW RESB WR BYTE END	6
12	a)	7 features-memory,registers,instruction format,data format,addressing mode,Instruction set, I/O	8
	b)	Compare any 6 features---1 mark each	6
Module -2			

<table>
<tr><td>13</td><td>a)</td><td>

```
PGM2      START    1000
          LDA      BETA
          LDS      GAMMA
          DIVR      S, A
          STA      ALPHA
          MULR     S, A
          LDS      BETA
          SUBR      A, S
          STS      DELTA

BETA      WORD     20
GAMMA     WORD     7
ALPHA     RESW     1
DELTA     RESW     1
          END      1000
```

</td><td>5</td></tr>
<tr><td></td><td>b)</td><td>

	Column	Contents
Header Record	1	H
	2-7	Program name
	8-13	Starting address of object program (HEX)
	14-19	Length of object program in bytes (HEX)
Text Record	1	T
	2-7	Starting address for object code in this record (HEX)
	8-9	Length of object code in this record in bytes (HEX)
	10-69	Object code (HEX, 2 columns per byte of object code)
End Record	1	E
	2-7	Address of first executable instruction in object program (HEX)

1 mark for each record

</td><td>3</td></tr>
<tr><td></td><td>c)</td><td>Location Counter(LOCCTR) ---2 marks
Operation Code Table(OPTAB) ---2 marks
Smbol Table(SYMTAB) ---2 marks</td><td>6</td></tr>
</table>

14	a)	PGM2	START	1000	6
			LDS	#3	
			LDT	#300	
			LDX	#0	
		LOOP	LDA	#0	
			STA	ALPHA, X	
			ADDR	S, X	
			COMPR	X, T	
			JLT	LOOP	
		ALPHA	RESW	100	
			END	1000	

b) 8

```
Algorithm Pass1
{       Read the input line
        If OPCODE='START'
        {       starting address = #OPERAND
                LOCCTR = starting address
                Write line to intermediate file
                Read next input line
        }
        Else
                LOCCTR = 0
        While OPCODE != 'END' do
        {       If this is not a comment line
                {       Write line to intermediate file along with LOCCTR
                        If there is a symbol in the label field
                        {       Search SYMTAB for LABEL
                                If found
                                        Set error flag(Duplicate Symbol)
                                Else
                                        Insert (LABEL, LOCCTR) into SYMTAB
                        }
                        Search OPTAB for OPCODE
                        If found
                                LOCCTR = LOCCTR + 3
                        Else if OPCODE = 'WORD'
                                LOCCTR = LOCCTR + 3
                        Else if OPCODE = 'RESW'
                                LOCCTR = LOCCTR + 3 x #[OPERAND]
                        Else if OPCODE = 'RESB'
                                LOCCTR = LOCCTR + #[OPERAND]
                        Else if OPCODE = 'BYTE'
                                LOCCTR = LOCCTR + length of constant in bytes
                        Else
                                Set error flags
                }
                Read next input line
        }
        Write last line to intermediate file
        Save (LOCCTR – starting address) as program length.
}
```

Module -3

15 a) List the features of MASM 7

b) 7

A		**800**	**Ø**
B		**1000**	**Ø**
C		**2034**	**Ø**
D		**1034**	**Ø**
E		**1033**	**Ø**

This is the final symtab after executing all statements--- 3 marks

4 marks can be given to the procedure/steps

16	a)	▪ **Load-and-go Single Pass Assembler Algorithm**
		Begin
		Read 1st input line
		If *OPCODE = 'START'* then
		{
		Starting address = #OPERAND
		LOCCTR = Starting address
		Read the next input line
		}
		Else
		LOCCTR = 0
		While *OPCODE != 'END'* do
		{
		If *there is not a comment line* then
		{
		If *there is a symbol in the LABEL field* then
		{
		Search SYMTAB for LABEL
		If *found* then
		{
		If *symbol value as null* then
		{
		Symbol value = LOCCTR
		Search the linked list with corresponding operand
		Generate operand addresses as corresponding to symbol value
		Delete the linked list
		}
		}
		Else
		Insert (LABEL, LOCCTR) into SYMTAB
		}
		Search OPTAB for OPCODE
		If *found* then
		{
		Search SYMTAB for OPERAND address
		If *found* then
		{
		If *symbol value != null* then
		OPERAND address = symbol value
		Else
		Insert a node at the end of the linked list with address as LOCCTR+1
		}
		Else
		{ Insert (symbol name, null) into SYMTAB
		Create a linked list with address as LOCCTR+1
		}
		Generate object code and load it in memory location LOCCTR
		
		LOCCTR = LOCCTR +3
		}
		Else if *OPCODE = 'WORD'* then
		{
		Object code = #OPERAND
		load this object code in memory location LOCCTR
		LOCCTR = LOCCTR +3
		}
		Else if *OPCODE = 'RESW'* then
		LOCCTR = LOCCTR +3x#OPERAND
		Else if *OPCODE = 'RESB'* then
		LOCCTR = LOCCTR + #OPERAND
		Else if *OPCODE = 'BYTE'* then
		{
		Convert constant to object code and load it in memory location LOCCTR
		LOCCTR = LOCCTR +length of the constant
		}
		Else
		Set error flag
		}
		Read the next input line
		}
		If there are still SYMTAB entries indicated undefined symbols
		Reports the error
		Else
		Jump to the location specified in END statement.
		End
			7

	b)	Definition- 2 marks	7

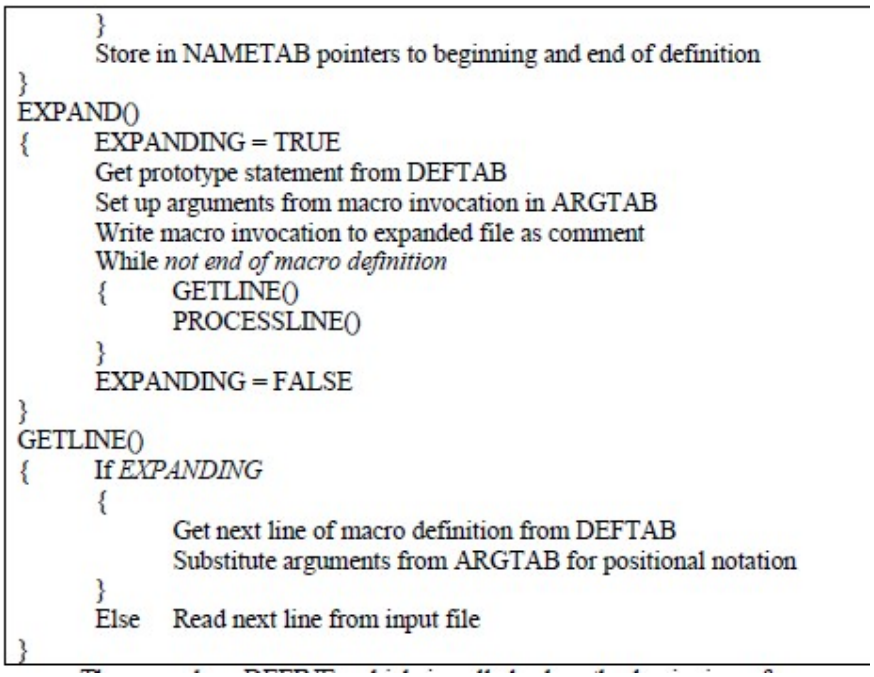
		Example- 5 marks	
		Module -4	
17	a)	Automatic Library Search-4 marks Loader options- 4 marks	8

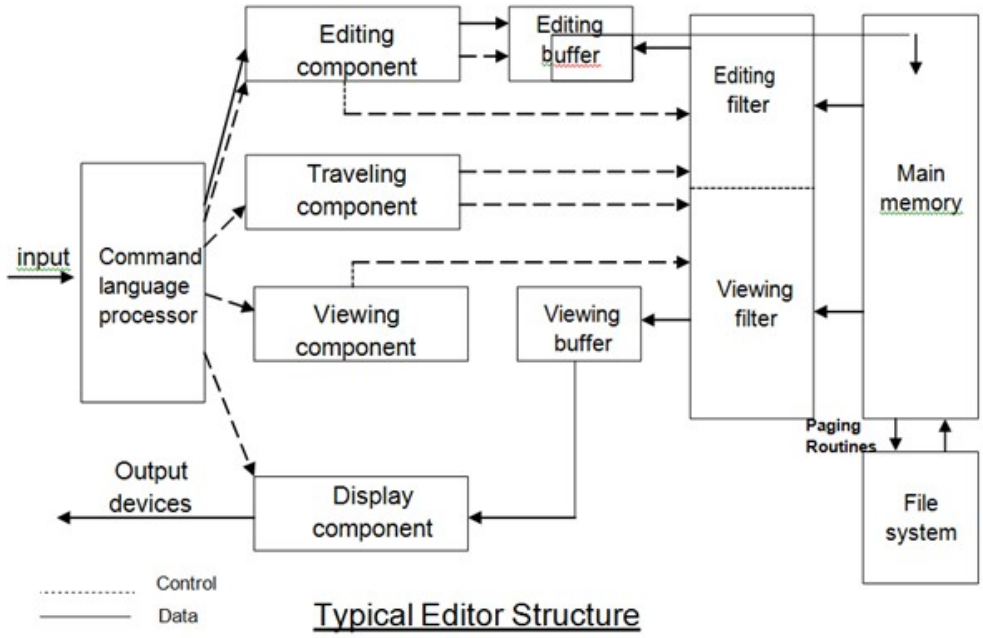
	b)	Data structures---External Symbol Table(ESTAB)-2 marks
		
		Pass 2 Linking Loader Algorithm
		CSADDR = PROGADDR \\ for the 1st control section
		EXECADDR = PROGADDR
		While *not end of input* do
		{
		Read the next input record \\ header record for the control section
		CSLTH = control section length
		While *record_type != 'E'* do
		{
		Read the next input record
		If *record_type = 'T'* then
		{
		If *the object code is in character form* then
		Convert it into internal representation
		Move the object code from record to location (CSADDR + specified address)
		}
		Else if *record_type = 'M'* then
		{
		Search ESTAB for modification symbol name
		If *found* then
		Add or subtract the symbol value at location (CSADDR + specified address)
		Else
		Set error flag
		}
		}
		If *an address is specified in End record* then
		EXECADDR = CSADDR + specified address
		CSADDR = CSADDR + CSLTH
		}
		Jump to location given by EXECADDR to start execution of the program
		
		---4 marks
			6

18	a)	**begin**
		read Header record
		verify program name and length
		read first Text record
		while record type ≠ 'E' **do**
		begin
		{if object code is in character form, convert into internal representation}
		move object code to specified location in memory
		read next object program record
		end
		jump to address specified in End record
		end
			6

	b)	4 loader options-2 marks each Bootstrap loader Linking loader Linkage editor Dynamic linking	8
		Module -5	

19	a)	• **Algorithm for One Pass Macro Preprocessor**
		ONE_PASS_MACRO()
		{ EXPANDING= FALSE
		while *OPCODE != 'END'*
		{
		GETLINE()
		PROCESSLINE()
		}
		}
		PROCESSLINE()
		{ Search NAMETAB for OPCODE
		If *found* then EXPAND()
		Else if *OPCODE='MACRO'* then DEFINE()
		Else Write source line to expanded file
		}
		DEFINE()
		{ Enter macro name into NAMTAB
		Enter macro prototype into DEFTAB
		LEVEL = 1
		While *LEVEL > 0*
		{ GETLINE()
		If *this is not a comment line*
		{ Substitute positional notation for parameters
		Enter line into DEFTAB
		If *OPCODE='MACRO'* LEVEL = LEVEL+1
		Else If *OPCODE='MEND'* LEVEL = LEVEL-1
		}
			6

			
	b)	Simple macro---2 marks Parametrized macro---2 marks Nested macro---2 marks Recursive macro---2 marks	8
20	a)	Text editor-structure&components,types---4 marks Diagram 3 marks	7

		 Typical Editor Structure	
	b)	Definition-1mark Functions—2 marks, types—2 marks, anatomy—2 marks	7
		*********	