

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Internal Assessment Test 1 Answer Key– Sep. 2019

Sub:	System Software					Sub Code:	18MCA34	Branch:	MCA
Date:	07/09/2019	Duration:	90 min's	Max Marks:	50	Sem	III		

Q1 a) Define system software? List and explain any four assembler directives with examples.

System Software consists of a variety of programs that support the operation of a computer. It makes possible for the user to focus on an application or other problem to be solved, without needing to know the details of how the machine works internally.

They are usually related to the architecture of the machine on which they are to run.

Example: Assembler, Compiler, text editor, loader and linkers etc.

Assembler Directives

In addition to the mnemonic machine instructions assembler uses following assembler directives. These statements are not translated into machine instructions. Instead they provide instructions to assembler itself.

1) START

START specify the name and starting address of the program.

Example: START 1000

2) END

Indicate the end of the source program and (optionally) specify the first executable instruction in the program.

Example: END FIRST

3) BYTE

Generate character or hexadecimal constant, occupying as many bytes as needed to represent the constant.

Example: BYTE X'F1'

4) WORD

Generate one-word integer constant

Example: THREE WORD 3

5) RESB

Reserve the indicate number of bytes for a data area.

Example: BUFFER RESB 4096

6) RESW

Reserve the indicate number of words for a data area.

Example: LENGTH RESW 1

Q1 b) Write general description of pass 1 and pass 2 for two pass assembler

Pass 1 (define symbols)

- Assign addresses to all statements in the program
- Save the addresses assigned to all labels for use in Pass 2
- Perform some processing of assembler directives, (including those for address assignment, such as BYTE and RESW

Pass 2 (assemble instructions and generate object program)

- Assemble instructions (translate opcodes and look up addresses)
- Generate data values defined by BYTE, WORD etc.
- Perform processing of assembler directives not done during Pass 1
- Write the object program and the assembly listing

Q2 a) Differentiate between application software and system software.

System Software	Application Software
Intended to support the operation and use of the computer	An application program is primarily concerned with the solution of some problem, using the computer as tool
Focus is on the Computer system and not on the application	The focus is on the application not on the computing system.
It depends on the structure of the machine on which it is executed.	It does not depend on the structure of the machine it works
Ex. Operating system, Loader, Linkers, assembler, compiler, text editors etc.	Ex. Banking system, Inventory system.

Q2 b) Write Simple object program format. (Header, Text, End Record)

The simple object program format contains three types of records: Header record, Text record and end record. The header record contains the starting address and length. Text record contains the translated instructions and data of the program, together with an indication of the addresses where these are to be loaded.

The end record marks the end of the object program and specifies the address where the execution is to begin.

The format of each record is as given below.

Header record:

Col 1 H

Col. 2-7 Program name

Col 8-13 Starting address of object program (hexadecimal)

Col 14-19 Length of object program in bytes (hexadecimal)

Text record:

Col. 1 T

Col 2-7. Starting address for object code in this record (hexadecimal)

Col 8-9 Length of object code in this record in bytes (hexadecimal)

Col 10-69 Object code, represented in hexadecimal (2 columns per byte of object code)

End record:

Col. 1 E

Col 2-7 Address of first executable instruction in object program (hexadecimal)

Q3) Describe SIC (standard model) Machine Architecture.

1) Memory

- Memory consists of 8-bit bytes.
- 3 consecutive bytes form a word (24 bits).
- All the address in SIC are byte addresses.
- Words are addressed by the location of their lowest numbered byte.
- There are total of 32,768 (2¹⁵) bytes in the computer memory.

2) Registers

There are five registers, each 24 bits in length.

Mnemonic

	Number	Use
A	0	Accumulator; used for arithmetic operations
X	1	Index register; used for addressing
L	2	Linkage register, the jump to subroutine instruction stores the return address in this register.
PC	8	Program counter, contains the address of the next instruction to be fetched for execution.
SW	9	Status word, contains a variety of information, including a Condition Code.

3) Data Formats

- Integers are stored as 24 bit binary numbers; 2's complement representation is used for negative values.
- Characters are stored using their 8-bit ASCII codes.

- There is no floating point hardware on the standard version of SIC.

4) Instruction Formats

All machine instructions on the standard version of SIC have the following 24-bit format 8 1 15

opcode	x	address
8	1	15

The flag bit x is used to indicate indexed addressing mode.

5) Addressing Modes

There are two addressing modes, indicated by the setting of the x bit in the instruction.

Mode	Indication	Target address calculation
Direct	x = 0	TA = address
Indexed	x = 1	TA = address + (x)

Parentheses are used to indicate the contents of a register or a memory location. For example, (X) represents the contents of register X.

6) Instruction Set

SIC provides a basic set of instructions that are sufficient for most simple task.

- Data transfer instruction: This include instructions that load and store registers. Eg. LDA, LDX, STA, STX.
- Arithmetic operation instruction: Basic arithmetic operations that involves register A Eg. ADD, SUB, MUL, DIV, COMP.
- Conditional Branching: Conditional jump instructions test the settings of conditional code and jump accordingly. Eg. JLT, JGT, JEQ.
- Subroutine call Instructions: Perform subroutine linkage. Eg. JSUB, RSUB. Return address is stored in linkage(L) register.

7) Input and Output

Input and Output are performed by transferring 1 byte at a time to or from the rightmost 8 bits of register A (accumulator).

Each device is assigned a unique 8bit code.

There are 3 I/O instructions.

The Test Device (TD) instruction tests whether the addressed device is ready to send or receive a byte of data. Read Data (RD), Write Data (WD) are used for reading or writing the data.

Q4) Describe SIC/XE Machine Architecture.

1) Memory

- Memory consists of 8-bit bytes.
- 3 consecutive bytes form a word (24 bits).
- All the address in SIC/XE are byte addresses.
- Words are addressed by the location of their lowest numbered byte.
- Maximum memory available on a SIC/XE system is 1 megabyte (220 bytes).

- This increase leads to a change in instruction formats and addressing modes

2) Registers

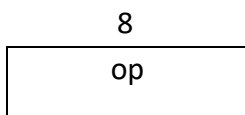
Five registers of SIC machine remains same in SIC/XE. The additional registers provided by SIC/XE are as follows.

Mnemonic	Number	Use
B	3	Base register; used for addressing.
S	4	General working register – no special use.
T	5	General working register – no special use.
F	6	Floating-point accumulator (48 bits).

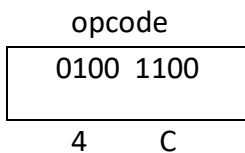
3) Instruction Formats

- SIC/XE has larger memory hence instruction format of standard SIC version is no longer suitable.
- SIC/XE provide two possible options; using relative addressing (Format 3) and extend the address field to 20 bit (Format 4).
- In addition SIC/XE provides some instructions that do not reference memory at all. (Format 1 and Format 2) .
- The new set of instruction format is as follows. Flag bit e is used to distinguish between format 3 and format 4. (e=0 means format 3, e=1 means format 4)

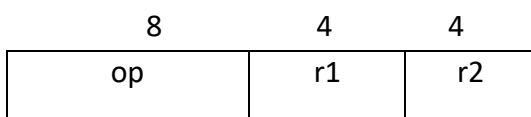
1. Format 1 (1 byte)



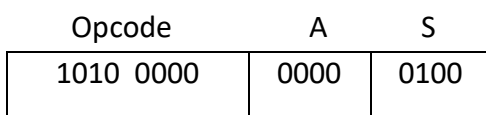
Example RSUB (return to subroutine)



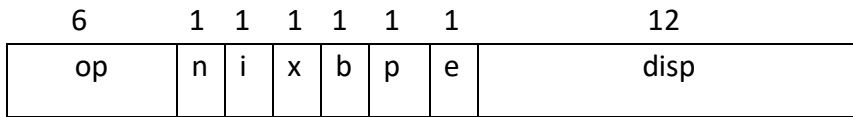
2. Format 2 (2 bytes)



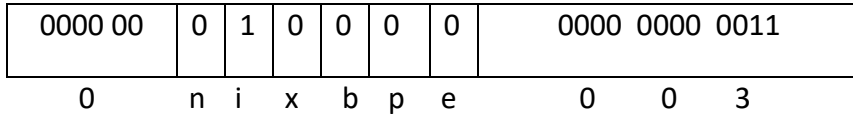
Example COMPR A, S (Compare the contents of register A & S)



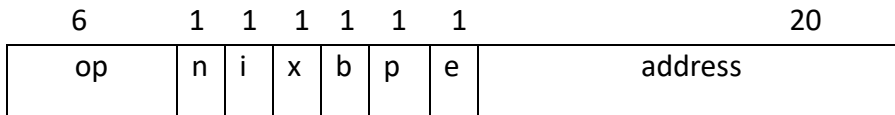
3. Format 3 (3 bytes)



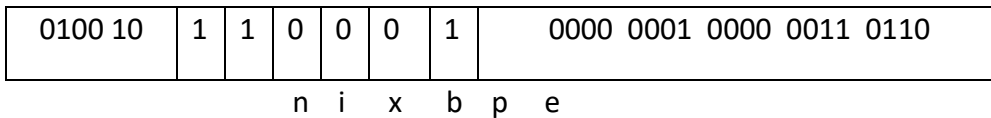
Example LDA #3(Load 3 to Accumlator A)



4. Format 4 (4 bytes)



Example JSUB RDREC(Jump to the address, 1036)



4) Addressing Modes

Two new relative addressing modes are available for use with instructions assembled using Format 3

Mode	Indication	Target address calculation
Base Relative	b=1, p=0	TA = (B) + disp (0 ≤ disp ≤ 4095)
Program-counter relative	b=0, p=1	TA = (PC)+disp (-2048 ≤ disp ≤ 2047)

b represents for base relative addressing where as p represents program counter relative addressing. If both the bits b and p are 0 then target address is taken form the address field of the instruction (i.e displacement)

SIC/XE also support addressing modes that are assembled using Format 4.

Mode	Indication	Target address calculation
Direct	b=0, p=0, x=0	TA = disp
Indexed	x=1	TA = (x)+disp
Immediate	i=1, n=0	TA = operand value
Indirect	i=0, n=1	TA = address of operand value
simple	i=1, n=1 i=0, n=0	TA = location of the operand value

6) Instruction Set

- SIC/XE provides all of the instructions that are available on the standard version.

- In addition we have, Instructions to load and store the new registers LDB, STB, etc,
- Floating-point arithmetic operations, ADDF, SUBF, MULF, DIVF,
- Register move instruction : RMO,
- Register-to-register arithmetic operations, ADDR, SUBR, MULR, DIVR and,
- Supervisor call instruction : SVC

7) Input and Output

- There are I/O channels that can be used to perform input and output while the CPU is executing other instructions.
- Allows overlap of computing and I/O, resulting in more efficient system operation.
- The instructions SIO, TIO, and HIO are used to start, test and halt the operation of I/O channels.

Q5) Write pass 1 Algorithm for pass two assembler

Assembler Pass 1:

```

begin
  read first input line
  if OPCODE='START' then
    begin
      save #[OPERAND] as starting address
      initialize LOCCTR to starting address
      write line to intermediate file
      read next input line
      end {if START}
  else
    initialize LOCCTR to 0
  while OPCODE != 'END' do
    begin
      if this is not a comment line then
        begin
          if there is a symbol in the LABEL field then
            begin
              search SYMTAB for LABEL
              if found then
                set error flag (duplicate symbol)
              else
                insert (LABEL,LOCCTR) into SYMTAB
            end {if symbol}
          search OPTAB for OPCODE
          if found then
            add 3 {instruction length} to LOCCTR
          else if OPCODE='WORD' then
            add 3 to LOCCTR
          else if OPCODE = 'RESW' then
            add 3 * #[OPERAND] to LOCCTR
          else if OPCODE = 'RESB' then
            add #[OPERAND] to LOCCTR
          else if OPCODE = 'BYTE' then
            begin
              find length of constant in bytes
              add length to LOCCTR
            end {if BYTE}
          else
            set error flag (invalid operation code)
          end {if not a comment}
          write line to intermediate file
          read next input line
        end {while not END}
      write last line to intermediate file
      save (LOCCTR – starting address) as program length
    end {Pass 1}

```

Q6) Write pass 2 Algorithm for pass two assembler

Assembler Pass2:

```
begin
read first input line (from intermediate file)
if OPCODE ='START' then
  begin
    write listing line
    read next input line
  end (if START)
write Header record to object program
initialize first Text record
while OPCODE != 'END' do
  begin
    if this is not a comment line then
      begin
        search OPTAB for OPCODE
        if found then
          begin
            if there is a symbol in OPERAND field then
              begin
                search SYMTAB for OPERAND
                if found then
                  store symbol value as operand address
                else
                  begin
                    store 0 as operand address
                    set error flag (undefined symbol)
                  end
                end (if symbol)
              else
                store 0 as operand address
                assemble the object code instruction
            end (if opcode found)
            else if OPCODE ='BYTE' or 'WORD' then
              convert constant to object code
            if object code will not fit into the current Text record then
              begin
                write Text record to object program
                initialize new Text record
              end
            add object code to Text record
            end (if not comment)
          write listing line
          read next input line
        end(while not END)
        write last Text record to object program
        write End record to object program
        write last listing line
      end{Pass 2}
```

Q7 a) Give the target address generated for following machine instruction.

if (B)=006000 (PC)=003000 (x)=000090 i)75101000 ii)032026

i)75101000

000110	1	0	0	0	1	0	0000 0000 0000
--------	---	---	---	---	---	---	----------------

TA= operand value
TA = 01000

ii) 032026

000000	1	1	0	0	1	0	0000 0010 0110
--------	---	---	---	---	---	---	----------------

TA= disp+(PC)
TA= 026+003000
TA=3026

Q7 b) Write a program for SIC/XE machine to copy a string "SYSTEM SOFTWARE" from LOC1 to LOC2

```

                LDT      #15
                LDX      #0
MOVECH         LDCH     LOC1,X
                STCH     LOC2,X
                TIXR     T
                JLT      MOVECH
LOC1           BYTE     C'SYSTEM SOFTWARE'
LOC2           RESB     15

```

Q8 a) Write a program for SIC machine to perform BETA = BETA / 5 + 4

```

                LDA      BETA
                DIV      FIVE
                ADD      FOUR
                STA      BETA
FIVE           WORD     5
FOUR          WORD     4

```

Q8 b) Give the target address generated for following machine instruction.

if (B)=006000 (PC)=003000 (x)=000090 i)03C300 ii)0310C303 iii)010030

i) 03C300

000000	1	1	1	1	0	0	0011 0000 0000
--------	---	---	---	---	---	---	----------------

TA=disp + (x) + (b)
TA= 6390

ii) 0310C303

000000	1	1	0	0	0	1	0000 1100 0011 0000 0011
--------	---	---	---	---	---	---	--------------------------

TA= C303

iii) 010030

000000	0	1	0	0	0	0	0000 0011 0000
--------	---	---	---	---	---	---	----------------

TA = 30

Q9 a) List and describe data structures used by two-pass assembler

1) OPTAB:

- ☒ It is used to lookup mnemonic operation codes and translates them to their machine language equivalents.
- ☒ In more complex assemblers the table also contains information about instruction format and length
- ☒ In pass 1 the OPTAB is used to look up and validate the operation code in the source program.
- ☒ In pass 2, it is used to translate the operation codes to machine language.
- ☒ In simple SIC machine this process can be performed in either in pass 1 or in pass 2.
- ☒ But for machine like SIC/XE that has instructions of different lengths, we must search OPTAB in the first pass to find the instruction length for incrementing LOCCTR.
- ☒ In pass 2 we take the information from OPTAB to tell us which instruction format to use in assembling the instruction, and any peculiarities of the object code instruction.
- ☒ OPTAB is usually organized as a hash table, with mnemonic operation code as the key.
- ☒ The hash table organization is particularly appropriate, since it provides fast retrieval with a minimum of searching.
- ☒ Most of the cases the OPTAB is a static table- that is, entries are not normally added to or deleted from it. In such cases it is possible to design a special hashing function or other data structure to give optimum performance for the particular set of keys being stored.

2) SYMTAB:

- ☒ This table includes the name and value for each label in the source program, together with flags to indicate the error conditions (e.g., if a symbol is defined in two different places).
- ☒ During Pass 1: labels are entered into the symbol table along with their assigned address value as they are encountered. All the symbols address value should get resolved at the pass 1.

☒ During Pass 2: Symbols used as operands are looked up the symbol table to obtain the address value to be inserted in the assembled instructions.

☒ SYMTAB is usually organized as a hash table for efficiency of insertion and retrieval. Since entries are rarely deleted, efficiency of deletion is the important criteria for optimization.

3) LOCCTR:

☒ Apart from the SYMTAB and OPTAB, this is another important variable which helps in the assignment of the addresses.

☒ LOCCTR is initialized to the beginning address mentioned in the START statement of the program.

☒ After each statement is processed, the length of the assembled instruction is added to the LOCCTR to make it point to the next instruction.

☒ Whenever a label is encountered in an instruction the LOCCTR value gives the address to be associated with that label.

Q9 b) Generate object code for the given SIC program
 OPCODES: LDA - 00 ADD - 18 STA - 0C

		START	0	
0000	FIRST	LDA	FIVE	000015
0003		STA	ALPHA	0C001B
0006		LDA	TWO	000018
0009		STA	BETA	0C001E
000C		LDA	ALPHA	00001B
000F		ADD	BETA	18001E
0012		STA	RESULT	0C0021
0015	FIVE	WORD	5	
0018	TWO	WORD	2	
001B	ALPHA	RESW	1	
001E	BETA	RESW	1	
0021	RESULT	RESW	1	

Q 10) Explain following with example

i) LDA ii) TIX

i) LDA

LDA is a mnemonic that stands for Load Accumulator with the contents from memory.
 Eg. LDA ALPHA

ii) TIX

Increments the index register and then compare the index register with the given operand

Eg TIX ELEVEN

Q10 b) Generate object code for the given SIC/XE program

OPCODES: LDA-00 STX-10 LDX-04 LDS-6C ADDR-90 STA-0C

	QUIZ	START	1000	
1000	FIRST	LDA	#3	010003
1003		STX	THREE	102011
1006		LDX	#0	070000
1009		+LDS	THREE	6F300A
100D		ADDR	A,X	9001
1010		+STA	RESULT,X	0FB000
1014	RESULT	RESW	1	
1017	THREE	RESW	1	
101A		END	FIRST	