14-MAY-2021

PAGE NO. 06

ASSIGNMENT - 1

QUESTIONS :-

1. a. Determine the Pass 1 and Pass 2 algorithm of 2 pass assembler. (CK3)

1. b. Obtain SIC and SIC/XE Machine Architecture. (CK3)

1. c. Identify the working of Absolute Loader and Bootstrap Loader, also write the algorithms. (K3)

1. d. Pedagogy - Module - 1 Quiz (CK3)

2. a. Make use of statement position = initial + rate * 60 to show translations at different phases of compiles. (K3)

2. b. Determine the role of sentinels in input buffering along with the look ahead code. (K3)

ANSWERS :-

1. a. ALGORITHM for 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 line
End

```
else
    Initialize LOCCTR to 0
    ~~write~~
while OPCODE != 'END' do
Begin
    if this is not a comment line then
    Begin
        if there is a symbol ~~to~~ in LABEL field then
        Begin
            Search SYMTAB for LABEL
            if found then
                Set Error flag
            else
                Search OPTAB for OPCODE
            if found then
                Add 3 to LOCCTR
            else if OPCODE = 'WORD' then
                Add 3 to LOCCTR
            else if OPCODE = 'RESW' then
                Add 3 * #[Operand] to LOCCTR
            else if OPCODE = 'BYTE' then
            Begin
                find length of constant inputs
                Add length to LOCCTR
            End
            else if OPCODE = 'RESB' then
                Add #[Operand] to LOCCTR
            else
                Set Error flag
End
```

```
    Writ line to intermediate file
    Read next input line
  End
  Writ last line to intermediate file
  Save {LOCCTR - Starting address} as program length
End
```

ALGORITHM FOR PASS -2

```
Begin
  Read 1st input line
  if OPCODE = 'START' then
  Begin
    Write listing line
    Read next input line
  End
  Writ header record to object program
  Initialize 1st text record
  While OPCODE != 'END' do
  Begin
    if there is no comment line then
    Begin
      Search OPTAB for OPCODE
      if found then
      Begin
        if there is symbol in OPERAND field then
        Begin
          Search SYMTAB for OPERAND
          it found then
          Store symbol value as operand address
```

```
else
Begin
    Store 0 as operand address
    Set error flag
End
End
else
    store 0 as operand address
    Assemble object code instruction
End
else if OPCODE = 'BYTE' or 'WORD' then
    convert constant to object code
    Initialize new text record
    End
    Add object code to text record
    End
    Writ listing line
    Read next input line
    End
    writ last Text record to object program.
    write last listing line.
End.
```

1.b. SIC MACHINE ARCHITECTURE:

i) MEMORY:

a) Memory storage in SIC consists of 8 bit-bytes and all memory address in SIC are byte addresses.

b) Total of 2^{15} bytes of total memory.

c> Any consecutive bytes from a word

* All addresses in SIC are byte addres.

* Words are addressed by location of their numbered byte.

ii> REGISTERS

REGISTER	NUMBER	FUNCTION	USES
A	0	Accumulator	Arithematic operations
X	1	Index Register	Addresing
L	2	Linkage Register	storing return address for sub routine jump
PC	8	Program counter	contain Address of next instruction to be fetched for execution.
SW	9	status word	Flags, other information including condition code (CC).

~~DA~~

iii) DATA FORMAT:

a) Integers are represented by 24 bit.

b) Negative numbers are represented in 2's complement.

c) Characters are represented by 8 bit ASCII value.

d) NO floating point representation is available.

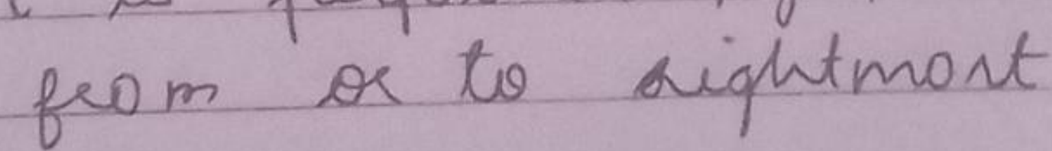
iv) ADDRESING MODES

a) Direct $x=0$ TA = address

b) Indexed $x=1$ TA = adress + X)

v) INSTRUCTION FORMAT

All instructions in SIC have 24 bit format. If $x=0$ it means direct addressing, If $x=1$ it means indexed addressing mode.



vi) INSTRUCTION SET

a) Load and store instructions, comparison instructions, arithematic instructions, conditional jump, subroutine linkage.

vii) INPUT AND OUTPUT

It is performed by transfering 1 byte at a time from or to rightmost 8 bit of

accumulator. Each device has 8 bit unique code. There are three I/O instruction

a) Test Device
b) Read data
c) Write data

SIC/XE ARCHITECTURE:

i) MEMORY:

a) Number of address lines are 20.
b) Total of 2^{20} bytes of memory is required.

ii) REGISTERS:

REGISTER	NUMBER / CODE	FUNCTION
A	0	Accumulator
X	1	Index Register
L	2	Linkage Register
B	3	Base Register
S	4	General purpose Register
T	5	General purpose Register
F	6	Floating point integer
PC	7	Program counter
SW	9	Status word.

iii) DATA FORMAT:

a) Integer - 3 bytes
b) Character - 1 byte
c) Floating point - 6 byte

iv) INSTRUCTION FORMAT:

a) FORMAT-1 : 8 bits
opcode

b) FORMAT-2

8	4	4
opcode	R1	R2

c) FORMAT-3

6	1 1 1 1 1 1	12
opcode	n i x b p e	Disp/data

d) FORMAT-4

6	1 1 1 1 1 1	20
opcode	n i x b p e	Disp/data

v) INSTRUCTION SET:

a) Load instructions, store instructions, Arithematic instructions, compare instructions, jump instructions, subroutines instructions, move instructions, clear instructions.

vi) INPUT AND OUTPUT:

a) Test Drive, Read data and write data.
b) Start I/O device Operation (SIO), Test

I/O operations (TIO), Halt I/O operations (HIO).

1.C. 3. ABSOLUTE LOADER:

i) It is a loader in which relocated object files are readed, loader writes there files and places them at a specified location in the memory.

ii) The starting address of every module is known to the programmer, this corresponding starting address is stated in the object file then the task of loader becomes very simple that is to simply place the executable form of the machine instructions at the locations.

iii) In this scheme, the programmer or assembler should have knowledge of memory management, the program should take care of 2 things.

iv) Specification of starting address of each module to be used if same modification is done in same module, then the length of that module may vary.

v) This causes change in starting address of paradiate next modules.

vi) While branching from one segment to another the absolute starting address of respective module is important instruction.

BOOTSTRAP LOADER:

i) Bootstrap is a special type of absolut loader.

ii) It begins at address 0.

iii) It loads the OS starting address 0x80.

2.a. i) The given tree shows the ~~address~~ order in which operations in assigned.

ii) position = initial + rate * 60 are to be performed. The tree has an interior node labeled * with (id,3) as its left child and integer 60 as its right child.

iii) The node (id,3) represents identifier rate.

iv) The node labeled * makes it explicit that we must first multiply value of rate 60.

v) The node labelled + indicates that we must add result to initial value.

vi) '=' indicates that we must store the result of this into location for identifier position.

2.b. i) Sentinel is an extra key inserted at the end of the array.

ii) It is a special dummy character that can't be a part of source program.

CODE:

```
Switch (* forward ++) {
    case eof :
        if (forward is at end of first buffer) {
            reload second buffer;
            forward = beginning of second buffer;
        }
        else if (forward is at end of second buffer) {
            reload first buffer;
            forward = beginning of first buffer;
        }
        else
            break;
    }
```

— x — x —