# Unit-I

## Q.) Discuss the SIC machine Architecture.

→ **Memory:**

Memory consists of 8-bit bytes. ~~ary~~ All addresses on SIC are byte addresses. Words are ~~associated~~ located by the address of their lowest numbered byte. There are $2^{15}$ byte in computer memory.

→ **Registers:**

Five registers, each 24 bits in length.

~~Mnemonic Number Special Use~~
~~A 0~~

| Mnemonic | No. | Special Use |
|---|---|---|
| A | 0 | Accumulator, used for arithematic op$^n$ |
| X | 1 | Index Register, used for addressing |
| L | 2 | Linkage register; the jump to subroutine (JSUB) instruction stores the return address in this register |
| PC | 8 | Program counter; Contains the address of next inst$^n$ to be fetched ~~f~~ for execution |
| SW | 9 | Status word; contains a variety of info, including a Cond$^n$ code (CC) |

## → Data formats

Integers are stored as 24 bit binary numbers. 2's complement rep. is used for negative values. Characters are stored using their 8 bit ASCII codes. There is no floating point hardware on standard version of SIC.

## → Instruction formats

| 8 | 1 | 15 |
|---|---|---|
| opcode | x | address |

flag bit x is used to indicate indeped addressing mode.

## → Addressing Modes

| Mode | Indication | Target add. calc$^n$ |
|---|---|---|
| Direct | X=0 | TA = address |
| Indexed | X=1 | TA = address + (X) |

## → Instr$^n$ set

(LDA, LDX, STA, STX etc) basic task.

integer arithmetic op$^n$ (ADD, SUB, MUL, DIV)

All arithematic operations involve register A & a word in memory, with the result being left in the register.

→ Input & Output

On the standard version of SIC, input & output are performed by transferring 1 byte at a time, to or from the rightmost 8 bits of register A. Each device is assigned a unique 8-bit code. There are 3 three I/O instr$^n$, each of which specifies the device code as an operand

Test Device (TD) instr$^n$ tests wheather the addressed device is ready to send or receive a byte of data.
CC is set to indicate the result of this test.

A program needing to transfer data must

## Q.) Instr$^n$ format of SIC/XE

→ **Format 1 (1 byte) :**

| 8 |
|---|
| op |

**Format 2 (2bytes) :**

| 8 | 4 | 4 |
|---|---|---|
| op | r1 | r2 |

**Format 3 (3bytes) :**

| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 12 |
|---|---|---|---|---|---|---|---|
| op | n | i | x | b | p | e | disp |

**Format 4 (4 bytes) :**

| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 20 |
|---|---|---|---|---|---|---|---|
| op | n | i | x | b | p | e | address |

## Q.) Addressing modes

Two more addressing modes are available for use ~~with instr$^n$ ass~~.

| Mode | Indication | Target address calc$^n$ |
|---|---|---|
| Base relative | $b=1, p=0$ | $TA=(B)+disp$ $(0<disp<4095)$ |
| Program-counter relative | $b=0, p=1$ | $TA=(PC)+disp$ $(-2048<disp<2047)$ |

Scanned with CamScanner

Q.) Discuss first pass

→ first pass does little more than scan the source program for label definitions & assign addresses

→ General description of the functions

Pass1 (define symbols):

1.) Assign addresses to all statements in the program.

2.) Save the values (addresses) assigned to all labels for use in pass 2.

3.) Perform some processing of assembler directives.

Pass2 (assemble instrn & generate obj. program):

1) Assemble instructions

2) Generate data values defined by BYTE, WORD etc.

3.) Perform processing of assembler directives ~~of~~ not done during Pass 1.

4) Write the object program & the assembly Listing.