**Prepared By**
**CHETAN D S**
**Asst. Prof. CSE**
**RNSIT**

# SYSTEM SOFTWARE

## SYLLABUS

## PART – A

**UNIT – 1** **6 Hours**
**Machine Architecture:** Introduction, System Software and Machine Architecture, Simplified Instructional Computer (SIC) - SIC Machine Architecture, SIC/XE Machine Architecture, SIC Programming Examples.

**UNIT – 2** **6 Hours**
**Assemblers -1:** Basic Assembler Function - A Simple SIC Assembler, Assembler Algorithm and Data Structures, Machine Dependent Assembler Features - Instruction Formats & Addressing Modes, Program Relocation.

**UNIT – 3** **6 Hours**
**Assemblers -2:** Machine Independent Assembler Features – Literals,Symbol-Definition Statements, Expression, Program Blocks, Control Sections and Programming Linking, Assembler Design Operations - One- Pass Assembler, Multi-Pass Assembler, Implementation Examples – MASM Assembler.

**UNIT – 4** **8 Hours**
**Loaders and Linkers:** Basic Loader Functions - Design of an Absolute Loader, A Simple Bootstrap Loader, Machine-Dependent Loader Features – Relocation, Program Linking, Algorithm and Data Structures for a Linking Loader; Machine-Independent Loader Features - Automatic Library Search, Loader Options, Loader Design Options - Linkage Editor, Dynamic Linkage, Bootstrap Loaders, Implementation Examples - MS-DOS Linker.

## PART – B

**UNIT – 5** **6 Hours**
**Editors and Debugging Systems:** Text Editors - Overview of Editing Process, User Interface, Editor Structure, Interactive Debugging Systems - Debugging Functions and Capabilities, Relationship With Other Parts Of The System, User-Interface Criteria

**UNIT – 6** **8 Hours**
**Macro Processor:** Basic Macro Processor Functions - Macro Definitions and Expansion, Macro Processor Algorithm and Data Structures, Machine- Independent Macro Processor Features - Concatenation of Macro Parameters, Generation of Unique Labels, Conditional Macro Expansion, Keyword Macro Parameters, Macro Processor Design Options - Recursive Macro
Expansion, General-Purpose Macro Processors, Macro Processing Within Language Translators, Implementation Examples - MASM Macro Processor, ANSI C Macro Processor.

**UNIT – 7** **6 Hours**

**Lex and Yacc – 1:** Lex and Yacc - The Simplest Lex Program, Recognizing Words With LEX, Symbol Tables, Grammars, Parser-Lexer Communication, The Parts of Speech Lexer, A YACC Parser, The Rules Section, Running LEX and YACC, LEX and Hand- Written Lexers, Using LEX – Regular Expression, Examples of Regular Expressions, A Word Counting Program,
Parsing a Command Line.

**UNIT – 8** **6 Hours**

**Lex and Yacc - 2 :** Using YACC – Grammars, Recursive Rules, Shift/Reduce Parsing, What YACC Cannot Parse, A YACC Parser - The Definition Section, The Rules Section, Symbol Values and Actions, The LEXER, Compiling and Running a Simple Parser, Arithmetic Expressions and Ambiguity, Variables and Typed Tokens.

**Text Books:**

1. Leland.L.Beck: System Software, 3rd Edition, Pearson Education, 1997.
(Chapters 1.1 to 1.3, 2 (except 2.5.2 and 2.5.3), 3 (except 3.5.2 and 3.5.3), 4 (except 4.4.3))

2. John.R.Levine, Tony Mason and Doug Brown: Lex and Yacc, O'Reilly, SPD, 1998.
(Chapters 1, 2 (Page 2-42), 3 (Page 51-65))

**Reference Books:**

1. D.M.Dhamdhere: System Programming and Operating Systems, 2nd Edition, Tata McGraw - Hill, 1999.

# UNIT 1: MACHINE ARCHITECTURE

The Software is set of instructions or programs written to carry out certain task on digital computers. It is classified into system software and application software. System software consists of a variety of programs that support the operation of a computer. Application software focuses on an application or problem to be solved. System software consists of a variety of programs that support the operation of a computer.

Examples for system software are Operating system, compiler, assembler, macro processor, loader or linker, debugger, text editor, database management systems (some of them) and, software engineering tools. These software’s make it 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.

Difference between System Software and Application Software

| **System Software** | **Application Software** |
|---|---|
| System Software intended to support the operation and use of computer | Application Software is primarily concerned with the solution of some problem using computer as a tool |
| Related to Machine Architecture | Not related to machine architecture |
| Machine Dependent<br>Example: Compilers, Assemblers, OS etc | Machine Independent<br>Example: Payroll System, Games etc |

## The Simplified Instructional Computer (SIC):

Simplified Instructional Computer (SIC) is a hypothetical computer that includes the hardware features most often found on real machines. There are two versions of SIC, they are, standard model (SIC), and, extension version (SIC/XE) (extra equipment or extra expensive).

## SIC Machine Architecture:

We discuss here the SIC machine architecture with respect to its Memory and Registers, Data Formats, Instruction Formats, Addressing Modes, Instruction Set, Input and Output

- **Memory:**

There are a total of 32,768($2^{15}$)bytes in the computer memory. It uses Little Endian format to store the numbers, 3 consecutive bytes form a word, and each location in memory contains 8-bit bytes.

- **Registers:**

There are five registers, each 24 bits in length. Their mnemonic, number and use are given in the following table.

| **Mnemonic** | **Number** | **Use** |
|---|---|---|
| A | 0 | Accumulator; used for arithmetic operations |
| X | 1 | Index register; used for addressing |
| L | 2 | Linkage register; JSUB |
| PC | 8 | Program counter |
| SW | 9 | Status word, including CC |

- **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. No floating-point hardware on the standard version of SIC.

- **Instruction Formats:**

All machine instructions on the standard version of SIC have the 24-bit format as shown above

| 8 | 1 | 15 |
|---|---|---|
| Opcode | x | Address |

- **Addressing Modes:**

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

There are two addressing modes available, which are as shown in the above table. Parentheses are used to indicate the contents of a register or a memory location.

- **Instruction Set :**

1. SIC provides, load and store instructions (LDA, LDX, STA, STX, etc.). Integer arithmetic operations: (ADD, SUB, MUL, DIV, etc.).
2. All arithmetic operations involve register A and a word in memory, with the result being left in the register. Two instructions are provided for subroutine linkage.
3. COMP compares the value in register A with a word in memory, this instruction sets a condition code CC to indicate the result. There are conditional jump instructions: (JLT, JEQ, JGT), these instructions test the setting of CC and jump accordingly.
4. JSUB jumps to the subroutine placing the return address in register L, RSUB returns by jumping to the address contained in register L.

- **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). 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.

**Data movement and Storage Definition**

LDA, STA, LDL, STL, LDX, STX (A- Accumulator, L – Linkage Register, X – Index Register)**,** all uses3-byte word. LDCH, STCH associated with characters uses 1-byte. There are no memory-memory move instructions.

Storage definitions are

- WORD - ONE-WORD CONSTANT
- RESW - ONE-WORD VARIABLE
- BYTE - ONE-BYTE CONSTANT
- RESB - ONE-BYTE VARIABLE

## Example Programs (SIC):

**Example 1: Simple data and character movement operation**

To store the value 5 in a variable ALPHA and character Z in a variable C1

```
            LDA  FIVE
            STA  ALPHA
            LDCH      CHARZ
            STCH      C1
ALPHA       RESW      1
FIVE        WORD      5
CHARZ       BYTE      C'Z'
C1          RESB      1
```

**Example 2: Arithmetic operations**

BETA=ALPHA+INCR+1

```
            LDA   ALPHA

            ADD   INCR

            SUB   ONE

            STA   BETA

            ……..

            ……..

            ……..

ONE         WORD   1

ALPHA       RESW   1

BEETA       RESW   1

INCR        RESW   1
```

**Example 3: Looping and Indexing operation**

To perform STR2=STR1 where STR1 is a string of 11 characters.

```
          LDX     ZERO          ;  X = 0

 MOVECH   LDCH    STR1, X       ;  LOAD A FROM STR1

          STCH    STR2, X       ;  STORE A TO STR2

          TIX     ELEVEN        ;  ADD 1 TO X, TEST

          JLT     MOVECH

         .

         .
```

```
                .

STR1        BYTE     C ‘HELLO WORLD’

STR2        RESB     11

ZERO        WORD     0

ELEVEN      WORD     11
```

**Example 4: Input and Output operation**

To read a character from the input device and to write a character to the output device.

```
INLOOP      TD      INDEV          : TEST INPUT DEVICE

            JEQ     INLOOP         : LOOP UNTIL DEVICE IS READY

            RD      INDEV          : READ ONE BYTE INTO A

            STCH    DATA           :  STORE A TO DATA

            .

            .

OUTLOOP     TD      OUTDEV         : TEST OUTPUT DEVICE

            JEQ     OUTLP          : LOOP UNTIL DEVICE IS READY
            LDCH    DATA           : LOAD DATA INTO A

            WD      OUTDEV         : WRITE A TO OUTPUT DEVICE

            .

            .

INDEV       BYTE     X ‘F5’        : INPUT DEVICE NUMBER

OUTDEV      BYTE     X ‘08’        : OUTPUT DEVICE NUMBER
```

**Example 5: To transfer two hundred bytes of data from input device to memory**

```
         LDX   ZERO
CLOOP    TD    INDEV
         JEQ   CLOOP
         RD    INDEV
         STCH  RECORD, X
         TIX   B200
         JLT   CLOOP
         .
         .
INDEV    BYTE  X ‘F5’
RECORD   RESB  200
ZERO     WORD  0
B200     WORD  200
```

## SIC/XE Machine Architecture:

- **Memory**

Maximum memory available on a SIC/XE system is 1 Megabyte ($2^{20}$ bytes).

- **Registers**

Additional B, S, T, and F registers are provided by SIC/XE, in addition to the registers of SIC.

| Mnemonic | Number | Special use |
|---|---|---|
| B | 3 | Base register |
| S | 4 | General working register |
| T | 5 | General working register |
| F | 6 | Floating-point accumulator (48 bits) |

- **Data Formats**

There is a 48-bit floating-point data type, $F*2^{(e-1024)}$

| 1 | 11 | 36 |
|---|---|---|
| s | exponent | fraction |

- **Instruction Formats:**

The new set of instruction formats fro SIC/XE machine architecture are as follows.

- Format 1 (1 byte): contains only operation code (straight from table).
- Format 2 (2 bytes): first eight bits for operation code, next four for register 1 and following four for register 2. The numbers for the registers go according to the

numbers indicated at the registers section (ie, register T is replaced by hex 5, F is replaced by hex 6).

- Format 3 (3 bytes): First 6 bits contain operation code, next 6 bits contain flags, last 12 bits contain displacement for the address of the operand. Operation code uses only 6 bits, thus the second hex digit will be affected by the values of the first two flags (n and i). The flags, in order, are: n, i, x, b, p, and e. Its functionality is explained in the next section. The last flag e indicates the instruction format (0 for 3 and 1 for 4).
- Format 4 (4 bytes): same as format 3 with an extra 2 hex digits (8 bits) for addresses that require more than 12 bits to be represented.

**Format 1 (1 byte)**

| 8 |
|---|
| op |

**Format 2 (2 bytes)**

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

Formats 1 and 2 are instructions do not reference memory at all

**Format 3 (3 bytes)**

| **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** |

- Addressing modes & Flag Bits

  Five possible addressing modes plus the combinations are as follows.

1. **Direct** (x, b, and p all set to 0): operand address goes as it is. n and i are both set to the same value, either 0 or 1. While in general that value is 1, if set to 0 for format 3 we can assume that the rest of the flags (x, b, p, and e) are used as a part of the address of the operand, to make the format compatible to the SIC format.

2. **Relative** (either b or p equal to 1 and the other one to 0): the address of the operand should be added to the current value stored at the B register (if b = 1) or to the value stored at the PC register (if p = 1)

3. **Immediate**(i = 1, n = 0): The operand value is already enclosed on the instruction (ie. lies on the last 12/20 bits of the instruction)

4. **Indirect**(i = 0, n = 1): The operand value points to an address that holds the address for the operand value.

5. **Indexed** (x = 1): value to be added to the value stored at the register x to obtain real address of the operand. This can be combined with any of the previous modes except immediate.

The various flag bits used in the above formats have the following meanings e - > e = 0 means format 3, e = 1 means format 4.

- **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.

- **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.

## Example Programs (SIC/XE)

**Example 1: Simple data and character movement operation**

To store the value 5 in a variable ALPHA and character Z in a variable C1

```
            LDA      #5
            STA     ALPHA

            LDA      #90
            STCH    C1

               .
               .
ALPHA       RESW      1
C1          RESB      1
```

**Example 2: Arithmetic operations**

BETA=ALPHA+INCR+1

```
LDS     INCR

LDA     ALPHA

ADDR S,A

SUB     1

STA     BETA

.

.
```

```
ALPHA       RESW        1
INCR        RESW        1
BETA        RESW        1
```

**Example 3: Looping and Indexing operation**

To perform STR2=STR1 where STR1 is a string of 11 characters.

```
            LDT     #11
            LDX     #0
MOVECH      LDCH    STR1, X    :  LOAD A FROM STR1
            STCH    STR2, X    :  STORE A TO STR2
            TIXR    T          :  ADD 1 TO X, TEST (T)
            JLT     MOVECH
            ……….
  STR1      BYTE    C 'HELLO WORLD'
  STR2      RESB    11
```

Difference between SIC and SIC/XE

| | **SIC** | **SIC/XE** |
|---|---|---|
| Memory | $2^{15}$ bytes | $2^{20}$ bytes |
| Registers | 5 (A,X,L,PC & SW) | 9(A,X,L,B,S,T,F,PC & SW) |
| Data Formats | No Floating Point Hardware | Supports Floating Point Hardware |
| Instruction Format | One | Four |
| Addressing Mode | Two | Five and its combination |

***