Unit 1

Machine Architecture

1.1 Introduction:

System Software consists of programs that supports the operation of a computer. This software makes its possible for the user to focus on an application without needing to know the detail of how the machine works internally.

Difference between system software & application software:

System Software	Application Software
It is a program or group of programs written for a computer system management.	It is a program or collection of programs written to solve a particular problem.
These are developed by the manufacturers.	These are developed by users.
To write system software the programmer needs to understand the architecture & hardware details and hence are Machine Dependent.	To write the application software the programmer need not worry about the architecture & hardware details & hence are Machine Independent.
System software control & manage the hardware.	Application software uses the services of the system software to interact with the hardware.
Development of system software is complex task.	Development of application software is relatively easier.
Ex: Operating System, Compilers, Text Editors, Assemblers, Loaders, Linkers	Ex: MS-WORD, MS-EXCEL, Payroll inventory system, Student management system, Library Management System

Most system software is machine-dependent, we must include real machines & real pieces of software.

1.2 The Simplified Instructional Computer (SIC):

SIC is a hypothetical computer that has been carefully designed to include the hardware features most often found on real machines, while avoiding unusual or irrelevant complexities.

They are two versions :

1. SIC - Standard Model
2. SIC/XE – Extra equipment or extra expensive.

The two versions are designed to be upward compatible i.e. An object program for the standard SIC machine will also executed properly on a SIC/XE machine.

1.3 SIC Machine Architecture (The Standard Model)

1.3.1 Memory:

1. Memory consists of 8 bit bytes.
2. Three consecutive bytes form a word.
3. All addresses on SIC are byte addresses.
4. These are a total of 32,768 (215) bytes.

1.3.2 Register:

There are 5 Register, each register is 24 bits in length.

Mnemonic	Number	Special Use
A	0	Accumulator, used for arithmetic operation.
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 the next instruction to be fetched for execution.
SW	9	Status word, contains a variety of information including a Condition Code (CC).

1.3.3 Data Formats:

1. Integer are stored as 24 bit binary number.
2. 2's Complement representations is used for negative values.
3. Characters are stored as 8 bit ASCII Code.
4. There is a no floating point hardware.

1.3.4 Instruction Formats:

All machine instruction have 24 bit formats.

8	1	15
opcode	x	address

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

1.3.5 Addressing Modes:

1. There are 2 types of addressing mode.
2. It indicate by the setting of the x bit in the instruction.

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

Direct Addressing mode :

Example: LDA TEN (opcode of LDA = 00)

Opcode (8)	x(1)	address(15)
0000 0000	0	001 0000 0000 0000

Effective address (EA)=1000

Content of the address 1000 is loaded to accumulator.

Indexed Addressing Mode:

Example: STCH BUFFER,X (opcode of STCH = 54)

Opcode (8)	x(1)	address(15)
0101 0100	1	001 0000 0000 0000

Effective address (EA) = 1000 + (x)

Accumulator contains the content of a calculated address.

1.3.6 Instruction Set:

1. SIC provides a basic set of instruction.
2. Instruction to load and store register (LDA,LDX,STA,STX).
3. It includes integer arithmetic operation (ADD,SUB,MUL,DIV).
4. There is an instruction COMP it compares the value in register A with a word in memory.
5. This instruction sets a Condition Code(CC) to indicate the result(>,=,<).
6. Conditional jumps instructional are used JLT,JEQ,JGT are used.
7. JSUB jumps to the subroutine placing the return address in Register "L".
8. RSUB returns by jumping to the address contained in register L.

1.3.7 Input and Output:

1. Input and output are performed by transferring one byte at a time.
2. Each device is assigned a unique eight bit code.
3. There are 3 IO instructions that uses device code as an operand.

- Text device(TD) : Instruction test whether the addressed device is ready to send or receive a byte of data. This can be done by setting a Condition Code(CC).

  < means the device is ready to send or receive.

  = means the device is not ready.

  A program needing to transfer data must wait until the device is ready, then execute.
- Read Data(RD) : Reads a byte of data from the addressed device.
- Write Data(WD) : Writes a byte of data to the addressed device.

1.4 SIC/XE Machine Architecture

1.4.1 Memory:

1. Memory consists of 8 bit bytes.
2. Three consecutive byte form a word.
3. All address on SIC/XE are byte address.
4. The maximum memory available on SIC/XE is 1 Mega Byte (2^{20} bytes).

1.4.2 Register:

There are 9 registers among which first 8 are 24bit in length.

Mnemonic	Number	Special Use
A	0	Accumulator, used for arithmetic operation.
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 the next instruction to be fetched for execution.
SW	9	Status word, contains a variety of information including a Condition Code (CC).
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)

1.4.3 Data Formats:

1. Integers are stored as 24 bit binary number.
2. 2's Complement representation is used for negative value.
3. Characters are stored as 8 bit ASCII code.
4. There is a 48 floating point data types having a following format.

1	11	36
s	exponent	fraction

fraction → It is interpreted as a value between 0 & 1. The high-order bit of fraction must be 1.

exponent → It is interpreted as an unsigned binary number between 0 & 2047.

s → It is interpreted as sign of the floating point number (s=0 Positive number s = 1 negative number).

1.4.4 Instruction formats:

1. SIC/XE machine support four instruction format.
2. Format 1 and format 2- that do not reference memory.
3. Format 3 and format 4- that reference memory.
4. If e is set 0 then it is Format 3 & if e is set 1 then it is Format 4.

Format 1 (1 byte) :

8
op

Ex : RSUB (opcode = 4C)

8 (op)
0100 1100
4 C

Format 2 (2 bytes) :

8	4	4
op	r1	r2

Ex: COMPR A, S (opcode COMPR= A0)

8 (op)	4 (r1)	4 (r2)
1010 0000	0 000	0 100

Format 3 (3 bytes):

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

Ex: LDA #3

Format 4 (4 bytes):

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

Ex: +JSUB RDREC

1.4.5 Addressing modes:

1. **Direct :** If b& p are both set to 0 then disp field is taken to be target address.
2. **Base relative :** b=1 & p=0, the displacement field is interpreted as a 12-bit unsigned integer. (0 <= disp <= 4095)
3. **Program Counter relative :** b=0 & p=1, the displacement field is interpreted as a 12-bit signed integer. (-2048 <= disp <= 2047)

4. **Indxed :** If x=1 then it is called as indexed addressing It can be combined with base-relative & program counter relative.
5. **Immediate :** If bit i=1 & n=0, the target address itself is used as the operand value. It is represented in the code with # symbol.
6. **Indirect:** If bit i=0 & n=1, the word at the location given by the TA is fetched, the value contained in this word is taken as the address of the operand value. It is represented in the code with @ symbol.

Name	Bits Set n	i	x	b	p	e	Format	Example	Target Address
Direct	1	1	0	0	0	0	Format 3	LDA LENGTH	TA = disp
	1	1	0	0	0	1	Format 4		TA = address
Base-Relative	1	1	0	1	0	0	Format 3	STX LENGTH	TA = disp + (B)
	1	1	0	1	0	1	Format 4		TA = address + (B)
Program Counter Relative	1	1	0	0	1	0		STL RETADR	TA = disp + (PC)
	1	1	0	0	1	1	Format 4		TA = address + (PC)
Indexed Base-Relative	1	1	1	1	0	0	Format 3	STCH BUFFER,X	TA = disp + (B) + (X)
	1	1	1	1	0	1	Format 4		TA = address + (B) + (X)
Indexed Program Counter Relative	1	1	1	0	1	0		LDCH BUFFER,X	TA = disp + (PC) + (X)
	1	1	1	0	1	1	Format 4		TA = address +(PC)+ (X)
Immediate	0	1	0	0	0	0	Format 3	LDA #10	TA = disp
	0	1	0	0	0	1	Format 4		TA = address
Immediate with Base-Relative	0	1	0	1	0	0/1	Format 3/4		TA = disp + (B) / TA = address + (B)
Immediate with Program Counter	0	1	0	0	1	0/1	Format 3/4		TA = disp + (PC) / TA = address + (PC)
Indirect	1	0	0	0	0	0	Format 3	J @RETADR	TA = disp
	1	0	0	0	0	1	Format 4		TA = address
Indirect with Base-Relative	1	0	0	1	0	0/1	Format 3/4		TA = disp + (B) / TA = address + (B)
Indirect with Program Counter	1	0	0	0	1	0/1	Format 3/4		TA = disp + (PC) / TA = address + (PC)

- Indexing cannot be used with immediate or indirect addressing modes.

- SIC/XE instructions that specify neither immediate nor indirect addressing are assembled with bits n & i both set to 1.

1.4.6 Instruction set:

1. Instruction to load and store register (LDA,LDX,STA,STX, LDB, STB, LDS, STS, LDT, STT etc).
2. It includes integer arithmetic operation (ADD,SUB,MUL,DIV).
3. There is an instruction COMP it compares the value in register A with a word in memory.
4. This instruction sets a Condition Code(CC) to indicate the result(>,=,<).
5. Conditional jumps instructional are used JLT,JEQ,JGT are used.
6. JSUB jumps to the subroutine placing the return address in Register "L".
7. RSUB returns by jumping to the address contained in register L.
8. Floating point arithmetic operation instruction are also available(ADDF,SUBF,MULF,DIVF)
9. Register to register arithmetic operation are also available. (ADDR,SUBR,MULR,DIVR)
10. Supervisor cal instruction are also available(SVC) to communicate with OS.

1.4.7 Input and Output:

1. Input and output are performed by transferring one byte at a time.
2. Each device is assigned a unique eight bit code.
3. There are 3 IO instructions that uses device code as an operand.
    - Text device(TD) : Instruction test whether the addressed device is ready to send or receive a byte of data. This can be done by setting a Condition Code(CC).
      < means the device is ready to send or receive.
      = means the device is not ready.
      A program needing to transfer data must wait until the device is ready, then execute.
    - Read Data(RD) : Reads a byte of data from the addressed device.
    - Write Data(WD) : Writes a byte of data to the addressed device.
4. There are IO channels that can be used to perform input and output while the CPU is executing other instructions.
5. The instructions SIO , PIO , HIO are used to Start, Test and Halt the operation of IO channels.

1.5 Problems on Target Address Calculation :

Generate the target address for the following object codes:

i) 032600 ii)010030 iii) 03C300h iv) 022030 v) 0310C303

Content of X=000090; Content of B=006000; Content of PC=003000;

Ans: I) 032600

Hex	opcode	n	i	x	b	p	e	disp/address
0 32600	0 000 0 0	1	1	0	0	1	0	0110 0000 0000

Program- Counter Relative Addressing Mode :

TA = (PC) + disp

= 003000 + 600

= 003600

Ans : ii) 010030

Hex	opcode	n	i	x	b	p	e	disp/address
0 10030	0 000 0 0	0	1	0	0	0	0	0000 0011 0000

Immediate Addressing:

TA = disp

= 0030

Ans : iii) 03C300H

Hex	opcode	n	i	x	b	p	e	disp/address
0 3C300	0 000 0 0	1	1	1	1	0	0	0011 0000 0000

Base Indexed Relative Addressing:

TA = (B) + disp +(X)

= 006000 + 300 + 000090

= 006390

Ans : iv) 022030

Hex	opcode	n	i	x	b	p	e	disp/address
0 22030	0 000 0 0	1	0	0	0	1	0	0000 0011 0000

Indirect Program Counter Relative Addressing Mode :

TA = (PC) + disp

= 003000 + 030

= 003030

Ans: v) 0310C303

Hex	opcode	n	i	x	b	p	e	address
0 310C303	0 000 0 0	1	1	0	0	0	1	0000 1100 0011 0000 0011

Simple Addressing mode:

TA = address

= 0C303

1.6 Simple SIC & SIC/XE Examples:

- No memory-memory move instruction
- 3-byte word:

  LDA, STA, LDL, STL, LDX, STX
- 1-byte:

  LDCH, STCH
- Storage definition

  WORD : Generate one word integer constant.

  RESW : Reserve the indicated number of words for a data area.

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

  RESB : Reserve the indicated number of bytes for a data area.
- All arithmetic operations are performed using register A, with the result being left in register A.

1.6.1. Data movement

```
        LDA     FIVE
        STA     ALPHA
        LDCH    CHRZ
        STCH    C1

ALPHA   RESW    1
FIVE    WORD    5
CHRZ    BYTE    C'Z'
C1      RESB    1
```

1.6.2 Arithmetic Operations:

2. Write a sequence of instrcutions for SIC and SIC/XE to set BETA=(ALPHA+INCR-1) & GAMMA = (DELTA+INCR-1).

SIC Example:

```
        LDA     ALPHA
        ADD     INCR
        SUB     ONE
        STA     BETA
        LDA     GAMMA
        ADD     INCR
        SUB     ONE
        STA     DELTA

ONE     WORD    1
ALPHA   RESW    1
BETA    RESW    1
```

```
GAMMA    RESW    1
DELTA    RESW    1
INCR     RESW    1
```

SIC/XE Example:

```
         LDS     INCR
         LDA     ALPHA
         ADDR    S, A
         SUB      #1
         STA     BETA
         LDA     GAMMA
         ADDR    S, A
         SUB     #1
         STA     DELTA

ALPHA    RESW    1
BETA     RESW    1
GAMMA    RESW    1
DELTA    RESW    1
INCR     RESW    1
```

1.6.3 Looping & Indexing Operations:

TIX instruction : First it increments the value of x by 1 then it tests the value with its operand value.

TIXR instruction : First it increments the value of x by 1 then it tests the value with its operand value register.

3. Write a sequence of instructions for SIC and SIC/XE to copy the string “system software” into another string.

SIC Example:

```
         LDX     ZERO
MOVECH   LDCH    STR1, X
         STCH    STR2, X
         TIX     FIFTEEN
         JLT     MOVECH

STR1     BYTE    C 'system software'
STR2     RESB    15
ZERO     WORD    0
FIFTEEN  WORD    15
```

SIC/XE Example:

```
         LDT     #15
         LDX     #0
MOVECH   LDCH    STR1, X
         STCH    STR2, X
         TIXR    T
         JLT     MOVECH
```

```
STR1        BYTE        C 'system software'
STR2        RESB        15
```

4. Write a sequence of instruction for SIC to clear 20 bytes strings to all blanks.

SIC Example:

```
            LDX         ZERO
MOVECH      LDCH        CHRZ
            STCH        STR2, X
            TIX         TWENTY
            JLT         MOVECH

CHRZ        BYTE        C ' '
STR2        RESB        20
ZERO        WORD        0
TWENTY      WORD        20
```

Note:

TIX instruction adds 1 to register so it is not suitable for next program where the value of index register must be incremented by 3 byte.

5. Write a sequence of instructions for SIC & SIC/XE to add two array elements namely ALPHA & BETA & store the result in GAMMA.

SIC Example:

```
            LDA         ZERO
            STA         INDEX
ADDLP       LDX         INDEX
            LDA         ALPHA, X
            ADD         BETA, X
            STA         GAMMA, X
            LDA         INDEX
            ADD         THREE
            STA         INDEX
            COMP            K300
            JLT         ADDLP

INDEX       RESW            1
ALPHA       RESW            100
BETA        RESW            100
GAMMA       RESW            100
ZERO        WORD            0
K300        WORD            300
```

SIC/XE Example:

```
            LDS         #3
            LDT         #300
```

```
            LDX     #0
ADDLP       LDA     ALPHA, X
            ADD     BETA, X
            STA     GAMMA, X
            ADDR    S, X
            COMPR   X, T
            JLT     ADDLP

ALPHA       RESW        100
BETA        RESW        100
GAMMA       RESW        100
```

1.6.4 Input & Output:

6. Write a sequence of instructions for SIC to read a 1-byte of data from the device 'F1' & copy it to the device '05'.

```
INLOOP      TD      INDEV
            JEQ     INLOOP
            RD      INDEV
            STCH    DATA

OUTLP       TD      OUTDEV
            JEQ     OUTLP
            LDCH    DATA
            WD      OUTDEV

INDEV       BYTE        X 'F1'
OUTDEV      BYTE        X '05'
DATA        RESB        1
```

7. Write a subroutine for SIC & SIC/XE to read a 100-byte record from the device 'F1' into BUFFER.

SIC Example:

```
            JSUB READ
            .
READ        LDX     ZERO
RLOOP       TD      INDEV
            JEQ     RLOOP
            RD      INDEV
            STCH    BUFFER, X
            TIX     K100
            JLT     RLOOP
            RSUB

INDEV       BYTE        X 'F1'
BUFFER      RESB        100
ZERO        WORD        0
K100        WORD        100
```

SIC/XE Example:

```
            JSUB READ
            .
READ        LDX   #0
            LDT   #100
RLOOP       TD    INDEV
            JEQ   RLOOP
            RD    INDEV
            STCH  BUFFER, X
            TIXR  T
            JLT   RLOOP
            RSUB
            .
INDEV       BYTE        X 'F1'
BUFFER      RESB        100
```

8. Write a subroutine for SIC & SIC/XE to write a 100-byte record from BUFFER into the device '05'.

SIC Example:

```
            JSUB WRITE
            .           .
WRITE       LDX   ZERO
WLOOP       TD    OUTDEV
            JEQ   WLOOP
            LDCH  BUFFER, X
            WD    OUTDEV
            TIX   K100
            JLT   WLOOP
            RSUB
            .
OUTDEV      BYTE        X '05'
BUFFER      RESB        100
ZERO        WORD        0
K100        WORD        100
```

SIC/XE Example:

```
            JSUB WRITE
            .
WRITE       LDX   #0
            LDT   #100
WLOOP       TD    OUTDEV
            JEQ   WLOOP
            LDCH  BUFFER, X
            WD    OUTDEV
            TIXR  T
            JLT   WLOOP
            RSUB
```

```
.
OUTDEV    BYTE      X '05'
BUFFER    RESB      100
```

1.7 Pentium Pro Architecture

1.7.1 Memory:

Memory can be defined in 2 ways:

Physical Level:

1. Memory consists of 8 bit bytes
2. All addresses used are byte address.
3. Two consecutive byte form a word.
4. Four byte form a double-word.(dword)

Logical View: Collection of segments

1. Memory address consists of two parts: a segment number & an offset.
2. Segments can be of different sizes & often used for different purposes.
3. A segment can also be divided into pages.
4. The segment/offset address specified by the programmer is automatically translated into a physical byte address by the x86 Memory Management Unit (MMU).

1.7.2 Register:

1. These are 8 general-purpose register,which are named EAX,EBX,ECX,EDX,ESI,EDI,EBP and ESP.
2. Each general purpose register is 32 bits long(One double word)
3. Register EAX,EBX,ECX, and EDX are generally used for data manipulation.
4. Register ESI, EDI, EBP & ESP are generally used to hold addresses.
5. EIP is a 32-bit special purpose register that contains a pointer to the next instruction to be executed.
6. FLAGS is a 32-bit register that contains many different bit flags which indicates the status of the processor, some contains the results of comparisons & arithmetic operations.
7. There are 6 16-bit segment register which are named DS,CS,SS,ES,GS,FS.
8. Floating point computations are performed using a special floating point unit (FPU). This unit contains eight 80 bit data registers and several other control and status registers.

1.7.3 Data formats:

1. Integers are normally stored as 8,16 or 32 bits binary numbers.
2. 2's complement is used for negative values.
3. Integers can also be stored in binary coded decimal(BCD).
   - In unpacked BCD format, each byte represents one decimal digit.
   - In packed BCD format, each byte represents two decimal digits with each encoded using 4 bits of the byte.
4. Characters are stored one per byte using their 8 bit ASCII codes.
5. Strings may consists of bits, bytes, words or doublewords, special instructions are provided to handle each type of string.
6. There are three different floating point data formats.

a. The single-precision format is 32 bit long.

1	7	24
s	exponent	fraction

b. The double-precision format is 64 bit long.

1	10	53
s	exponent	fraction

c. Extended-precision format is 80 bit long.

1	15	64
s	exponent	fraction

1.7.4 Instruction formats:

1. All of the x86 machine instructions use variations of the same basic format.
2. This format begins with optional prefixes containing flags that modify the operation of the instructions.
3. Following the prefixes is an opcode(1 or 2 bytes);
4. Following the opcode are a number of bytes that specify the operands & addressing modes to be used.
5. The opcode is the only elements that is always present in every instructions.
6. Other elements may or may not be present, and may be of different lengths,depending on the operations and the operands involved.

7. There are a large number of different potential instruction formats, varying in length from 1 byte to 10 bytes or more.

1.7.5 Addressing Modes:

1. An operand value may be specified as part of the instruction itself (immediate mode), or it may be in a register (register mode).
2. Operands stored in memory are often specified using variations of the general target address calculations.

   TA = (base register) + (index register) * (scale factor) + displacement
3. Any general-purpose register may be used as base register & any general-purpose register except ESP can be used as index register.
4. The scale factor may have the value 1, 2, 4 or 8 and displacement may be at 8-, 16- or 32-bit value.
5. Various combinations of these items may be omitted resulting in eight addressing mode.
6. The address of an operand in memory may also be specified as an absolute location (direct mode) or as a location relative to the EIP register (relative mode).

1.7.6 Instruction set:

1. There are more than 400 different machine instructions available.
2. An instruction may have zero, one,two,or three operands.
3. There are register-to-register, registers-to-memory instructions, and a few memory-to-memory instructions.
4. In some cases, operands may also be specified in the instructions as immediate values.
5. Most data movements and integers arithmetic instructions can use operands that are 1,2, or 4 byte.
6. These are many instructions that perform logical and bit manipulations and support control of the processor and memory-management systems.

1.7.7 Input and Output:

1. Input is performed by instructions that transfer one byte,word or doubleword at a time from an I/O port into register EAX.
2. Output instructions transfer one byte word,or double word from EAX to an I/O port.