1) What is system software?

System software consists of variety of programs that supports the operations of a computer. This makes 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.

Examples of system software are text-editors, compilers, loaders or linkers, debuggers, assemblers , and operating systems.

2) How system software is different from application Software?

The most important characteristic in which most system software differ from application software is machine dependency.

An application program is primarily concerned with the solution to some problem ,using computer as a tool. The focus is on the application, not on the application system.

An application software cannot run on itself but it is dependent on system software to execute.

Ex:MS Word, a spreadsheet system.

System programs, on the other hand, are intended to support the operation and use of the computer itself, rather than any particular application. They are usually related to the architecture of the machine on which they are to run. For ex in the translation of mnemonic instructions into machine code the instruction formats and addressing modes of an assembler are directly concerned with the design of an assembler. Similarly compilers that generate the machine code are also depending on the number and types of registers and the machine instructions available.

3) Explain the machine dependency of system software with examples?

An assembler is a system software. It translates mnemonic instructions into machine code; the instruction formats,addressing modes ,etc,are of direct concern in assembler design.

Similarly, compilers must generate machine language code,taking into account such hardware characteristics as the number and the types of registers and machine instructions available

Operating systems are directly concerned with the management of nearly all of the resources of a computer system.

- Example:
- When you took the first programming course
  - **Text editor** - create and modify the program
  - **Compiler**- translate programs into machine language
  - **Loader or linker** - load machine language program into memory
- and prepared for execution
  - **Debugger** - help detect errors in the program
- When you wrote programs in assembler language
  - **Assembler** - translate assembly program into machine language
  - **Macro processor** - translate macros instructions into its definition
- When you control all of these processes
  - By interacting with the OS

4) What are the important machine structures used in the design of system software?

- Memory structure
- Registers
- Data formats
- Instruction formats
- Addressing modes
- Instruction set

5) What is SIC machine?

SIC refers to Simplified Instruction Computer which is a hypothetical computer that has been designed to include the hardware features most often found on real machines, while avoiding unusual and irrelevant complexities. This allows to clearly separate the central concepts of a system software from the implementation details associated with a particular machine.

6) Explain SIC Machine architecture.

SIC Machine Architecture

- **Memory**
  - All addresses are byte addresses with words associated by the location of their Lowest numbered byte.
  - Consist of 8-bit bytes.
  - 3 consecutive bytes form a word (24 bits)
  - Total size of the memory is 32768(2^{15}) bytes.

- **Registers**

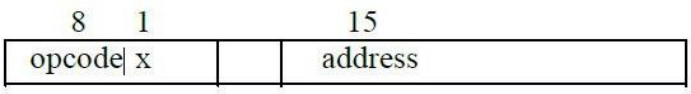
  There are 5 special purpose registers each of which are 24 bits in size as shown in table.

Mnemonic	Number	Special 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 as 8-bit ASCII codes.
  - No floating-point hardware.

- **Instruction Formats**

  All machine instructions have the following format:



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

- **Addressing Modes**

  There are two addressing modes, indicated by the setting of the X bit in the instruction.. The target address (TA) is calculated for each of the modes as shown below:

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

Parentheses are used to indicate the content of a register or memory location. In the above table (X) indicates the content of the register X.

- **Instruction Set**
  - **Integer arithmetic operations**: ADD, SUB, MUL, DIV, etc.
    - All arithmetic operations involve register A and a word in memory, with the result being left in the register
  - **comparison**: COMP
    - COMP compares the value in register A with a word in memory, this instruction sets a condition code CC to indicate the result
  - **conditional jump instructions**: JLT, JEQ, JGT
    - these instructions test the setting of CC and jump accordingly
  - **subroutine linkage**: JSUB, RSUB
    - JSUB jumps to the subroutine, placing the return address in register L
    - RSUB returns by jumping to the address contained in register L.
  - **Load and store registers**
    - Instructions to store and load registers are LDA, LDX, STA, STX.

- **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.
  - Each device is assigned a unique 8 bit code.
  - There are 3 I/O instructions
    1. The Test Device (TD) instruction tests whether the addressed device is ready to send or receive a byte of data.(CC is set to <,if it is ready and set to =, if its not ready).
    2. Read Data (RD), reads the data from an input device, if it is ready to send the data.
    3. Write Data (WD) writes the data onto ann output device, if its ready to receive the data.

7. Explain SIC/XE architecture

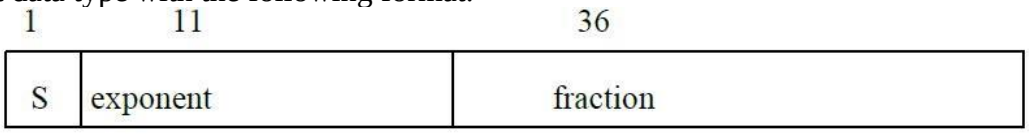
SIC/XE Machine Architecture

- **Memory**
  Memory structure is same as that of the SIC standard version. The maximum memory available is 1MegaByte(2^{20}) bytes.

- **Registers**
  In addition to the registers of SIC standard version, the following registers are provided in SIC/XE.

Mnemonic	Number	Special use
B	3	Base register; used for addressing
S	4	General working register
T	5	General working register
F	6	Floating-point acumulator (48bits)

- **Data Formats**
  In addition to the data formats provided in SIC Standard version, there is a 48 bit floating point data type with the following format.



The fraction is interpreted as a value between 0 & 1. For normalized floating-point numbers, the high order bit of the fraction must be 1. The exponent is interpreted as an unsigned binary number between 0 & 2047. If the exponent has value e & the fraction has value f, the absolute value of the number represented is $f*2^{(e-1024)}$. The sign of floating point number is indicated by the value of S (0 = +ve & 1 = -ve).

- **Instruction Formats**
  - SIC/XE memory available is larger, the instruction format used in the SIC is no longer available**.**
  - There are two possible options**:**
    1. Use relative addressing
    2. Extend the address field to 20 bits.
  - SIC/ XE provides some instructions that do not reference memory at all. Format 1 and 2 are used for such instructions.
  - Format 3 and 4 for new set of instructions. If e=0, it is format 3 and if e=4 it is format 4.

Format 1 (1 byte)		
op(8)		
Format 2 (2 bytes)		
op(8)	r1(4)	r2(4)

Format 3(3 bytes)

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

Format 4(4 bytes)

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

- **Instruction Set**
  In addition to the instruction set, provided in SIC Standard version, SIC/XE provides following instructions.
  - new registers: LDB, STB, etc.
  - floating-point arithmetic: ADDF, SUBF, MULF, DIVF
  - register move: RMO
  - register-register arithmetic: ADDR, SUBR, MULR, DIVR
  - supervisor call: SVC
    - generates an interrupt for OS
- **Input/Output**
  In addition to the I/O instructions provided in SIC Standard version, SIC/XE provides I/O channels used to perform input and output while CPU is executing other instructions. SIO, TIO, HIO: start, test, halt the operation of I/O channels respectively.

- **Addressing modes:**

Mode	Indication	TargetAddress Calculation
Base relative	b=1,p=0	TA= (B) + disp
Program counter relative	b=0,p=0	TA= (pc)+ disp

Base relative addressing: bit b=1and p=0 and disp field is interpreted as a 12 bit unsigned integer in format 3.
Program counter relative addressing: bit b=0and p=1 and disp field is interpreted as a 12 bit signed integer, with negative values represented in 2's complement notation.

Direct addressing : The displacement and address fields will be taken as the target address respectively in format 3 and format 4, if the bits b and p are both set to 0.

Indexed addressing: If the bit x is set to 1, the content of register X is also added in the target address calculation.

Immediate addressing : if the bits i=1 and n=0, the target address itself is the operand value and no memory is referenced.

Indirect addressing: if the bits i=0, and n=1, the word at the location given by the target address contains the address of the operand value.

Simple addressing: if the bits i =n=0 or i=n=1, the target address is taken as the location of the operand.

- **Addressing modes**
  - **Base relative (n=1, i=1, b=1, p=0)**
  - **Program-counter relative (n=1, i=1, b=0, p=1)**
  - **Direct (n=1, i=1, b=0, p=0)**
  - **Immediate (n=0, i=1, x=0)**
  - **Indirect (n=1, i=0, x=0)**
  - **Indexing (both n & i = 0 or 1, x=1)**
  - **Extended (e=1 for format 4, e=0 for format 3)**

Problem Calculate the target address generated for the following machine instruction

I. 032600 h
II. 03C300 h
III. 022030 h
IV. 010030 h
V. 003600 h
VI. 0310C303 h

If (B)= 006000, (PC)=003000, (X)=000090

1. **032600 h**
   0-0000
   3- 0011
   2-0010
   6-0110
   0-0000
   0-0000

op(6 bits)	n(1)	i(1)	x(1)	b (1)	p(1)	e (1)	disp(12 bits)
0000 00	**1**	**1**	**0**	**0**	**1**	**0**	**0110 0000 0000**

Since p=1 it is program counter relative addressing add content of (PC) to disp value.

TA=disp+ (PC)
= 600+003000
= 3600.

---

1. **03C300 h**
   0-0000
   3-0011
   C-1100
   3-0011
   0-0000
   0-0000

op(6 bits)	n(1)	i(1)	x(1)	b (1)	p(1)	e (1)	disp(12 bits)
0000 00	**1**	**1**	**1**	**1**	**0**	**0**	**0011 0000 0000**

Since b and p are set to 1, Add content of both (X) and (B) contents to disp
TA= disp+(B)+(X)
= 300+006000+000090
=6390.

2. **022030 h**

op(6 bits)	n(1)	i(1)	x(1)	b (1)	p(1)	e (1)	disp(12 bits)
0000 00	1	0	0	0	1	0	0000 0011 0000

p is set to 1.

TA=disp+(PC)
=030+003000
=3030.

3. **010030 h**

op(6 bits)	n(1)	i(1)	x(1)	b (1)	p(1)	e (1)	disp(12 bits)
0000 00	0	1	0	0	0	0	0000 0011 0000

b=p=0 ; the disp field is taken as TA for format 3(direct addressing)

TA=030.

4. **003600 h**

op(6 bits)	n(1)	i(1)	x(1)	b (1)	p(1)	e (1)	disp(12 bits)
0000 00	0	0	0	0	1	1	0110 0000 0000

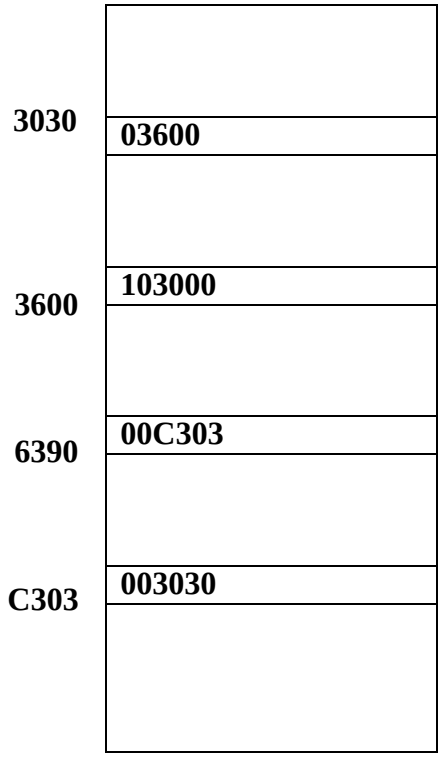
TA=disp+(PC)
=600+003000
=3600

5. **0310C303**

op(6 bits)	n(1)	i(1)	x(1)	b (1)	p(1)	e (1)	address(20 bits)
0000 00	1	1	0	0	0	1	0000 1100 0011 0000 0011

b=p=0 ; the address field is taken as TA for format 4(direct addressing)

TA= C303.



Note: To caluclate value loaded into register A

1. 032600 h

op(6 bits)	n(1)	i(1)	x(1)	b (1)	p(1)	e (1)	disp(12 bits)
0000 00	1	1	0	0	1	0	0110 0000 0000

n=i=1: Simple addressing; TA is taken as the location of the operand

TA= 3600
Value loaded= 103000 // value stored in the mem location 3600

2. 03C300 h

op(6 bits)	n(1)	i(1)	x(1)	b (1)	p(1)	e (1)	disp(12 bits)
0000 00	1	1	1	1	0	0	0011 0000 0000

n=i=1: Simple addressing; TA is taken as the location of the operand

TA= 6390
Value loaded= 00C303 // value stored in the mem location 6390

3. 022030 h

op(6 bits)	n(1)	i(1)	x(1)	b (1)	p(1)	e (1)	disp(12 bits)
0000 00	1	0	0	0	1	0	0000 0011 0000

n=1 and i=0; indirect addressing ;
The word at the location given by the target address contains the address of the operand value.

TA= 3030
Value stored in 3030 mem loc = 003600
003600 is the address of the operand value
Value stored in 3600 = 103000
So value loaded = 103000.

4. 010030 h

op(6 bits)	n(1)	i(1)	x(1)	b (1)	p(1)	e (1)	disp(12 bits)
0000 00	0	1	0	0	0	0	0000 0011 0000

i=1 and n=0; immediate addressing ; TA is taken as operand value. No memory reference is done.

TA=030
Value loaded= 000030

5. 003600 h

op(6 bits)	n(1)	i(1)	x(1)	b (1)	p(1)	e (1)	disp(12 bits)
0000 00	0	0	0	0	1	1	0110 0000 0000

i=n=0; simple addressing; TA is taken as the location of the operand

TA= 3600
Value loaded= 103000

6. 0310C303

op(6 bits)	n(1)	i(1)	x(1)	b (1)	p(1)	e (1)	address(20 bits)
0000 00	1	1	0	0	0	1	0000 1100 0011 0000 0011

i=n=1; simple addressing; TA is taken as the location of the operand
TA= C303
Value loaded= 00303.