BHCS15B: System Programming

Introduction

Mahesh Kumar
(maheshkumar@andc.du.ac.in)

Course Web Page
(www.mkbhandari.com/mkwiki)

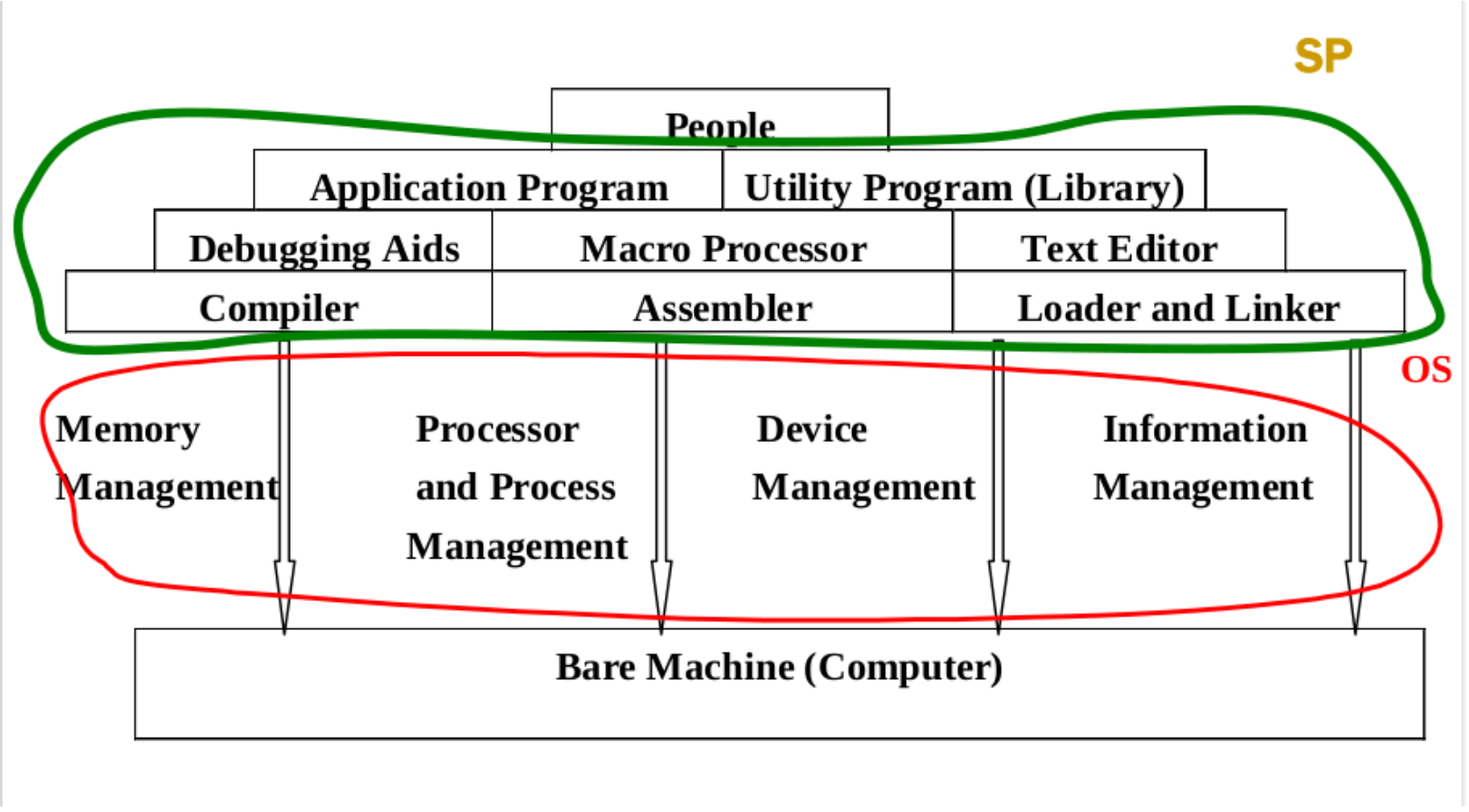
Outline

1. Introduction
2. System Software and Machine Architecture
3. Simplified Instructional Computer (SIC)
4. Traditional (CISC) Machines and RISC Machines [self study]

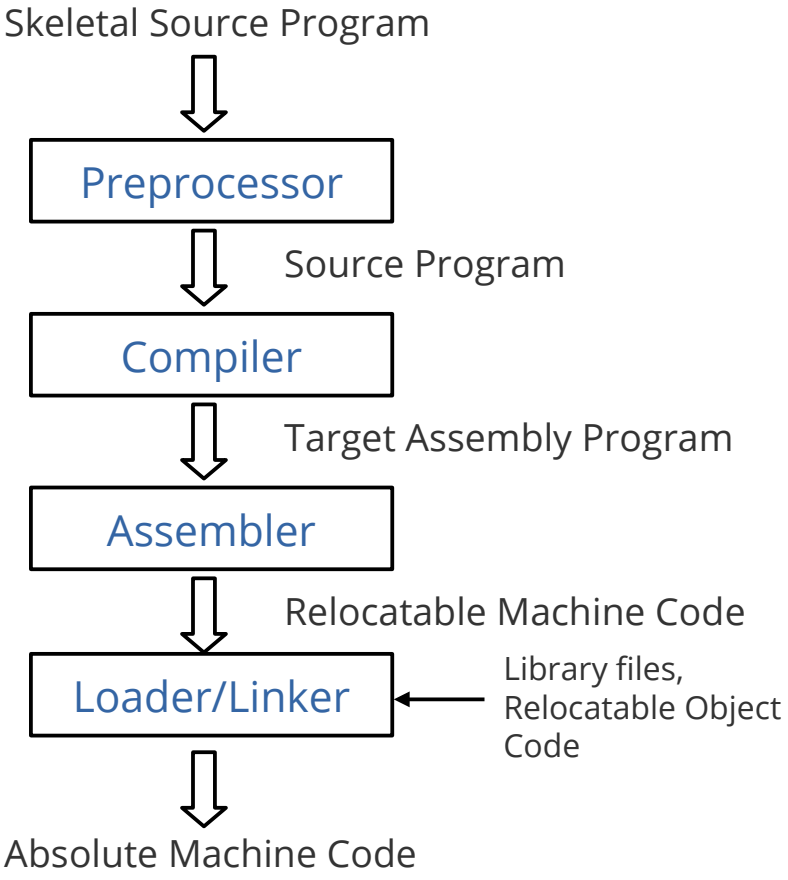
Introduction

- *System Software*:
  - Consists of a variety of programs that support the operation of a computer.
  - The software makes it possible for the user to focus on an application without needing to know the details of how the machine works internally.

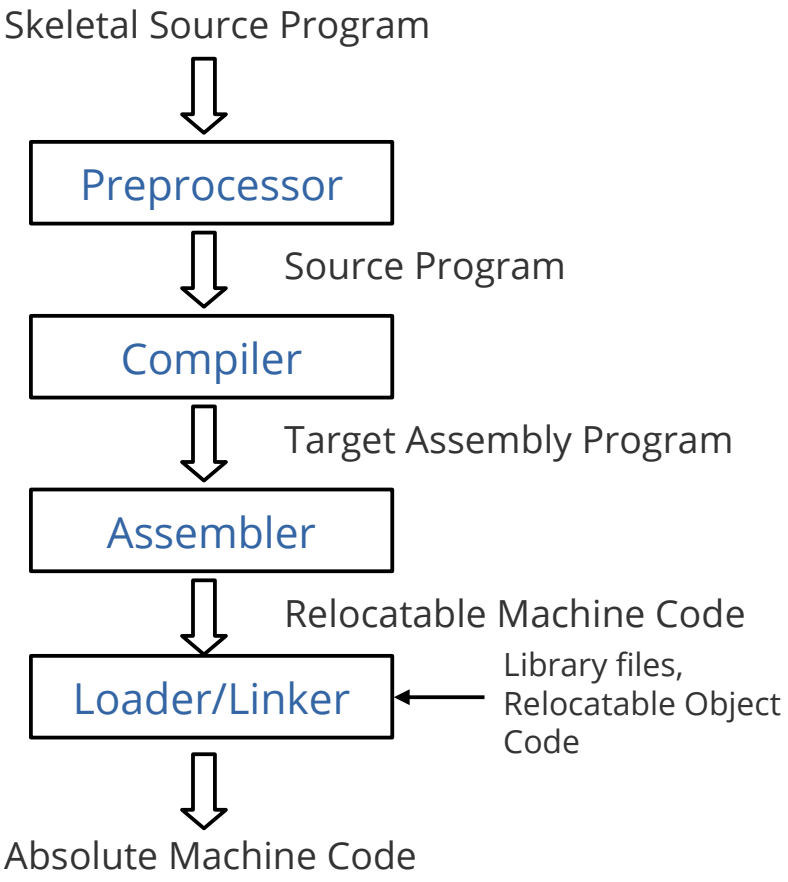
Introduction (2)



Introduction (3)



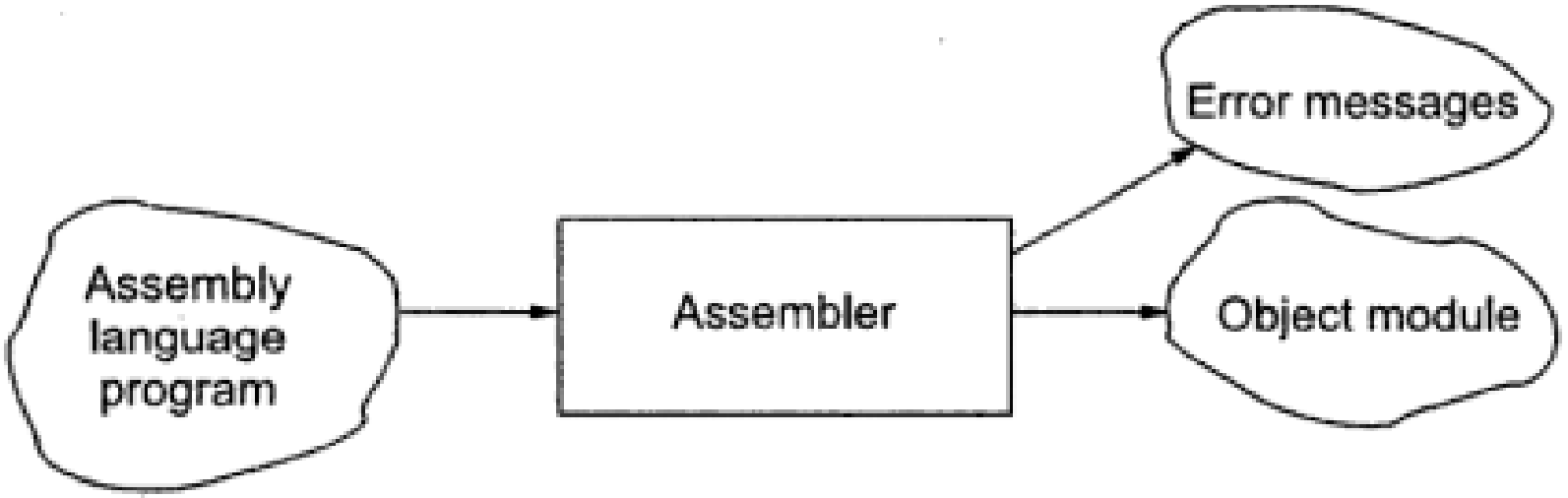
Introduction (4)



- *Different types of System Software*:
  - ***Text editor:*** create and modify the programs
  - ***Compiler:*** translate programs into machine language
  - ***Linker:*** performs the linking task
  - ***Loader:*** load machine language program into memory and prepares for execution
  - ***Assembler:*** translate assembly program into machine language
  - ***Macro processor:*** translate macros instructions into its definition
  - ***Debugger:*** detect errors in the program
  - ***OS:*** *You* control all the above by interacting with the operating system.

Introduction (5)

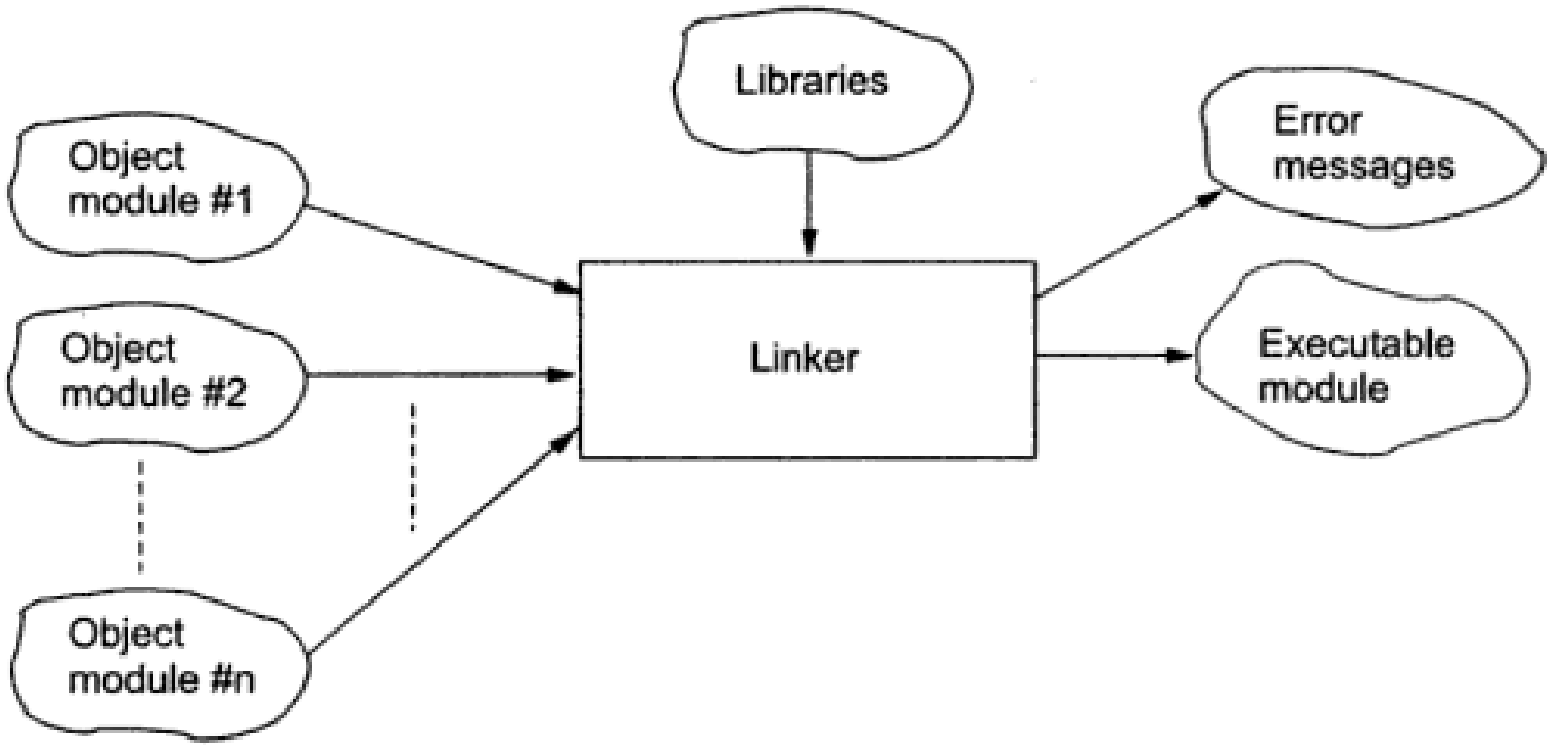
- *Input-Output of an Assembler*



[Source 3]

Introduction (6)

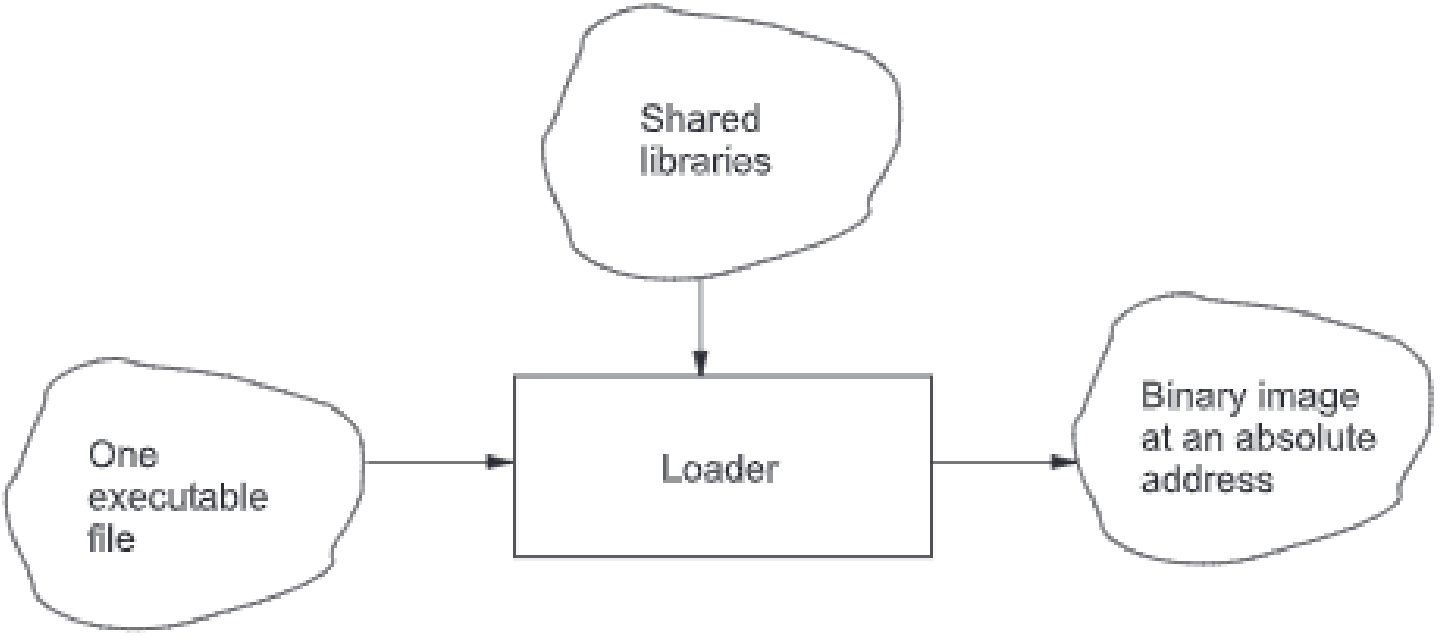
- *Input-Output of an Linker*



[Source 3]

Introduction (7)

- *Input-Output of an Loader*



[Source 3]

- *Program Development Flow ->*

[Source 2]

System Software and Machine Architecture

- One characteristic in which most system software differ from application software is *machine dependency.*
  - System programs are intended to support the operation and use of the computer itself, rather than any particular application.
  - Application programs are primary concerned with the solution of some problem, using the computer as a tool.
- *Example:*
  - Assembler translates mnemonic instructions into machine code.
    - *instruction formats, addressing modes, etc..*
  - Compilers must generate machine language code.
    - *number and type of registers, machine instructions available, etc..*
  - OS is directly concerned with the management of nearly all of the resources of a computing system.

System Software and Machine Architecture (2)

- Important machine structures to the design of system software:
  - Memory Structure
  - Registers
  - Data Formats
  - Instruction Formats
  - Addressing Modes
  - Instruction set

System Software and Machine Architecture (3)

- *Some aspects of system software* that do not directly depend upon the machine architecture are:
  - The general design and logic of an *assembler* is basically the same on most computers.
  - Some of the code optimization techniques used by *compilers* are independent of the target machine.
  - The process of linking together independently assembled subprograms do not usually depend on the computer being used.

System Software and Machine Architecture (4)

- Simplified Instructional Computer (SIC)
  - SIC is a hypothetical computer that includes the hardware features most often found on real machines, while avoiding unusual or irrelevant complexities.

- *While understanding any system software, we should identify:*
  - Features that are fundamental
  - Features that are architecture dependent
  - Extended features that are relatively machine independent
  - Major design options for structuring the software
  - Optional features

Simplified Instructional Computer (SIC)

- SIC is a hypothetical computer that includes the hardware features most often found on real machines, while avoiding unusual or irrelevant complexities.

- *SIC comes in two version:*
  1. **The Standard Model**
  2. **XE version (Extra Equipment)**
     - ”extra equipment” , “extra expensive”

- These two versions has been designed to be ***upward compatible***
  - An object program for the standard **SIC** will also execute properly on a **SIC/XE** system

SIC Machine Architecture

- ***Memory***
  - 1 byte = 8-bit
  - 1 word=3 consecutive bytes
    - Addressed by the location of their lowest numbered byte
  - Total 32,768 (2^{15}) bytes, **32KB** of memory.
  - Memory is *byte addressable*

SIC Machine Architecture (2)

- ***Registers***
  - Five Registers, all of which have a special use.
  - Each register is 24 bits in length.

Mnemonic	*Number*	*Special Use*
A	0	Accumulator
X	1	Index register
L	2	Linkage register(JSUB)
PC	8	Program counter
SW	9	Status word(Condition Code)

SIC Machine Architecture (3)

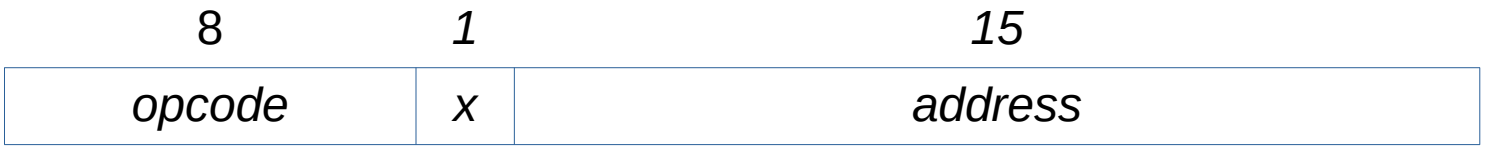
- ***Status Word register contents***

Bit positions	***Field name***	***Use***
0	MODE	**0**=user mode, **1**=supervisor mode
1	IDLE	**0**=running, **1**=idle
2-5	ID	Process identifier
6-7	CC	Condition code
8-11	MASK	Interrupt mask
12-15		Unused
16-23	ICODE	Interruption code

SIC Machine Architecture (4)

- ***Data Formats***
  - **Integers**: stored as 24-bit binary numbers;
    - 2's complement representation is used for negative values
  - **Characters**: stored as 8-bit ASCII codes
  - No floating-point hardware

- ***Instruction Formats***
  - 24-bit format
  - The flag bit **x** is used to indicate ***indexed-addressing mode***



SIC Machine Architecture (5)

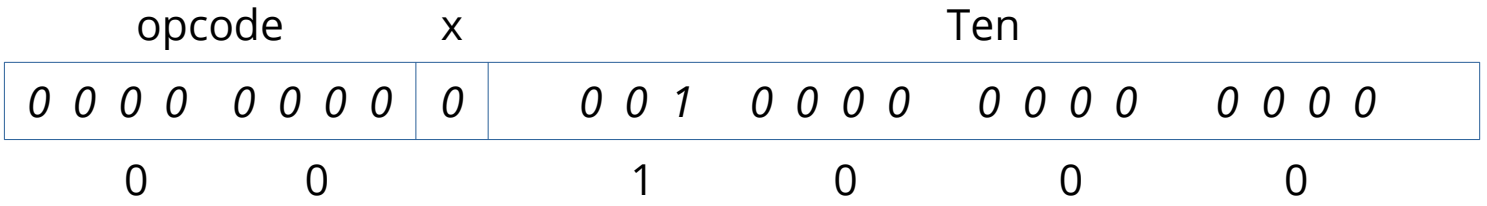
- ***Addressing Modes***
  - There are two addressing modes available
    - indicated by the setting of x bit in the instruction

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

- (X) represents the contents of a register or a memory location

SIC Machine Architecture (6)

- ***Addressing Modes*** (Direct)
  - Example: LDA TEN

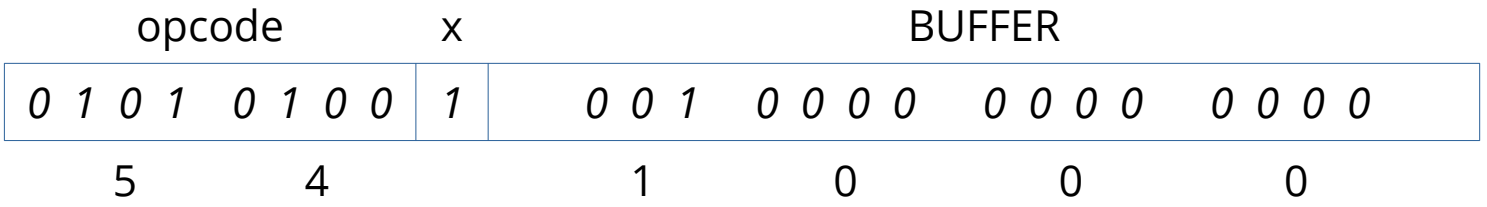


- Effective Address (EA) = 1000

- Contents of the address 1000

SIC Machine Architecture (7)

- ***Addressing Modes*** (Indexed)
  - Example: STCH BUFFER, X



- Effective Address (EA) = 1000 + [X]
  = 1000 + contents of the index register X

- The Accumulator content, the character is loaded to the Effective address.

SIC Machine Architecture (8)

- ***Instruction Set*** (*SIC provides a basic set of instructions sufficient for simple tasks*)
  - load and store instructions: **LDA, LDX, STA, STX**, etc.
  - integer arithmetic operations: **ADD, SUB, MUL, DIV,** etc.
    - All arithmetic operations involve a *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 in SW (Status Word) to indicate the result (**<,=,or >**)
  - 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

SIC Machine Architecture (9)

- ***Input and Output***
  - Input and output are performed by transferring 1 byte at a time to/from the rightmost 8 bits of register A.
  - Each device is assigned a unique 8-bit code.
  - ***Three I/O instructions:***
    1. Test Device (TD)
       - Tests whether the addressed device is ready to *send or receive* a byte of data
       - *Condition code* is set to indicate the result (<: *ready*, =: *not ready*)
    2. Read Data (RD)
    3. Write Data (WD)

SIC/XE Machine Architecture

- ***Memory***
  - Almost the same as that previously described for SIC.
  - However, **1 MB** (2^{20} bytes) maximum memory available.
- ***Registers***
  - More additional registers are provided by SIC/XE

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)

SIC/XE Machine Architecture (2)

- Data Formats
  - The same data format as the standard version
  - However, provide an addition 48-bit floating-point data type
    - fraction: 0~1
    - exponent: 0~2047
    - sign: 0=positive, 1=negative

$$\textbf{Value} = (-1)^{S}\ 0.f * 2^{(exp-1024)}$$

[Source 1 & 4]

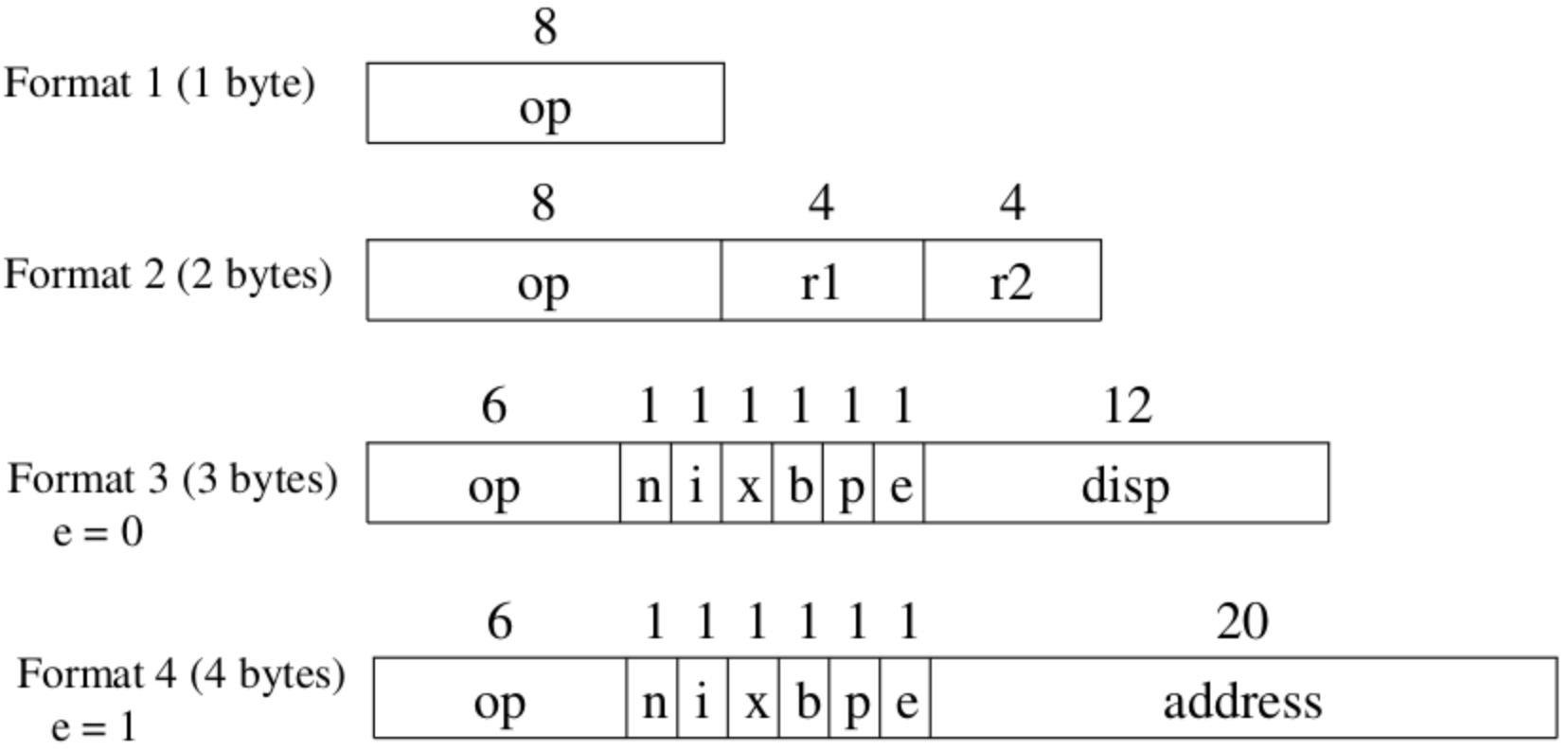
SIC/XE Machine Architecture (3)

- Instruction Formats
  - Larger memory means an address cannot fit into a 15-bit field
  - Extend addressing capacity
    - Use some form of *relative addressing* -> instruction format 3
    - Extend the address field to *20 bits* -> instruction format 4
  - Additional instructions do not reference memory
    - Instruction format 1 & 2

[Source 1 & 4]

SIC/XE Machine Architecture (4)

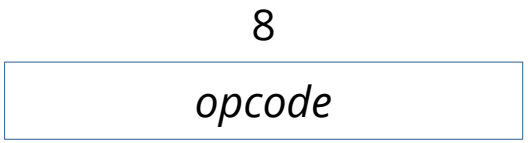
- Instruction Formats (Cont.)



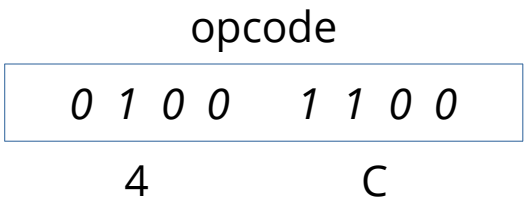
[Source 1 & 4]

SIC/XE Machine Architecture (4)

- Format 1 (1 byte)



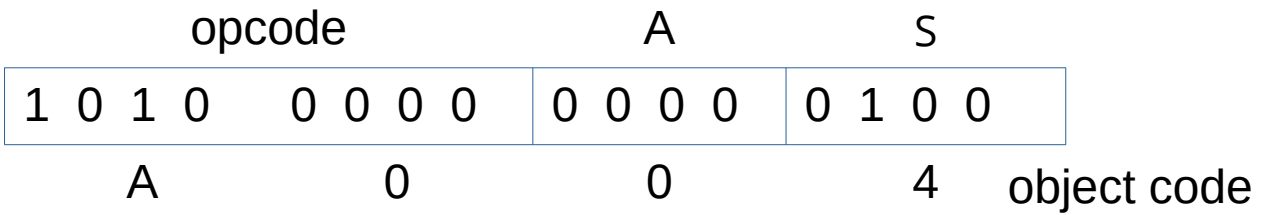
- Example : RSUB (Return to subroutine)



SIC/XE Machine Architecture (4)

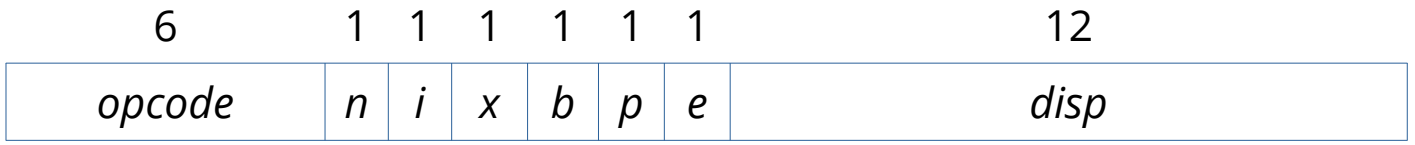
- Format 2 (2 bytes)

- Example : COMPR A, S (Compare the contents of register A and S)

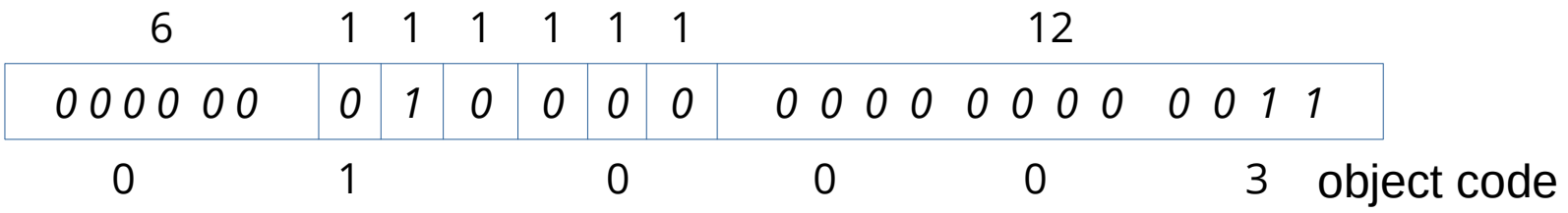


SIC/XE Machine Architecture (4)

- Format 3 (3 bytes)



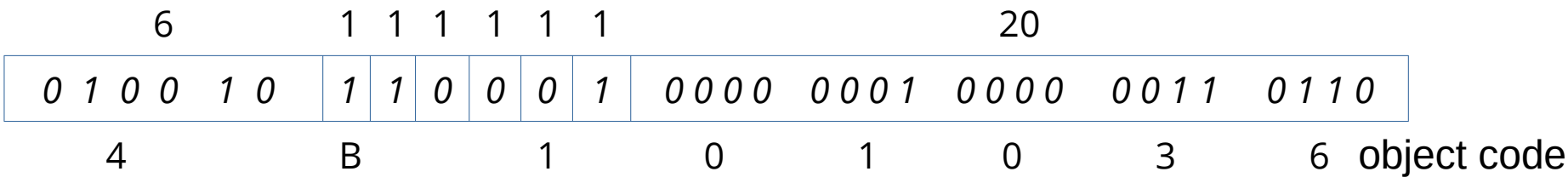
- Example : LDA #3 (Load 3 to Accumlator)



SIC/XE Machine Architecture (4)

- Format 4 (4 bytes)

- Example : +JSUB RDREC (Jump to the address, 1036)



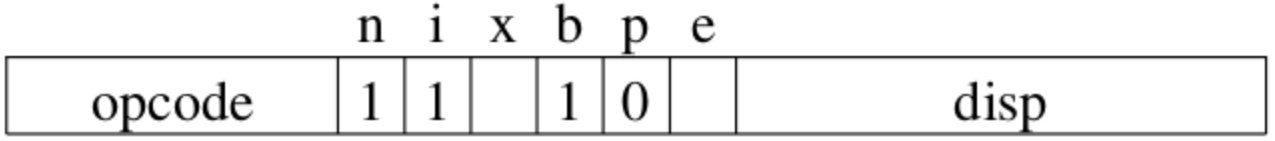
SIC/XE Machine Architecture (5)

- Addressing Modes
  - ***Base relative addressing*** **- format 3 only**
    - n = 1, i = 1, ***b=1, p=0***
  - ***Program-counter relative addressing*** **- format 3 only**
    - n = 1, i = 1, ***b=0, p=1***
  - ***Direct addressing*** **– format 3 and 4**
    - n = 1, i = 1, ***b=0, p=0***
  - ***Indexed addressing*** **– format 3 and 4**
    - n = 1, i = 1, ***x = 1*** or n = 0, i = 0, ***x = 1***
  - ***Immediate addressing*** **– format 3 and 4**
    - **n = 0, i = 1,** x = 0 // cannot combine with *indexed*
  - ***Indirect addressing*** **– format 3 and 4**
    - **n = 1, i = 0,** x = 0 // cannot combine with *indexed*
  - ***Simple addressing*** **– format 3 and 4**
    - **n = 0, i = 0 or n = 1, i = 1**

[Source 1 & 4]

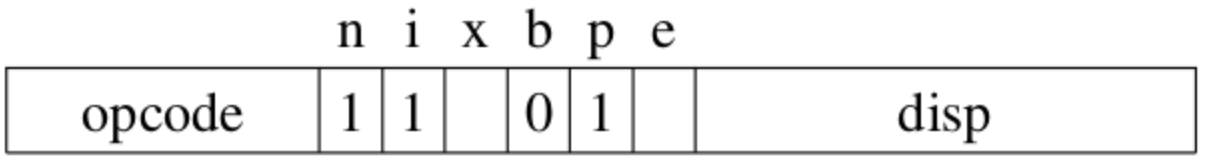
SIC/XE Machine Architecture (6)

- ***Base Relative Addressing***



n=1, i=1, ***b=1, p=0***, TA=(**B**)+disp (0≤disp ≤4095)

- ***Program-Counter Relative Addressing***

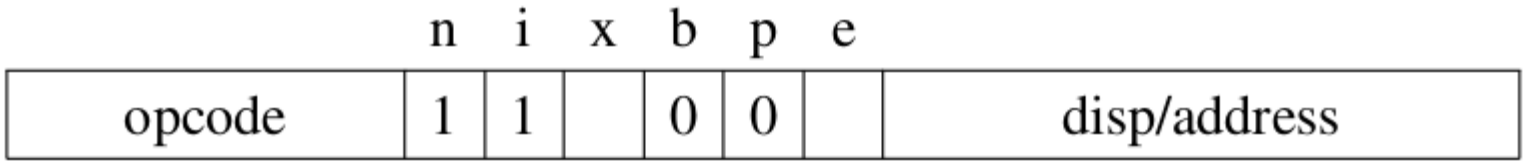


n=1, i=1, ***b=0, p=1***, TA=(**PC**)+disp (-2048≤disp ≤2047)

[Source 1 & 4]

SIC/XE Machine Architecture (7)

- ***Direct Addressing***
  - The target address is taken directly from the *disp* or *address* field



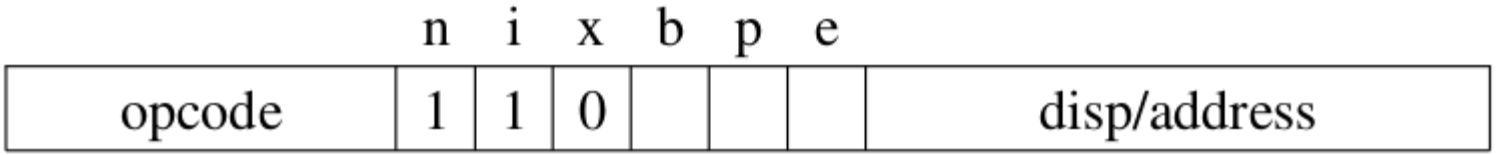
Format 3 (e=0): n=1, i=1, ***b=0, p=0***, TA=disp (0≤disp ≤4095)

Format 4 (e=1): n=1, i=1, ***b=0, p=0***, TA=address

[Source 1 & 4]

SIC/XE Machine Architecture (8)

- ***Indexed Addressing***
  - The term (X) is added into the target address calculation



n=1, i=1, ***x=1***

Ex. *Direct Indexed Addressing*

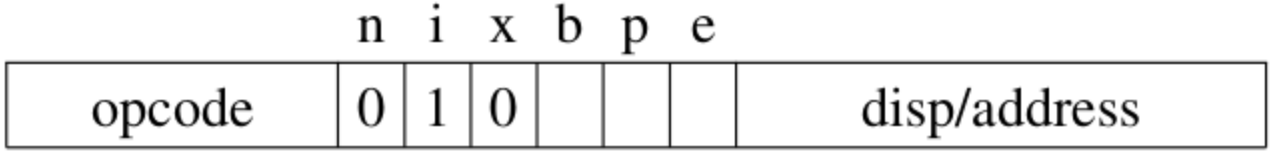
Format 3, TA=(X)+disp

Format 4, TA=(X)+address

[Source 1 & 4]

SIC/XE Machine Architecture (9)

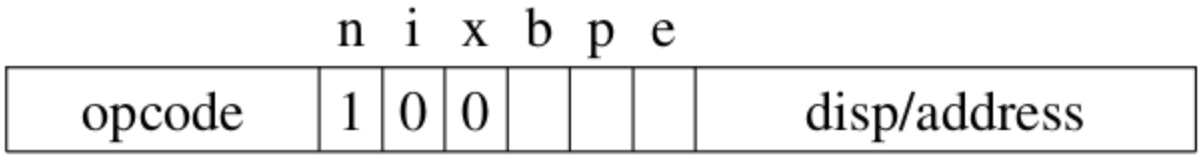
- ***Immediate Addressing – no memory access***



n=0, i=1, x=0, **operand**=disp //format 3

n=0, i=1, x=0, **operand**=address //format 4

- ***Indirect Addressing***



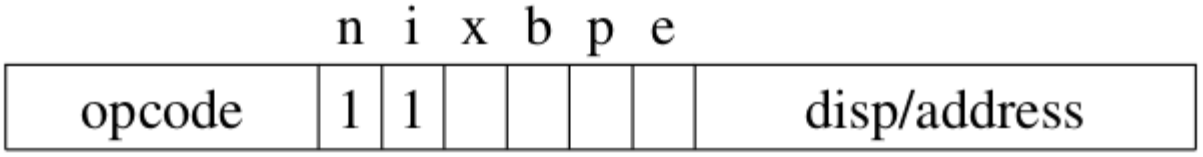
n=1, i=0, x=0, TA=(disp), **operand** = (TA) = ((disp))

n=1, i=0, x=0, TA=(address), **operand** = (TA) = ((address))

[Source 1 & 4]

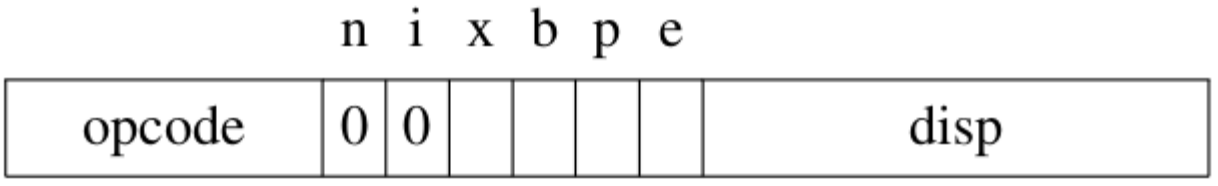
SIC/XE Machine Architecture (10)

- ***Simple Addressing Mode***



Format 3: ***i=1, n=1***, TA=disp, **operand** = (disp)

Format 4: ***i=1, n=1***, TA=address, **operand** = (address)



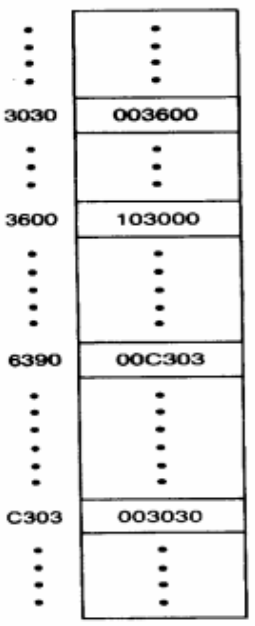
i=0, n=0, TA=b/p/e/disp (SIC standard)

[Source 1 & 4]

Addressing type	Flag bits n i x b p e	Assembler language notation	Calculation of target address TA	Operand	Notes
Simple	1 1 0 0 0 0	op c	disp	(TA)	D
	1 1 0 0 0 1	+op m	addr	(TA)	4 D
	1 1 0 0 1 0	op m	(PC) + disp	(TA)	A
	1 1 0 1 0 0	op m	(B) + disp	(TA)	A
	1 1 1 0 0 0	op c,X	disp + (X)	(TA)	D
	1 1 1 0 0 1	+op m,X	addr + (X)	(TA)	4 D
	1 1 1 0 1 0	op m,X	(PC) + disp + (X)	(TA)	A
	1 1 1 1 0 0	op m,X	(B) + disp + (X)	(TA)	A
	0 0 0 - - -	op m	b/p/e/disp	(TA)	D S
	0 0 1 - - -	op m,X	b/p/e/disp + (X)	(TA)	D S
Indirect	1 0 0 0 0 0	op @c	disp	((TA))	D
	1 0 0 0 0 1	+op @m	addr	((TA))	4 D
	1 0 0 0 1 0	op @m	(PC) + disp	((TA))	A
	1 0 0 1 0 0	op @m	(B) + disp	((TA))	A
Immediate	0 1 0 0 0 0	op #c	disp	TA	D
	0 1 0 0 0 1	+op #m	addr	TA	4 D
	0 1 0 0 1 0	op #m	(PC) + disp	TA	A
	0 1 0 1 0 0	op #m	(B) + disp	TA	A

[Source 1 & 4]

Example of SIC/XE instructions and addressing modes.



(a)

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

LDA instructions – student exercise

Machine instruction									Value loaded into register A
Hex	Binary							Target address	
	op	n	i	x	b	p	e	disp/address		
032600	000000	1	1	0	0	1	0	0110 0000 0000	3600	103000
03C300	000000	1	1	1	1	0	0	0011 0000 0000	6390	00C303
022030	000000	1	0	0	0	1	0	0000 0011 0000	3030	103000
010030	000000	0	1	0	0	0	0	0000 0011 0000	30	000030
003600	000000	0	0	0	0	1	1	0110 0000 0000	3600	103000
0310C303	000000	1	1	0	0	0	1	0000 1100 0011 0000 0011	C303	003030

(b)

[Source 1 & 4]

SIC/XE Machine Architecture (11)

- ***Instruction Set*** (*SIC/XE add the following new instructions in addition to SIC instructions*)
  - Load and store new instructions:
    LDB, STB, etc.
  - Floating-point arithmetic:
    ADDF, SUBF, MULF, DIVF
  - Register move:
    RMO
  - Register-to-register arithmetic:
    ADDR, SUBR, MULR, DIVR
  - Supervisor call instruction
    SVC - Generates an interrupt for communicating with OS

SIC/XE Machine Architecture (12)

- ***Input and Output***
  - The I/O instructions for SIC are also available on SIC/XE.
  - 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 operations.
  - The instructions SIO, TIO, and HIO are used to start, test, halt the operation of I/O chennels.

Traditional (CISC) Machines and RISC Machines

- ***CISC***
  - VAX Architecture
  - Pentium Pro Architecture
- ***RISC***
  - UltraSPARC Architecture
  - PowerPC Architecture
  - Cray T3E Architecture

Previous Year Questions?

1. Differentiate between System Software and Application Software 3M
2. List the format of 3 byte and 4 byte instructions available in SIC/XE machine 3M
3. List the various instruction formats of SIC/XE machine. 4M

References

Reference for this topic

- **Book:** System Software: An Introduction to System Programming, *Third Edition*, Leland L. Beck and D. Manjula, Pearson Education.
- **Book:** System Software, S. Chattopadhyaya (2011), PHI Learning.
- **Book:** Alfred V. Aho, Monica S. Lam, Ravi Sethi, J D Ullman, *Compilers: Principles, Techniques, and Tools*, 2nd Edition, Prentice Hall, 2006.
- **PPT:** Hsung-Pin Chang, Department of Computer Science, National Chung Hsing University, **Chapter 1: Background**