SWE4001 – System Programming
Module 1: An Overview of System Programming
Lesson 1 of 7: Introduction to System Software

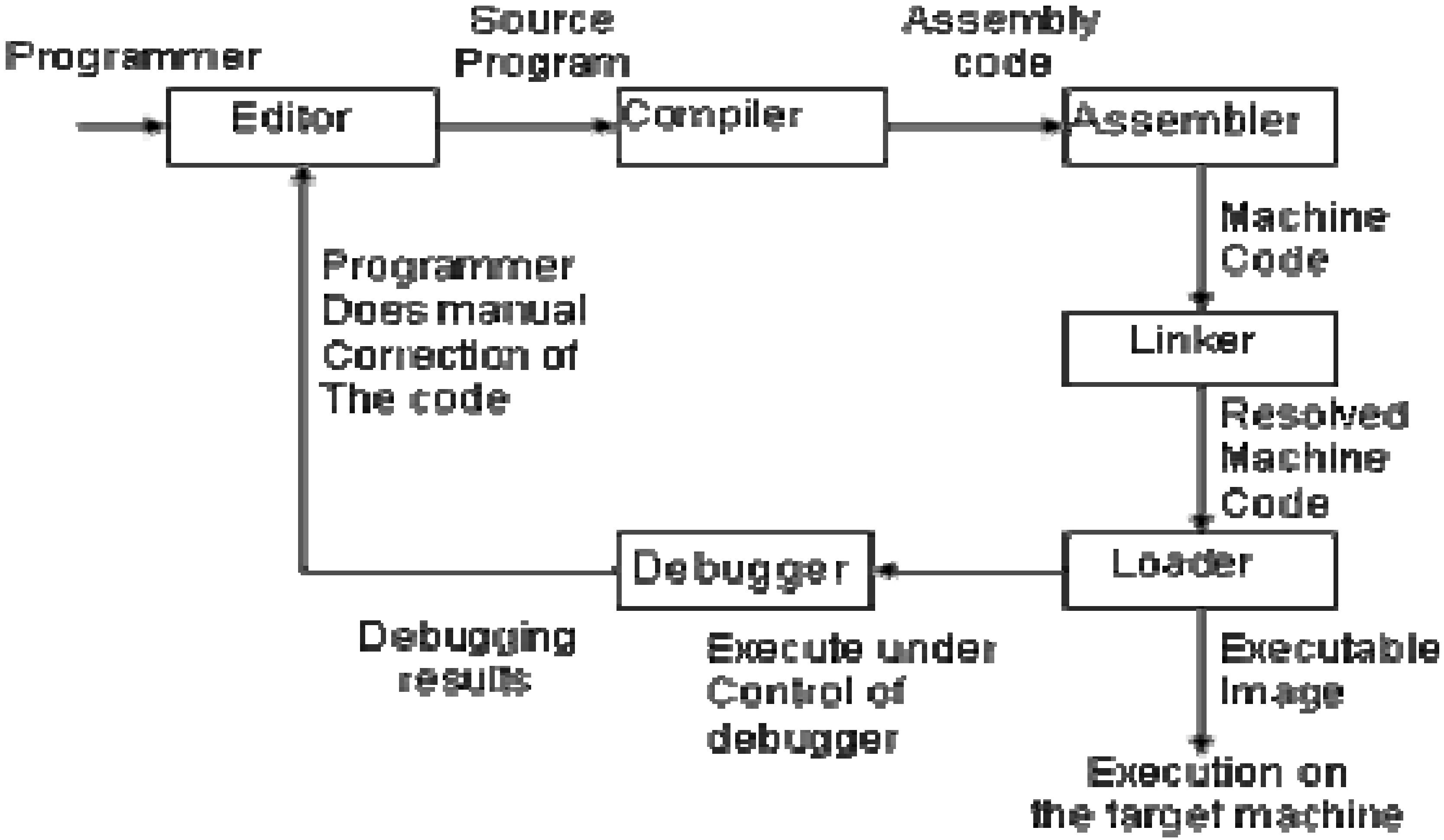
Introduction

- Definition
  - System software consists of a variety of programs that support the operation of a computer
  - One characteristic in which most differ from is machine dependency

Examples

- Text Editor
- Compiler
- Linker
- Loader
- Debugger
- Assembler
- Macro processor
- Operating system

Program Development

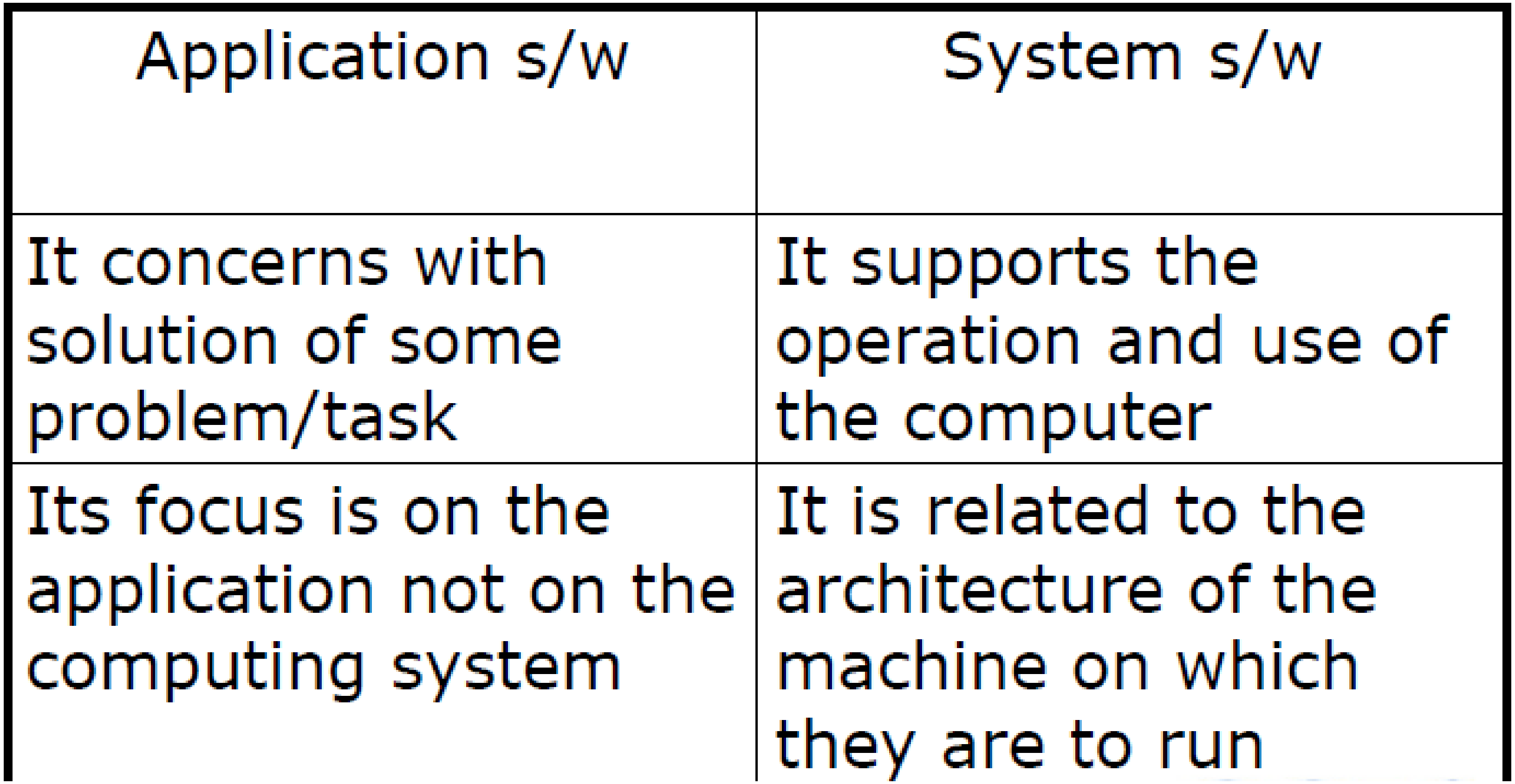


System Software vs. Machine Architecture

- Machine Dependent:
  - The important characteristic in which most system s/w differ from application s/w is machine dependency
  - e.g. assembler translate mnemonic instructions into machine code
  - e.g. compilers must generate machine language code

- Machine independent
  - There are aspects of system software that do not directly depend upon the type of computing system
  - e.g. general design and logic of an assembler
  - e.g. code optimization techniques

Difference between System s/w and Application s/w



Application s/w	System s/w
It concerns with solution of some problem/task	It supports the operation and use of the computer
Its focus is on the application not on the computing system	It is related to the architecture of the machine on which they are to run

Machine dependency

- Assembler

– Translates mnemonic instruction into machine code

– Involves different instruction set, instruction format, addressing modes, etc

– Varies from architecture to architecture

–Machine dependent

Machine dependency

- Compiler
  - Lexical analysis – separates source programs to tokens
  - Syntactic analysis – groups tokens to statements, expression, etc
  - Intermediate code generation – breaks the statements to simple three address instructions
  - Code optimization – usage of registers, using faster instructions – machine dependent
  - Code generation – write mnemonic equivalent of three address code – machine dependent

Machine dependency

- Operating system

–Directly concerns with the management of all the resources

–Machine dependent

- Loader

– Loads object program into memory

–Machine dependent

Machine independency

- Assembler

– General design and logic of assembly program is machine independent

- Compiler

– 1st three phases are machine independent

– Code optimization techniques like Dead code elimination, constant folding, common sub-expression elimination are machine independent

- Linker

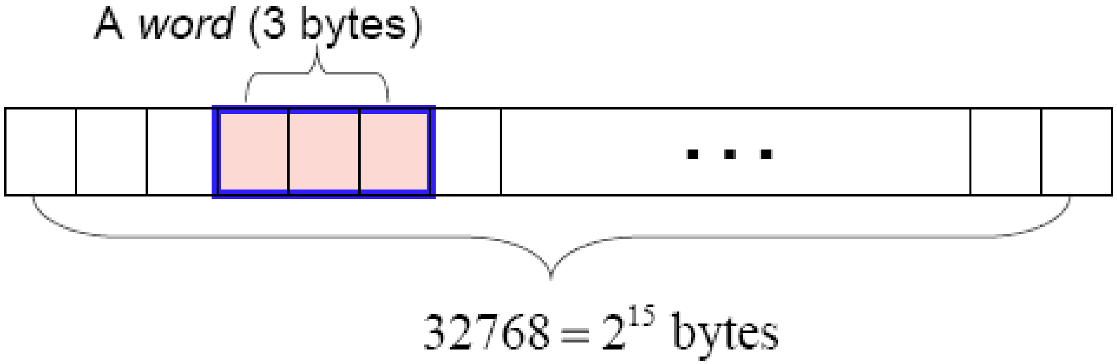
– Combines independently assembled programs into single module, machine independent

The Simplified Instructional Computer (SIC)

- SIC is a hypothetical computer that includes the hardware features most often found on real machines
- Two versions of SIC
  - standard model
  - XE version

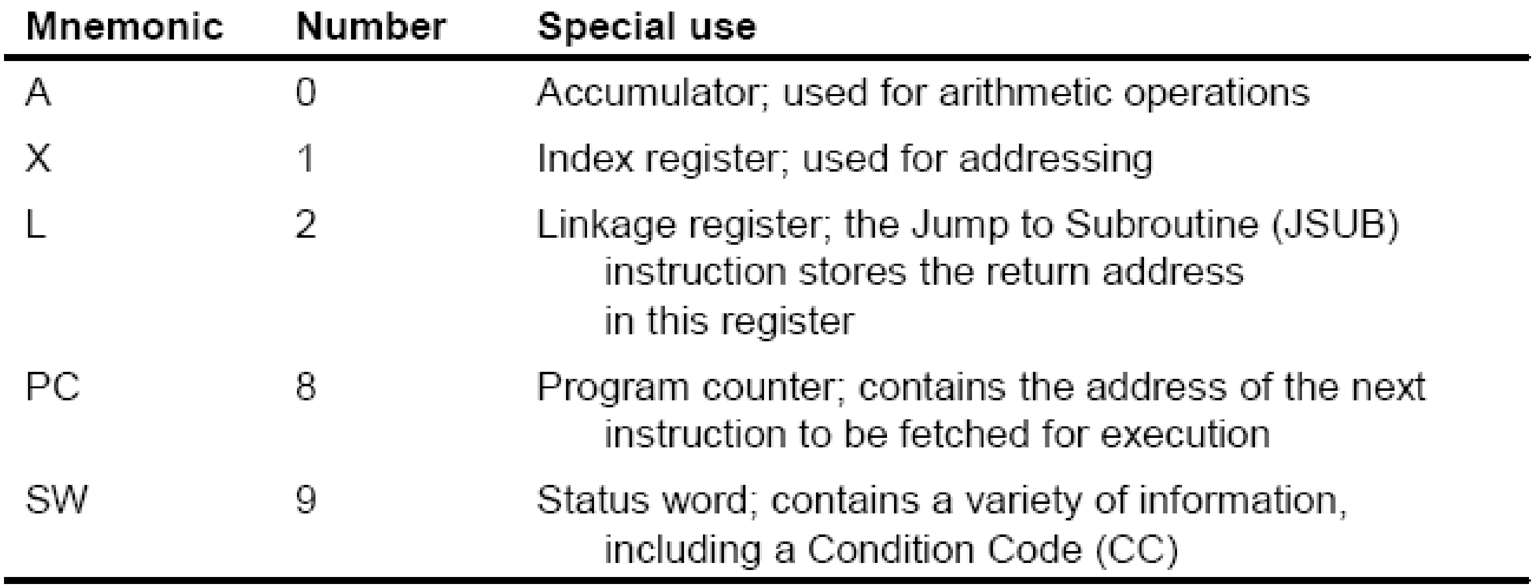
SIC Machine Architecture

- Memory
  - 8-bit bytes
  - 3 consecutive bytes form a word
  - 2^{15} bytes in the computer memory



SIC Machine Architecture

- Registers(5 register/each 24-bits)



Mnemonic	Number	Special use
A	0	Accumulator; used for arithmetic operations
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)

SIC Machine Architecture

- Data Formats
  - Characters:
    - 8-bit ASCII format
  - Integers:
    - 24-bit binary numbers;
    - 2’s complement for negative values
      - **$-N \Leftrightarrow 2^n - N$**
      - **e.g., if n = 4, $-1 \Leftrightarrow 2^4 - 1 = (1111)_2$.**
  - No floating-point hardware

SIC Machine Architecture

- Instruction Formats
  - 24 bit format

opcode (8)	x	address (15)

- Addressing Modes

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

SIC Machine Architecture

- Instruction Set
  - load and store: LDA, LDX, STA, STX, etc.
    - Ex: LDA ALPHA ⇔ (A) ← (ALPHA)

      STA ALPHA ⇔ (ALPHA) ← (A)
  - integer arithmetic operations: ADD, SUB, MUL, DIV, etc.
    - involves register A and a word in memory
    - Ex: ADD ALPHA ⇔ (A) ← (A) + (ALPHA)
  - comparison: COMP
    - involves register A and a word in memory
    - save result in the condition code (CC) of SW
    - Ex: COMP ALPHA ⇔ CC ←(<,=,>) of (A)?(ALPHA)

SIC Machine Architecture

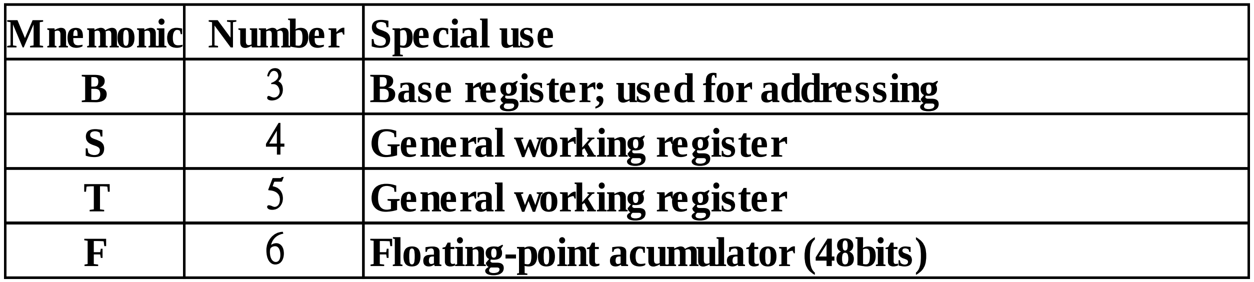
- Instruction Set
  - conditional jump instructions: JLT, JEQ, JGT
    - these instructions test the setting of CC and jump accordingly
  - subroutine linkage: JSUB, RSUB
    - JSUB jumps to the subroutine, placing the return address in register L
    - RSUB returns by jumping to the address contained in register L

SIC Machine Architecture

- 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
  - Three I/O instructions
    - The Test Device (TD) instruction
      - tests whether the addressed device is ready to send or receive a byte of data
      - CC : < : ready
      - CC : = : busy
    - Read Data (RD)
    - Write Data (WD)

SIC/XE Machine Architecture

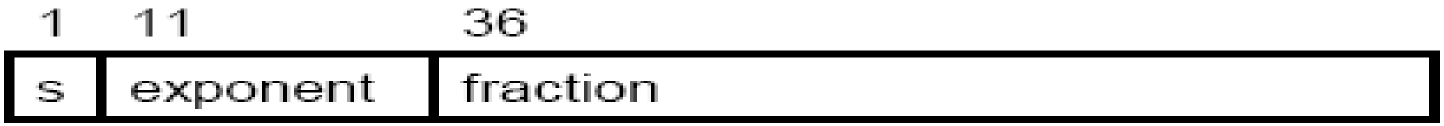
- Memory
  - 2^{20} bytes in the computer memory
- More Registers



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)

SIC/XE Machine Architecture

- Data formats
  - There is a 48-bit floating-point data type

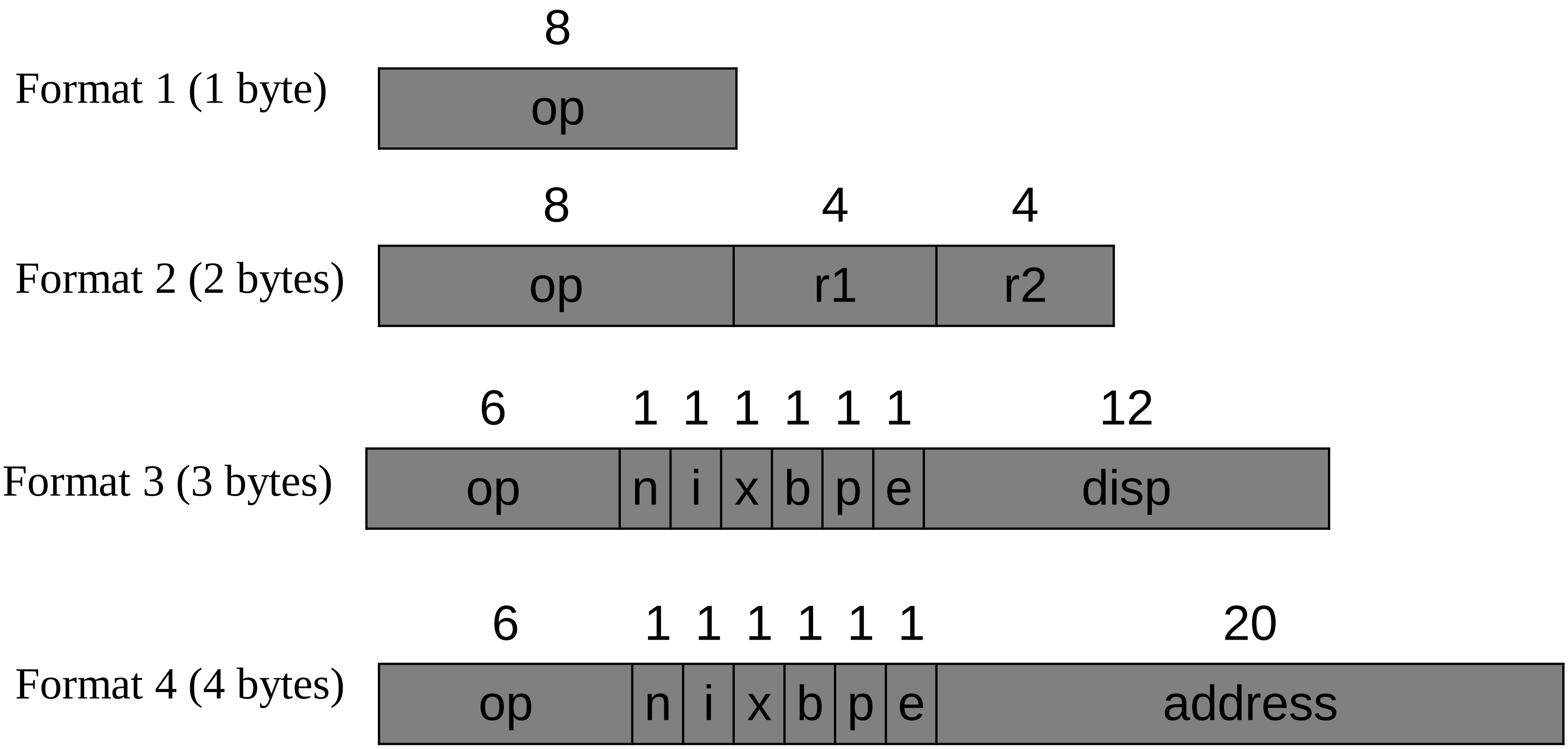


1	11	36
s	exponent	fraction

  - sign bit s (0: +, 1: -)
  - fraction f: a value between 0 and 1
  - exponent e: unsigned binary number between 0 and 2047
  - value: $s * f * 2^{(e-1024)}$
  - Ex: $5 = 2^2+2^0=(2^{-1}+2^{-3})*2^3= (2^{-1}+2^{-3})*2^{1027-1024}$
    - 0,10000000011,1010000….0

SIC/XE Machine Architecture

Instruction formats



Format 1 (1 byte)

8
op

Format 2 (2 bytes)

8	4	4
op	r1	r2

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

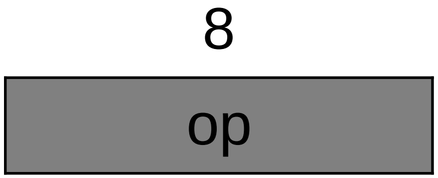
Formats 1 and 2 do not reference memory at all

Bit e distinguishes between format 3 and 4

SIC/XE Machine Architecture

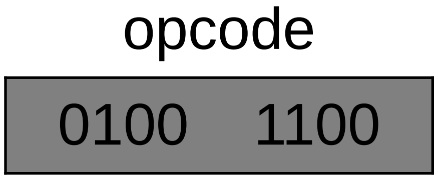
Instruction formats

Format 1 (1 byte)



8

op



opcode

0100 1100

RSUB 4 C

SIC/XE Machine Architecture

Instruction formats

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

COMPR A,S

Opcode	A	S
1010 0000	0000	0100
A 0	0	4

SIC/XE Machine Architecture

Base Relative Addressing Mode

	n	i	x	b	p	e	
opcode				1	0		disp

b=1, p=0, TA=(B)+disp (0≤disp ≤4095)

SIC/XE Machine Architecture

Program-Counter Relative Addressing Mode

	n	i	x	b	p	e	
opcode				0	1		disp

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

SIC/XE Machine Architecture

Direct Addressing Mode

	n	i	x	b	p	e	
opcode				0	0		disp

b=0, p=0, TA=disp (0≤disp ≤4095)

	n	i	x	b	p	e	
opcode			1	0	0		disp

b=0, p=0, TA=(X)+disp

(with index addressing mode)

SIC/XE Machine Architecture

Immediate Addressing Mode

	n	i	x	b	p	e	
opcode	0	1	0				disp

n=0, i=1, x=0, operand=disp

SIC/XE Machine Architecture

Indirect Addressing Mode

	n	i	x	b	p	e	
opcode	1	0	0				disp

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

SIC/XE Machine Architecture

Simple Addressing Mode

	n	i	x	b	p	e	
opcode	0	0					disp

i=0, n=0, TA=bpe+disp (SIC standard)

opcode+n+i = SIC standard opcode (8-bit)

	n	i	x	b	p	e	
opcode	1	1					disp

i=1, n=1, TA=disp (SIC/XE standard)

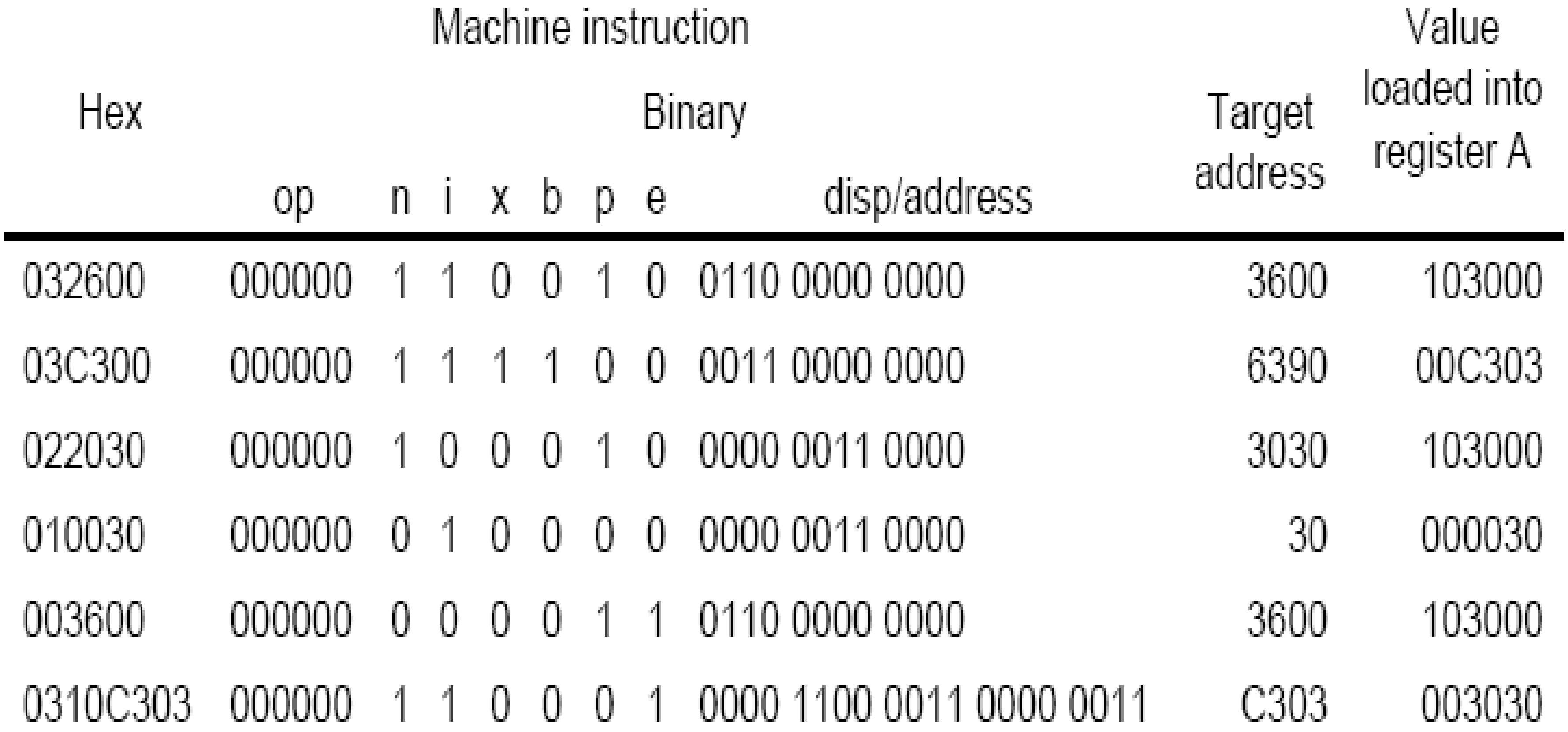
Addressing mode example

. . .	. . .
3030	003600
. . .	. . .
3600	103000
. . .	. . .
6390	00C303
. . .	. . .
C303	003030
. . .	. . .

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

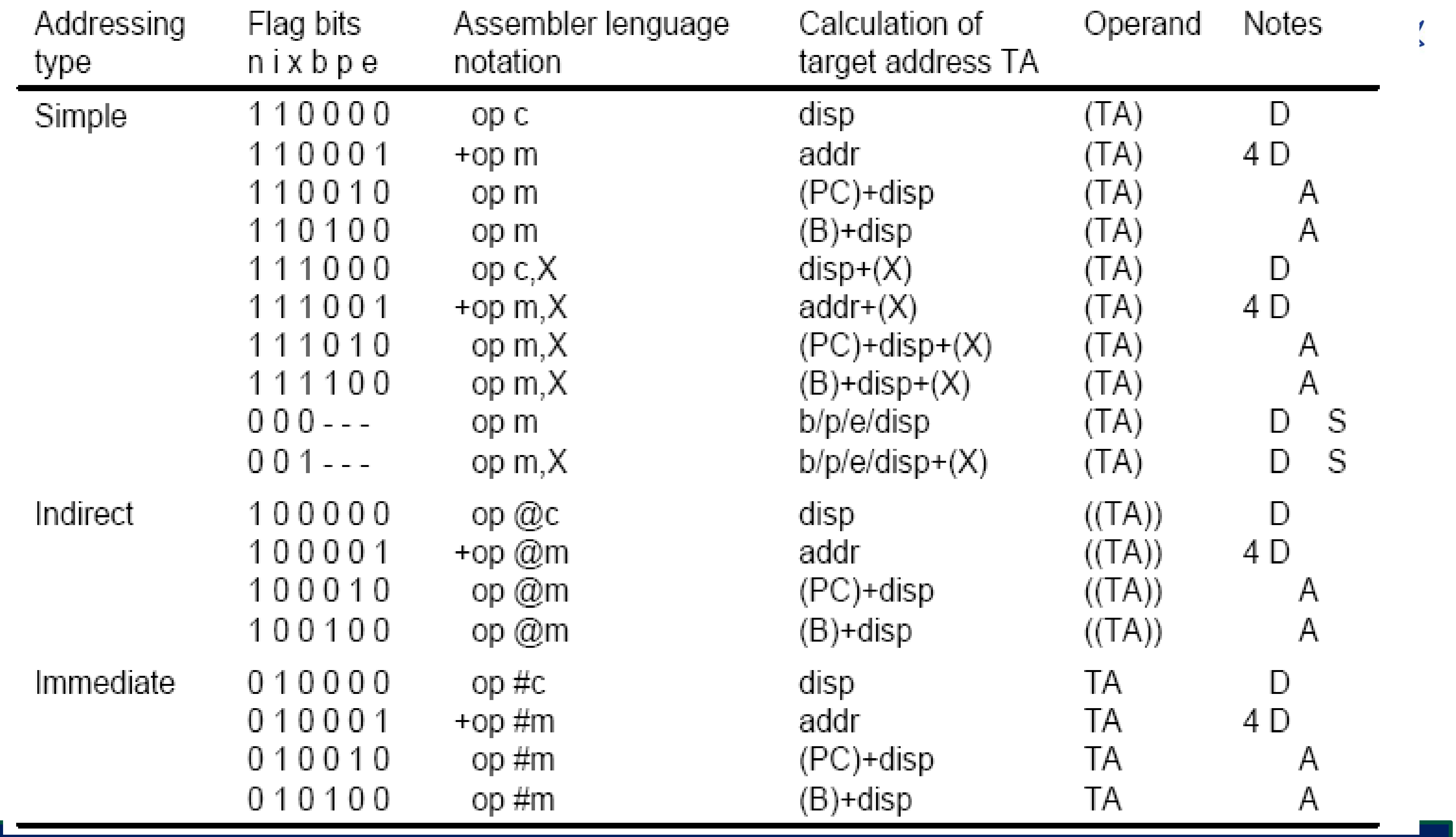
Hex
032600
03C300
022030
010030
003600
0310C303

Addressing mode example



Machine instruction										
Hex	Binary								Target address	Value loaded into register A
	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

Addressing mode summary



Addressing	Flag bits	Assembler lenguage	Calculation of	Operand	Notes
type	n i x b p e	notation	target address TA		
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

SIC/XE Machine Architecture

- Instruction Set
  - 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
  - SIO, TIO, HIO: start, test, halt the operation of I/O device