Chapter 1: Background

Hsung-Pin Chang

Department of Computer Science

National Chung Hsing University

Outline

1.1 Introduction

- Definition: ***System Software***
  - Consist of a variety of programs that support the operation of a computer
  - Make it possible for the user to focus on an application *without needing to know the details of how the machine works internally*

1.1 Introduction (Cont.)

- Example:
  - When you took the first <u>programming course</u>
    - 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 <u>assembler language</u>
    - 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

1.2 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
  - *Application programs* are primary concerned with the solution of some problem
- Example
  - Assembler translates mnemonic instructions into machine code
  - Compilers must generate machine language code
  - OS is directly concerned with the management of nearly all of the resources of a computing system

1.2 System Software and Machine Architecture (Cont.)

- Important machine structures to the design of system software
  - Memory structure
  - Registers
  - Data formats
  - Instruction formats
  - Addressing modes
  - Instruction set

1.2 System Software and Machine Architecture (Cont.)

- This textbook discusses system software basing on a *Simplified Instructional Computer (SIC)*
  - SIC is a hypothetical computer that includes the hardware features most often found on real machines
- Study 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

1.3 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 (extra equipment)

1.3 The Simplified Instructional Computer (SIC)

- SIC comes in two versions
  - The standard model
  - An XE version
    - "extra equipments", "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

1.3.1 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
  - Memory is ***byte addressable***

1.3.1 SIC Machine Architecture (Cont.)

- Registers
  - Five registers
  - Each register is 24 bits in length

Mnemonic	Number	Special use
A	0	Accumulator
X	1	Index register
L	2	Linkage register
PC	8	Program counter
SW	9	Status word

1.3.1 SIC Machine Architecture (Cont.)

- Status Word register contents

Bit position	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

1.3.1 SIC Machine Architecture (Cont.)

- 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 - standard version of SIC

8	1	15
opcode	x	address

The flag bit x is used to indicate ***indexed-addressing mode***

1.3.1 SIC Machine Architecture (Cont.)

- 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)

( ): the contents of a register or a memory location

1.3.1 SIC Machine Architecture (Cont.)

- Instruction Set
  - 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
  - Appendix A gives a complete list of all SIC instructions

1.3.1 SIC Machine Architecture (Cont.)

- Input and Output
  - Input and output are performed by transferring 1 byte at a time to/from the rightmost 8 bits of register A
  - Three I/O instructions:
    - 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)
    - Read Data (RD)
    - Write Data (WD)

1.3.2 SIC/XE Machine Architecture

- Memory
  - Almost the same as that previously described for SIC
  - However, 1 MB (2^{20} bytes) maximum memory available

1.3.2 SIC/XE Machine Architecture (Cont.)

- Registers
  - More 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)

1.3.2 SIC/XE Machine Architecture (Cont.)

- 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

1	11	36
s	exponent	fraction

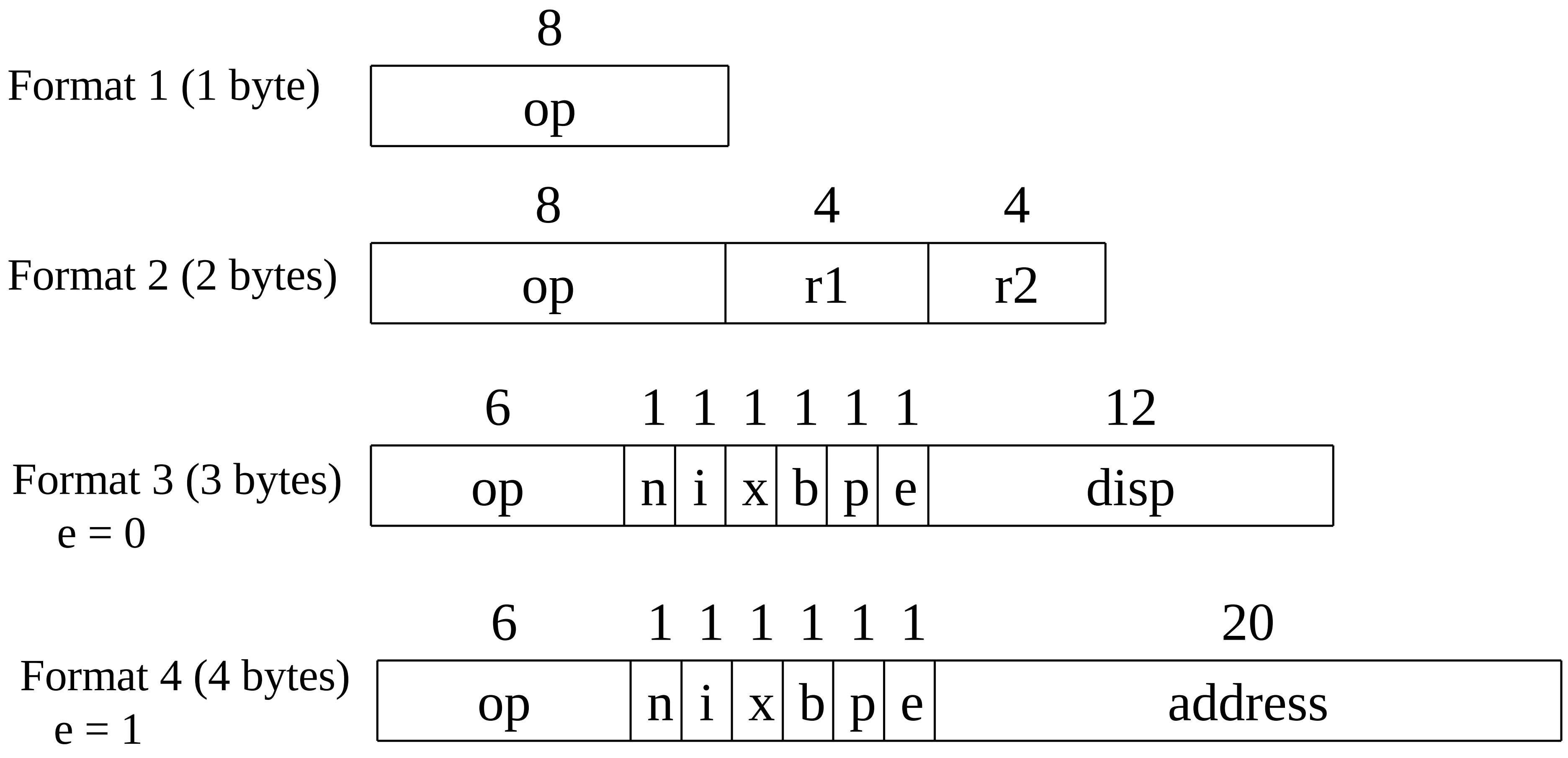
$$\textbf{Value} = (-1)^{S}\, 0.f * 2^{(exp-1024)}$$

1.3.2 SIC/XE Machine Architecture (Cont.)

- 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

1.3.2 SIC/XE Machine Architecture (Cont.)

- Instruction Formats (Cont.)



1.3.2 SIC/XE Machine Architecture (Cont.)

- 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**

1.3.2 SIC/XE Machine Architecture (Cont.)

- ***Base Relative Addressing***

	n	i	x	b	p	e	
opcode	1	1		1	0		disp

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

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

	n	i	x	b	p	e	
opcode	1	1		0	1		disp

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

1.3.2 SIC/XE Machine Architecture (Cont.)

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

	n	i	x	b	p	e	
opcode	1	1		0	0		disp/address

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

1.3.2 SIC/XE Machine Architecture (Cont.)

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

	n	i	x	b	p	e	
opcode	1	1	1				disp/address

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

Ex. *Direct Indexed Addressing*

Format 3, TA=(X)+disp

Format 4, TA=(X)+address

1.3.2 SIC/XE Machine Architecture (Cont.)

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

	n	i	x	b	p	e	
opcode	0	1	0				disp/address

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

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

- ***Indirect Addressing***

	n	i	x	b	p	e	
opcode	1	0	0				disp/address

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

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

1.3.2 SIC/XE Machine Architecture (Cont.)

- ***Simple Addressing Mode***

		n	i	x	b	p	e	
Case 1	opcode	1	1					disp/address

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

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

		n	i	x	b	p	e	
Case 2	opcode	0	0					disp

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

See the following slide

1.3.2 SIC/XE Machine Architecture (Cont.)

- Equal with a special hardware feature to provide the upward compatibility
  - If bits $n = i = 0$,
    - Neither immediate nor indirect
    - A special case of ***Simple Addressing***
    - Bits *b*, *p*, *e* are considered to be ***part*** *of the address field* of the instruction
    - Make Instruction Format 3 identical to the format used in SIC
    - Thus, provide the desired compatibility

1.3.2 SIC/XE Machine Architecture (Cont.)

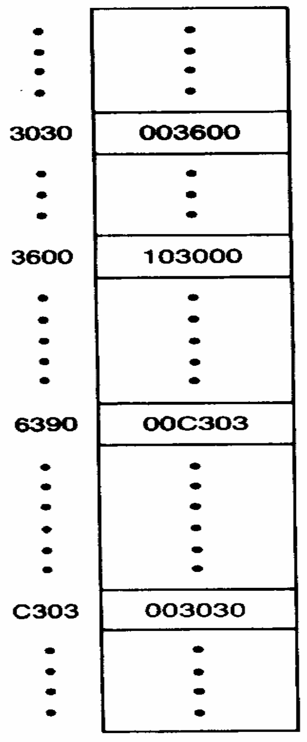
- Specify how to calculate the target address value
  - *Base relative addressing*
  - *Program-counter relative addressing*
  - *Direct addressing*
  - *Indexed addressing*
    - Combined with above any addressing mode
    - But cannot combined with immediate or indirect addressing modes

- Specify how the target address is used
  - *Immediate addressing*
  - *Indirect addressing*
  - *Simple addressing*

1.3.2 SIC/XE Machine Architecture (Cont.)

- SIC/XE
  - n=1, i=1
  - Except the *immediate (i=1)* and *indirect (n=1)* addressing
- SIC
  - n=0, i=0
  - Ignore the b, p, e fields and consider to be part of address field
- Format 4
  - b = 0, p = 0
  - Not allow *base relative* and *PC relative* addressing

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	



(a)

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

LDA instructions – student exercise

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

(b)

Figure 1.1 Examples of SIC/XE instructions and addressing modes.

1.3.2 SIC/XE Machine Architecture (Cont.)

- Instruction Set – *add the following new instructions*
  - load and store new registers
    - 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, often called ***software interrupt***
  - Appendix A gives a complete list of all SIC instructions

1.3.2 SIC/XE Machine Architecture (Cont.)

- Input/Output
  - There are I/O channels that can be used to perform input and output while the CPU is executing other instructions
    - Allow overlap of computing and I/O
  - SIO, TIO, HIO: start, test, halt the operation of I/O device

1.3.3 SIC Programming Examples

- Constructions of assembly language program
  - Instruction: Label mnemonic operand
    - Operand
      - **Direct addressing**   A
        - E.g. LDA   ZERO
      - **Immediate addressing**   #A
        - E.g. LDA #0
      - **Indexed addressing**   A, X
        - E.g. STCH BUFFER, X
      - **Indirect addressing**   @A
        - E.g J @RETADR

1.3.3 SIC Programming Examples (Cont.)

- Constructions of assembly language program (Cont.)
  - Data

```
Label   BYTE    value
Label   WORD    value
Label   RESB    value
Label   RESW    value
```

- E.g. EOF   BYTE   C'EOF'
- E.g. FIVE WORD   5

name of operand

integer, character

1.3.3 SIC Programming Examples (Cont.)

- Data movement [fig 1.2]
  - [fig 1.2 (a)] SIC assembler language programming
  - No *memory-to-memory* move instructions
    - ***All data movement must be done using registers***
  - Move 3-byte word: LDA, STA, LDL, STL, LDX, STX
    - By means of register A, L, or X
  - Move 1-byte: LDCH, STCH
  - Define storage for data items
    - **WORD, RESW // WORD: initialized, RESW: uninitialized**
    - **BYTE, RESB // BYTE: initialized, RESB: uninitialized**

1.3.3 SIC Programming Examples (Cont.)

- Data movement [fig 1.2] (Cont.)
  - [fig 1.2 (b)] **SIC/XE** versions of fig. 1.2 (a)
  - Make use of the ***immediate addressing*** of SIC/XE
    - E.g. LDA #5

```
        LDA     FIVE        LOAD CONSTANT 5 INTO REGISTER A
        STA     ALPHA       STORE IN ALPHA
        LDCH    CHARZ       LOAD CHARACTER 'Z' INTO REGISTER A
        STCH    C1          STORE IN CHARACTER VARIABLE C1
         .
         .
         .
ALPHA   RESW    1           ONE-WORD VARIABLE
FIVE    WORD    5           ONE-WORD CONSTANT
CHARZ   BYTE    C'Z'        ONE-BYTE CONSTANT
C1      RESB    1           ONE-BYTE VARIABLE
```

(a)

```
        LDA     #5          LOAD VALUE 5 INTO REGISTER A
        STA     ALPHA       STORE IN ALPHA
        LDA     #90         LOAD ASCII CODE FOR 'Z' INTO REG A
        STCH    C1          STORE IN CHARACTER VARIABLE C1
         .
         .
         .
ALPHA   RESW    1           ONE-WORD VARIABLE
C1      RESB    1           ONE-BYTE VARIABLE
```

(b)

Figure 1.2 Sample data movement operations for (a) SIC and (b) SIC/XE.

1.5 RISC Machines

- RISC (Reduced Instruction Set Computers)
  - Simplify the design of processors
  - Faster and less expensive processor development
  - Greater reliability
  - Faster instruction execution times

1.4 Traditional (CISC) Machines

- Complex Instruction Set Computers (CISC)
  - Relatively large and complicated instruction set
  - Several different instruction formats and lengths
  - Many different addressing modes
  - e.g. VAX or PDP-11 from DEC
  - e.g. Intel x86 family

- Reduced Instruction Set Computer (RISC)

1.5 RISC Machines

- RISC (Reduced Instruction Set Computers)
  - Simplify the design of processors
  - Faster and less expensive processor development
  - Greater reliability
  - Faster instruction execution times

1.5 RISC Machines (Cont.)

- Features
  - Standard, fixed instruction format
  - Single-cycle execution of most instructions
  - Memory access is done by ***load and store instructions only***
  - All instructions are *register-to-register* operations
    - Except load and store
  - A small number of machine instructions and instruction format,
  - A large number of general-purpose registers
  - A small number of addressing modes

1.5 RISC Machines (Cont.)

- Examples
  - Sun SPARC family
  - IBM PowerPC series
  - ARM corp. ARM family