Department of Electrical Engineering, Feng-Chia University

Chapter 1: Background

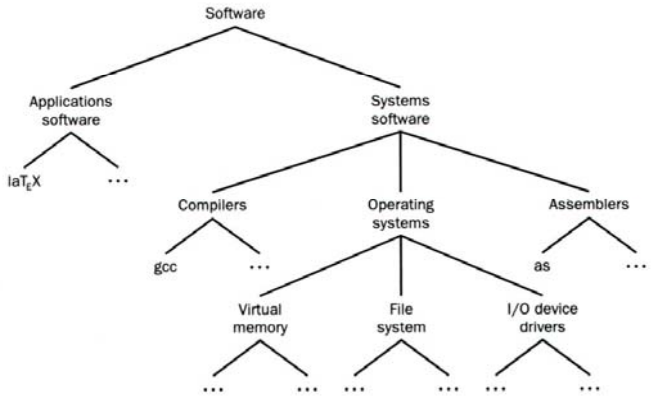
王振傑 (Chen-Chieh Wang)
ccwang@mail.ee.ncku.edu.tw

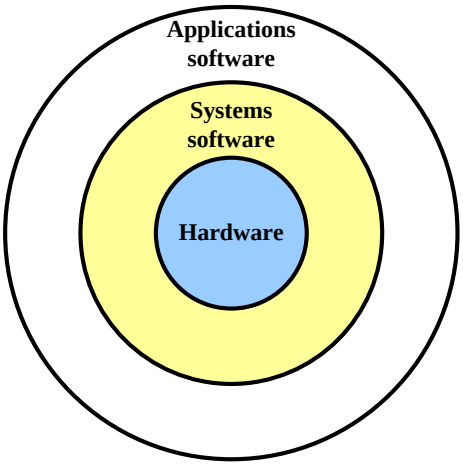
Department of Electrical Engineering, Feng-Chia University

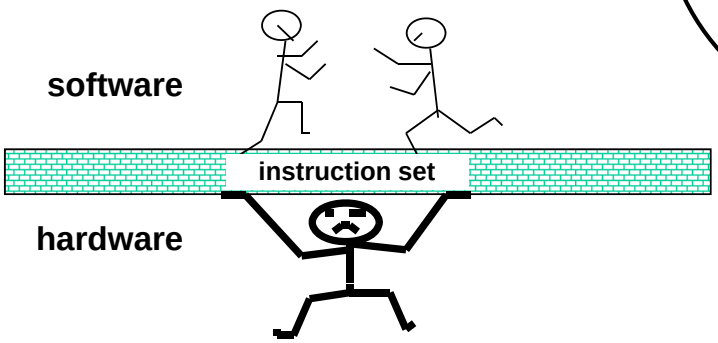
Outline

1.1 Introduction
1.2 System Software and Machine Architecture
1.3 The Simplified Instructional Computer (SIC)
- 1.3.1 SIC Machine Architecture
- 1.3.2 SIC/XE Machine Architecture
- 1.3.3 SIC Programming Examples

Abstraction of layers

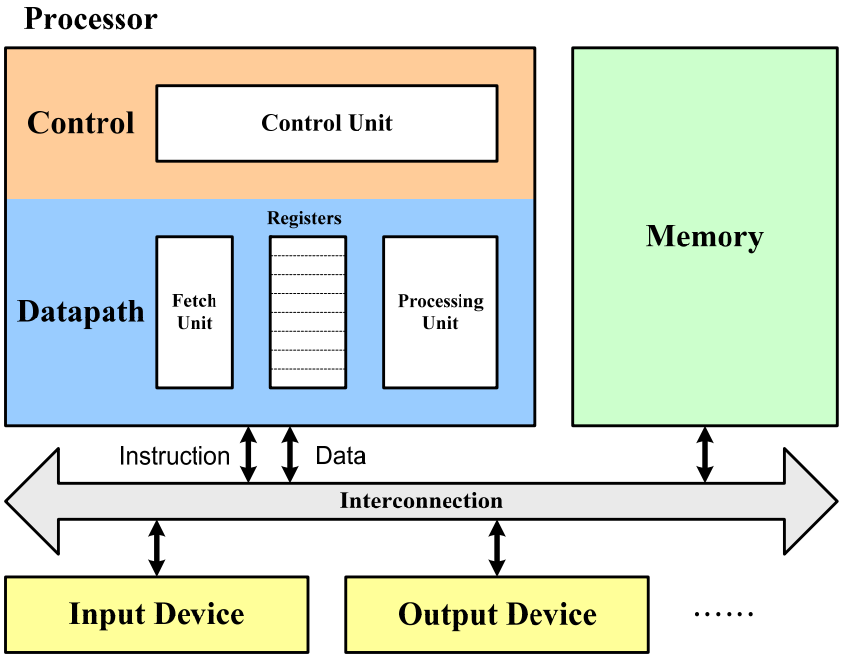






Department of Electrical Engineering, Feng-Chia University

Hardware



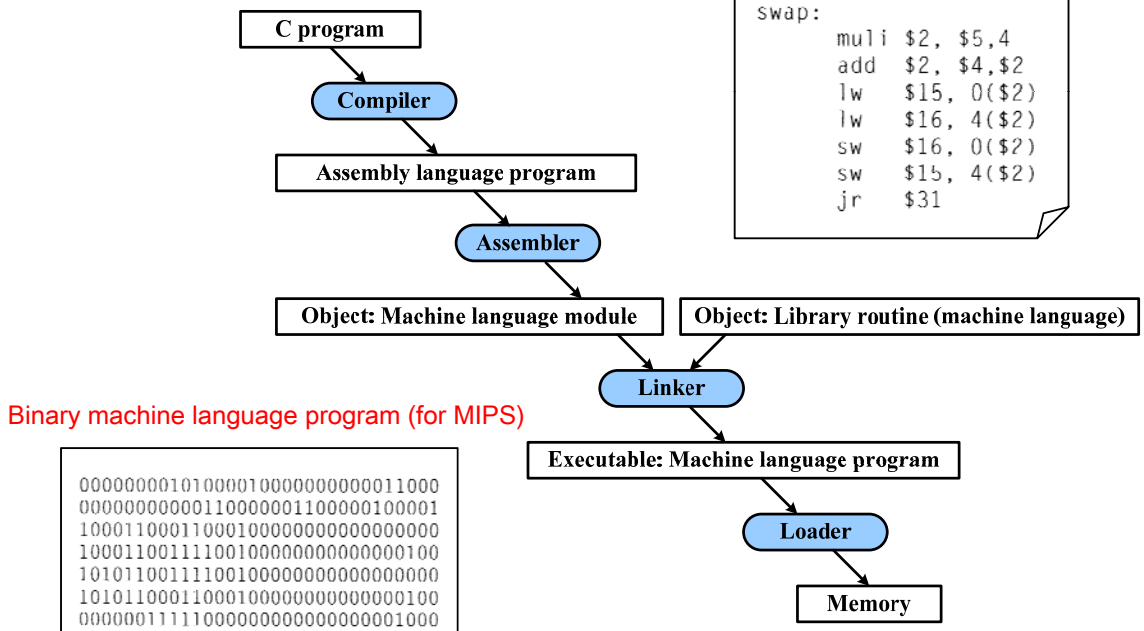
High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

An abstraction omits unneeded detail, helps us cope with complexity

Assembly language program (for MIPS)

```
swap:
        muli $2, $5,4
        add  $2, $4,$2
        lw   $15, 0($2)
        lw   $16, 4($2)
        sw   $16, 0($2)
        sw   $15, 4($2)
        jr   $31
```



Binary machine language program (for MIPS)

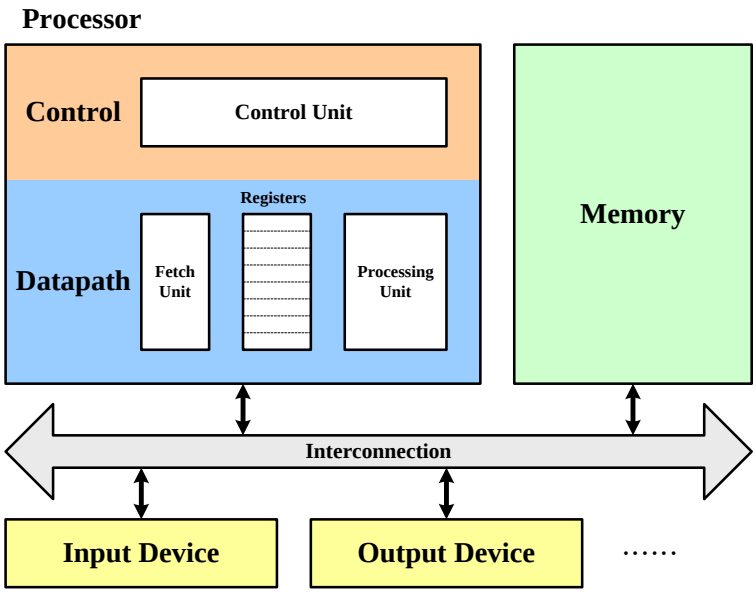
```
00000000101000010000000000011000
00000000000110000011000001000001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```


Software

- **Source Program**
  - Any sequence of statements and/or declarations written in some human-readable computer programming language (e.g. C++, Assembly program)
  - Usually created using a text editor (ASCII file)
- **Object Program**
  - Produced from a source program by compiling/assembling to intermediate machine code
  - Also contain data for use by the code at runtime, relocation information, program symbols for linking and/or debugging purposes, and other debugging information
- **Executable Program**
  - Machine code directly executed by a computer's CPU

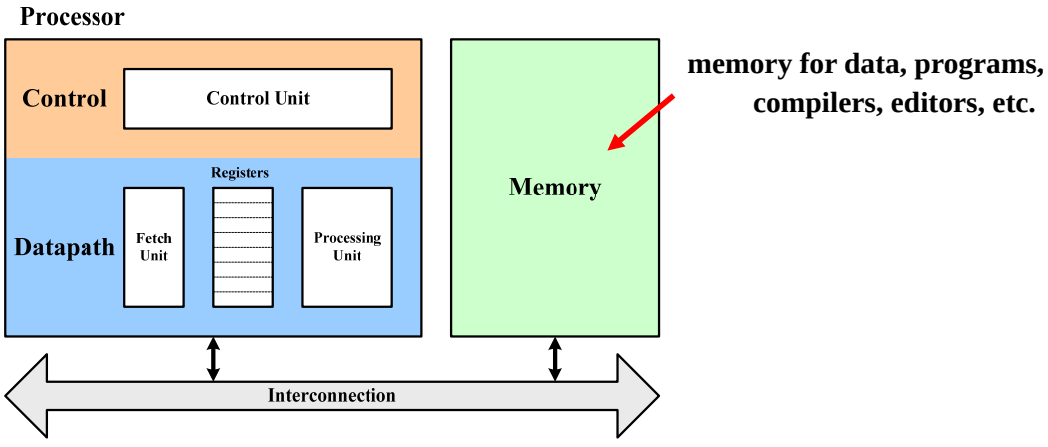
Registers vs. Memory

- Compiler associates variables with registers
- What about programs with lots of variables



Stored Program Concept

- Instructions are bits
- Programs are stored in memory — to be read or written just like data



- Fetch & Execute Cycle
  - Instructions are fetched and put into a special register.
  - Bits in the register are inputs to a control unit that controls the subsequent actions
  - Fetch the “next” instruction using address in the program counter (PC) and continue.

Outline

1.1 Introduction
1.2 System Software and Machine Architecture
1.3 The Simplified Instructional Computer (SIC)
1.3.1 SIC Machine Architecture
1.3.2 SIC/XE Machine Architecture
1.3.3 SIC Programming Examples

Department of Electrical Engineering, Feng-Chia University

Simplified Instructional Computer (SIC)

- This machine has been designed to illustrate the most commonly encountered hardware features and concepts, while avoiding most of the idiosyncrasies that are often found in real machines.
- SIC comes in two versions:
  - SIC (standard)
  - SIC/XE (“eXtra Equipment” or “eXtra Expensive”)
- SIC and SIC/XE are upward compatible
  - An object program for the standard SIC machine will also execute properly on a SIC/XE system.

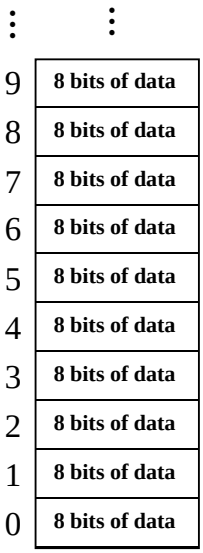
Outline

1.1 Introduction
1.2 System Software and Machine Architecture
1.3 The Simplified Instructional Computer (SIC)
1.3.1 SIC Machine Architecture
1.3.2 SIC/XE Machine Architecture
1.3.3 SIC Programming Examples

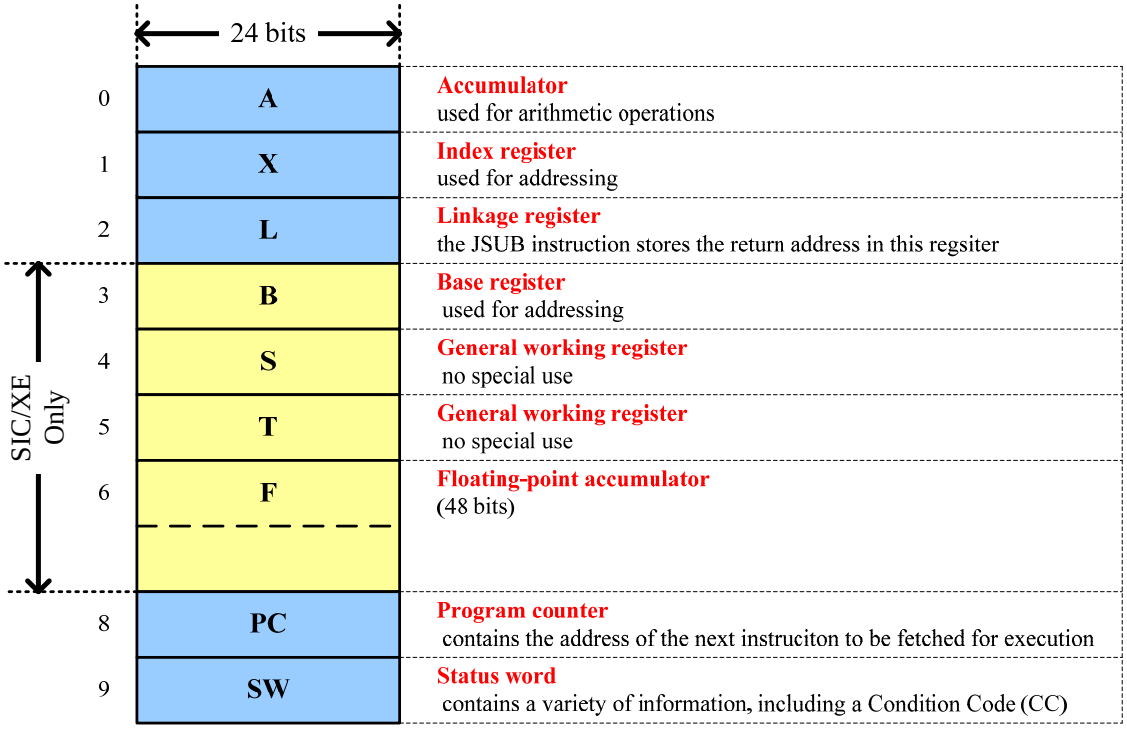
Department of Electrical Engineering, Feng-Chia University

Memory

- Memory consists of 8-bit bytes
- Any 3 consecutive bytes form a word (24 bits)
- All addresses are byte addresses
- Words are addressed by the location of their lowest numbered byte
- Maximum Memory Size
  - SIC: 2^{15} bytes = 32KB
  - SIC/XE: 2^{20} bytes = 1MB



Registers



Department of Electrical Engineering, Feng-Chia University

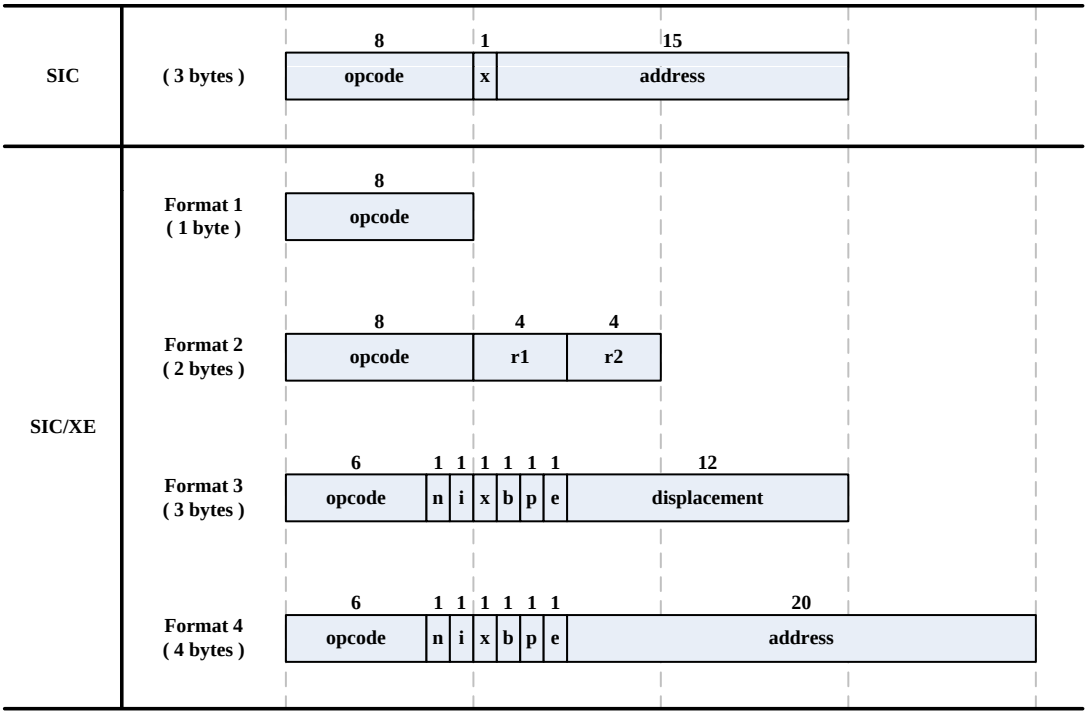
Data Formats

- **Integers**
  - 24-bit binary numbers (2's complement)
- **Characters**
  - 8-bit ASCII codes (see Appendix B, p501)
- **Floating Point** (SIC/XE Only)

  - S: sign (0 = positive, 1 = negative)
  - Exponent: an unsigned binary number between 0 and 2047
  - Fraction: interpreted as a value between 0 and 1

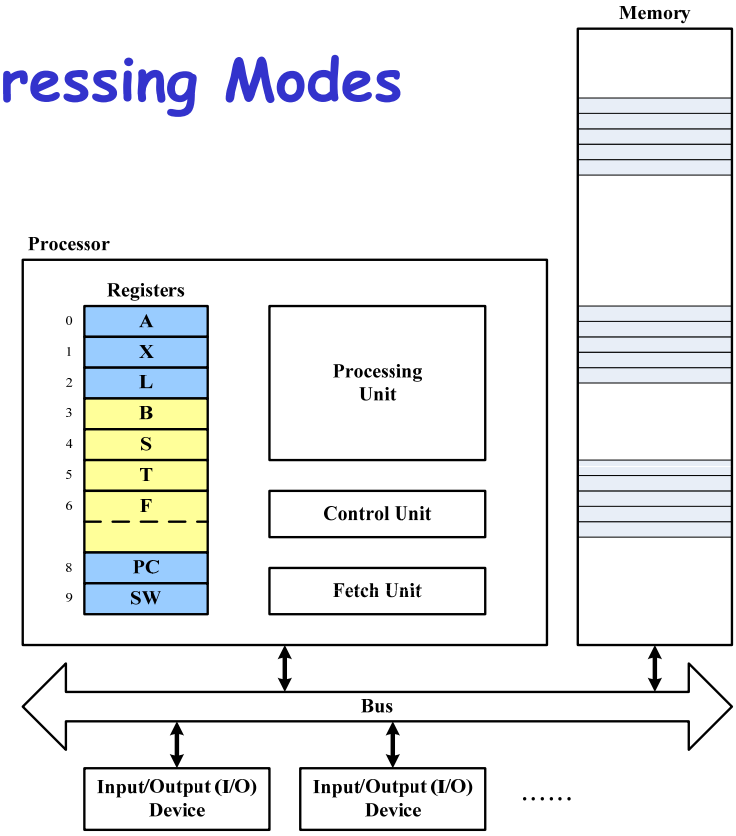
$$value = (-1)^s \times 0.f \times 2^{(e-1024)}$$

Instruction Formats



Addressing Modes

- SIC
  - Direct addressing
  - Indexed addressing
- SIC/XE
  - Direct addressing
  - Indexed addressing
  - Base relative addressing
  - Program-counter relative addressing
  - Immediate addressing
  - Indirect addressing



Addressing Modes (1/2)

[p499]

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

Department of Electrical Engineering, Feng-Chia University

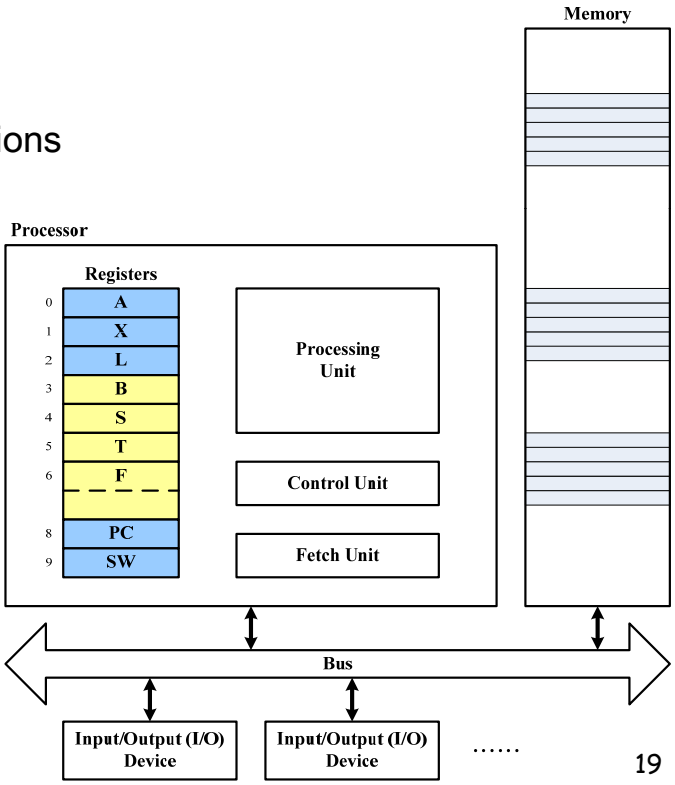
Addressing Modes (2/2)

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

Instruction Set (SIC)

- Load and store registers
  - LDA, LDX, STA, STX
- Integer arithmetic operations
  - ADD, SUB, MUL, DIV
- Compare
  - COMP
- Conditional jump
  - JLT, JEQ, JGT
- Subroutine linkage
  - JSUB, RSUB
- …

Instruction Set
[Appendix A, p495]

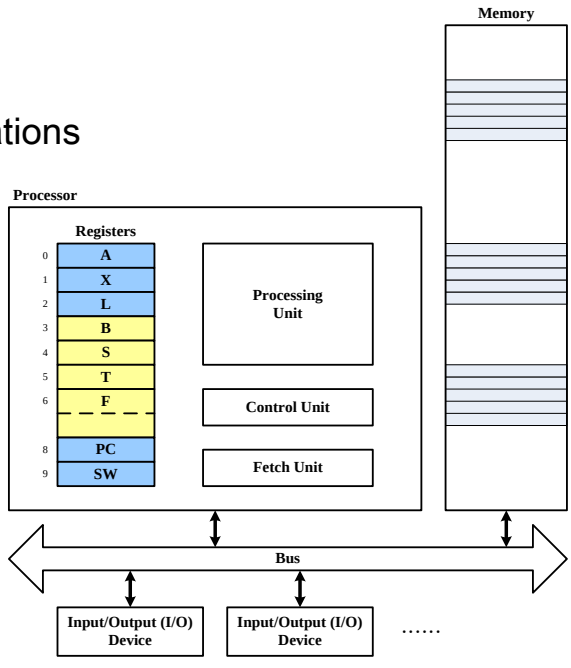


Department of Electrical Engineering, Feng-Chia University

Instruction Set (SIC/XE)

- All SIC Instructions
- Load and store registers
  - LDB, STB, etc.
- Floating-point arithmetic operations
  - ADDF, SUBF, MULF, DIVF
- R-R arithmetic operations
  - ADDR, SUBR, MULR, DIVR
- Supervisor call
  - SVC
- …

Instruction Set
[Appendix A, p495]

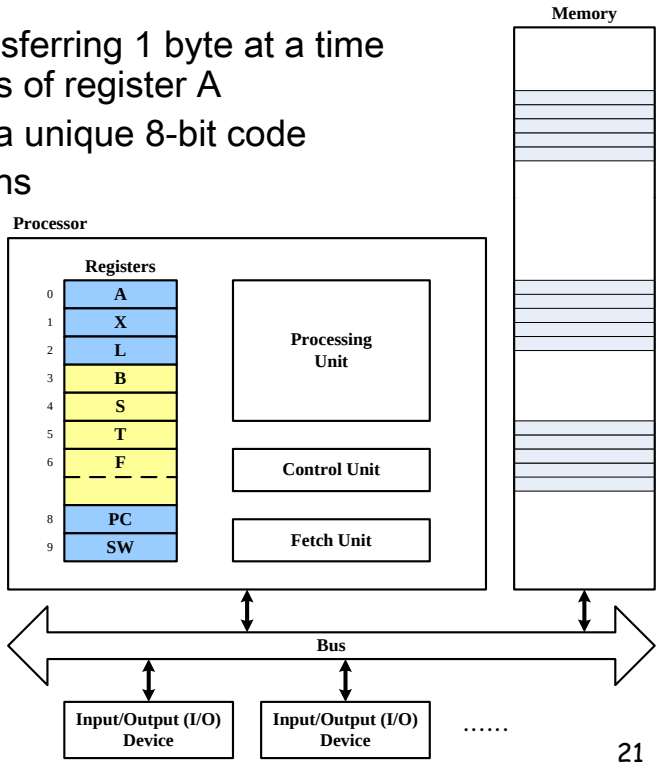


Input and Output

- I/O are performed by transferring 1 byte at a time to/from the rightmost 8bits of register A
- Each device is assigned a unique 8-bit code
- SIC : Three I/O instructions
  - Test Device (TD)
  - Read Data (RD)
  - Write Data (WD)

TD ➔ RD, WD
ready
(CC = '<')

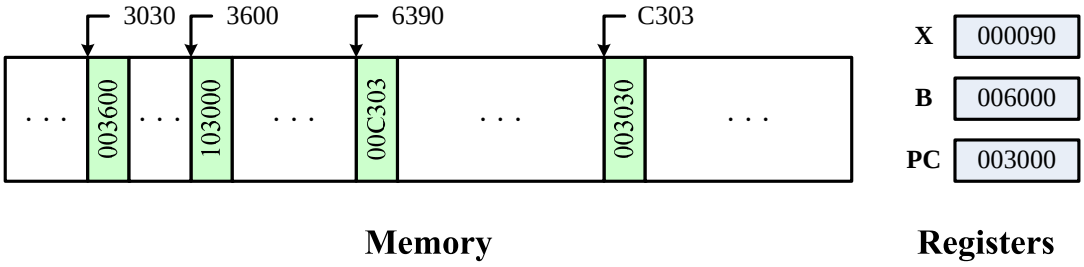
- SIC/XE:
  - Start I/O (SIO)
  - Test I/O (TIO)
  - Halt I/O (HIO)



Department of Electrical Engineering, Feng-Chia University

Addressing Examples

[Figure 1.1]



Machine instruction			Target address	Value loaded into register A
Hex	Binary			
	op	n i x b p e disp/address		
0 3 2 6 0 0	0 0 0 0 0 0	1 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0	3600	103000
0 3 C 3 0 0	0 0 0 0 0 0	1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0	6390	00C303
0 2 2 0 3 0	0 0 0 0 0 0	1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0	3030	103000
0 1 0 0 3 0	0 0 0 0 0 0	0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0	30	000030
0 0 3 6 0 0	0 0 0 0 0 0	0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0	3600	103000
0 3 1 0 C 3 0 3	0 0 0 0 0 0	1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1	C303	003030

Outline

1.1 Introduction
1.2 System Software and Machine Architecture
1.3 The Simplified Instructional Computer (SIC)
1.3.1 SIC Machine Architecture
1.3.2 SIC/XE Machine Architecture
1.3.3 SIC Programming Examples

Sample data movement operations

[Figure 1.2]

SIC

```
        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)

SIC/XE

```
        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)

Sample arithmetic operations (1/2)

[Figure 1.3(a)]

SIC

```
        LDA     ALPHA       LOAD ALPHA INTO REGISTER A
        ADD     INCR        ADD THE VALUE OF INCR
        SUB     ONE         SUBTRACT 1
        STA     BETA        STORE IN BETA
        LDA     GAMMA       LOAD GAMMA INTO REGISTER A
        ADD     INCR        ADD THE VALUE OF INCR
        SUB     ONE         SUBTRACT 1
        STA     DELTA       STORE IN DELTA
          .
          .
          .
ONE     WORD    1           ONE-WORD CONSTANT
.                           ONE-WORD VARIABLES
ALPHA   RESW    1
BETA    RESW    1
GAMMA   RESW    1
DELTA   RESW    1
INCR    RESW    1
```


Department of Electrical Engineering, Feng-Chia University

Sample arithmetic operations (2/2)

[Figure 1.3(b)]

SIC/XE

```
        LDS     INCR        LOAD VALUE OF INCR INTO REGISTER S
        LDA     ALPHA       LOAD ALPHA INTO REGISTER A
        ADDR    S,A         ADD THE VALUE OF INCR
        SUB     #1          SUBTRACT 1
        STA     BETA        STORE IN BETA
        LDA     GAMMA       LOAD GAMMA INTO REGISTER A
        ADDR    S,A         ADD THE VALUE OF INCR
        SUB     #1          SUBTRACT 1
        STA     DELTA       STORE IN DELTA
          .
          .
          .
.                           ONE WORD VARIABLES
ALPHA   RESW    1
BETA    RESW    1
GAMMA   RESW    1
DELTA   RESW    1
INCR    RESW    1
```

Sample looping and indexing operations

[Figure 1.4]

SIC

```
            LDX     ZERO        INITIALIZE INDEX REGISTER TO 0
MOVECH      LDCH    STR1,X      LOAD CHARACTER FROM STR1 INTO REG A
            STCH    STR2,X      STORE CHARACTER INTO STR2
            TIX     ELEVEN      ADD 1 TO INDEX, COMPARE RESULT TO 11
            JLT     MOVECH      LOOP IF INDEX IS LESS THAN 11
            .
            .
            .
STR1        BYTE    C'TEST STRING'   11-BYTE STRING CONSTANT
STR2        RESB    11               11-BYTE VARIABLE
.                                    ONE-WORD CONSTANTS
ZERO        WORD    0
ELEVEN      WORD    11
```

SIC/XE

```
            LDT     #11         INITIALIZE REGISTER T TO 11
            LDX     #0          INITIALIZE INDEX REGISTER TO 0
MOVECH      LDCH    STR1,X      LOAD CHARACTER FROM STR1 INTO REG A
            STCH    STR2,X      STORE CHARACTER INTO STR2
            TIXR    T           ADD 1 TO INDEX, COMPARE RESULT TO 11
            JLT     MOVECH      LOOP IF INDEX IS LESS THAN 11
            .
            .
            .
STR1        BYTE    C'TEST STRING'   11-BYTE STRING CONSTANT
STR2        RESB    11               11-BYTE VARIABLE
```


Department of Electrical Engineering, Feng-Chia University

Sample looping and indexing operations (1/2) [Figure 1.5(a)]

SIC

```
            LDA     ZERO        INITIALIZE INDEX VALUE TO 0
            STA     INDEX
ADDLP       LDX     INDEX       LOAD INDEX VALUE INTO REGISTER X
            LDA     ALPHA,X     LOAD WORD FROM ALPHA INTO REGISTER A
            ADD     BETA,X      ADD WORD FROM BETA
            STA     GAMMA,X     STORE THE RESULT IN A WORD IN GAMMA
            LDA     INDEX       ADD 3 TO INDEX VALUE
            ADD     THREE
            STA     INDEX
            COMP    K300        COMPARE NEW INDEX VALUE TO 300
            JLT     ADDLP       LOOP IF INDEX IS LESS THAN 300
            .
            .
            .
INDEX       RESW    1           ONE-WORD VARIABLE FOR INDEX VALUE
.                               ARRAY VARIABLES--100 WORDS EACH
ALPHA       RESW    100
BETA        RESW    100
GAMMA       RESW    100
.                               ONE-WORD CONSTANTS
ZERO        WORD    0
K300        WORD    300
THREE       WORD    3
```

Sample looping and indexing operations (2/2) [Figure 1.5(b)]

SIC/XE

```
            LDS     #3          INITIALIZE REGISTER S TO 3
            LDT     #300        INITIALIZE REGISTER T TO 300
            LDX     #0          INITIALIZE INDEX REGISTER TO 0
ADDLP       LDA     ALPHA,X     LOAD WORD FROM ALPHA INTO REGISTER A
            ADD     BETA,X      ADD WORD FROM BETA
            STA     GAMMA,X     STORE THE RESULT IN A WORD IN GAMMA
            ADDR    S,X         ADD 3 TO INDEX VALUE
            COMPR   X,T         COMPARE NEW INDEX VALUE TO 300
            JLT     ADDLP       LOOP IF INDEX VALUE IS LESS THAN 300
            .
            .
            .
.                               ARRAY VARIABLES--100 WORDS EACH
ALPHA       RESW    100
BETA        RESW    100
GAMMA       RESW    100
```


Department of Electrical Engineering, Feng-Chia University

Sample input and output operations [Figure 1.6]

SIC

```
INLOOP      TD      INDEV       TEST INPUT DEVICE
            JEQ     INLOOP      LOOP UNTIL DEVICE IS READY
            RD      INDEV       READ ONE BYTE INTO REGISTER A
            STCH    DATA        STORE BYTE THAT WAS READ
            .
            .
            .
OUTLP       TD      OUTDEV      TEST OUTPUT DEVICE
            JEQ     OUTLP       LOOP UNTIL DEVICE IS READY
            LDCH    DATA        LOAD DATA BYTE INTO REGISTER A
            WD      OUTDEV      WRITE ONE BYTE TO OUTPUT DEVICE
            .
            .
            .
INDEV       BYTE    X'F1'       INPUT DEVICE NUMBER
OUTDEV      BYTE    X'05'       OUTPUT DEVICE NUMBER
DATA        RESB    1           ONE-BYTE VARIABLE
```

Sample Subroutine call and record input operations (1/2) [Figure 1.7(a)]

SIC

```
            JSUB    READ        CALL READ SUBROUTINE
            .
            .
            .
.                               SUBROUTINE TO READ 100-BYTE RECORD
READ        LDX     ZERO        INITIALIZE INDEX REGISTER TO 0
RLOOP       TD      INDEV       TEST INPUT DEVICE
            JEQ     RLOOP       LOOP IF DEVICE IS BUSY
            RD      INDEV       READ ONE BYTE INTO REGISTER A
            STCH    RECORD,X    STORE DATA BYTE INTO RECORD
            TIX     K100        ADD 1 TO INDEX AND COMPARE TO 100
            JLT     RLOOP       LOOP IF INDEX IS LESS THAN 100
            RSUB                EXIT FROM SUBROUTINE
            .
            .
            .
INDEV       BYTE    X'F1'       INPUT DEVICE NUMBER
RECORD      RESB    100         100-BYTE BUFFER FOR INPUT RECORD
.                               ONE-WORD CONSTANTS
ZERO        WORD    0
K100        WORD    100
```


Department of Electrical Engineering, Feng-Chia University

Sample Subroutine call and record input operations (2/2) [Figure 1.7(b)]

SIC/XE

```
            JSUB    READ        CALL READ SUBROUTINE
            .
            .
            .
.                               SUBROUTINE TO READ 100-BYTE RECORD
READ        LDX     #0          INITIALIZE INDEX REGISTER TO 0
            LDT     #100        INITIALIZE REGISTER T TO 100
RLOOP       TD      INDEV       TEST INPUT DEVICE
            JEQ     RLOOP       LOOP IF DEVICE IS BUSY
            RD      INDEV       READ ONE BYTE INTO REGISTER A
            STCH    RECORD,X    STORE DATA BYTE INTO RECORD
            TIXR    T           ADD 1 TO INDEX AND COMPARE TO 100
            JLT     RLOOP       LOOP IF INDEX IS LESS THAN 100
            RSUB                EXIT FROM SUBROUTINE
            .
            .
            .
INDEV       BYTE    X'F1'       INPUT DEVICE NUMBER
RECORD      RESB    100         100-BYTE BUFFER FOR INPUT RECORD
```