Chapter 1 Background

Professor Gwan-Hwan Hwang
Dept. Computer Science and Information Engineering
National Taiwan Normal University
9/17/2009

Outlines

- 1.1 Introduction
- 1.2 System Software and Machine Architecture
- 1.3 The Simplified Instructional Computer (SIC)
  - SIC Machine Architecture
  - SIC/XE Machine Architecture
  - SIC Programming Examples
- 1.4 Traditional (CISC) Machines
- 1.5 RISC Machines

Introduction

- Definition of System software
  - System software consists of a variety of programs that support the operation of a computer
- Examples
  - Text editor, compiler, loader or linker, debugger, macro processors, operating system, database management systems, software engineering tools, etc.

System Software and Machine Architecture

- One characteristic in which most system software differs 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.
  - E.g. Assemblers, compilers, operating systems

System Software and Machine Architecture (Cont'd)

- There are some aspects of system software that do not directly depend upon the type of computing system being supported
  - The second para. of Section 1.2

System Software and Machine Architecture (Cont'd)

- Because most system software is machine-dependent, we must include real machines and real pieces of software in our study.
- Simplified Instructional Computer (SIC)
  - SIC is a hypothetical computer that has been carefully designed to include the hardware features most often found on real machines, while avoiding unusual or irrelevant complexities

The Simplified Instructional Computer (SIC)

- Like many other products, SIC comes in two versions
  - The standard model
  - An XE version
    - “extra equipments”, “extra expensive”
- The two versions has been designed to be *upward compatible*
  - An object program for the standard SIC machine will also execute properly on a SIC/XE system

SIC Machine Architecture

- Memory
  - Memory consists of 8-bit bytes
  - Any 3 consecutive bytes form a word (24 bits)
  - Total of 32768 (2^{15}) bytes in the computer memory

SIC Machine Architecture (Cont'd)

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

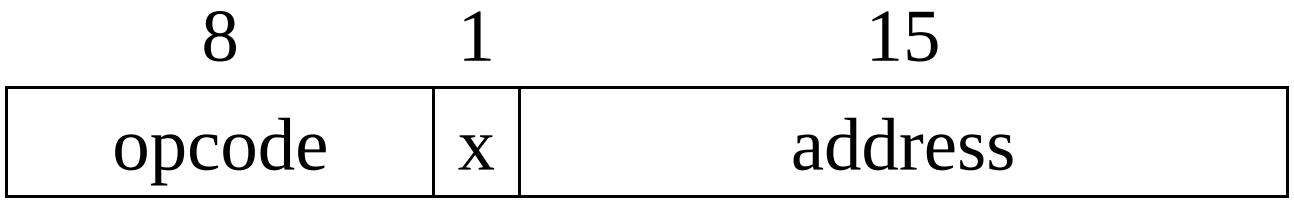
Mnemonic	Number	Special use
A	0	Accumulator
X	1	Index register
L	2	Linkage register
PC	8	Program counter
SW	9	Status word

SIC Machine Architecture (Cont'd)

- Data Formats
  - Integers are stored as 24-bit binary number
  - 2's complement representation for negative values
  - Characters are stored using 8-bit ASCII codes
  - No floating-point hardware on the standard version of SIC

SIC Machine Architecture (Cont'd)

- Instruction Formats
  - Standard version of SIC
  - 24 bits



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

SIC Machine Architecture (Cont'd)

- Addressing Modes
  - There are two addressing modes available
    - Indicated by x bit in the instruction

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

(X): the contents of register X

SIC Machine Architecture (Cont'd)

- Instruction Set
  - Load and store registers
    - LDA, LDX, STA, STX, etc.
  - Integer arithmetic operations
    - ADD, SUB, MUL, DIV
    - All arithmetic operations involve register A and a word in memory, with the result being left in A
  - COMP
  - Conditional jump instructions
    - JLT, JEQ, JGT
  - Subroutine linkage
    - JSUB, RSUB
- See appendix A, Pages 495-498

SIC Machine Architecture (Cont'd)

- 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
    - Test Device TD instruction
    - Read Data (RD)
    - Write Data (WD)

SIC/XE Machine Architecture

- Memory
  - Maximum memory available on a SIC/XE system is 1 megabyte (2^{20} bytes)

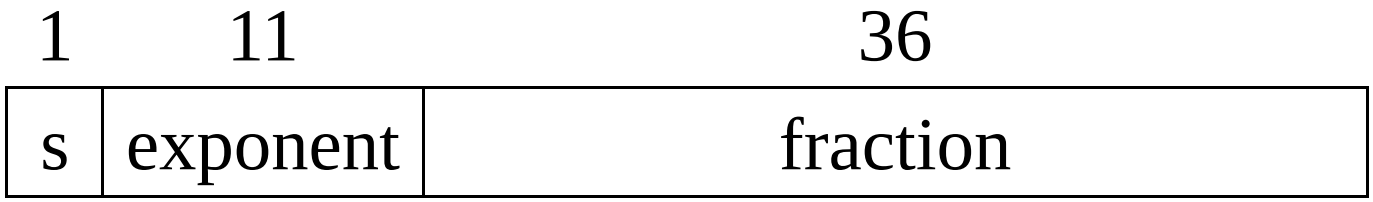
SIC/XE Machine Architecture (Cont'd)

- Registers
  - 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 (Cont'd)

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

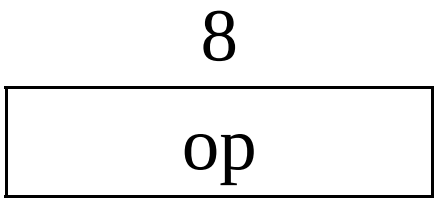


$F*2^{(e-1024)}$

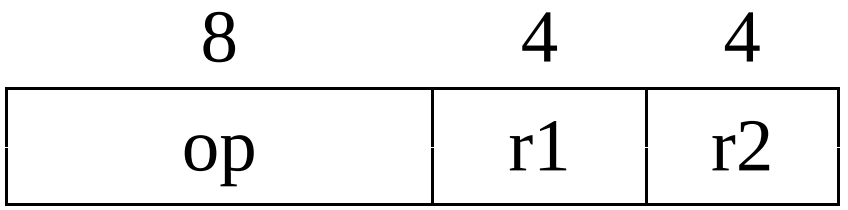
SIC/XE Machine Architecture (Cont'd)

- Instruction Formats
  - 15 bits in (SIC), 20 bits in (SIC/XE)

Format 1 (1 byte)



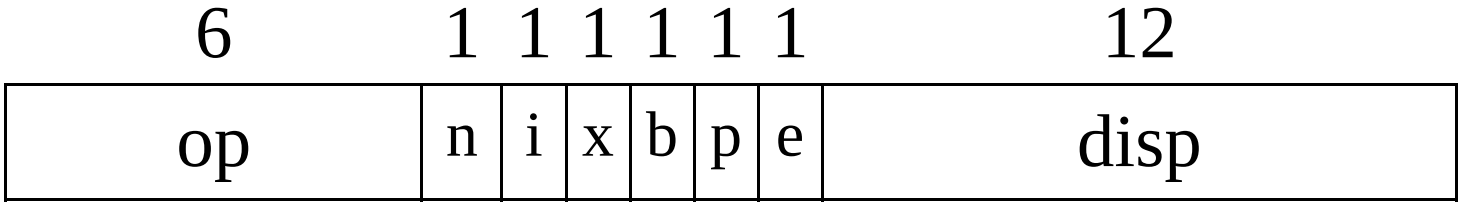
Format 2 (2 bytes)



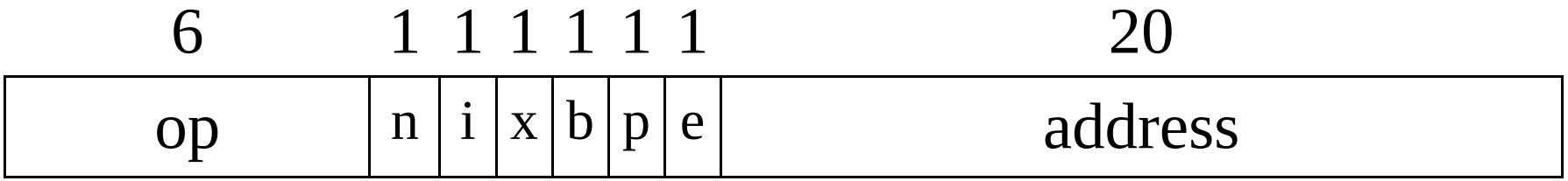
Formats 1 and 2 are instructions that do not reference memory at all

SIC/XE Machine Architecture (Cont'd)

Format 3 (3 bytes)



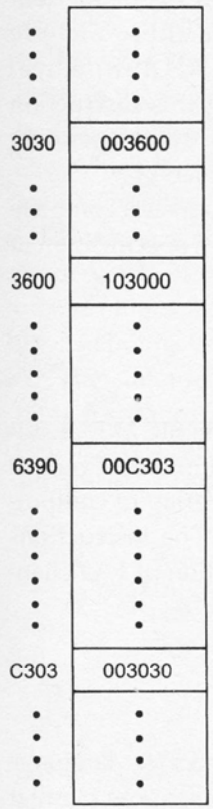
Format 4 (5 bytes)



Mode	Indication	Target address calculation	
Base relative	b=1,p=0	TA=(B)+disp	(0≤disp ≤4095)
Program-counter relative	b=0,p=1	TA=(PC)+disp	(-2048≤disp ≤2047)

SIC/XE Machine Architecture (Cont'd)

- Instruction Formats
  - See Figure 1.1, P. 11.



(a)

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

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.

SIC/XE Machine Architecture (Cont'd)

- Instruction Set
  - Instructions to load and store the new registers
    - LDB, STB, etc.
  - Floating-point arithmetic operations
    - ADDF, SUBF, MULF, DIVF
  - Register move instruction
    - RMO
  - Register-to-register arithmetic operations
    - ADDR, SUBR, MULR, DIVR
  - Supervisor call instruction
    - SVC

SIC/XE Machine Architecture (Cont'd)

- Input and Output
  - There are I/O channels that can be used to perform input and output while the CPU is executing other instructions

SIC Programming Examples

- Figure 1.2
  - Sample data movement operations
- Figure 1.3
  - Sample arithmetic operations
- Figure 1.4
  - Sample looping and indexing operations
- Figure 1.5
  - Sample looping and indexing operations
- Figure 1.6
  - I/O
- Figure 1.7
  - Subroutine call

```
        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.

```
        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
```

(a)

```
        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
```

(b)

Figure 1.3 Sample arithmetic operations for (a) SIC and (b) SIC/XE.

```
            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
```

(a)

```
            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
```

(b)

Figure 1.4 Sample looping and indexing operations for (a) SIC and (b) SIC/XE.

```
        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
```

(a)

```
        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
```

(b)

Figure 1.5 Sample indexing and looping operations for (a) SIC and (b) SIC/XE.

```
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
```

Figure 1.6 Sample input and output operations for 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
```

(a)

```
          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
```

(b)

Figure 1.7 Sample subroutine call and record input operations for (a) SIC and (b) SIC/XE.