CS2422 Assembly Language and System Programming

System Software and Machine Architecture

國立清華大學
National Tsing Hua University

Chapter Outline

Chapter 1 of Beck's "*System Software*" book

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

System Software

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

System Software & Architecture

- System software differs from application software in machine dependency
  - System programs are intended to support the operation and use of the computer itself, rather than any particular application.
- Thus, we must include real machines and real pieces of software in our study
- **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

Simplified Instructional Computer

- Like many other products, SIC comes in two versions
  - The standard model
  - An XE version
    - "extra equipments", "extra expensive"
- The two versions have been designed to be upward compatible

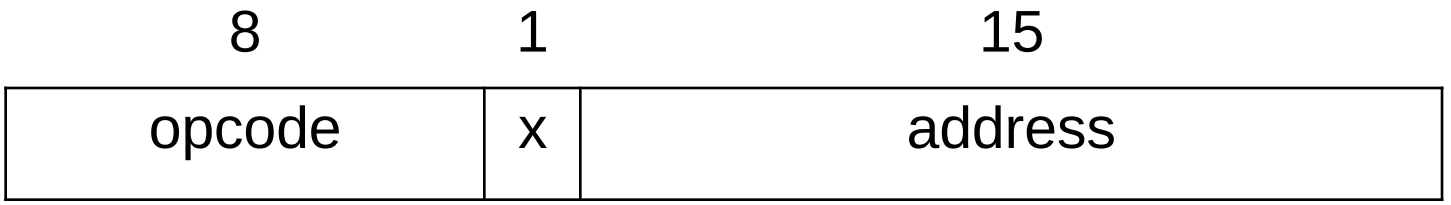
SIC Machine Architecture (1/5)

- Memory
  - Memory consists of 8-bit bytes, 15-bit addresses
  - Any 3 consecutive bytes form a word (24 bits)
  - Total of 32768 (2^{15}) bytes in the computer memory
- Registers
  - Five registers, each 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 (2/5)

- Data formats
  - 24-bit integer representation in 2's complement
  - 8-bit ASCII code for characters
  - No floating-point on standard version of SIC
- Instruction formats
  - Standard version of SIC



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

SIC Machine Architecture (3/5)

- Addressing modes
  - Two addressing modes
    - Indicated by the 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 (4/5)

Instruction set: (Appendix A, Page 495)

- Load/store registers: LDA, LDX, STA, STX
- Integer arithmetic: ADD, SUB, MUL, DIV
  - All involve register A and a word in memory, result stored in register A
- COMP
  - Compare value in register A with a word in memory
  - Set a *condition code* CC (<, =, or >)
- Conditional jump instructions
  - JLT, JEQ, JGT: test CC and jump

SIC Machine Architecture (5/5)

- Subroutine linkage
  - JSUB, RSUB: return address in register L
- Input and output
  - Performed by transferring 1 byte at a time to or from the rightmost 8 bits of register A
  - Each device is assigned a unique 8-bit code, as an operand of I/O instructions
  - Test Device (TD): < (ready), = (not ready)
  - Read Data (RD), Write Data (WD)

SIC Programming Example (Fig 1.2a)

- Data movement

```
        LDA     FIVE      load 5 into A
        STA     ALPHA     store in ALPHA
        LDCH    CHARZ     load 'Z' into A
        STCH    C1        store in C1
        .
        .
        .
ALPHA   RESW    1         reserve one word space
FIVE    WORD    5         one word holding 5
CHARZ   BYTE    C'Z'      one-byte constant
C1      RESB    1         one-byte variable
```

SIC Programming Example (Fig 1.3a)

- Arithmetic operations: BETA = ALPHA+INCR-1

```
          LDA     ALPHA
          ADD     INCR
          SUB     ONE
          STA     BETA
          LDA     GAMMA
          ADD     INCR
          SUB     ONE
          STA     DELTA
          ...
ONE       WORD    1         one-word constant
ALPHA     RESW    1         one-word variables
BETA      RESW    1
GAMMA     RESW    1
DELTA     RESW    1
INCR      RESW    1
```

SIC Programming Example (Fig 1.4a)

- Looping and indexing: copy one string to another

```
         LDX    ZERO        initialize index register to 0
MOVECH   LDCH   STR1,X      load char from STR1 to reg A
         STCH   STR2,X
         TIX    ELEVEN      add 1 to index, compare to 11
         JLT    MOVECH      loop if "less than"
         .
         .
         .
STR1     BYTE   C'TEST STRING'
STR2     RESB   11
ZERO     WORD   0
ELEVEN   WORD   11
```

SIC Programming Example (Fig 1.5a)

```
            LDA     ZERO        initialize index value to 0
            STA     INDEX
ADDLP       LDX     INDEX       load index value to reg X
            LDA     ALPHA,X     load word from ALPHA into reg A
            ADD     BETA,X
            STA     GAMMA,X     store the result in a word in GAMMA
            LDA     INDEX
            ADD     THREE       add 3 to index value
            STA     INDEX
            COMP    K300        compare new index value to 300
            JLT     ADDLP       loop if less than 300
            ...
            ...
INDEX       RESW    1
ALPHA       RESW    100         array variables—100 words each
BETA        RESW    100
GAMMA       RESW    100
ZERO        WORD    0           one-word constants
THREE       WORD    3
K300        WORD    300
```

SIC Programming Example (Fig 1.6)

- Input and output

```
INLOOP  TD      INDEV   test input device
        JEQ     INLOOP  loop until device is ready
        RD      INDEV   read one byte into register A
        STCH    DATA
        .
        .
OUTLP   TD      OUTDEV  test output device
        JEQ     OUTLP   loop until device is ready
        LDCH    DATA
        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
```

SIC/XE Machine Architecture (1/11)

- Memory
  - Maximum memory available on a SIC/XE system is 1 megabyte (2^{20} bytes)
  - An address (20 bits) cannot be fitted into a 15-bit field as in SIC Standard
  - Must change instruction formats and addressing modes

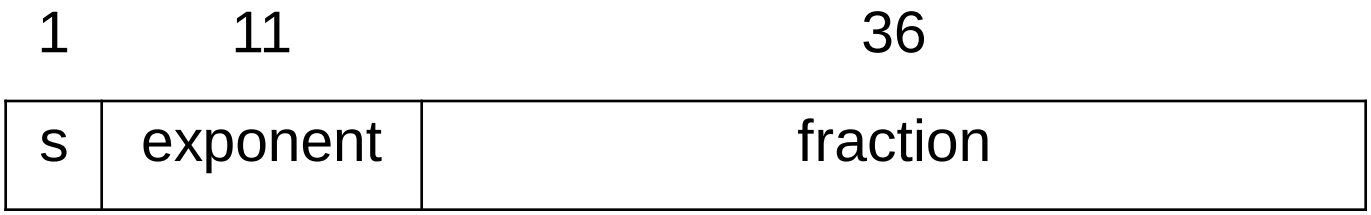
SIC/XE Machine Architecture (2/11)

- 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 (3/11)

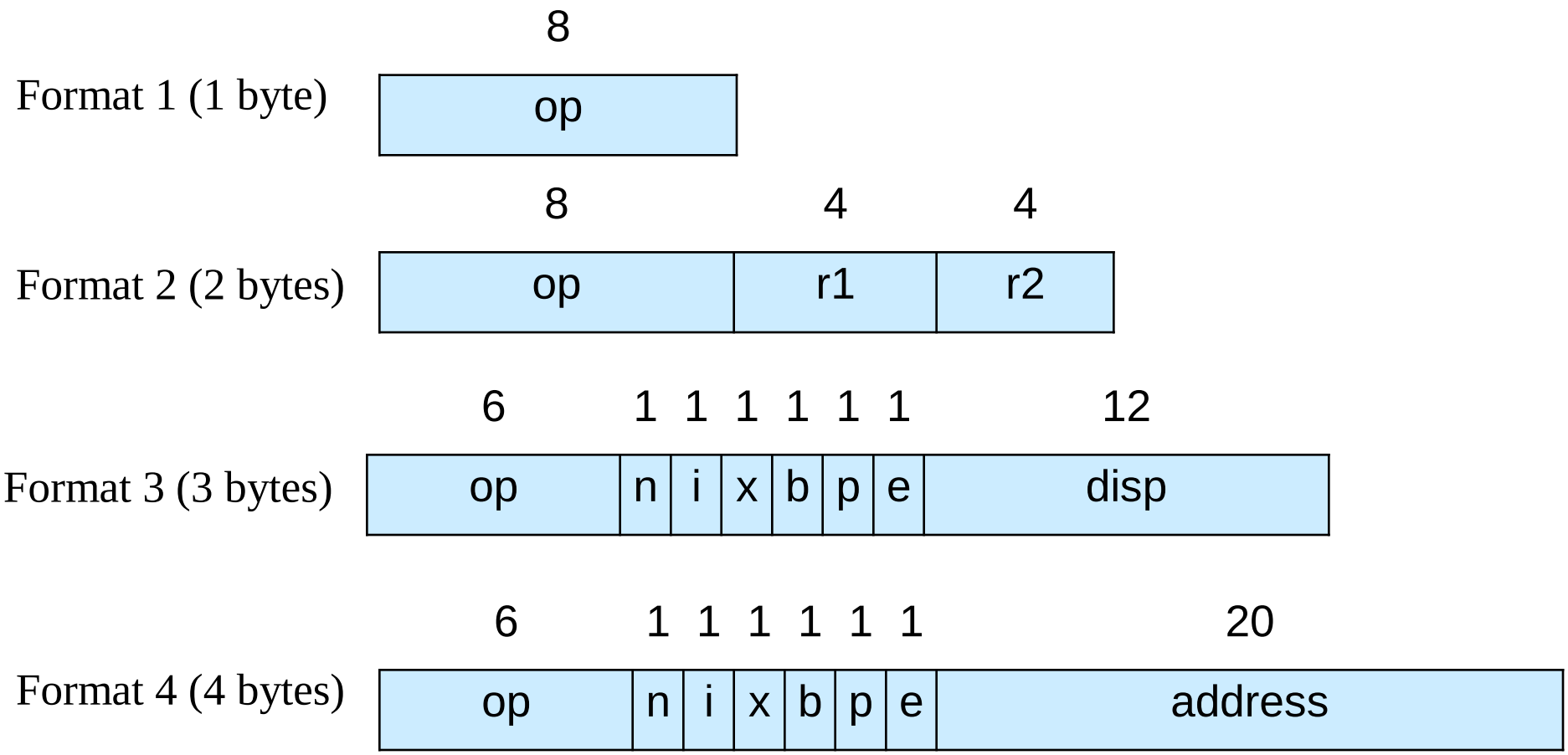
- There is a 48-bit floating-point data type
  - *fraction* is a value between 0 and 1
  - *exponent* is an unsigned binary number between 0 and 2047
  - zero is represented as all 0



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

SIC/XE Machine Architecture (4/11)

- Instruction formats

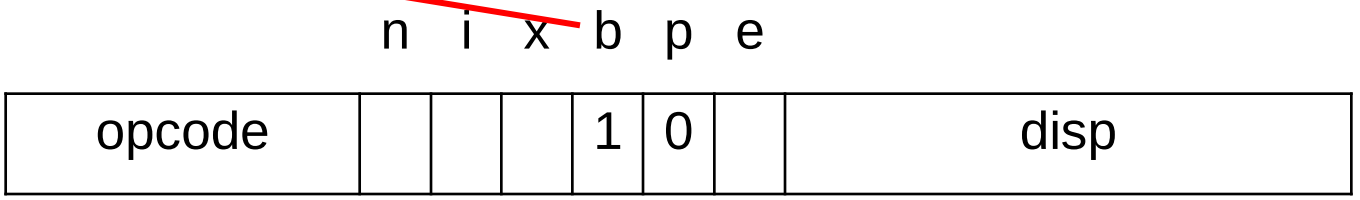


Formats 1 and 2 do not reference memory at all

Bit e distinguishes between format 3 and 4

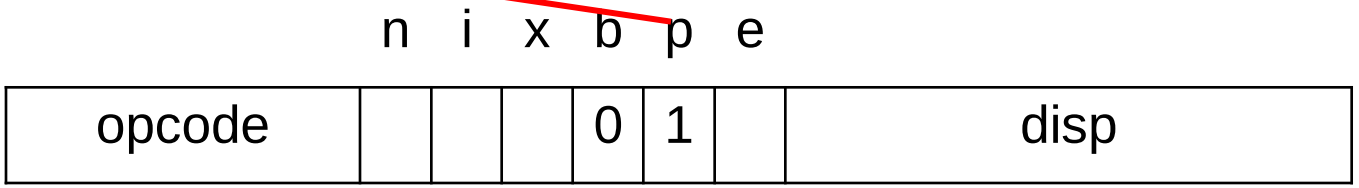
SIC/XE Machine Architecture (5/11)

- Base Relative Addressing Mode



b=1, p=0, TA=(B)+disp (0≤disp ≤4095)

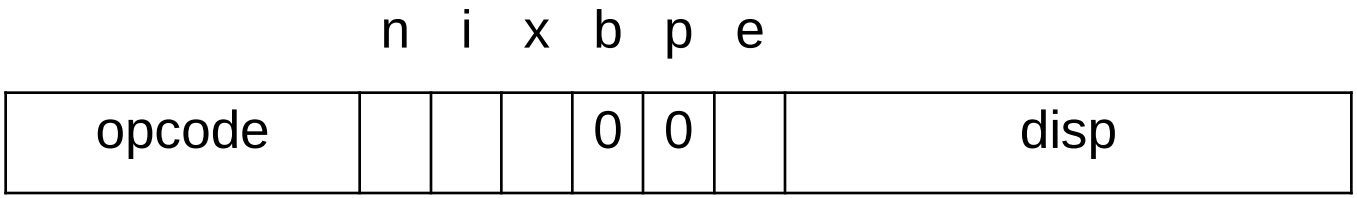
- Program-Counter Relative Addressing Mode



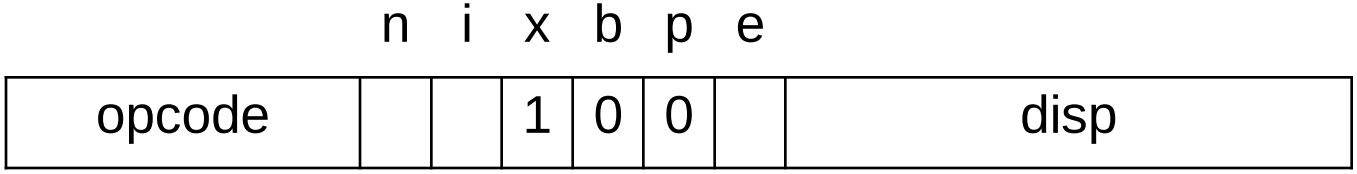
b=0, p=1, TA=(PC)+disp (-2048≤disp ≤2047)

SIC/XE Machine Architecture (6/11)

- Direct Addressing Mode



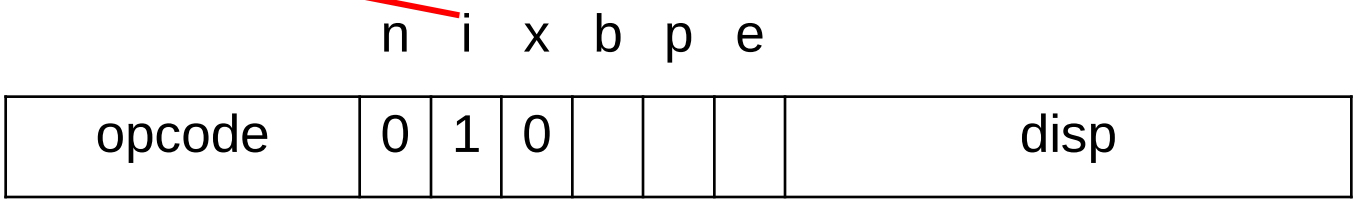
b=0, p=0, TA=disp (0≤disp ≤4095)



b=0, p=0, TA=(X)+disp
(with index addressing mode)

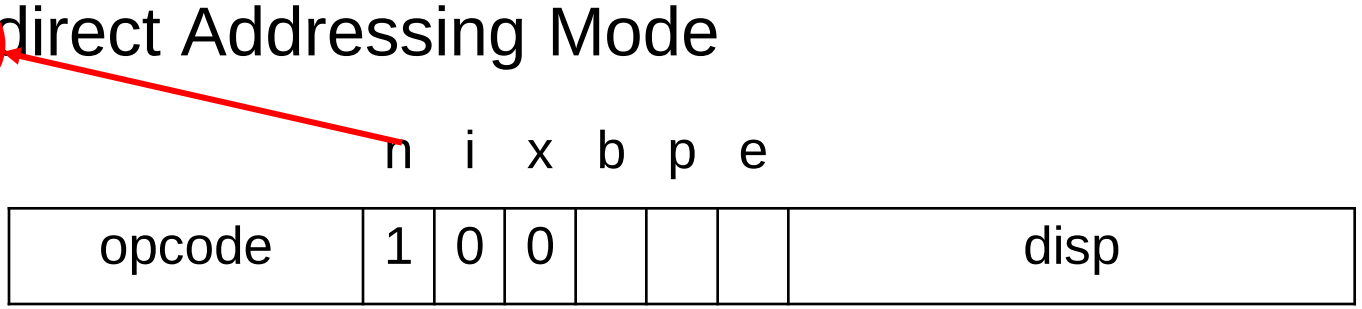
SIC/XE Machine Architecture (7/11)

- Immediate Addressing Mode



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

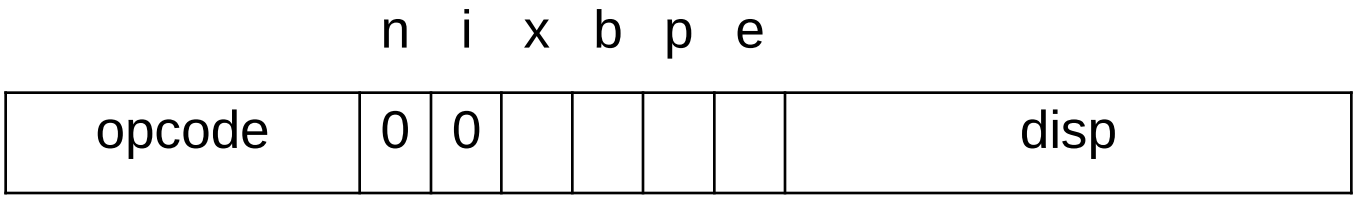
- Indirect Addressing Mode



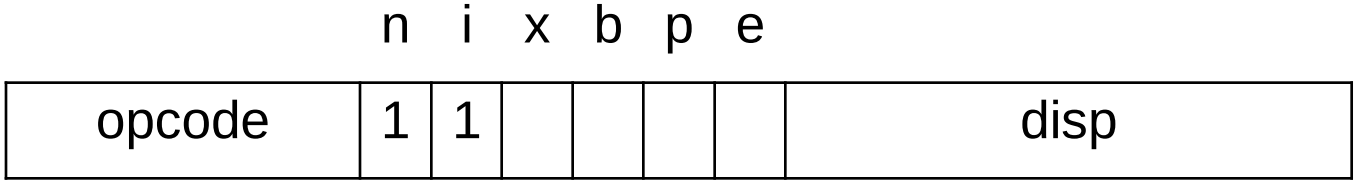
n=1, i=0, x=0, TA=(disp)

SIC/XE Machine Architecture (8/11)

- Simple Addressing Mode



i=0, n=0, TA=bpe+disp (SIC standard)
opcode+n+i = SIC standard opcode (8-bit)



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

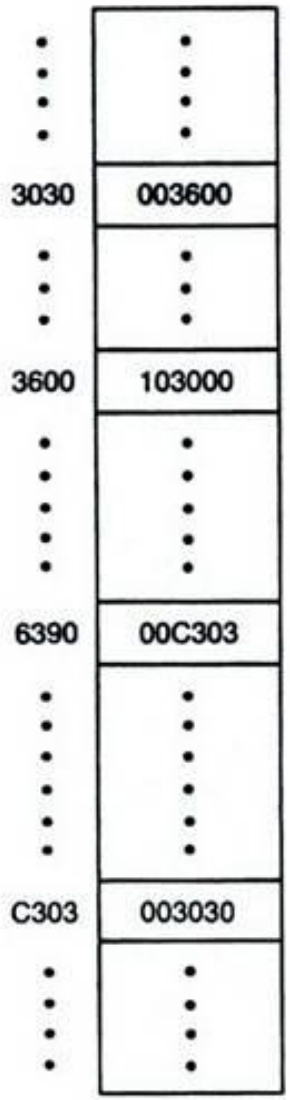
SIC/XE Machine Architecture (9/11)

- Addressing Modes Summary (p.499)

Assembler decides which format to use

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

SIC/XE Machine Architecture (10/11)



(a)

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

Example Instruction Format

(PC) + disp
(B) + disp + (X)
((PC) + disp)
disp
b/p/e + disp
addr

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)

SIC/XE Machine Architecture (11/11)

- Instruction set:
  - load and store the new registers: LDB, STB, etc.
  - Floating-point arithmetic operations
    - ADDF, SUBF, MULF, DIVF
  - Register move: RMO
  - Register-to-register arithmetic operations
    - ADDR, SUBR, MULR, DIVR
  - Supervisor call: SVC
- Input and output:
  - I/O channels to perform I/O while CPU is executing other instructions: SIO, TIO, HIO

SIC/XE Programming Example (Fig 1.2b)

SIC version

```
        LDA     FIVE
        STA     ALPHA
        LDCH    CHARZ
        STCH    C1
        .
        .
        .
ALPHA   RESW    1
FIVE    WORD    5
CHARZ   BYTE    C'Z'
C1      RESB    1
```

SIC/XE version

```
        LDA     #5
        STA     ALPHA
        LDCH    #90
        STCH    C1
        .
        .
        .
ALPHA   RESW    1
C1      RESB    1
```

SIC/XE Programming Example (Fig 1.3b)

```
        LDS     INCR
        LDA     ALPHA   BETA=ALPHA+INCR-1
        ADDR    S,A
        SUB     #1
        STA     BETA
        LDA     GAMMA   DELTA=GAMMA+INCR-1
        ADDR    S,A
        SUB     #1
        STA     DELTA
        ...
        ...
ALPHA   RESW    1       one-word variables
BETA    RESW    1
GAMMA   RESW    1
DELTA   RESW    1
INCR    RESW    1
```

SIC/XE Programming Example (Fig 1.4b)

- Looping and indexing: copy one string to another

```
        LDT     #11         initialize register T to 11
        LDX     #0          initialize index register to 0
MOVECH  LDCH    STR1,X      load char from STR1 to reg A
        STCH    STR2,X      store char into STR2
        TIXR    T           add 1 to index, compare to 11
        JLT     MOVECH      loop if "less than" 11
        .
        .
        .
STR1    BYTE    C'TEST STRING'
STR2    RESB    11
```

SIC/XE Programming Example (Fig 1.5b)

```
        LDS     #3
        LDT     #300
        LDX     #0
ADDLP   LDA     ALPHA,X     load from ALPHA to reg A
        ADD     BETA,X
        STA     GAMMA,X     store in a word in GAMMA
        ADDR    S,X         add 3 to index value
        COMPR   X,T         compare to 300
        JLT     ADDLP       loop if less than 300
        ...
        ...
ALPHA   RESW    100         array variables—100 words each
BETA    RESW    100
GAMMA   RESW    100
```

SIC/XE Programming Example (Fig 1.7b)

```
            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
```