SYSTEM PROGRAMMING

Prof. S.G.Gollagi

Chapter 1:

System Software vs. Machine Architecture

- Machine dependent
  - The most important characteristic in which most system software differ from application software
  - e.g. assembler translate mnemonic instructions into machine code
  - e.g. compilers must generate machine language code
  - Machine architecture differs in:
    - Machine code
    - Instruction formats
    - Addressing mode
    - Registers
- 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

System Software and Architecture

- System software will be discussed:
  - The basic functions
  - Machine-dependent functions
  - Machine-independent functions
  - Design options (single-pass vs. multi-pass)

The Simplified Instructional Computer (SIC)

- SIC is a hypothetical computer that includes the hardware features most often found on real machines.
- Why the simplified instructional computer
  - To avoid various unique features and idiosyncrasies of a particular machine.
  - To focus on central, fundamental, and commonly encountered features and concepts.
- Two versions of SIC
  - standard model (SIC)
  - extension version (SIC/XE)
- Upward compatible
    - Programs for SIC can run on SIC/XE

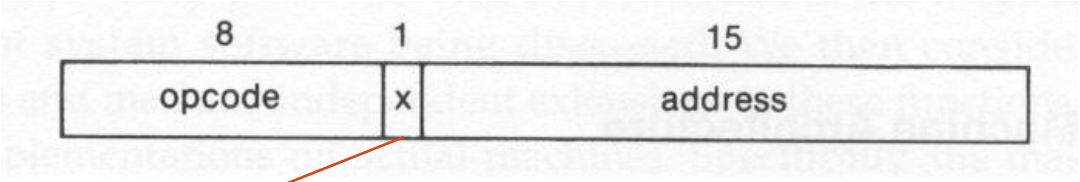
SIC Machine Architecture

- Memory
  - 2^{15} (32,768) bytes in the computer memory
  - 3 consecutive bytes form a word
  - 8-bit bytes
- Registers

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
  - Integers are stored as 24-bit binary numbers; 2' s complement representation is used for negative values
  - 8-bit character support.
  - No floating-point hardware
- Instruction Formats



x: indicate indexed-addressing mode

- Addressing Modes

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

() are used to indicate the content of a register.

SIC Machine Architecture

- Instruction Set
  - load and store: LDA, LDX, STA, STX, etc.
  - integer arithmetic operations: ADD, SUB, MUL, DIV, etc.
    - All arithmetic operations involve 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 to indicate the result

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
  - The Test Device (TD) instruction tests whether the addressed device is ready to send or receive a byte of data
  - Read Data (RD)
  - Write Data (WD)

SIC Programming Examples

Ex. Data movement

```
        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
```

Address labels

Assembler directives for defining storage

-- Arithmetic operation

```
        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
```

BETA=ALPHA+INCR-ONE
DELTA=GAMMA+INCR-ONE

(a) All arithmetic operations are performed using register A, with the result being left in register A.

-- Looping and indexing

```
        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)

```
        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
```

GAMMA[I]=ALPHA[I]+BETA[I]
I=0 to 100

SIC/XE Machine Architecture

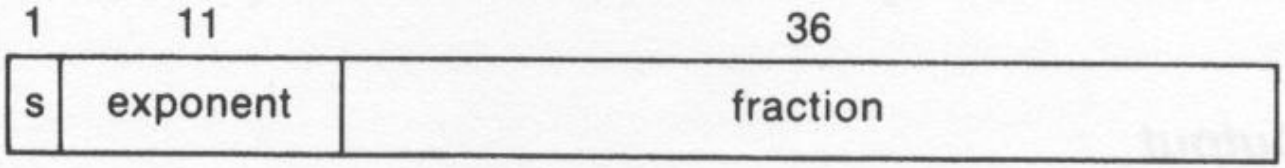
- Memory
  - 2^{20} bytes in the computer memory
- More Registers

Mnemonic	Number	Special use
B	3	Base register; used for addressing
S	4	General working register—no special use
T	5	General working register—no special use
F	6	Floating-point accumulator (48 bits)

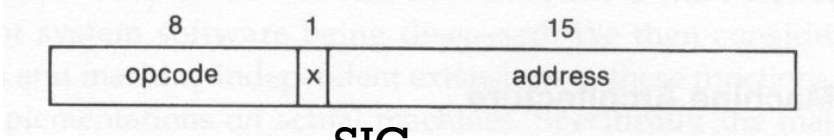
SIC/XE Machine Architecture

- Data Formats
  - Floating-point data type: frac*$2^{(exp-1024)}$
    - frac: 0~1
    - exp: 0~2047

For normalized floating-point numbers,
the high-order bit must be 1.

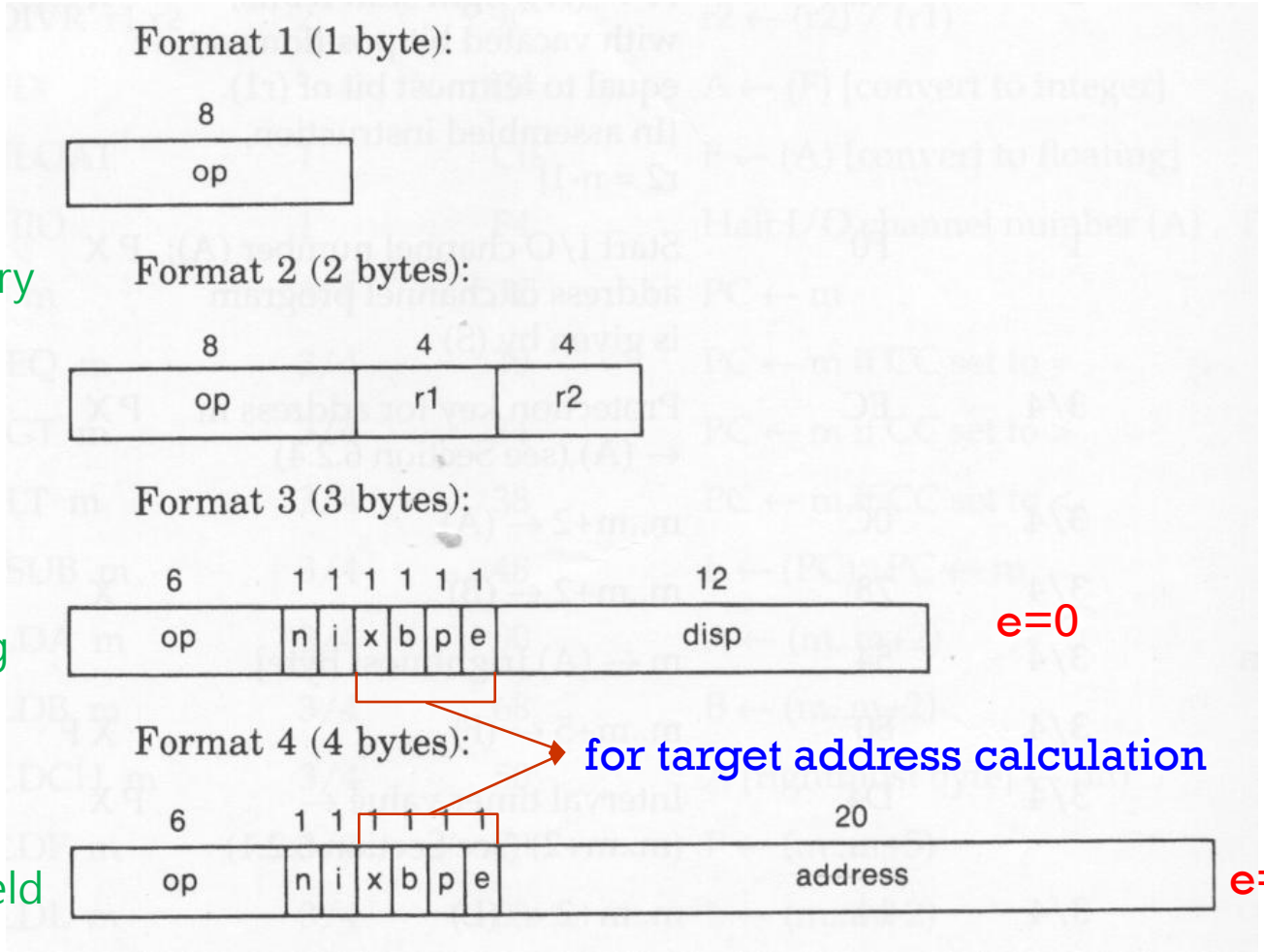


SIC/XE Machine Architecture (3)



SIC

- Instruction formats



No memory reference

Relative addressing

Extended address field

e=0

for target address calculation

e=1

SIC/XE Machine Architecture

- Addressing modes:
  - two new relative addressing for format 3

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

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)

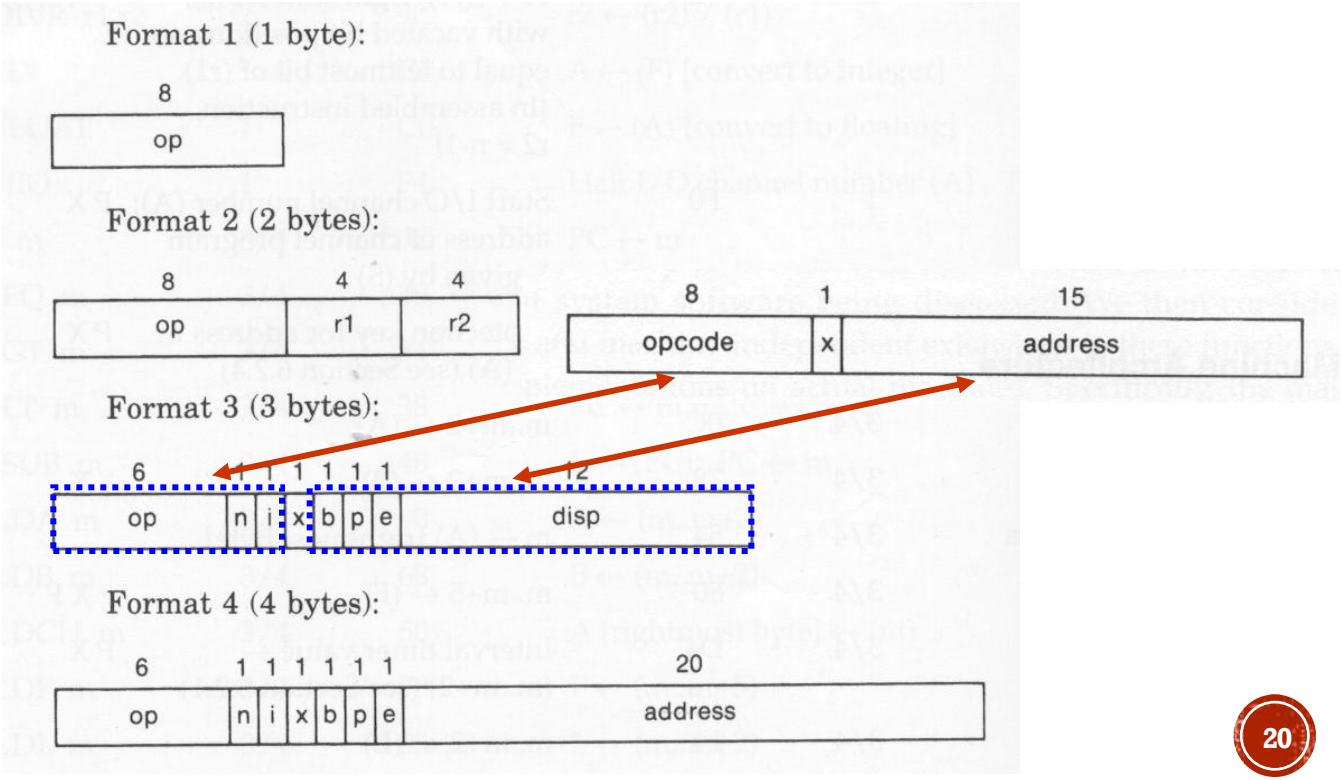
  - Direct addressing for formats 3 and 4 if b=p=0
  - Indexed addressing can be combined if x=1:
    - the term (x) should be added

SIC/XE Machine Architecture

- Bits x,b,p,e: how to calculate the target address
  - relative, direct, and indexed addressing modes
- Bits i and n: how to use the target address (TA)
  - i=1, n=0: immediate addressing
    - TA is used as the operand value, no memory reference
  - i=0, n=1: indirect addressing
    - The word at the TA is fetched
    - Value in this word is taken as the address of the operand value
  - i=0, n=0 (in SIC), or
  - i=1, n=1 (in SIC/XE): simple addressing
    - TA is taken as the address of the operand value
- Any of these addressing modes can also be combined with indexed addressing.

SIC/XE Machine Architecture

- For upward compatibility
  - 8-bit binary codes for all SIC instructions end in 00
  - If n=i=0, bits b,p,e are considered as part of the 15-bit address field



SIC/XE Machine Architecture

- How to compute TA?

Mode	**Indication**	**Target address calculation**	operand
Base relative	b=1, p=0	TA=(B)+disp (0<=disp<=4095)	(TA)
PC-relative	b=0, p=1	TA=(PC)+disp (-2048<=disp<=2047)	(TA)
Direct	b=0, p=0	TA=disp (format 3) or address (format 4)	(TA)
Indexed	x=1	TA=TA+(X)	(TA)

- How the target address is used?

Mode	**Indication**	**operand value**
immediate addressing	i=1, n=0	TA
indirect addressing	i=0, n=1	((TA))
simple addressing	i=0, n=0	SIC instruction (all end with 00)
	i=1, n=1	SIC/XE instruction

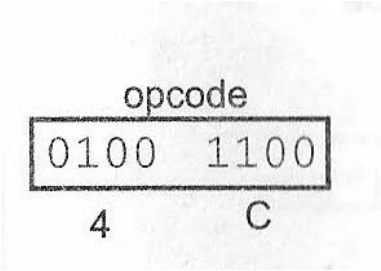
  - Note: Indexing cannot be used with immediate or indirect addressing modes

SIC/XE Machine Architecture

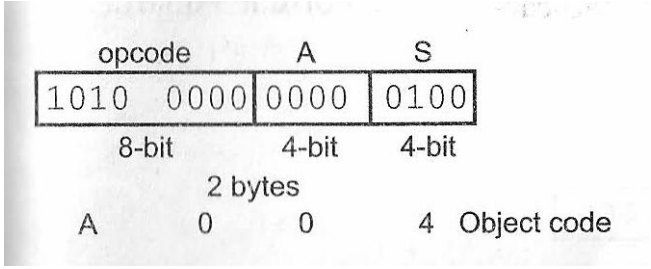
- Instruction Set
  - new registers: LDB, STB, etc.
  - floating-point arithmetic: ADDF, SUBF, MULF, DIVF
  - register move: RMO
  - register-register arithmetic: ADDR, SUBR, MULR, DIVR
- Input/Output
  - SIO, TIO, HIO: start, test, halt the operation of I/O device

SIC/XE Machine Architecture

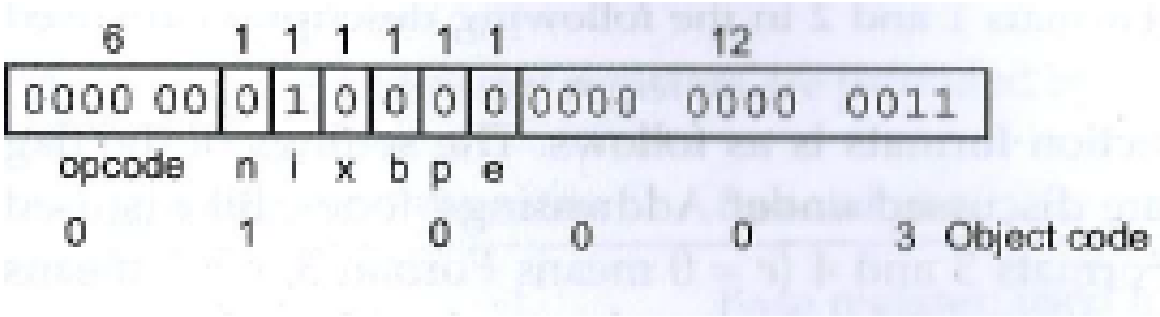
- Example. RSUB



- Example. COMPR A, S

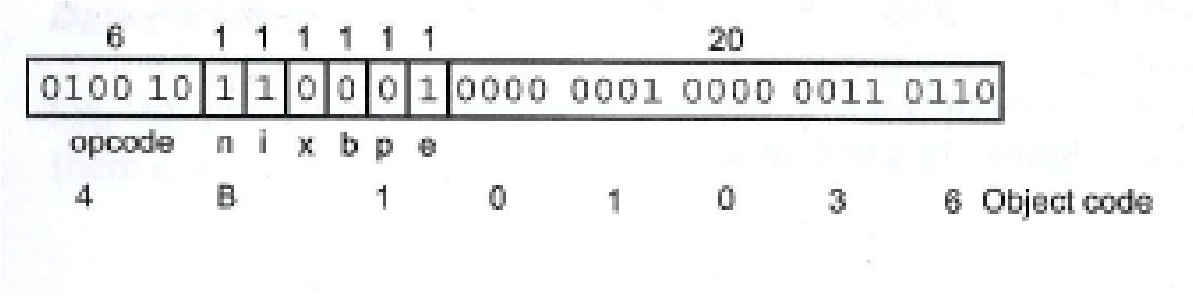


- Example. LDA #3 (Format 3)

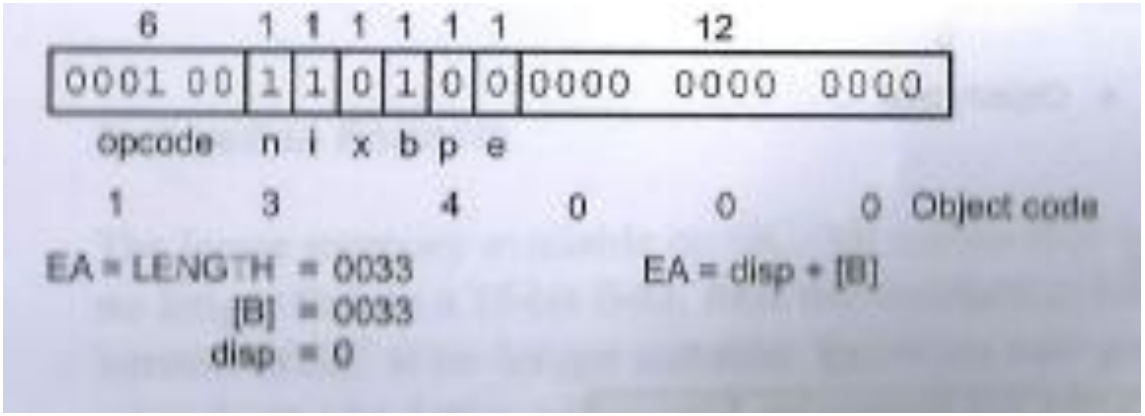


SIC/XE Machine Architecture

- Example. +JSUB RDREC (Format 4)

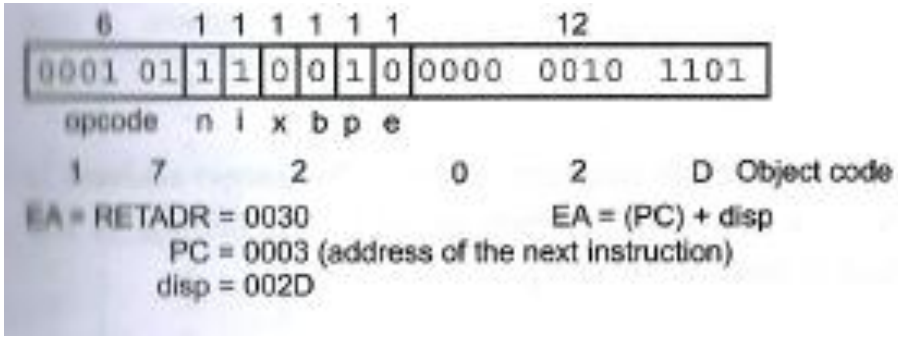


- Example. 1056 STX LENGTH

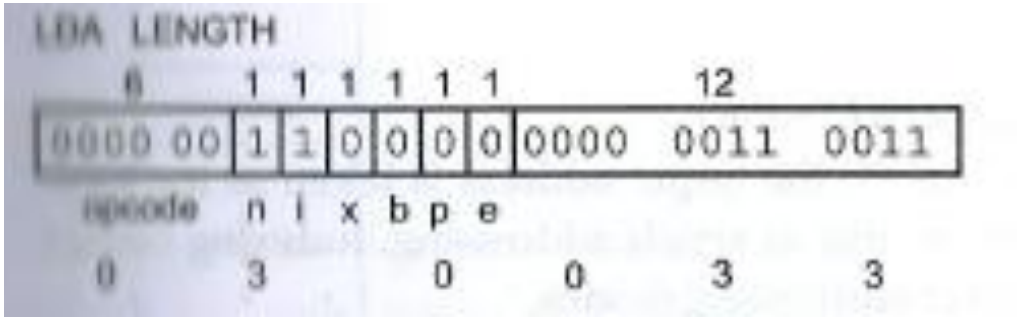


SIC/XE Machine Architecture

- Example. 0000 STL RETADR

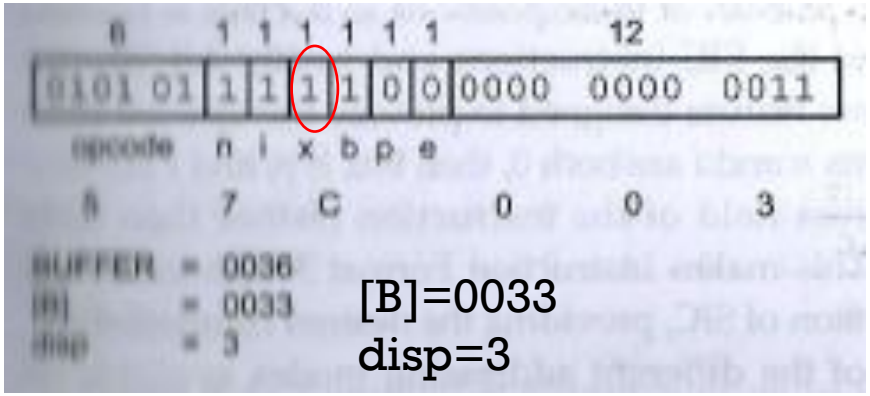


- Example. LDA LENGTH (direct addressing)

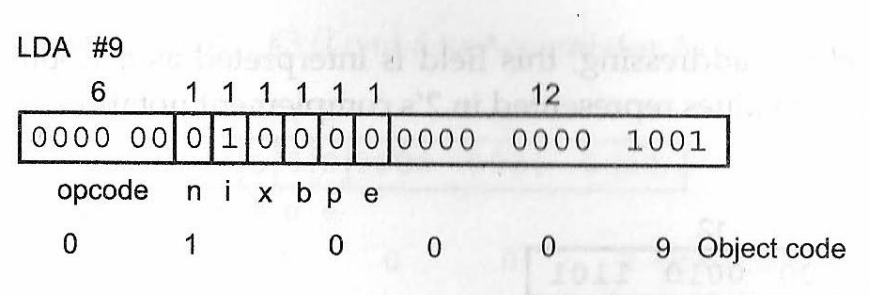


SIC/XE Machine Architecture

- Example. STCH BUFFER, X

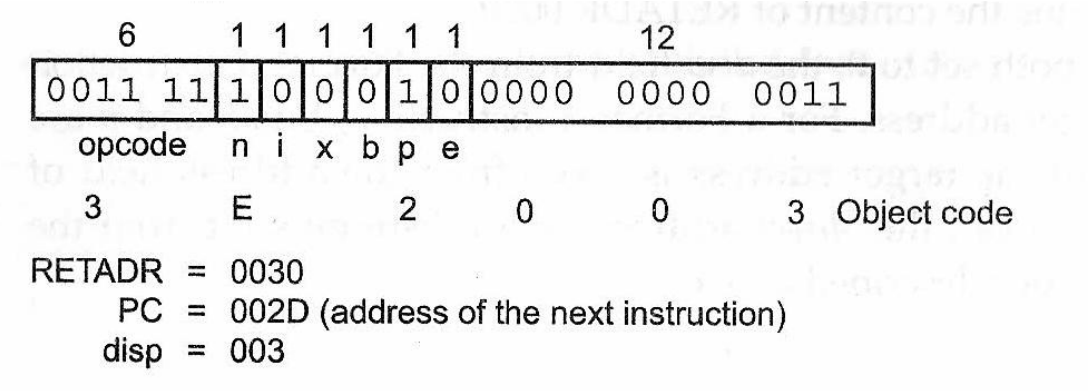


- Example. LDA #9



SIC/XE Machine Architecture

- Example. 002A J @RETADR (indirect 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

c: constant between 0 and 4095
m: memory address or constant larger than 4095

4: Format 4

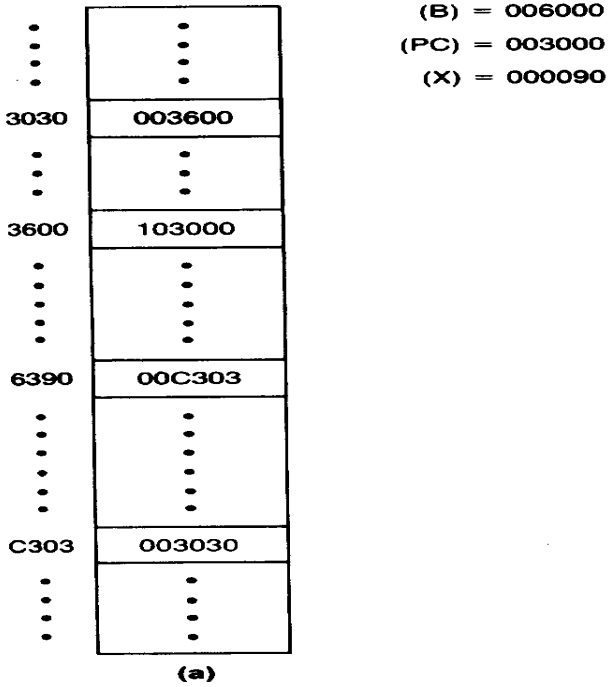
A: Relative addressing

D: Direct addressing

S:Compatible with SIC

SIC/XE Machine Architecture

SIC/XE Machine Architecture



(a)

Machine instruction									Target address	Value loaded into register A
Hex	Binary									
	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

- Instruction set
  - Load and store registers
    - LDA, LDX, STA, STX, LDB, STB, …
  - Integer arithmetic operations
    - ADD, SUB, MUL, DIV, ADDF, SUBF, MULF, DIVF, ADDR, SUBR, MULR, DIVR
  - Comparison COMP
  - Conditional jump instructions (according to CC)
    - JLE, JEQ, JGT
  - Subroutine linkage
    - JSUB
    - RSUB
  - Register move
    - RMO
  - Supervisor call (for generating an interrupt)
    - SVC

SIC/XE Machine Architecture

- Input and output
  - IO device
    - Three instructions:
      - Test device (TD)
      - Read data (RD)
      - Write data (WD)
  - IO channels
    - Perform IO while CPU is executing other instructions
    - Three instructions:
      - SIO: start the operation of IO channel
      - TIO: test the operation of IO channel
      - HIO: halt the operation of IO channel

SIC/XE Machine Architecture I/O Mechanisms

- Polling I/O
- Interrupt-Driven I/O
- DMA (Direct Memory Access) I/O

SIC/XE Instruction

P	Privileged instruction
X	Instruction available only on XE version
F	Floating-point instruction
C	Condition code CC set to indicate result of operation (<, =, or >)

Mnemonic	Format	Opcode	Effect	Notes
ADD m	3/4	18	A ← (A) + (m..m+2)	
ADDF m	3/4	58	F ← (F) + (m..m+5)	X F
ADDR r1,r2	2	90	r2 ← (r2) + (r1)	X
AND m	3/4	40	A ← (A) & (m..m+2)	
CLEAR r1	2	B4	r1 ← 0	X
COMP m	3/4	28	(A) : (m..m+2)	C
COMPF m	3/4	88	(F) : (m..m+5)	X F C
COMPR r1,r2	2	A0	(r1) : (r2)	X C
DIV m	3/4	24	A ← (A) / (m..m+2)	
DIVF m	3/4	64	F ← (F) / (m..m+5)	X F
DIVR r1,r2	2	9C	r2 ← (r2) / (r1)	X
FIX	1	C4	A ← (F) [convert to integer]	X F
FLOAT	1	C0	F ← (A) [convert to floating]	X F
HIO	1	F4	Halt I/O channel number (A)	P X

X: only for XE

C: set CC

F: floating-point

P: privileged

J m	3/4	3C	PC ← m	
JEQ m	3/4	30	PC ← m if CC set to =	
JGT m	3/4	34	PC ← m if CC set to >	
JLT m	3/4	38	PC ← m if CC set to <	
JSUB m	3/4	48	L ← (PC); PC ← m	
LDA m	3/4	00	A ← (m..m+2)	
LDB m	3/4	68	B ← (m..m+2)	X
LDCH m	3/4	50	A [rightmost byte] ← (m)	
LDF m	3/4	70	F ← (m..m+5)	X F
LDL m	3/4	08	L ← (m..m+2)	
LDS m	3/4	6C	S ← (m..m+2)	X
LDT m	3/4	74	T ← (m..m+2)	X
LDX m	3/4	04	X ← (m..m+2)	
LPS m	3/4	D0	Load processor status from information beginning at address m (see Section 6.2.1)	P X
MUL m	3/4	20	A ← (A) * (m..m+2)	

for interrupt

Mnemonic	Format	Opcode	Effect	Notes
MULF m	3/4	60	F ← (F) * (m..m+5)	X F
MULR r1, r2	2	98	r2 ← (r2) * (r1)	X
NORM	1	C8	F ← (F) [normalized]	X F
OR m	3/4	44	A ← (A) | (m..m+2)	
RD m	3/4	D8	A [rightmost byte] ← data from device specified by (m)	P
RMO r1,r2	2	AC	r2 ← (r1)	X
RSUB	3/4	4C	PC ← (L)	
SHIFTL r1,n	2	A4	r1 ← (r1); left circular shift n bits. {In assembled instruction, r2 = n–1}	X
SHIFTR r1,n	2	A8	r1 ← (r1); right shift n bits, with vacated bit positions set equal to leftmost bit of (r1). {In assembled instruction, r2 = n-1}	X
SIO	1	F0	Start I/O channel number (A); address of channel program is given by (S)	P X

SSK m	3/4	EC	Protection key for address m ← (A) (see Section 6.2.4)	P X
STA m	3/4	0C	m..m+2 ← (A)	
STB m	3/4	78	m..m+2 ← (B)	X
STCH m	3/4	54	m ← (A) [rightmost byte]	
STF m	3/4	80	m..m+5 ← (F)	X F
STI m	3/4	D4	Interval timer value ← (m..m+2) (see Section 6.2.1)	P X
STL m	3/4	14	m..m+2 ← (L)	
STS m	3/4	7C	m..m+2 ← (S)	X
STSW m	3/4	E8	m..m+2 ← (SW)	P
STT m	3/4	84	m..m+2 ← (T)	X
STX m	3/4	10	m..m+2 ← (X)	
SUB m	3/4	1C	A ← (A) – (m..m+2)	
SUBF m	3/4	5C	F ← (F) – (m..m+5)	X F

Set Storage Key for memory protection

Mnemonic	Format	Opcode	Effect	Notes
SUBR r1,r2	2	94	r2 ← (r2) – (r1)	X
SVC n	2	B0	Generate SVC interrupt. {In assembled instruction, r1 = n}	X
TD m	3/4	E0	Test device specified by (m)	P C
TIO	1	F8	Test I/O channel number (A)	P X C
TIX m	3/4	2C	X ← (X) + 1; (X): (m..m+2)	C
TIXR r1	2	B8	X ← (X) + 1; (X): (r1)	X C
WD m	3/4	DC	Device specified by (m) ← (A) [rightmost byte]	P

SIC/XE Programming Example (1)

```
        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)

SIC/XE Programming Example (2)

```
        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)

SIC/XE Programming Example (3)

```
        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)

SIC/XE Programming Example (4)

```
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.