Spring 2020

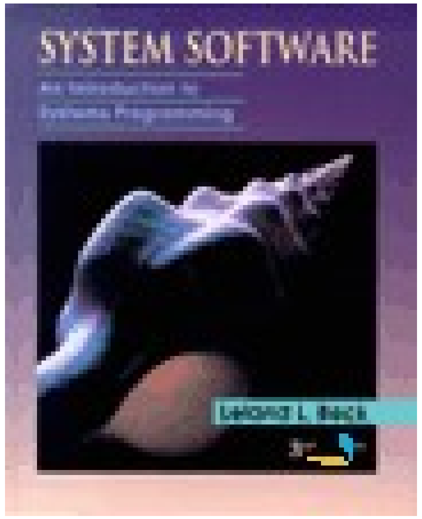
Systems Programming

Jaeyoung Choi

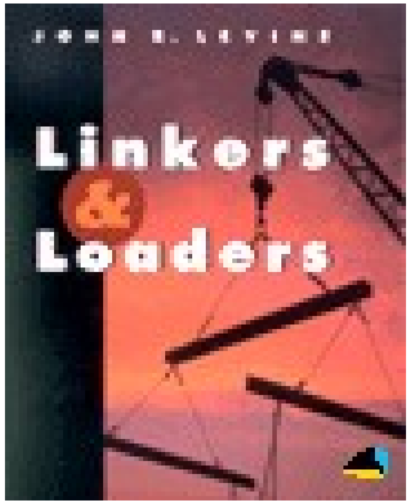
choi@ssu.ac.kr

Text

- System Software 3rd Edition
  - Leland L. Beck
  - Addison-Wesley, 1997



- Linkers & Loaders
  - John & Levine
  - Morgan Kaufman, 2000



Spring 2020

Chap.1 Background

Jaeyoung Choi

choi@ssu.ac.kr

Chap.1 Background

1. Introduction
2. Machine Architecture
3. SIC
4. CISC
5. RISC

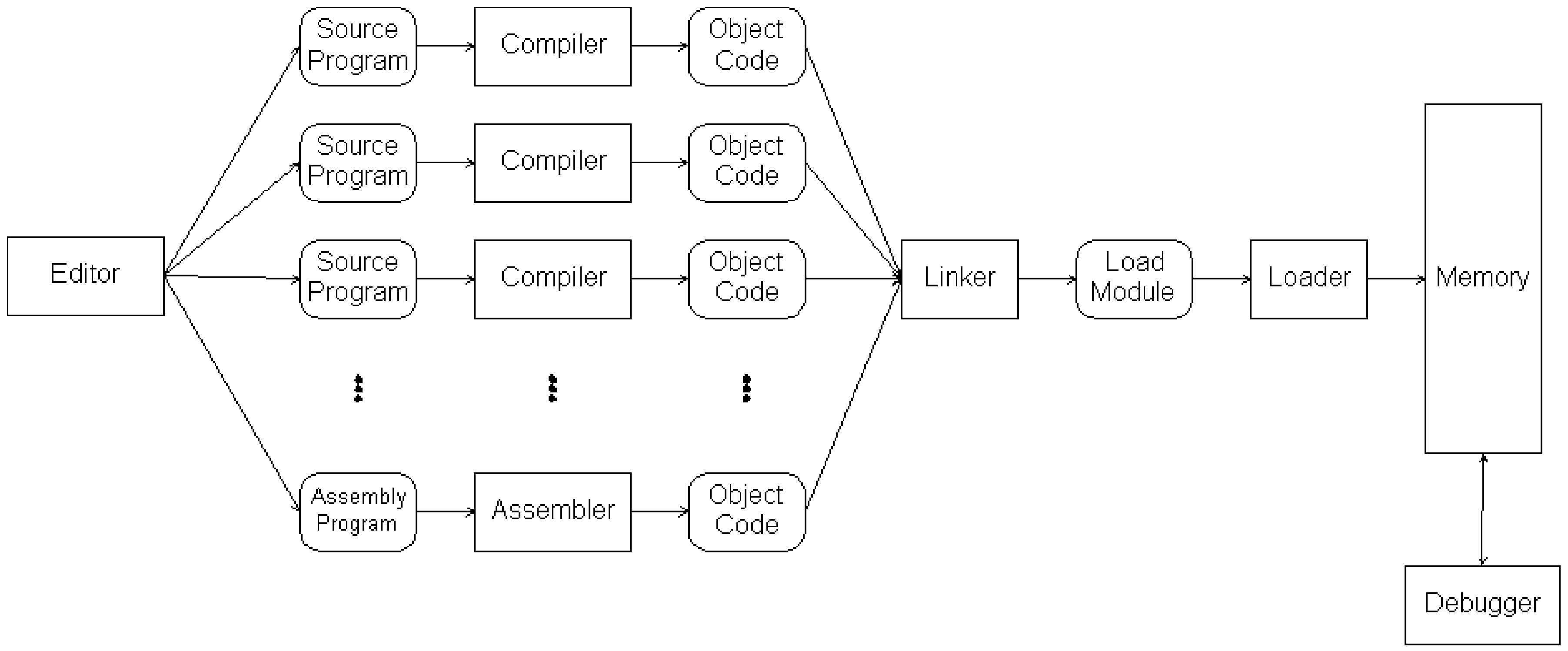
Introduction

- **What is System Software** ?
  - consists of a variety of programs
    that support the operation of a computer
  - support the operation and use of the computer itself,
    rather than any particular application
  - related to the **architecture** of a computer

Introduction

- Examples
  - Assembler
    - Translate mnemonic instructions into machine code;
      - ✓ the instruction formats, addressing modes, etc
  - Compiler
    - Generate machine language code
      - ✓ Taking into account such H/W characteristics
        as the number and type of registers
        and the machine instructions available
  - Operating system
    - the management of all of the resources of a computing system

How to Programming ?



Purpose of the Class

- Learn how to design and implement system software
  - Assembler
  - Loader/Linker
  - Macro processor
  - etc.

The study of Translators will involve

- Symbolic Languages
- Machine Languages
- Computer Organization/Computer Architecture
- Software Engineering

The Computer Architecture

- Memory
- Registers
- Data Format
- Instruction Format
- Addressing Mode
- Instruction Set
- Input/Output

SIC(/XE) Machine Architecture

- SIC (Simplified Instructional Computer)
  - Most commonly encountered H/W features and concepts,
    while avoiding most of the idiosyncrasies
    that are often found in real machines

- SIC/XE
  - XE version of **SIC** Machine
    - Extra Equipment
    - Extra Expensive

Memory

- Structure
  - 1 byte - 8 bit
  - 1 word - 3 bytes (24 bits)

  - All addresses on SIC are byte address
    - Words are addressed by lowest numbered byte

  - Max memory size - 32,768 (2^{15}) bytes

Registers

- 5 registers
  - all have their own special purpose
  - 24-bit length

- Register name

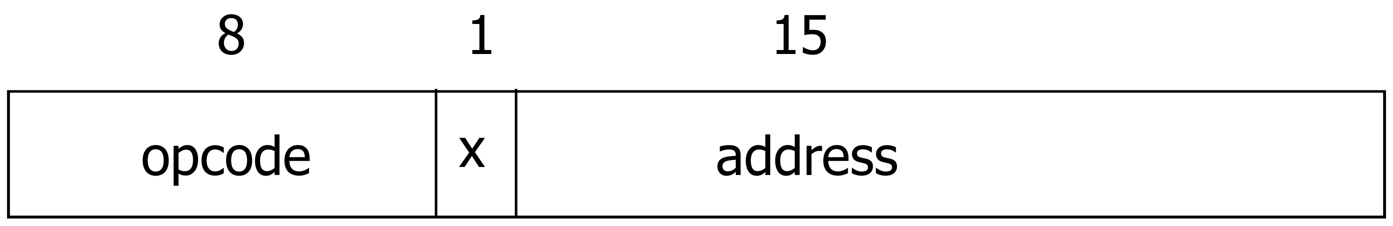
Mnemonic	Number	Special Use
A	0	**A**ccumulator; for arithmetic operations
X	1	Inde**x** register; for addressing
L	2	**L**inkage register; store return address
PC	8	**P**rogram **C**ounter; contain the address of the next instruction
SW	9	**S**tatus **W**ord; contain a variety of info, including a Condition Code (CC)

Data Formats

- Integer: 24-bit binary number
  - 2’s complement representation
  - $-2^{23} \sim 2^{23} - 1$
- Character: 8-bit ASCII code
  - See Appendix B
- No floating-point H/W on SIC

Instruction Formats

- All instructions on SIC have 24-bit format (3 bytes, 1 word)
  - Opcode - First 8 bits
  - Index Code - Next bit
  - Address - Last 15 bits



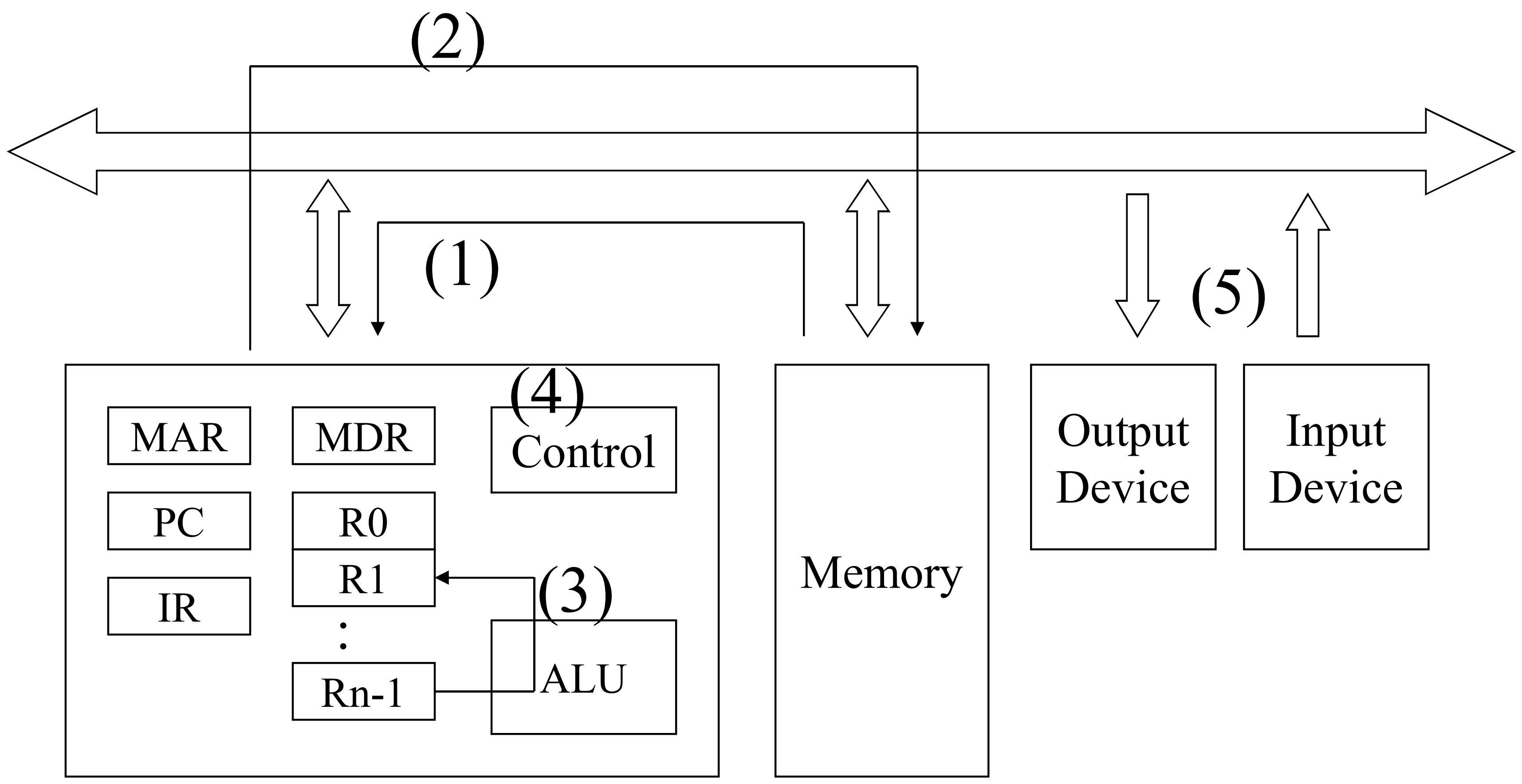
x: indicate indexed-addressing mode

Addressing Modes

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

TA : Target Address
(X) : the contents of register X

Basic Structure of the Computer System



Instruction Set

- Load & Store
  - (1) `LDA, LDX, LDL, LDCH`
  - (2) `STA, STX, STL, STCH, STSW`
- Arithmetic & Logical
  - (3) `ADD, SUB, MUL, DIV, AND, OR, TIX`
- Compare
  - (4) `COMP` – compare the value in A with a word in memory & set a condition code CC to indicate the result(<, =, >)
- Control
  - (4) `(J), JLT, JEQ, JGT` – conditional jump
  - (4) `JSUB, RSUB` – jump to/return from subroutine in L
- I/O
  - (5) `TD, RD, WD`

See Appendix A for a complete list of SIC instructions

Input and Output

- Transfer one byte at a time to/from the rightmost 8 bits of A
  - **TD** (Test Device) tests whether the addressed device ready to send/receive a byte of data.

    The result of this test is:

    $<$ means ready to send/receive

    $=$ means device busy

    $>$ means device not operational
  - **RD** (Read Data) m
  - **WD** (Write Data) m

SIC/XE Machine Architecture

- Max memory size : 1 Mbyte (2^{20} bytes)
- Registers (Additional)

Mnemonic	Number	Special Use
B	3	**B**ase register; used for addressing (24 bits)
S	4	General working register (24 bits)
T	5	General working register (24 bits)
F	6	**F**loating-point accumulator (48 bits)

Data Formats (SIC/XE)

1	11	36
S	exponent	fraction

- 48-bit floating-point data type
  - S: sign of fraction (0=positive, 1=negative)
  - Fraction: the high-order bit of the fraction must be 1
    - $0 \le \text{fraction} < 1$
  - Exponent: unsigned binary number (0 ~ 2047)

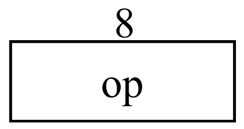
  - Absolute value = fraction * $2^{(\text{exp} - 1024)}$
  - Value of zero: setting all bits to 0

Instruction Formats (SIC/XE - I)

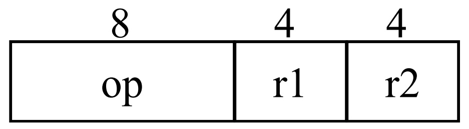
- How to support 1 Mbyte addressing
  - 2 possible options
    - Use relative addressing
    - Extend the address field to 20 bits

Instruction Formats (SIC/XE - II)

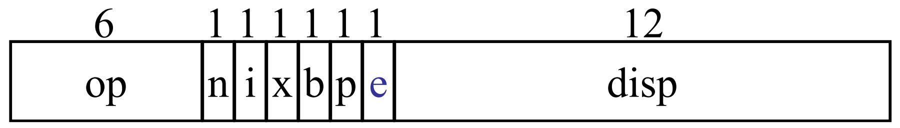
Format 1 (1 byte)



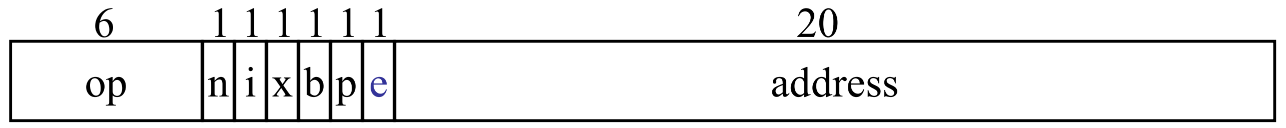
Format 2 (2 byte)



Format 3 (3 byte) (e=0)



Format 4 (4 byte) (e=1)



Addressing Modes (SIC/XE - I)

Format 3

Mode	Indication	Target address calculation
Base relative	b = 1, p = 0	TA = (B) + disp $(0 \leq \text{disp} \leq 4095)$
PC relative	b = 0, p = 1	TA = (PC) + disp $(-2048 \leq \text{disp} \leq 2047)$
Direct mode	b = 0, p = 0	TA = disp

Format 4

Direct mode	b = 0, p = 0	TA = Address (20 bits)	*direct addressing*

These addressing modes can be combined with *indexed addressing* mode
- if bit x is set to 1, (X) is added in the target address calculation

Addressing Modes (SIC/XE - II)

n=0, i=1 $\longrightarrow$ Immediate addressing

No memory reference

- target address is used as the operand value

n=1, i=0 $\longrightarrow$ Indirect addressing

Memory reference

- value in the target address is taken

$$\left.\begin{array}{l}\text{n=0, i=0}\\ \text{n=1, i=1}\end{array}\right\}$$ Simple addressing

Target address is taken

n=0, i=0 in SIC,
n=1, i=1 in SIC/XE

e=0 → Format 3
e=1 → Format 4

Indexing cannot be used with *immediate* or *indirect* addressing modes

Instruction Set (SIC/XE)

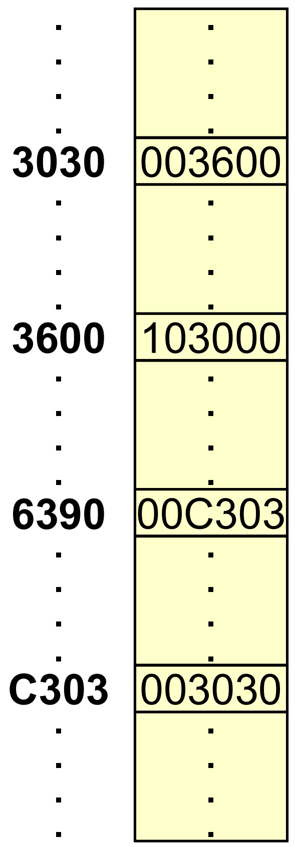
- SIC instruction set
  - Load & Store for new registers
    - LDB, STB, etc
  - Floating Point Arithmetic
    - ADDF, SUBF, MULF, DIVF
  - Register instruction
    - RMO, ADDR, SUBR, MULR, DIVR
  - Supervise Call
    - SVC

See Appendix A for a complete list of all SIC/XE instructions

Input and Output (SIC/XE)

- Instructions for I/O channels
  - Used to perform I/O while CPU is executing
    - Overlap CPU computing with I/O – more efficient operation
  - SIO (Start I/O)
  - TIO (Test I/O)
  - HIO (Halt I/O)

Example



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

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

Fig 1.1 Example of SIC/XE instructions and addressing modes

SIC Program Example (I-a)

- Data move operations
  (ALPHA=5, C1='Z')

```
        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 VARIBALE
```

Fig 1.2(a) Sample data movement operation for SIC

SIC Program Example (I-b)

```
       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 VARIBALE
```

Using immediate addressing

Fig 1.2(b) Sample data movement operation for SIC/XE

SIC Program Example (II-a)

- Arithmetic operations

  (BETA=ALPHA+INCR-1, DELTA=GAMMA+INCR-1)

```
        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
```

Fig 1.3(a) Sample arithmetic operations for SIC

SIC Program Example (II-b)

```
          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
```

Fig 1.3(b) Sample arithmetic operations for SIC/XE

SIC Program Example (III-a)

- Looping and indexing operations
  (STR2='TEST STRING'(STR1))

```
        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
```

Fig 1.4(a) Sample looping and indexing operations for SIC

SIC Program Example (III-b)

```
       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
```

Fig 1.4(b) Sample looping and indexing operations for SIC/XE

SIC Program Example (IV-a)

- Indexing and looping operations

  (GAMMA[I] = ALPHA[I] + BETA[I])

```
        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
```

Fig 1.5(a) Sample indexing and looping operations for SIC

SIC Program Example (IV-b)

```
        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
```

Fig 1.5(b) Sample indexing and looping operations for SIC/XE

SIC Program Example (V)

- I/O operations

  (RECORD ← F1 device)

```
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
```

Fig 1.6 Sample input and output operations for SIC

SIC Program Example (VI-a)

- Subroutine call and record input operations
  (RECORD ← F1 device)

```
        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
```

Fig 1.7(a) Sample subroutine call and record input operations for SIC

SIC Program Example (VI-b)

```
        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 REIGSTER 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 THA 100
        RSUB                      EXIT FROM SUBROUTINE
        .
        .
INDEV   BYTE    X'F1'             INPUT DEVICE NUMBER
RECORD  RESB    100               100-BYTE BUFFER FOR INPUT RECORD
```

Fig 1.7(b) Sample subroutine call and record input operations for SIC/XE

Classes of the Computer Architecture

- CISC - Complex Instruction Set Computers
  - VAX, Intel x86
- RISC - Reduced Instruction Set Computers
  - UltraSPARC, PowerPC, Cray T3E