System Software

Course Code: 17ECSC302

Credits: 3

Semester: V

School of Computer Science & Engineering

Chapter 01

Introduction to a Machine Architecture

Introduction

Practical computer systems divide software into two major classes:

1) Application Software
2) System Software

System Software:

- System software consists of a variety of programs that support the operation of a computer .
- To study "behind the scene"

Simplified Instructional Computer (SIC)

- **SIC** is a **hypothetical** computer that includes the hardware features most often found on real machines
- **Two** versions of SIC
  - standard model
  - extension version (XE version)

SIC Machine Architecture

Features

- Memory
- Register
- Data Format
- Instruction Format
- Addressing Mode
- Input/output
- Instruction Set

SIC Machine Architecture

Memory

- 8-bit bytes
- 3 consecutive bytes form a word (24 bits)
- All addresses in SIC are byte addresses
- 2^{15} bytes =32768 bytes in the computer memory

SIC Machine Architecture

- Registers

  There are 5 registers all of 24 bits in length

SIC Machine Architecture

- **Data Formats**
  - Integers are stored as 24-bit binary numbers
  - 2's complement representation is used for negative values
  - 8 bit ASCII code for characters
  - No floating-point hardware

SIC Machine Architecture

Instruction Formats

- All instructions are of 24 bit format
- Addressing Modes

SIC Machine Architecture

- **Instruction Set**
  - Data transfer group
  - Arithmetic group
  - Logical group
  - Branch group
  - Machine group

SIC Machine Architecture

- **Data transfer Instructions**

  LDA-load data into accumulator

  LDX- load data into index register

  LDL-load data into linkage register

  LDCH-load char into accumulator

  STA-store the contents of A into the memory

  STX-store the contents of X into the memory

  STL-store the contents of L into the memory

  STSW-store the contents of SW into the memory

SIC Machine Architecture

Arithmetic group of Instructions

- ADD
- SUB
- MUL
- DIV
- All arithmetic operations involve register A and a word in memory, with the result being left in the register

SIC Machine Architecture

Logical group of Instructions

- AND
- OR

Both involve register A and a word in memory, with the result being left in the register. Condition flag is not affected.

- 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

Branch group of Instructions

- Conditional jump instructions
- Unconditional jump instructions
- Subroutine linkage

SIC Machine Architecture

Instruction Set

- Conditional jump instructions:
  - JLT
  - JEQ
  - JGT
  - these instructions test the setting of CC (<.=.>)and jump accordingly.

SIC Machine Architecture

Instruction Set

- Unconditional jump instructions:
- J
  - This instruction without testing the setting of CC , jumps directly to assigned memory.

SIC Machine Architecture

Subroutine linkage:

- JSUB
  - JSUB jumps to the subroutine, placing the return address in register L
  - (L □ PC , PC □ subroutine address)
- RSUB
  - RSUB returns by jumping to the address contained in register L
  - (PC □ 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.

- Test Device TD instruction tests whether the addressed device is ready to send or receive a byte of data.

  if CC='<' the device is ready to send or receive.

  if CC='=' the device is not ready to send or receive.

SIC Machine Architecture

- **Input and Output**
  - Read Data RD
    - Data from the input device specified by the memory is read into A lower order byte.
  - Write Data WD
    - Data is sent to output device specified by the memory.

SIC Machine Architecture

- **Input and Output**
  - TIX
    - Increments the content of X and compares its content with memory
    - Depending on the result the conditional flags are updated

      if (X) < (m) then CC = '<'

      if (X) = (m) then CC = '='

      if (X) > (m) then CC = '>'

SIC/XE Machine Architecture

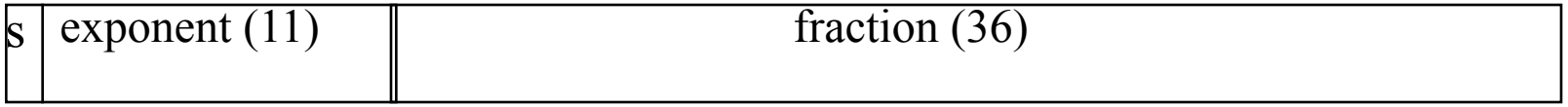
- **Memory**
  - Memory structure is same as that for SIC.
  - 2^{20} bytes in the computer memory.
  - This increase leads to a change in instruction format and addressing modes.

SIC/XE Machine Architecture

- More Registers

SIC/XE Machine Architecture

- **Data Formats**
  - Same data format as that of SIC
  - Floating-point data type of 48 bits



    - frac: 0~1
    - exp: 0~2047
    - S(0=+ve , 1=-ve)

The absolute value of the number is $frac*2^{(exp-1024)}$

SIC/XE Machine Architecture

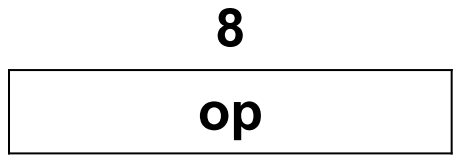
- **Instruction Formats**
  - The instruction format of SIC/XE is modified to suit the changes made in the hardware such as:
    - Enhancing the number of address lines
    - Increasing the number of registers
    - Providing floating point accumulator

SIC/XE Machine Architecture

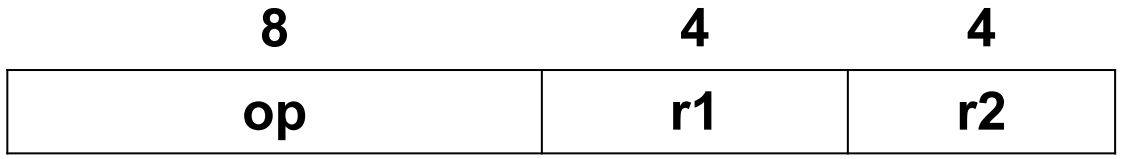
- **Instruction Formats**

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

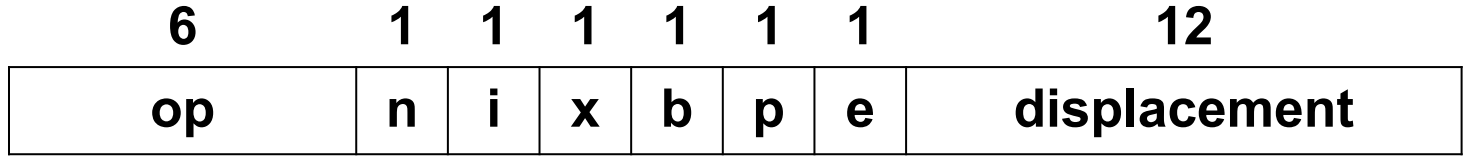
Format 1 (1 byte)



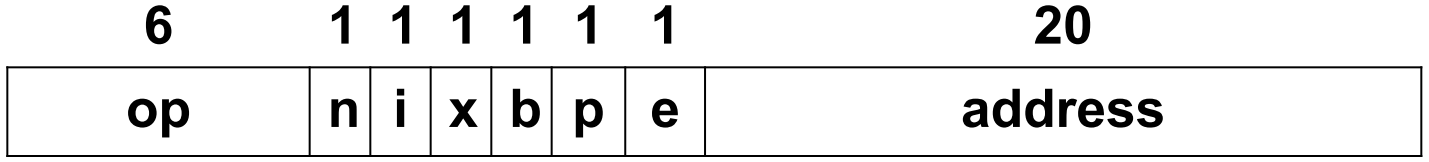
Format 2 (2 bytes)



Format 3 (3 bytes)



Format 4 (4 bytes)



SIC/XE Machine Architecture

- The Format 3 and Format 4 instructions have 6 flag bits:-

  n – indirect addressing

  I – immediate addressing

  x – indexed addressing

  b – base relative

  p – PC relative

  e – (0 – Format 3 1 – Format 4)

SIC/XE Machine Architecture

Addressing modes

Base relative (n=1, i=1, b=1, p=0)

Program-counter relative (n=1, i=1, b=0, p=1)

Direct (n=1, i=1, b=0, p=0)

Immediate (n=0, i=1, x=0)

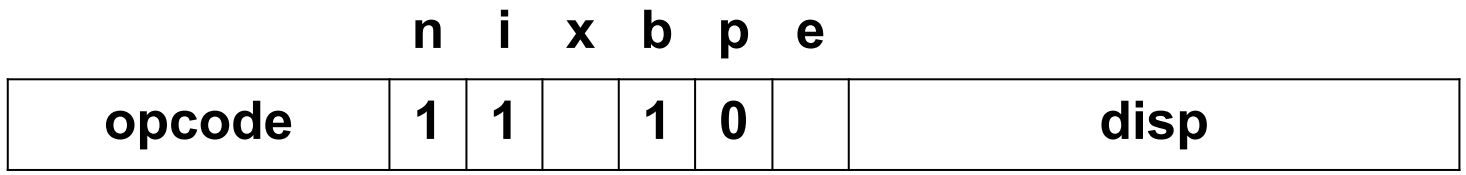
Indirect (n=1, i=0, x=0)

Indexing (both n & i = 0 or 1, x=1)

Extended (e=1)

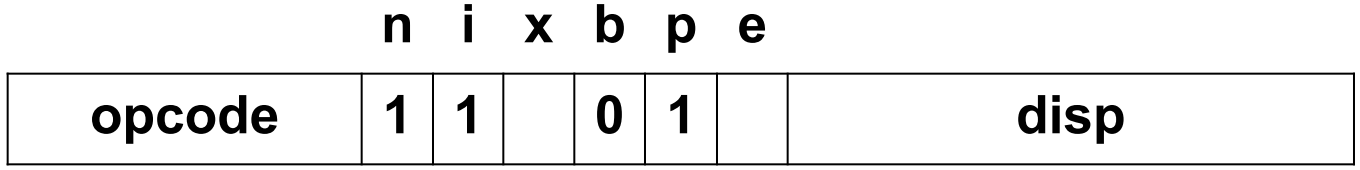
SIC/XE Machine Architecture

- **Base Relative Addressing Mode (STCH BUF,X)**



n=1, i=1, b=1, p=0, TA=(B)+disp (0≤disp ≤4095)

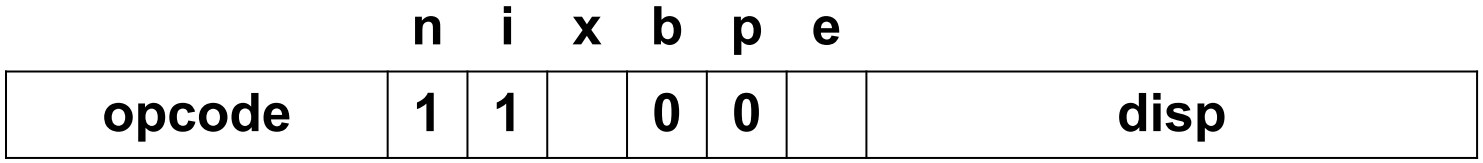
- PC Relative Addressing Mode (J Next)(-ve=2's comp



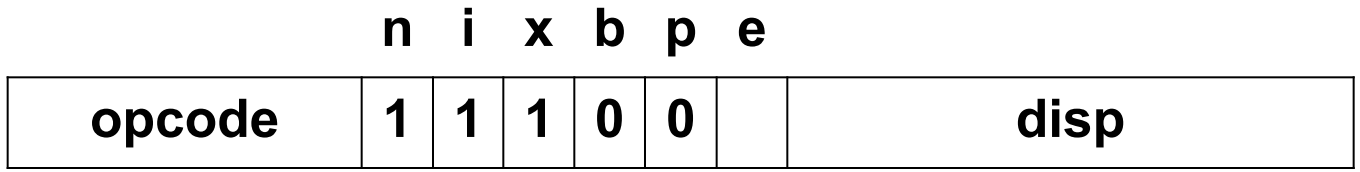
n=1, i=1, b=0, p=1, TA=(PC)+disp (-2048≤disp ≤2047)

SIC/XE Machine Architecture

- Direct Addressing Mode



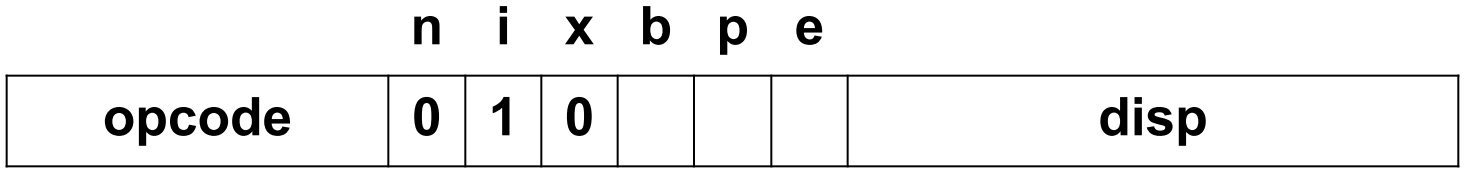
n=1, i=1, b=0, p=0, TA=disp (0≤disp ≤4095)



n=1, i=1, b=0, p=0, TA=(X)+disp (with index addressing mode)

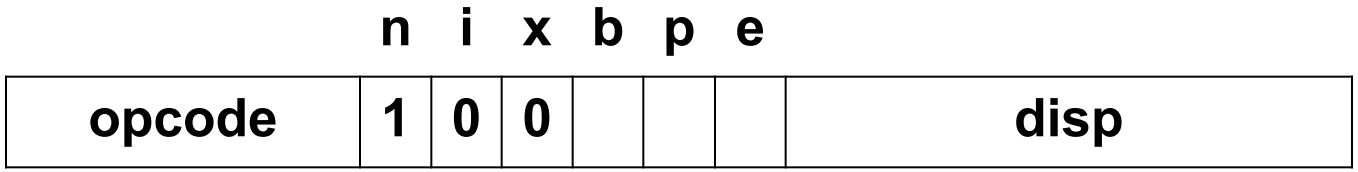
SIC/XE Machine Architecture

- Immediate Addressing Mode (ADD #30)



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

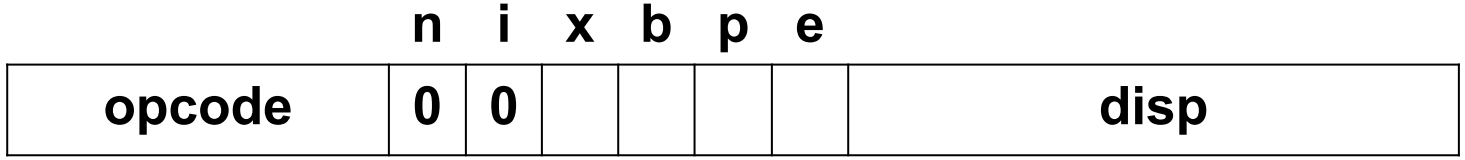
- Indirect Addressing Mode (ADD @2000)



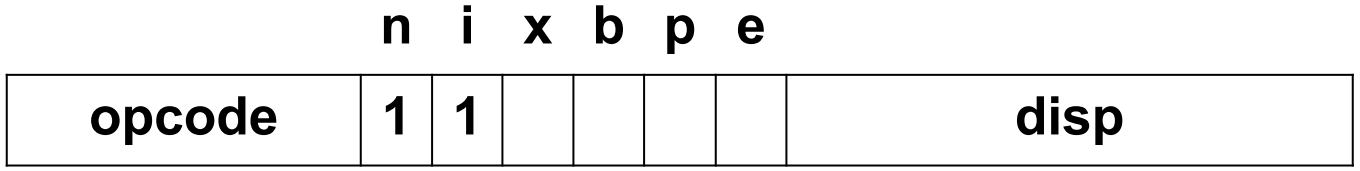
n=1, i=0, x=0, TA=(disp)

SIC/XE Machine Architecture

- Simple Addressing Mode (LDA NUM)



i=0, n=0, TA=bpe+disp (SIC standard)



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

SIC/XE Machine Architecture

- Addressing Modes

SIC/XE Machine Architecture

- **Addressing Modes**

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

Solve the following examples

i.LDA #3

GivenLDA=00H

ii.+JSUB RDREC

JSUB=4B

GIVEN RDREC =1036

Solve the following examples

I. **STX LENGTH**

Given EA=0033,[B]=0033
EA=[B]+DISP

I. **STL RETADR**

Given RETADR=0030 PC=0003
We know EA=[PC]+DISP

SIC/XE Machine Architecture

- **Data transfer Instructions**

  LDB(68)-load data into BASE register

  LDS(6E)- load data into S register

  LPS(D0)-load processor status

  LDT(04)-load data into T register

  LDF(70)-load data into F register

  STB-store the contents of B into the memory

  STS-store the contents of S into the memory

  STL(14)-store the contents of L into the memory

  STSW(E8)-store the contents of SW into the memory

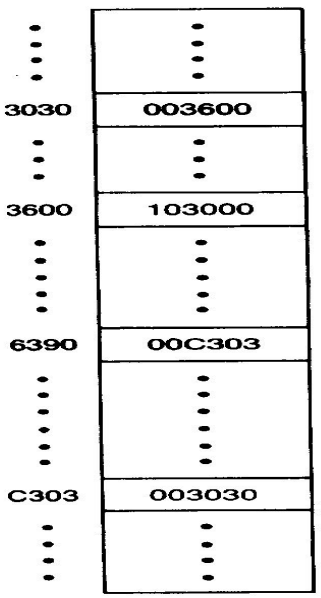
SIC/XE Machine Architecture

- **Instruction Set**
  - new registers: LDB,LDL,LDS, STB, etc.
  - floating-point arithmetic: ADDF, SUBF, MULF, DIVF
  - register move:data transfer from 1 reg to another reg RMO S,A (S□A)
  - register-register arithmetic: ADDR, SUBR, MULR, DIVR
  - Logical :
    - COMPR A,S
    - CLEAR X
    - SHIFTL T,n
    - SHIFTR T,n

SIC/XE Machine Architecture

- **Machine group of instructions**
  - TIO
    - To test I/O channel.
  - HIO
    - To halt the I/O channel.
    - Channel address is provided in A register
  - SIO
    - To start the I/O channel.

Example of SIC/XE instructions and addressing modes



(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 Programming Examples (Fig 1.2a)
To store 5 in ALPHA and z in C1

```
        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 Programming Examples

```
        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.

SIC Programming Example

```
            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
```

(a)

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

SIC/XE Programming Example

```
            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.

SIC Programming Example

to read 1 byte of data from device F1 and copy it to device 05

```
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.

SIC Programming Example

To read 100 bytes of record from an input device into memory

```
            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)

SIC/XE Programming Example

```
          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.

ANY QUESTIONS??