SWE4001 – System Programming

Module 2: Introduction

Lesson 6 of 7: SIC & SIC/XE Programming

SIC Programming Examples

- Data movement
  - No memory-memory move instruction
  - 3-byte word: LDA, STA, LDL, STL, LDX, STX
  - 1-byte: LDCH, STCH
  - Storage definition
    - **WORD/BYTE**

      Reserve one word/byte of storage
    - **RESW/RESB**

      Reserve one or more words/bytes of storage
    - **Example**

ALPHA	**RESW**	**1**
FIVE	**WORD**	**5**
CHARZ	**BYTE**	**C`Z’**
C1	**RESB**	**1**

SIC Programming Examples (Cont.)

- Arithmetic
  - Arithmetic operations are performed using register A, with the result being left in register A
- Looping (TIX)
  - (X)=(X)+1
  - compare with operand
  - set CC

SIC/XE Programming Example

- data movement
  - immediate addressing for SIC/XE
- arithmetic
- Looping (TIXR)
  - (X)=(X)+1
  - compare with register specified
  - set CC

SIC Programming Example

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

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

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

```
        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
```

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

I=0 to 100

SIC Programming Example

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 Programming Example

Subroutine call & record input operations

```
        JSUB    READ        call read subroutine
        .
        .
READ    LDX     ZERO        initialize index register to 0
RLOOP   TD      INDEV       test input device
        JEQ     RLOOP       loop until device is ready
        RD      INDEV       read one byte into register A
        STCH    RECORD,X    store data byte into record
        TIX     K100        add 1 to index, compare to 100
        JLT     RLOOP       loop if “less than”
        RSUB
        .
        .
INDEV   BYTE    X’F1’       input device number
RECORD  RESB    100         100-byte buffer for input record
ZERO    WORD    0
K100    WORD    100
```

SIC/XE Programming Example

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

```
        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

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

```
        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

Subroutine call & record input operations

```
         JSUB    READ        call read subroutine
         .
         .
READ     LDX     #0          initialize index register to 0
         LDT     #100        initialize register T to 100
RLOOP    TD      INDEV       test input device
         JEQ     RLOOP       loop until device is ready
         RD      INDEV       read one byte into register A
         STCH    RECORD,X    store data byte into record
         TIXR    T           add 1 to index, compare to 100
         JLT     RLOOP       loop if “less than”
         RSUB
         .
         .
INDEV    BYTE    X’F1’       input device number
RECORD   RESB    100         100-byte buffer for input record
```

Test your understanding

- Write a sequence of instructions for SIC to set ALPHA equal to the product of BETA and GAMMA. Assume ALPHA, BETA and GAMMA are defined as in slide.
- Write a sequence of instructions for SIC/XE to set ALPHA equal to 4 * BETA – 9. Assume ALPHA and BETA are defined as in slide.

Homework

Write a program for SIC/XE that contains routines. The routines read records from an input device (identified with device code F1) and copies them to an output device (code 05). This main routine calls subroutine RDREC to read a record into a buffer and subroutine WRREC to write the record from the buffer to the output device. Each subroutine must transfer the record one character at a time because the only I/O instructions available are RD and WD.

Homework

```
Program copy {
      save return address;
  cloop:   call subroutine RDREC to read one record;
            if length(record)=0 {
         call subroutine WRREC to write EOF;
      }  else {
     call subroutine WRREC to write one record;
               goto cloop;
            }
            load return address
            return to caller
  }
```

Homework (Cont.)

EOR:
character x‘00’

```
Subroutine RDREC {
        clear A, X register to 0;
 rloop:  read character from input device to A register
         if not EOR {
           store character into buffer[X];
           X++;
              if X < maximum length
                  goto rloop;
     }
         store X to length(record);
         return
}
```

Homework (Cont.)

```
Subroutine WDREC {
            clear X register to 0;
  wloop:    get character from buffer[X]
             write character from X to output device
             X++;
             if X < length(record)
                  goto wloop;
             return
 }
```