Page No.
Date

Sample Subroutine call in SIC

```
      JSUB   READ
       .
       .
       .
READ   LDX    ZERO
RLOOP  TD     INDEV
       JEQ    RLOOP
       RD     INDEV
       STCH   RECORD
       TIX    KTOO
       JLT    RLOOP
       RSUB
       .
       .
INDEV  BYTE   X'F1'   // input device number
RECORD RESB   100     // 100-Byte Buffer for input Record
                      one word constants
ZERO   WORD   0
KTOO   WORD   100
```

Subroutine

SAMPLE PROGRAM

1. Write a sequence of instruction for SIC to set ALPHA equal to the product of BETA & GAMMA. Assume that ALPHA, BETA are defined as in Fig 1.3(a)

Given

```
ALPHA   RESW  1
BETA    RESW  1
GAMMA   RESW  1
```

Pgm

```
      LDA   BETA
      MUL   GAMMA
      STA   ALPHA
ALPHA RESW  1
BETA  RESW  1
GAMMA RESW  1
```

2. Write SIC instn to swap values of ALPHA & BETA

```
      LDA   ALPHA
      STA   GAMMA     t = α
      LDA   BETA      α = β
      STA   ALPHA     β = t
      LDA   GAMMA
      STA   BETA
ALPHA RESW  1
BETA  RESW  1
GAMMA RESW  1
```

Scanned by CamScanner

SIC/XE

(a) **Memory**

— Memory structure same as SIC

— But Maximum memory available on SIC/XE is 1 megabyte (2^{20} bytes)

(b) **Registers**

— Additional registers are provided by SIC/XE

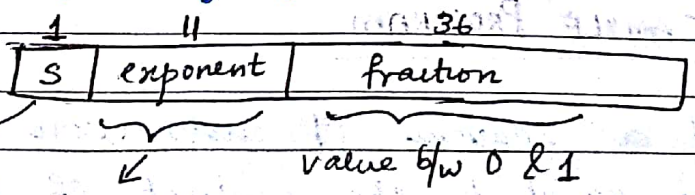
Mnemonic	Number	Special Use
B	3	Base register, used for addressing
S	4	General working register – no special use
T	5	General working register – no special use
F	6	Floating Point accumulator (48 bits)

(c) **Data Format**

int – 24 bit
-ve – 2's comp
char – 8 bit ASCII

— Same data format as SIC standard version

— In addition, there is a 48 bit floating data type with the following format.



S = 0 Positive
S = 1 Negative

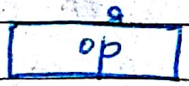
Unsigned binary number b/w 0 & 2047

If exponent 'e' and fraction 'f' then

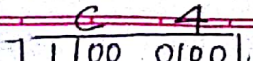
Absolute value of number = $f * 2^{(e-1024)}$

(d) **Instruction format**

(a) **Format 1** (1 byte)

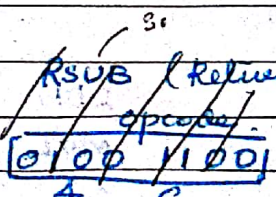


E.g. FIX



opcode = C4 // Fix convert floating point number to integer A ← (F)

E.g. RSUB (Return from subroutine)



Scanned by CamScanner

DIVR R1, R2 ($R_2 \leftarrow R_2 / R_1$)

MULR R1, R2 ($R_2 \leftarrow R_2 * R_1$)

Eg: SUBR R1, R2 ($R_2 \leftarrow R_2 - R_1$)

Format 2 (2 bytes)

8	4	4
op	r1	r2

Eg COMPR A, S

opcode	A	S
1010 0000	0000	0100

A 0 0 4 → object code

format 3 (3 bytes)

6	1	1	1	1	1	1	12
op	n	i	x	b	p	e	disp

Eg a) LDA m (m = any address)

b) LDA #3

0	1	0	0	0	0
1000 000	0	1	0	0	0	0

0 000 0000 0011

LDA = 00

format 4 (4 bytes)

6	1	1	1	1	1	1	20
op	n	i	x	b	p	e	address

Eg

+JSUB RDREC (Extended format)

JSUB = 48 (opcode)

n i x b p e

0100 10	1	1	0	0	0	1	0000 0001	0000 0011 0110

4 B 1 0 1 0 3 6

object code of +JSUB RDREC

Note

'p' used to distinguish b/w format 3 & 4

If e=0 → format 3

e=1 → format 4

format 3 eg

0001 01	1	1	0	0	1	0	0000 0010 1101

Addressing modes

1 7 2 0 2D

P.T.O

```
0000  COPY   START  1000
0000  FIRST  STL    RETADR  17202D
0003         LDB    #LENGTH 69202D
             BASE   LENGTH
0006  CLOOP  +JSUB  RDREC   4B101036
0030  RETADR RESW   1
1036  RDREC  CLEAR  X
```

```
0000 COPY START 0
0000 FIRST STL RETA
0003 RLO
0006 CLOOP +JSUB RDREC
0093 RET CLEA
1036 RDREC CLEAR X
     LEND FIRST
```

10 - A
11 - B
12 - C
13 - D

Opcode of STL = 14

Mnemonic Assembler directive

0030 − 0003 = 2D

(TA) CPC

$\frac{12-10}{2} = \frac{2}{2} = 1$

$\frac{10+12}{2} = \frac{22}{2} = 11$

$\frac{a-b}{2}$

Scanned by CamScanner

Addressing mode

1. Relative addressing modes
   - Base relative
   - Program Counter relative
2. Direct addressing mode

Other addressing modes:

3. Indexed addressing mode
4. Immediate " "
5. Indirect addressing mode
6. Simple addressing mode

1. Relative addressing Mode

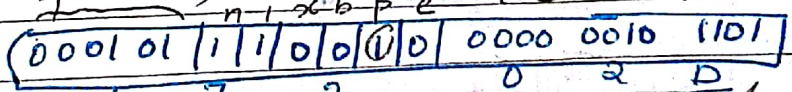
Mode	Indication	Target address Calc	
Base relative	b=1, p=0	TA = (B) + disp	0 ≤ disp ≤ 4095
Pgm Counter relative	b=0, P=1	TA = (pc) + disp	-2048 ≤ disp ≤ 2047

Base relative — displacement field of format 3 is interpreted as 12 bit unsigned integer

P.C relative — displacement field is interpreted as 12-bit signed integer with negative value represented in 2's compliment

Target adde
DISP = TA - PC → Progra counter

opcode (6bit) n i x b p e



Eg

P.C Relative

Item			
10	0000	FIRST	STL RETADR
	0003		
		.	
		.	
	0030	RETADR	

Opcode of STL = 14

17 2D.

DISP = 1251₇

Addres of RETADR = (PC) + displacement
= 0003 + 002D
= 0030

Here (pc) = 0003

Scanned by CamScanner

Page No.
Date

Pg 58 Pgm

Eg Base Relative

```
0036 BUFFER  —    —
1041E    STCH  BUFFER, X    57C003
```

LDB #length → Load value into register B

Calculation Step

(B) = 0033

Disp = 0003

(Buffer) = (B) + Disp

= 0033 + 0003 = 0036

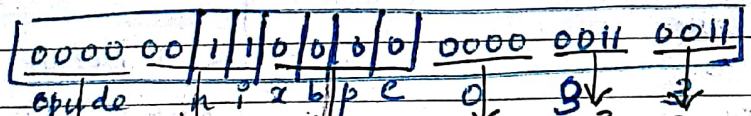
```
0003  LDB #Length
 .
 .
0033  LENGTH  RESW 1
```

II) Direct Addressing

If b & p = 0, then disp field from format 3 is taken to be target address.

for format 4, b=0 and p=0, target address is taken from address field of instn

Eg LDA LENGTH



⇒ 03033

address & length = 0033

address directly specified in the instn

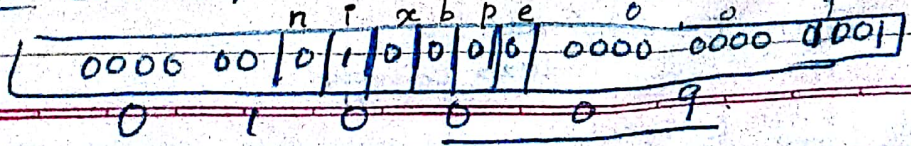
Immediate addressing

— Both i & n in format 3 & 4 are used to specify how the target address is used.

— If i=1 & n=0 ⇒ target address itself used as operand value

No mly reference performed.

Eg LDA #9



Scanned by CamScanner

Indirect addressing mode

If i=0 & n=1, the word at the location given by target address is fetched, the value contained in this word is taken as address of operand value

Eg 002A J @RETADR 3E2003.

Simple addressing

If i & n = 0 or 1, target address is taken as the location of the operand. This is simple addressing [Indexing cannot be used with immediate or indirect addressing mode)

Sample pgms in SIC/XE

I) Simple data movement in SIC/XE

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

II Arithmetic

```
LDS INCR // Load value of INCR into regS
LDA ALPHA
ADDR S,A // Add the value of INCR
SUB #1
STA BETA
LDA GAMMA
ADDR S,A
SUB #1
.
.
ALPH RESW 1
BETA RESW 1
GAMMA " 1
INCR " 1
```

III Looping & Indexing

```
      LDT #11   // Initialize register T to 11
      LDX #0
MOVECH LDCH STR1,X
      STCH STR2,X
      TIXR T    --> // Add 1 to Index compare result to T
      JLT MOVECH
STR1  BYTE C'TEST'  // 11 Byte variable
STR2  RESB 11
```

Scanned by CamScanner

(4) Sample indexing & looping oper^n.

```
        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 to reg 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.
        :
ALPHA   RESW  100     } // Array Variable - 100 WORD EACH -
BETA    RESW  100     |
GAMMA   RESW  100     /
```

Input/output opn

V) READ

```
        LDX   #0      // Subroutine to read 100 byte record.
        LDT   #100    // Initialize index reg to 0
RLOOP   TD    INDEV   // Initialize T to 100.
        JEQ   RLOOP   // Test input device
        RD    INDEV   // loop if device busy.
        STCH  RECORD,X // Read 1 byte into reg A
        TIXR  T       // store data byte into record
        JLT   RLOOP   // Add 1 to Index & compare to 100
        RSUB          // loop if Index less than 100.
        :             // Exit from subroutine
INDEV   BYTE  X'F1'   // Input Device Number
RECORD  RESB  100     // 100-Byte Buffer for i/p record
```

(P.T.O)

Scanned by CamScanner

Sample pgms

1. Write a sequence of instn for SIC/XE to set ALPHA equal to 4 * BETA - 9. Assume ALPHA & BETA defined as in fig 1.3 (b). Use immediate addressing for constant

Given

($\alpha = 4 * \beta - 9$)

```
ALPHA   RESW  1
BETA    RESW  1
```

More details abt addressing mode & SIC/XE instn set Refer pg 495

Pgm

```
        LDS  #4
        LDT  #9
        LDA  BETA
        MULR S,A        // MULR R1 R2 ⇒ R2 = R1 * R2
        SUBR T,A        // SUBR R1 R2 ⇒ R2 ← (R2) - (R1)
        STA  ALPHA
ALPH    RESW 1
BETA    RESW 1
```

OR

```
LDA  BETA
LDS  #4
MULR S,A
SUB  #9
STA  ALPHA
```

(2) Write a sequence of instn for SIC/XE to set ALPHA equal to integer portion of BETA ÷ GAMMA. Assume ALPHA & BETA defined as in 1.3 (a)

```
        LDF  BETA
        DIVF GAMMA      // DIVF m ⇒ F ← F/(m...m+5)
        FIX             // FIX ⇒ A ← (F) // convert to integer
        STA  ALPHA
ALPHA   RESW 1
BETA    RESW 1
```

Scanned by CamScanner

Page No.
Date

4. Write a sequence of instⁿ in SIC to set ALPHA equal integer portion of BETA ÷ GAMMA.

```
      LDA   BETA
      DIV   GAMMA
      STA   ALPHA
       :
ALPHA RESW  1
BETA   "    "
GAMMA  "    "
```

5. Write a sequence of instruction of SIC/XE to divide BETA by GAMMA, setting ALPHA to integer portion of quotent and DELTA to remainder. Use register - register instⁿ to make calculation as efficient as possible.

```
      LDA   BETA          4/3 = 2.5
      LDS   GAMMA
      DIVR  S, A                  6 — 7
      STA   ALPHA.                |
      MULR  S, A  6               N
      LDS   BETA  7
      SUBR  A, S
      STS   DELTA
       :
ALPHA RESW  1
BETA  RESW  1
GAMMA  "    "
DELTA  "    "
```

Scanned by CamScanner

6. Write a sequence of instn for SIC/XE to divide BETA by GAMMA, setting ALPHA to the value of quotient, rounded to nearest integer. Use register to register instn

```
        LDF   BETA
        DIVF  GAMMA
        FIX
        STA   ALPHA
        :
ALPHA   RESW  1
BETA    RESW  1
GAMMA   "     "
```

7. Write a sequence of instn for SIC/XE to clear 20-byte string to blanks

```
        LDX   ZERO
LOOP    LDCH  BLANK
        STCH  STR1,X
        TIX   TWENTY
        JLT   LOOP
        :
STR1    RESW  20
BLANK   BYTE  C' '
ZERO    WORD  0
TWENTY  WORD  20.
```

8. Write sequence of instn for SIC/XE to clear a 20 byte string to all blanks. Use immediate addressing & register-register instn

```
        LDT   #20
        LDX   #0
LOOP    LDCH  #0
        STCH  STR1,X
        TIXR  T
```

```
        :
STR1    RESW  20
        :
```

Scanned by CamScanner

(9) Suppose that ALPHA is an array of 100 words write seq of instn for SIC to set all 100 elements of the array to 0

```
        LDA   ZERO
        STA   INDEX
LOOP    LDX   INDEX
        LDA   ZERO          A:=0
        STA   ALPHA,X
        LDA   INDEX
        ADD   THREE
        STA   INDEX
        COMP  K300
        TIX   TWENTY
        JLT   LOOP
        :
INDE    RESW  1
ALPHA   RESW  100
        :
ZERO    WORD  0
K300    WORD  100
THREE   WORD  3
```

correction in code

(10) Suppose that ALPHA is an array of 100 WORDS. write Sequence of instn for SIC/XE to set all 100 elements of array to 0. Use Immediate addressing

```
        LDS   #3
        LDT   #300
        LDX   #0
LOOP    LDA   #0
        STA   ALPHA,X
        ADDR  S,X
        COMPR X,T
        JLT   LOOP
ALPHA   RESW  100
```

Scanned by CamScanner

11. Suppose that ALPHA & BETA are 2 arrays of 100 words. Another array of GAMMA elements are obtained by multiplying the corresponding ALPHA element by 4 and adding corresponding BETA element

```
        LDS  #3
        LDT  #300
        LDX  #0
ADDLOOP ~~ADDLOOP~~ LDA  ALPHA,X
        MUL  #4
        ADD  BETA,X
        STA  GAMMA,X
        ADDR S,X        // ADDR R1,R2 ⇒ (R2) ← (R1)+(R2)
        COMPR X,T
        JLT  ADDLOOP
        .
        .
ALPHA   RESW 100
BETA    RESW 100
GAMMA   RESW 100
```

12. Suppose ALPHA is an array of 100 words. Write a sequence of instn. for SIC/XE to find the maximum element in an array and store result in MAX

```
        LDS  #3
        LDT  #300
        LDX  #0
CLOOP   LDA  ALPHA,X
        COMP MAX
        JLT  NOCH
        STA  MAX
NOCH    ADDR S,X
        COMPR X,T
        JLT  CLOOP
        .
ALPHA   RESW 100
MAX     WORD -32768
```

Scanned by CamScanner

13. Suppose RECORD contains a 100-byte record. Write a subroutine for SIC that will write this record on device 05

```
        JSUB   WRREC
          :
WRREC   LDX    ZERO
WLOOP   TD     OUTPUT
        JEQ    WLOOP
        LDCH   RECORD,X
        WD     OUTPUT
        TIX    LENGTH
        JLT    WLOOP
        RSUB
          :
ZERO    WORD   0
LENGTH  WORD   1
OUTPUT  BYTE   X'05'
RECORD  RESB   100
```

14. In SIC/XE

```
        JSUB   WRREC
          :
WRREC   LDX    #0
        LDT    #100
WLOOP   TD     OUTPUT
        JEQ    WLOOP
        LDCH   RECORD,X
        WD     OUTPUT
        TIXR   T
        JLT    WLOOP
        RSUB
          :
OUTPUT  BYTE   X'05'
RECORD  RESB   100
```

Scanned by CamScanner

15. Write a subroutine for SIC that will read a record into a buffer. The record may be any length from 1 to 100 bytes. The end of record is marked as "null" character (Ascii code 00). The subroutine should place the length of record read into variable named length

```
        JSUB    RDREC
          :
RDREC   LDX     ZERO
RLOOP   TD      INDEV
        JEQ     RLOOP
        RD      INDEV
        COMP    NULL
        JEQ     EXIT
        STCH    BUFFER,X
        TIX     K100
        JLT     RLOOP
        ~~EXIT    STX LENGTH~~
EXIT    STX     LENGTH
        RSUB
          :
ZERO    WORD    0
NULL    WORD    0
K100    ~~WORD    100~~  => K100  BYTE  100.
INDEV   BYTE    X'F1
LENGTH  RESW    1
BUFFER  RESB    100
```

Scanned by CamScanner

16 Write a subroutine for SIC/XE that will read a record into buffer. The record may be any length from 1 to 100 bytes. The end of record marked with 'null' character (ASCII code 00). The subroutine should place the length of record read into variable named LENGTH. Use immediate addressing & register - register instn

```
      JSUB   RDREC
        :
RDREC LDX    #0
      LDT    #100
      LDS    #0
RLOOP TD     INDEV
      JEQ    RLOOP
      RD     INDEV
      COMPR  A,S
      JEQ    EXIT
      STCH   BUFFER,X
      TIXR   T
      JLT    RLOOP
EXIT  STX    LENGTH
      RSUB
        :
INDEV BYTE   X'F1
LENGTH RESW  1
BUFFER RESB  100
```

Scanned by CamScanner

SAMPLE PGM WITH OBJECT CODE

	0000	COPY	START	0	
(eg for format 3)	0000	FIRST	STL	RETADR	17202D (opcode of STL = 14)
	0003		LDB	#LENGTH	69202D
			BASE	LENGTH	
(eg for format 4)	0006	CLOOP	+JSUB	RDREC	4B101036 (opcode of JSUB = 48)
	000A		LDA	LENGTH	032026
	000D		COMP	#0	290000
	002A		J	@RETADR	3E2003 // indirect addressing
	0030	RETADR	RESW	1	
	0033	LENGTH	RESW	1	
	1036	RDREC	CLEAR	X	B410
(eg for format 2)	1049		COMPR	A,S	A004 (opcode of COMPR = A0)
	104B		JEQ	EXIT	332008

opcode = 3C

-14 = 0001 0100

1110 1011 + 1

A = 10, B = 11, C = 12, D = 13, E = 14

1111 1100

FC

Eg

40	0017	J	CLOOP	3F2FEC
	001A	—	—	
	0006	CLOOP	RESW 1	

0011 1100 1010 1111 1110 1100

3 C 2 F E C

Displace = 0006 − 001A
(T.A) (PC)
= −14

Represent Displacement in 2's complement form

−14 = 0001 0100

= 1110 1011 + 1

1110 1100

FE C

0000 0001 0100

1111 1110 1100

F E C

Scanned by CamScanner