papergrid

Date: / /

System Software

* Simplified Instructional Computer (SIC)
→ Upward compatible
→ Memory consists of 8-bit bytes, 3 consecutive bytes form a word (24 bits)
→ There are a total of 32768 bytes (32 KB) in the computer memory
→ 5 registers, 24 bits in length

A	0	Accumulator
X	1	Index register
L	2	Linkage register (JSUB)
PC	8	Program Counter
SW	9	Status word (Condition Code)

* Data Formats
(no floating point) → Integers are stored as 24-bit binary number
→ 2's complement representation for negative values
→ Characters are stored using 8-bit ASCII codes
→ No floating-point hardware on ~~the stand~~ standard SIC

* Instruction Cycle

Fetch Instruction	Decoder	Fetch Operand	Computation	Store Result

Scanned by CamScanner

* Instruction Format

8	1	15
opcode	x	address

= 24 bits.

x = 0 Direct Address mode
(Target Addr.) TA = address

x = 1 ~~Indirect~~ Indexed
TA = address + (x)
(x) → contents of reg. x

→ No immediate addressing

* Instruction Set

→ Format 3

→ Load and store registers (LDA, LDX, STA, STX, etc.) (from memory to reg.) (CPU to memory)

→ Integer arithmetic operations (ADD, SUB, MUL, DIV)

→ Compare instruction (COMP)

→ Conditional jump instructions (JLT, JEQ, JGT)

→ JSUB jumps to the subroutine, placing the return address ~~to~~ in register L (Linkage)

→ RSUB returns by ~~jm~~ jumping to the address contained in register L.

Scanned by CamScanner

takes only one operand in an instruction

papergrid

Date: / /

* I/O
→ transfers only 1 byte at a time to or from the rightmost 8 bits of register
→ Each device is assigned a unique 8 bit code as an operand
→ Test Device (TD): Tests whether the addressed device is ready to send or receive
  < ready = not ready
Read Data (RD)
Write Data (WD)

1 Write an SIC program to demonstrate load and store instructions
i) to do operation ~~x = 5~~ alpha = 5
ii) ~~char x = z~~ C1 = z

```
FIVE  WORD  5
      LDA   FIVE      load constant 5 into reg. A
      STA   ALPHA     store in alpha
      LDCH  CHARZ     load character 'z' into reg. 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
```

Scanned by CamScanner

papergrid
Date: / /

Q Write seq. of instructions for SIC to set
~~α~~. ALPHA = 4 * BETA - 9

```
four  word  4
nine  word  9
LDA   FOUR
MUL   BETA
SUB   NINE
STA   ALPHA

ALPHA RESW  1
BETA  RESW  1
FOUR  WORD  4
NINE  WORD  9
```

Q Write seq. of instructions in SIC to store the value ALPHA + INCR - 1 in BETA and the value GAMMA + INCR - 1 in DELTA

```
one word 1
LDA  ALPHA
ADD  INCR
SUB  ONE
STA  BETA

LDA  GAMMA
ADD  INCR
```

Scanned by CamScanner

```
        SUB   ONE
        STA   DELTA
ALPHA   RESW  1
BETA    RESW  1
GAMMA   RESW  1
DELTA   RESW  1
ONE     WORD  1
INCR    RESW  1
```

Q. Write an SIC prg to copy one 11 byte character string str1 into another character string str2.

```
        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

STR1    BYTE  C'TEST STRING'
STR2    RESB  11
ZERO    WORD  0
ELEVEN  WORD  11
```

Scanned by CamScanner

papergrid
Date: / /

Q. Write an SIC prg. to clear 20 byte string to all blanks.

```
        ldx     zero
        LDCH    BLANK
label:  STCH    STR1, X
        TIX     TWENTY
        JLT     LABEL

BLANK   BYTE    C' '
STR1    BYTE    C'Test String Movement'
ZERO    WORD    0
TWENTY  WORD    20
```

Q. Write an SIC prg to add the corresponding values of 2 arrays ALPHA and BETA of 100 words each and store the result into another array GAMMA

```
        LDA     ZERO
        STA     INDEX
ADDLP   LDX     INDEX
        LDA     ALPHA, X
        ADD     BETA, X
```

Scanned by CamScanner

papergrid

Date: / /

```
STA GAMMA,X
LDA INDEX
ADD THREE
STA INDEX
COMP K300
JLT ADDLP

INDEX RESW 1
ALPHA RESW 100
BETA RESW 100
GAMMA RESW 100
ZERO WORD 0
K300 WORD 300
```

Q set all elements of an array of 100 words to zeroes.

```
      LDX ZERO
      LDA ZERO
LABEL: STA ARRAY1,X
      COMP K300
      JLT LABEL
ZERO BYTE 0
```

Scanned by CamScanner

* I/O program

if not ready returns 0.

```
INLOOP  TD    INDEV     Test i/p device
        JEQ   INLOOP    loop until device is ready
        RD    INDEV     read one byte into reg A
        STCH  DATA      store byte that was read

OUTLP   TD    OUTDEV    test o/p device
        JEQ   OUTLP
        LDCH  DATA
        WD    OUTDEV
INDEV   BYTE  X'F1'
OUTDEV  BYTE  X'05'
DATA    RESB  1
```

* Using subroutine read 100 bytes into a variable called record.

jump subroutine

```
        JSUB  READ      call read subroutine
READ    LDX   ZERO
RLOOP   TD    INDEV     Test i/p device
        JEQ   RLOOP
        RD    INDEV
        STCH  RECORD,X
```

Scanned by CamScanner

papergrid
Date: / /

```
        TIX     K100
        JLT     RLOOP
        RSUB
          └— return to subroutine

INDEX   byte    X'F1'
RECORD  RESB    100
ZERO    WORD    0
K100    WORD    100
```

Scanned by CamScanner

* SIC/XE Machine Architecture:

→ 1 megabytes (1024 KB) in memory
→ 3 additional registers, 24 bits in length

		reg. no.'s	
along with 5 reg of SIC	B	3	Base register, used for addressing
	S	4	General working register
	T	5	General working register

→ 1 additional register, 48 bits in length

F	6	floating-point accumulator (48 bits)

reg. no.'s in machine level language.

* Data format
→ 24 bit binary number for integers, 2's complement for negative values.
→ 48-bit floating-point data type
→ The exponent is between 0 and 2047

$f * 2^{(e-1024)}$ -5.23

0: set all bits to 0 -523×10^{-2}

1	11	36
s	exponent	fraction

↑ sign ↑ biased representation.

Scanned by CamScanner

* Instruction Format
→ Relative addressing - format 3 (e = 0)
→ Extend the address to 20 bits - format 4 (e = 1)

Format 2 (2 bytes):

8	4	4
op	r1	r2

Format 1 (1 byte):

8
op

Format 3 (3 bytes): — addressing modes

6	1	1	1	1 (base relative)	1	1	12
op	n	i	x	b	p	e	disp
	indirect	immed	index		pc relative	extended mode	

Format 4 (4 bytes):

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

LDA #5
↑
immediate addressing present only in SIC/XE

* Addressing modes
n i x b p e
→ Simple n = 0, i = 0 (SIC) or n = 1, i = 1 ← remaining bits are used as address directly
Immediate n = 0, i = 1 TA = values
Indirect n = 1, i = 0 TA = (Operand)

Scanned by CamScanner

Base relative $b = 1, p = 0$ $TA = (B) + disp$
$0 \le disp \le 4095$

PC relative $b = 0, p = 1$ $TA = (PC) + disp$
$-2048 \le disp \le 2047$

Consider the foll. contents of memory and registers

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

Memory Address
00000
vFFFFF (byte).

3030	003600
3600	103000
6360	00C303
C303	003030

For the following ~~which rep~~ which rep. LDA instructions find the target address and the value loaded into reg. A

Scanned by CamScanner

(i) 032600 — if n and i are 0 then SIC inst. else SIC/XE inst.

3 bytes Format 3 :

0000 0011 0010 0110 0000 0000

opcode | n i | x b p e | disp.

p is set to 1 TA = (PC) + disp.

TA = 3000 + 600 = 3600 PC relative addressing mode

Value loaded into accumulator : 103000

(ii) 03C300

0000 0011 1100 0011 0000 0000

opcode | n i | x b p e | disp.

TA = (B) + disp + (x)

index + base relative mode

TA = 6000 + 300 + 90

= 6390

Value loaded into accumulator : 00C303

(iii) 022030

0000 0010 0010 0000 0011 0000

n i | x b p e

n = 1 i = 0 ← indirect and PC relative

TA = (PC) + disp

= 3000 + 30 = 3030

indirect value = (3600) = 103000

Scanned by CamScanner

010030

0000 0001 0000 0000 0011 0000
n i x b p e
Immediate addressing mode

TA = value
= 030

Value loaded into accumulator : 30

003600 SIC

0000 0000 0011 0110 0000 0000
n i x b p e
opcode address

TA = 3600

Value loaded into accumulator : 103000

0310C303

0000 0011 0001 0000 1100 0011 0000 0011
n i x b p e

TA = 0C303

Value loaded into accumulator : 3030

Scanned by CamScanner

Q. Write an SIC/XE program to do the following operations
alpha = 5
C1 = 'Z'

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

Q. Write an SIC/XE program to compute
BETA = ALPHA + INCR - 1
DELTA = GAMMA + INCR - 1

```
LDS   INCR
LDA   ALPHA
ADDR  S,A
SUB   #1
STA   BETA
LDA   GAMMA
ADDR  S,A
SUB   #1
STA   DELTA
```

Scanned by CamScanner

```
INCR    RESW  1
ALPHA   RESW  1
BETA    RESW  1
GAMMA   RESW  1
DELTA   RESW  1
```

Q. Copy 11 byte character string ~~from~~ str1 to str2

```
        LDT  #11
        LDX  #0
MOVECH  LDCH STR1,X
        STCH STR2,X
        TIXR T
        JLT  MOVECH

STR1    BYTE C'TEST STRING'
STR2    RESB 11
```

takes 2 bytes but TIX #11 can be used which takes 3 bytes. (automatically add 1 to index)

not efficient.

Q. Write an SIC/XE program to clear a 20 byte string to all blanks.

```
        LDT  #20
        LDX  #0
        LDA  #32
MOVECH  STCH STR1,X
        TIXR T
        JLT  MOVECH
```

number is given

Scanned by CamScanner

```
STR   RESB   20
```

Q. Write an SIC/XE program to add the corresponding values of two arrays of 100 words each and store the result into the third array GAMMA.

```
        LDX   #0
        LDT   #3
        LDS   #300
COPY    LDA   ALPHA,X   <- memory location
        ADD   BETA,X
        STA   GAMMA,X
        ADDR  T,X       <- participates in addition
        COMPR X,S
        JLT   COPY
ALPHA   RESW  100
BETA    RESW  100
GAMMA   RESW  100.
```

Q. Set all the elements of an array containing 100 words to zero.

```
        LDX   #0
        LDT   #3
        LDS   #300.
```

Scanned by CamScanner

papergrid
Date: / /

```
        LDA  #0
COPY:   STA  ARRAY1,X
        ADDR T, X
        COMPR X,S
        JLT  COPY
ARRAY1  RESW 100
```

8 Write a seq. of instructions for SIC/XE to Divide BETA by GAMMA setting ALPHA to the integer portion of the quotient and DELTA to the remainder. Use register to register calculations to make it efficient

```
        LDA  BETA
        LDT  GAMMA
        DIVR T, A
        STA  ALPHA
        MULR T, A
        LDS  BETA
        SUBR A, S
        STS  DELTA
ALPHA   RESW 1
BETA    RESW 1
GAMMA   RESW 1
DELTA   RESW 1
```

Scanned by CamScanner

papergrid
Date: / /

Q Write an SIC/XE program to read 100 characters from the input device and store it in a string called record. Write a subroutine to do the same.

```
        JSUB  READ
READ    LDX   #0
        LDT   #100
RLOOP   TD    INDEV
        JEQ   RLOOP
        RD    INDEV
        STCH  RECORD, X
        TIXR  T
        JLT   RLOOP
        RSUB
INDEV   BYTE  X
RECORD  RESB  100
```

Q Suppose that a record contains 100 byte record. Write a subroutine for SIC/XE that will write this record onto device '05'.

Q Write a subroutine for SIC and SIC/XE to read a record into a buffer. The record maybe of any length from 1 to 100 bytes.

Scanned by CamScanner

papergrid

Date: / /

The end of the record is marked with a null character (ASCII code 00). The subroutine should should place the length of the record read ~~re~~ into a variable named 'LENGTH'.

Scanned by CamScanner

p-2

Assemblers

Source Program → [Assembler] → Object Code → [Linker] → Executable Code → [Loader]

Assembler directives

START, END, BYTE, WORD, RESB, RESW

→ These strits are not translated into machine instructions.

→ They provide instructions to the assembler itself

Q) Consider the following SIC program

loc				opcode
2000		Start	2000	
2000	BEG	LDA	FIVE	00 200F
2003		STA	ALPHA	0C 200C
2006		LDCH	CHARZ	50 2012
2009		STCH	C1	54 2013
200C	ALPHA	RESW	1	
200F	FIVE	WORD	5	000005

Scanned by CamScanner

papergrid
Date: / /

```
2012        CHAR2   BYTE  C'Z'        00005A
2013        C1      RESB  1
2014                END   BEG
```

Q2

```
0000        START   0
0000  BEG   LDX     ZERO        040025
0003  MOVECH LDCH   STR1,X      50800F
0006        STCH    STR2,X      54801A
0009        TIX     ELEVEN      2C0028
000C        JLT     MOVECH      380003

000F  STR1  BYTE    C'TEST STRING'  54455354205354524E47
001A  STR2  RESB    11
0025  ZERO  WORD    0           000000
0028  ELEVEN WORD   11          00000B
002B
```

Q3

```
        COPY  START 1000
1000  BEG   LDA   ZERO        0013A8
1003        STA   INDEX       0C1021
1006  ADDLP LDX   INDEX       041021
1009        LDA   ALPHA,X     009024
100C        ADD   BETA,X      189150
100F        STA   GAMMA,X     0C927C
```

Scanned by CamScanner

Date: / /

1012		LDA	INDEX	001021
1015		ADD	THREE	1813AE
1018		STA	INDEX	0C1021
101B		COMP	K300	2813AB
101E		JLT	ADDLP	381006
1021	INDEX	RESW	1	
1024	ALPHA	RESW	100	
1150	BETA	RESW	100	
127C	GAMMA	RESW	100	
13A8	ZERO	WORD	0	000000
13AB	K300	WORD	300	00012C
13AE	THREE	WORD	3	000003
13B1		END	BEG	

Object Program

Q2 H^ (name of prg.) ^000000^00002B

T^000000^1A^040025^50800F^54801A^2C0028^380003^54^45^53^54^20^53^54^52^49^4E^47

(1A: length)

T^000025^06^000000^000 0B

E^000000

Q3 H^copy^001000^0003B1

T^001000^1E^0013A8^0C1021^041021^009024^189150^0C927C^001021^1813AE^0C1021^2813A8

T^0010 1E^03^381006

Scanned by CamScanner

papergrid

Date: / /

1012		LDA	INDEX	001021
1015		ADD	THREE	1813AE
1018		STA	INDEX	0C1021
101B		COMP	K300	2813AB
101E		JLT	ADDLP	381006
1021	INDEX	RESW	1	
1024	ALPHA	RESW	100	
1150	BETA	RESW	100	
127C	GAMMA	RESW	100	
13A8	ZERO	WORD	0	000000
13AB	K300	WORD	300	000120
13AE	THREE	WORD	3	000003
13B1		END	BEG	

Object Program

Q2 H^ (name of prg) ^000000^00002B

T^000000^1A^040025^50800F^54801A^2C0028^
3800003^54^45^53^54^20^53^54^52^49^4E^
47

T^000025^06^000000^000000B

E^000000

Q3 H^COPY^001000^0003B1

T^001000^1E^0013A8^0C1021^041021^009024^
189150^0C927C^001021^1813AE^0C1021^2813A8^

T^0010 1E^03^381006

Scanned by CamScanner

T^001BA8^09^000000^00012C^000003
E^001000

* Three types of records
Header : Program name, starting address, length
Text : Starting address, length, object code
End : address of first executable instruction

Format of header program:
col 1 : H col 2-7 program name
8-13 = starting address (Hexa)
14-19 - length of the program (hexa)

Text record: (Max upto 30 bytes) more than 30 bytes next text rec
(col 10-69)
col 1 : T
col 2-7 : starting address for object code
col 8-9 : length of the object (Hex) (in bytes)
col 10-69 : object code (2 col per byte) in bytes

End record:
col 1 : E
col 2-7 : Address of first executable instruction (hex)

Scanned by CamScanner

Date: / /

* The functions of the two passes assembler
* Pass 1 (define symbol)
→ Assign addresses to all statements (generate LOC)
→ Save the values (address) assigned to all labels for Pass 2
→ Perform some processing of assembler directives.

* Pass 2
→ Assemble instructions
→ Generate data values defined by byte, word
→ Perform processing of assembler directives not done during pass 1
→ Write object program and assembly listing.

Assembler and logic:
2 internal tables - OPTAB and SYMTAB

Pass 1 algorithm:

Pass 1:
```
begin
  read first input line
  if OPCODE = 'START' then
    begin
      save #[OPERAND] as starting address
```

Scanned by CamScanner

papergrid

Date: / /

initialize LOCCTR to starting address
write line to intermediate file
read next input line
end (if start)
else
initialize loccti to 0

(continued in TB)

Pass 2

begin
read first input line (from intermediate file)
(refer textbook)

Scanned by CamScanner

papergrid
Date: / /

Write the object code for SIC/XE program

```
FIRST START 1000
1000    LDA  #5      010005   | opcode | n | i | x | b | p | e |
1003    STA  ALPHA   0F2006
        OC
1006    LDA  #90
1009    STCH C1
100C ALPHA RESW 1
100F  C1   RESB 1
1010       END  BEG
```

STA ALPHA

OC → 1100

opcode	n	i	x	b	p	e	disp
↓	1	1	0	0	1	0	006
000011							

disp = value − (PC)

= 100C − 1006

= 006

Scanned by CamScanner

Page:

Mangal

Date: / /20

1000	COPY	START	1000	
1000	FIRST	STL	RETADR	141033
1003	CLOOP	JSUB	RDREC	482039
1006		LDA	LENGTH	001036
1009		COMP	ZERO	281030
100C		JEQ	ENDFIL	301015
100F		JSUB	WRREC	48
1012		J	CLOOP	3C1003
1015	ENDFIL	LDA	EOF	00102A
1018		STA	BUFFER	0C1039
101B		LDA	THREE	00102D
101E		STA	LENGTH	0C1036
1021		JSUB	WRREC	48
1024		LDL	RETADR	081033
1027		RSUB		4C0000
102A	EOF	BYTE	C'EOF'	454F46
102D	THREE	WORD	3	000003
1030	ZERO	WORD	0	000000
1033	RETADR	RESW	1	
1036	LENGTH	RESW	1	
1039	BUFFER	RESB	4096	

Scanned by CamScanner

2039	RDREC	LDX	ZERO	041030
203C		LDA	ZERO	001030
203F	RLOOP	TD	INPUT	E0205D
2042		JEQ	RLOOP	30203F
2045		RD	INPUT	D8205D
2048		COMP	ZERO	281030
204B		JEQ	EXIT	302057
204E		STCH	BUFFER,X	549039
2051		TIX	MAXLEN	2C205E
2054		JLT	RLOOP	38203F
2057	EXIT	STX	LENGTH	101036
205A		RSUB		4C0000
205D	INPUT	BYTE	X'F1'	000F1
205E	MAXLEN	WORD	4096	001000
2061				

Scanned by CamScanner

Format 2 - end with R
if + was there format - 4
otherwise 3.

Mangal

Page:

Date: / /20

1000		START	1000	
1000	BEG	LDA	#5	010005
1003		STA	ALPHA	0F2006
1006		LDA	#90	01005A
1009		STCH	C1	57 2003
100C	ALPHA	RESW	1	
100F	C1	RESB	1	
1010		END	BEG	

STA → 0C → 1100 (consider only first two bits)

n i x b p e

2048 bytes closer then its PC relative

0000 1101 0010 0006

0F2006

disp = value - (PC)

100C - 1006 → address of next instr

n i x b p e

0101 0111 0010 003

disp = value - (PC)

100F - 100C

Scanned by CamScanner

Page: Date: / /20

0000		LDS	INCR	6F2022
0003		LDA	ALPHA	032013
0006		ADDR	S, A	9040
0008		SUB	#1	1D0001 (reg No)
000B		STA	BETA	0F200E
000E		LDA	GAMMA	032000E
0011		ADDR	S, A	9040
0013		SUB	#1	1D0001
0016		STA	DELTA	0F2009
0019	ALPHA	RESW	1	
001C	BETA	RESW	1	
001F	GAMMA	RESW	1	
0022	DELTA	RESW	1	
0025	INCR	RESW	1	
0028				

1) 0110 11 1 1 0 0 1 0 022 (n i x b p e)

disp = 0025 - 0003

2) 0000 0011 0010 0013

3) 90

4) 0001 1101 0000 001 (n i x b p e)

5) 0000 1111 0010 00E

001C - 000E

6) 0000 0011 0010 00E

Scanned by CamScanner

Mangal

Page:

Date: / /20

```
0000          LDT    #11          75000B
0003          LDX    #0           050000
0006  MOVECH  LDCH   STR1,X
0009          STCH   STR2,X
000C          TIXR   T
000E          JLT    MOVECH
0011  STR1    BYTE   C'TEST STRING'
001C  STR2    RESB   11
0027
```

n i x b pe

1) 0111 0101 0000 00B

2) 0000 0101 0000 0000

3) 0101 0011 1010

Scanned by CamScanner

Date: / /

Q.

Q1

0000		START	0	
0000	FIRST	+LDA	VAL	0310001D
0004		STA	BLOCK	0F2010
0007		LDX	#0	050000
000A	LOOP	ADD	BUF	1B2000
000D		TIX	#5	2D0005
0010		JLT	LOOP	3B2FF7
0013		+STA	VAL	0F10001D
0017	BLOCK	RESW	1	
001A	BUF	WORD	5	000005
001D	VAL	WORD	13	00000D
0020				

LDA 0000 0011 0001 0001D

n i x b p e

STA 0000 1111 0010 010

disp = 0017 − 0007

= 0010

n i x b p e

LDX 0000 0101 0000 000

ADD 0001 1011 0010 00D

1A − D

n i x b p e

TIX 0010 1101 0000 005

* JLT 0011 1011 0010 FF7 000A − 0013

STA 0000 1111 0001 001D

Scanned by CamScanner

@ ← indirect

papergrid

Date: / /

Q.

	SUM	START	0	—
0000	FIRST	CLEAR	S	B440
0002		LDX	#5	050005
0005		+JSUB	RESULT	4B100019
0009		STA	LOC,X	0FA00C
000C		J	@NUM	3E2003
000F		LDA	~~=C'AB'~~ NUM	032000
0012	NUM	RESW	1	—
0015	TEMP	WORD	20	000014
0018	LOC	BYTE	X'12'	12
0019	RESULT	RESB	2	—
001B		END	SUM	—

↳ literal constant

n i x b p e

LDX 0000 0100000 005

+JSUB 0100 101100010010

STA 0000 111110100 00C

0018 - 000C

LDA 0000 001100100 000

n i x b p e

J 0011 111000100 003

H

T^000000^12^B440^050005^4B100019^0FA00C^3E2003^032000

Scanned by CamScanner

papergrid

Date: / /

T^000015^04^000014^12

M^000006^05

E^FIRST

		SUM	START	0
0000		FIRST	LDX	#0	050000
0000			LDA	#0	010000
0003			+ LDB	#TABLE2	69101790
0006		directive	BASE	TABLE2	
000A		LOOP	ADD	TABLE,X	1BA013
000A			ADD	TABLE2,X	1BC000
000D			TIX	COUNT	2F200A
0010			JLT	LOOP	3B2FF4
0013			+ STA	TOTAL	0F102F00
0016			RSUB		4F0000
001A		COUNT	RESW	1	
001D		TABLE	RESW	2000	
0020		TABLE2	RESW	2000	
1790		TOTAL	RESW	1	
2F00			END	FIRST	
2F03					

ADD TABLE2,X 0001 10 1 1 1 (n i x b p e) 0000 1111 0001 2F00

disp = 1790 − 0010

= 1780 (X) = 6016 (D)

Not in range So it is base relative

disp = value − (B)

= 1790 − 1790 = 0

Scanned by CamScanner

papergrid

Date: / /

n. i x b pe.

TIX 0010 11 1 1 0010 00A

JLT 0011 10 1 1 0010 FF4

Scanned by CamScanner