

System Programming

< LAB 2 >

Prepared Date :
March, 25, 2019

Prepared by :
Hagar Usama 4970

Contents:

1.	SRCHAR	3
2.	read_write_rev	5
3.	upper_i_o	8
4.	bubble_char	12

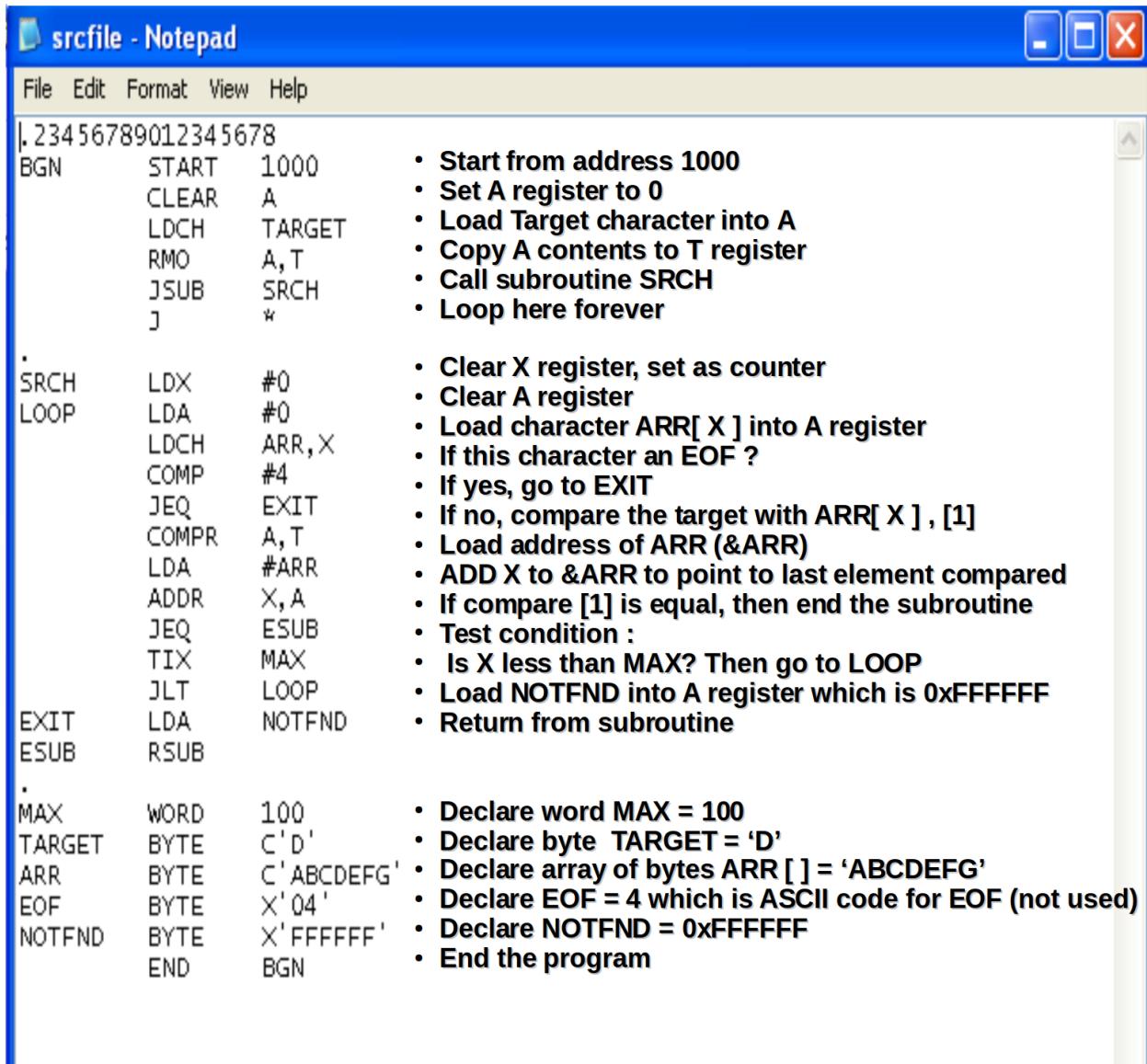
Code Description :

The lab is mainly divided into 4 problems. Each one has its own source file attached to the report :

- SCHAR
- read_write_dev
- upper_i_o
- bubble_char

SRCHAR:

Function : to search for a certain byte in a string, return the address of the byte else return 0xFFFFFFF into A register.



The screenshot shows a Notepad window titled "srcfile - Notepad". The code is written in a assembly-like language. To the right of the code, there are bullet points explaining the steps of the program:

BGN	START	1000	• Start from address 1000
	CLEAR	A	• Set A register to 0
	LDCH	TARGET	• Load Target character into A
	RMO	A, T	• Copy A contents to T register
	JSUB	SRCH	• Call subroutine SRCH
	J	*	• Loop here forever
SRCH	LDX	#0	• Clear X register, set as counter
LOOP	LDA	#0	• Clear A register
	LDCH	ARR, X	• Load character ARR[X] into A register
	COMP	#4	• If this character an EOF ?
	JEQ	EXIT	• If yes, go to EXIT
	COMPR	A, T	• If no, compare the target with ARR[X], [1]
	LDA	#ARR	• Load address of ARR (&ARR)
	ADDR	X, A	• ADD X to &ARR to point to last element compared
	JEQ	ESUB	• If compare [1] is equal, then end the subroutine
	TIX	MAX	• Test condition :
	JLT	LOOP	• Is X less than MAX? Then go to LOOP
	LDA	NOTFND	• Load NOTFND into A register which is 0xFFFFFFF
EXIT	RSUB		• Return from subroutine
MAX	WORD	100	• Declare word MAX = 100
TARGET	BYTE	C'D'	• Declare byte TARGET = 'D'
ARR	BYTE	C'ABCDEFG'	• Declare array of bytes ARR [] = 'ABCDEFG'
EOF	BYTE	X'04'	• Declare EOF = 4 which is ASCII code for EOF (not used)
NOTFND	BYTE	X'FFFFFFFFFF'	• Declare NOTFND = 0xFFFFFFFF
END		BGN	• End the program

Assumptions :

Maximum Size for our array is 100

Samples :

Case 1:

Shortcut to assemf3

```

01013 53A020      ldch arr,X
01016 290004      comp #4
01019 332010      jeq exit
0101C A005      compr A,T
0101E 012015      lda #arr
01021 9010      addr X,A
01023 332009      jeq esub
01026 2F2009      tix max
01029 3B2FE4      jlt loop
0102C 03200F exit    lda notfnd
0102F 4F0000 esub    rsub

01032 000064 max    word 100
01035 44 target    byte C'D'
01036 414243 arr    byte C'ABCDEFG'
444546
47
0103D 04 eof      byte X'04'
0103E FFFFFFFF notfnd byte X'FFFFFFFFFF'
01041           end    bgn

successfull assembly
end of program
symbol table

```

Shortcut to SICSIM

	A=001039	X=000003	L=00100A	B=FFFFFF	S=FFFFFF	T=000044	P=00100D	CC=EQ
--	----------	----------	----------	----------	----------	----------	----------	-------

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1000	B4	00	53	20	30	AC	05	4B	20	03	3F	2F	FD	05	00	00
1010	01	00	00	53	A0	20	29	00	04	33	20	10	A0	05	01	20
1020	15	90	10	33	20	09	2F	28	09	3B	2F	E4	03	20	0F	4F
1030	00	00	00	00	64	44	41	42	43	44	45	46	47	04	FF	FF
1040	FF															
1050	FF															
1060	FF															
1070	FF															
1080	FF															
1090	FF															
10A0	FF															
10B0	FF															
10C0	FF															
10D0	FF															
10E0	FF															
10F0	FF															

Press Esc to Quit, Up or Dn arrows Scrolling.

Case 2:

The screenshot shows a debugger window titled "Shortcut to assemf3". It displays assembly code and memory dump sections. The assembly code includes instructions like ldch, comp, jeq, compr, lda, addr, tix, jlt, and exit. The memory dump section shows memory starting at address 0000064, containing the word 100, followed by target byte C'Z', arr byte C'ABCDEFG', and EOF byte X'04'. The memory dump ends with FFFFFFFF at address 00000E0. The status bar at the bottom indicates a successful assembly end of program.

The screenshot shows a debugger window titled "Shortcut to SICSIM". It displays a memory dump with various memory addresses and their corresponding hex values. The status bar at the top shows registers A=FFFFFFFFFF, X=0000007, L=00100A, B=FFFFFFFFFF, S=FFFFFFFFFF, T=000005A, P=00100D, and CC=EQ. The memory dump starts at address 1000 and continues up to 10F0. The value at address 1030 is highlighted with a red box, showing 64 5A 41 42 43 44 45 46 47 04 FF FF. The status bar at the bottom indicates pressing Esc to quit or Up or Down arrows for scrolling.

Read_write_deu :

Function : to read a string from an input device and print it reversed to an output device.

srcfile - Notepad

File Edit Format View Help

```

.23456789012345678
.LABEL...OPCODE..OPERAND
    START 1000
    JSUB READ
    JSUB WRITE
    JSUB ENDFIL
    J      *
        • Start from address 1000
        • Call subroutine READ
        • Call subroutine WRITE
        • Not used (Call subroutine ENDFIL)
        • Loop here forever
        •

.SUBROUTINE FOR READING FROM DEV F3
        •

READ   CLEAR X
       CLEAR A
LPR     TD DEVF3
       JEQ LPR
       RD DEVF3
       COMP #04
       JEQ EXIT
       STCH DATA,X
       TIX MAX
       JLT LPR
EXIT   STX LENGTH
       RSUB
        • Clear register X to set as a counter
        • Clear register A
        • Test device DEVF3
        • Is Ready?
            • If no, go to LPR to test again
            • If yes, read a byte from the device into A
        • Compare the byte with EOF 0x04
        • If equal, go to EXIT
            • If no, store the character into array DATA[X]
        • Increment X and Compare to MAX
            • If less than, loop (go to LPR)
            • If not, store the X into LENGTH
        • Return from subroutine

DEVF3 BYTE X'F3'
LENGTH RESW 1
MAX WORD 100
DATA  RESB 100
        • Input device number
        • One-word variable
        • One-word constant = 100
        • Array variable 100-byte

.SUBROUTINE FOR WRITING INTO DEV
        •

WRITE  CLEAR A
       LDT #1
       LDS #0
       SUBR T,X
LOOP   TD DEV05
       JEQ LOOP
       LDCH DATA,X
       WD  DEV05
        • Clear register A
        • Load 1 into register T
        • Load 0 into register S (loop limit)
        •  $X = X - T$ 
        • Test output device
        • Is Ready?
            • If no, go to loop to test again
            • If yes, load a character from DATA[X] into A
        • Write one-byte into output device
        •

        •  $X = X - T$  (to read from last to first)
        • Compare register X to 0
            • If NOT less than 0
                • Go to loop (continue looping)
        • Return from subroutine
        •

.ENDFIL CLEAR A
.LOOP2  TD DEV05
        • ENDFIL : (not used) : to add EOF character
        • into the output device at the end
        •

        • Output device number
        • End program

```

```

0109E 9451      subr T,X
010A0 E32016    loop   td dev05
010A3 332FFA    jeq    loop
010A6 53AF89    ldch   data,X
010A9 DF200D    wd dev05
010AC 9451      subr T,X
010AE A014      compr X,S
010B0 372FED    jgt    loop
010B3 332FEA    jeq    loop
010B6 4F0000    rsub

.ENDFIL  CLEAR  A
.LOOP2  TD     DEV05
.      JEQ   LOOP2
.      LDCH  #4
.      WD    DEV05
.      RSUB

010B9 05       dev05 byte  X'05'
010BA          end

successful assembly
end of program
symbol table

```

Assumptions :

DATA: array to read from input device maximum 100-byte

Case 1:

```

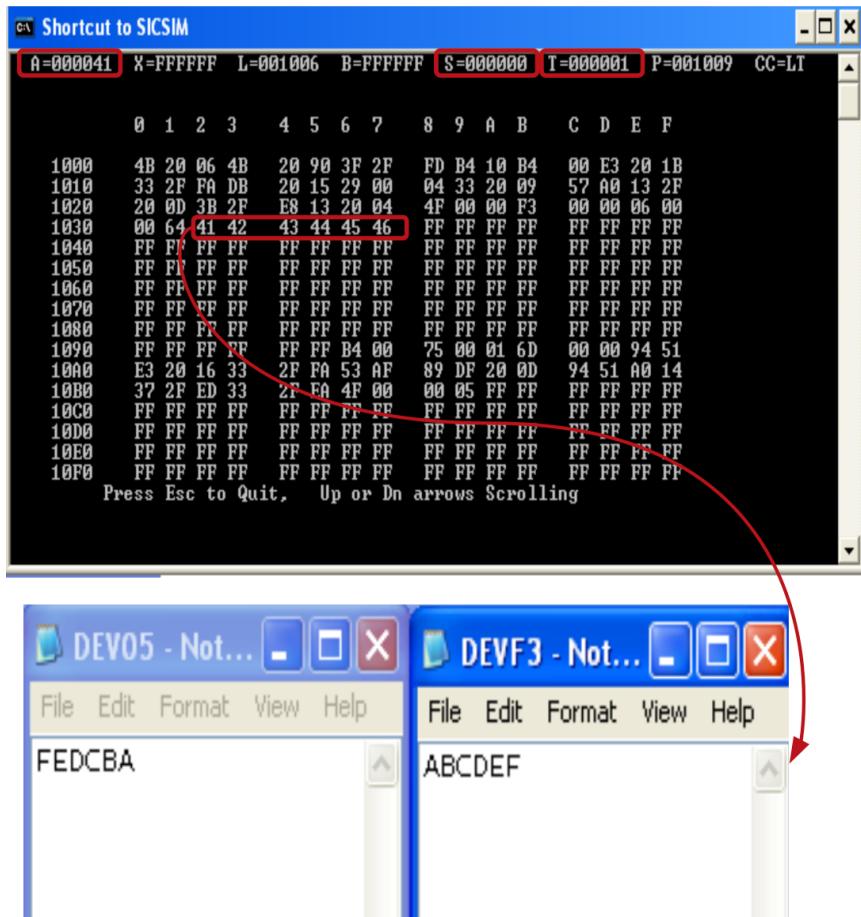
01000 4B 20 06 4B 20 90 3F 2F FD B4 10 B4 00 E3 20 1B
01010 33 2F FA DB 20 15 29 00 04 33 20 09 57 A0 13 2F
01020 20 0D 3B 2F E8 13 20 04 4F 00 00 F3 00 00 05 00
01030 00 64 21 75 65 65 6E FF FF FF FF FF FF FF FF FF
01040 FF FF
01050 FF FF
01060 FF FF
01070 FF FF
01080 FF FF
01090 FF FF FF FF FF FF B4 00 75 00 01 6D 00 00 94 51
010A0 E3 20 16 33 2F FA 53 AF 89 DF 20 0D 94 51 A0 14
010B0 37 2F ED 33 2F EA 4F 00 00 05 FF FF FF FF FF FF
010C0 FF FF
010D0 FF FF
010E0 FF FF
010F0 FF FF

Press Esc to Quit, Up or Dn arrows Scrolling

```

The screenshot shows the SICSIM debugger interface with a memory dump window at the top and two Notepad windows at the bottom. The memory dump window displays memory starting at address 01000. The first Notepad window, titled "DEV05 - Notepad", contains the text "neeuq". The second Notepad window, titled "DEVF3 - Notepad", contains the text "queen". A red arrow points from the memory dump window to the "DEV05 - Notepad" window.

Case 2:



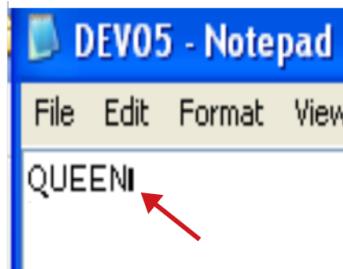
upper_i_o:

Function : to read a string from an input device and print the string to an output device capitalized.

Assumptions :

Assume input data are lowercase characters (special characters excluded)

NOTE : ENDFIL subroutine should be used to end output device file by EOF
(both in read_write_dev & upper_i_o) , however the EOF character seems to be visible in windows XP, so I discarded it.

A screenshot of a terminal window titled "Shortcut to assemf3". The window displays assembly code and its corresponding assembly output. The assembly code includes instructions like .ENDFIL, .LOOP3, and various data definitions. The assembly output shows the assembly language being converted into machine code. At the bottom of the assembly output, there is a message: "successful assembly" followed by "end of program" and "symbol table".

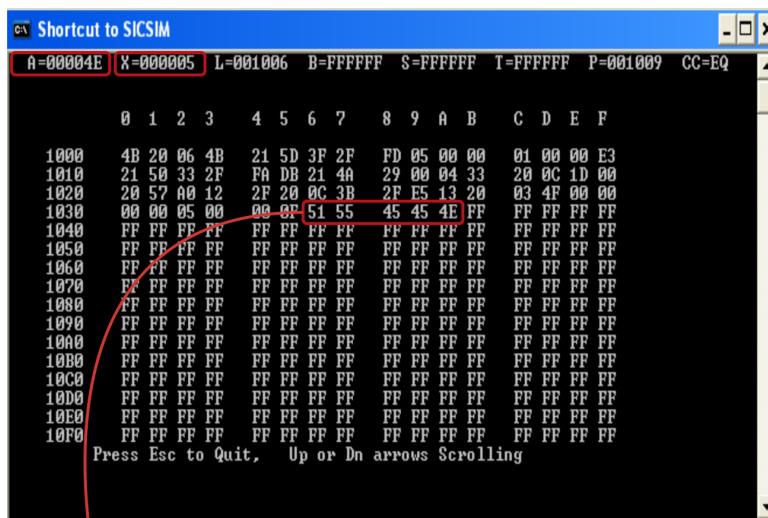
```
01163 B400    write    clear A
01165 B410    clear X
01167 E32012  loop2   td      dev05
0116A 332FFA   jeq     loop2
0116D 53AE06   ldch    data,X
01170 DF2009   wd      dev05
01173 2F2EBA   tix     length
01176 3B2FEE   jlt     loop2
01179 4F0000   rsub

.ENDFIL  CLEAR  A
.LOOP3   TD      DEU05
.        JEQ    LOOP3
.        LDCH   #4
.        WD      DEU05
.        RSUB

0117C 05    dev05  byte  X'05'
0117D       end    bgn

successful assembly
end of program
symbol table
```

Case 1 :



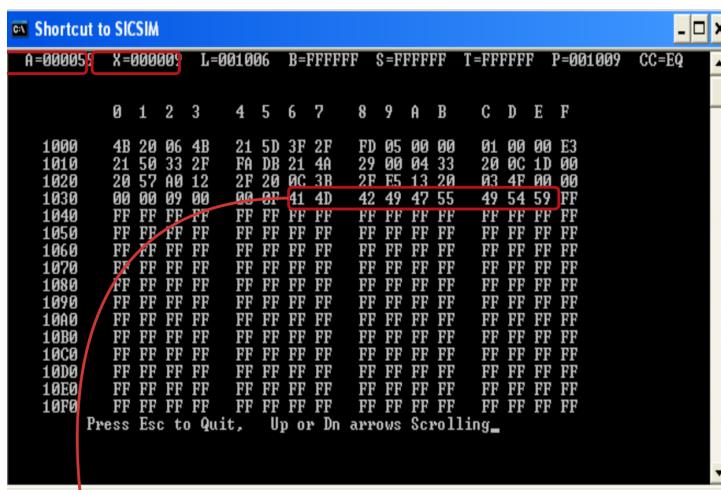
A=00004E X=000005 L=001006 B=FFFFFF S=FFFFFF T=FFFFFF P=001009 CC=EQ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1000	4B	20	06	4B	21	5D	3F	2F	FD	05	00	00	01	00	00	E3
1010	21	50	33	2F	FA	DB	21	4A	29	00	04	33	20	0C	1D	00
1020	20	57	A0	12	2F	20	0C	3B	2F	E5	13	20	03	4F	00	00
1030	00	00	05	00	00	00	00	51	55	45	45	4E	FF	FF	FF	FF
1040	FF															
1050	FF															
1060	FF															
1070	FF															
1080	FF															
1090	FF															
10A0	FF															
10B0	FF															
10C0	FF															
10D0	FF															
10E0	FF															
10F0	FF															

Press Esc to Quit, Up or Dn arrows Scrolling



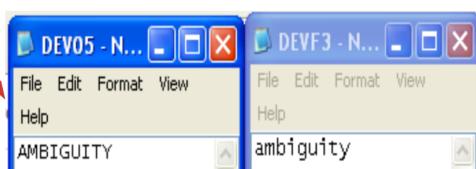
Case 2 :



A=000051 X=000005 L=001006 B=FFFFFF S=FFFFFF T=FFFFFF P=001009 CC=EQ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
1000	4B	20	06	4B	21	5D	3F	2F	FD	05	00	00	01	00	00	E3	
1010	21	50	33	2F	FA	DB	21	4A	29	00	04	33	20	0C	1D	00	
1020	20	57	A0	12	2F	20	0C	3B	2F	E5	13	20	03	4F	00	00	
1030	00	00	09	00	00	00	00	41	4D	42	49	47	55	49	54	59	FF
1040	FF																
1050	FF																
1060	FF																
1070	FF																
1080	FF																
1090	FF																
10A0	FF																
10B0	FF																
10C0	FF																
10D0	FF																
10E0	FF																
10F0	FF																

Press Esc to Quit, Up or Dn arrows Scrolling



bubble_char :

The algorithm is :

```
for( i=0; i<n-1; i++)
    for(j=0; j<n-i-1 ; j++)
        if (arr[j] > arr[j+1])
            swap(arr[j] , arr[j+1])
```

Where :

i : X register

&arr[j] : P1

&arr[j+1] : P2

n -1 : N

n -i-1 : M

srcfile - Notepad

File Edit Format View Help

```
.23456789012345678
.LABEL...opcode..OPERAND
    START 1000
    LDT #0
    LDA #0
    LDS #0 } Initial values to iterators
OUTLP   LDX #0
        LDA #ARR
        STA P1
        ADD INC
        STA P2
LOOP    CLEAR A
        LDCH @P1
        RMO A,T
        LDCH @P2
        COMPR T,A
        JEQ NOSWP
        JLT NOSWP
        JSUB SWAP
NOSWP   LDA N
        SUBR S,A
        STA M
        LDA P1
        ADD INC
        STA P1
        ADD INC
        STA P2
ENDNLP  TIX M
        JLT LOOP } Inner Loop Condition
        LDA #1
        ADDR A,S
        LDA N
        COMPR S,A
        JLT OUTLP } Outer Loop Condition
EXIT    J *
.
SWAP    LDCH @P1
        STCH TEMP
```

Diagram annotations:

- A red bracket labeled "Initial values to iterators" groups the first four lines of code: START 1000, LDT #0, LDA #0, and LDS #0.
- A red bracket labeled "Inner Loop" groups the lines from LDCH @P1 to JSUB SWAP.
- A red bracket labeled "Outer Loop" groups the lines from TIX M to OUTLP, including the entire inner loop structure.
- A red bracket labeled "Inner Loop Condition" groups the conditional part of the inner loop: JLT LOOP, LDA #1, ADDR A,S, LDA N, COMPR S,A, and JLT OUTLP.
- A red bracket labeled "Outer Loop Condition" groups the conditional part of the outer loop: TIX M, JLT LOOP, LDA #1, ADDR A,S, LDA N, COMPR S,A, and JLT OUTLP.

```

    .. COMPR S,A
    JLT OUTLP
EXIT   J *
.

SWAP   LDCH @P1
       STCH TEMP
       LDCH @P2
       STCH @P1
       LDCH TEMP
       STCH @P2
       RSUB
P1     RESW 1
P2     RESW 1
TEMP   RESW 1
INC    WORD 1
M      RESW 1
N      WORD 4
ARR   byte c'QUIZZ'
END

```

Swap Subroutine

String to be sorted

Assumptions :

Assume the length of the string to be sorted is already given, so $n-1$ (N) is given.

A string is declared in the program, and not read from input device.

Case 1:

Shortcut to assemf3

```

01052 A040      compr S,A
01054 3B2FB2    jlt   outlp
01057 3F2FFD exit   j    *
0105A 522012 swap   ldch  @p1
0105D 572015    stch  temp
01060 52200F    ldch  @p2
01063 562009    stch  @p1
01066 53200C    ldch  temp
01069 562006    stch  @p2
0106C 4F0000    rsub
0106F          p1    resw  1
01072          p2    resw  1
01075          temp  resw  1
01078 000001 inc   word  1
0107B          m    resw  1
0107E 000004 n    word  4
01081 515549 arr   byte  c'QUIZZ'
01086          5A7A end
successfull assembly
end      of program
symbol table

```

Shortcut to SICSIM

```

A=000004 X=000001 L=00102D B=FFFFFF S=000004 T=000049 P=00105A CC=EQ

```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1000	75	00	00	01	00	00	6D	00	00	05	00	00	01	20	72	0F
1010	20	5D	1B	20	63	0F	20	5A	B4	00	52	20	52	AC	05	52
1020	20	50	A0	50	33	20	06	3B	20	03	4B	20	2D	03	20	4E
1030	94	40	0F	20	46	03	20	37	1B	20	3D	0F	20	31	1B	20
1040	37	0F	20	2E	2F	20	34	3B	2F	CE	01	00	01	90	04	03
1050	20	2C	A0	40	3B	2F	B2	3F	2F	FD	52	20	12	57	20	15
1060	52	20	0F	56	20	09	53	20	0C	56	20	06	4F	00	00	00
1070	10	82	00	10	83	51	FF	FF	00	00	01	00	00	01	00	00
1080	04	49	51	55	5A	7A	FF									
1090	FF															
10A0	FF															
10B0	FF															
10C0	FF															
10D0	FF															
10E0	FF															
10F0	FF															

Press Esc to Quit, Up or Dn arrows Scrolling.

Case 2 :

Shortcut to assemf3

```

01054 3B2FB2      jlt    outlp
01057 3F2FFD exit   j      *
0105A 522012 swap   ldch   @p1
0105D 572015        stch   temp
01060 52200F        ldch   @p2
01063 562009        stch   @p1
01066 53200C        ldch   temp
01069 562006        stch   @p2
0106C 4F0000        rsub
0106F      p1      resw  1
01072      p2      resw  1
01075      temp    resw  1
01078 000001 inc    word   1
0107B      m      resw  1
0107E 000007 n      word   7
01081 5A5958 arr   byte   c'ZYXXWUUTS'
      575655
      5453
01089          end

successful assembly
end of program
symbol table

```

Shortcut to SICSIM

```

A=0000007 X=0000001 L=00102D B=FFFFFF S=0000007 T=0000054 P=00105A CC=EQ

      0 1 2 3 4 5 6 7 8 9 A B C D E F
1000 75 00 00 01 00 00 6D 00 00 05 00 00 01 20 72 0F
1010 20 5D 1B 20 63 0F 20 5A B4 00 52 20 52 AC 05 52
1020 20 50 A0 50 33 20 06 3B 20 03 4B 20 2D 03 20 4E
1030 94 40 0F 20 46 03 20 37 1B 20 3D 0F 20 31 1B 20
1040 37 0F 20 2E 2F 20 34 3B 2F CE 01 00 01 90 04 03
1050 20 2C A0 40 3B 2F B2 3F 2F FD 52 20 12 57 20 15
1060 52 20 0F 56 20 09 53 20 0C 56 20 06 4F 00 00 00
1070 10 82 00 10 83 54 FF FF 00 00 01 00 00 01 00 00
1080 07 53 54 55 56 57 58 59 5A FF FF FF FF FF FF FF FF
1090 FF FF
10A0 FF FF
10B0 FF FF
10C0 FF FF
10D0 FF FF
10E0 FF FF
10F0 FF FF

Press Esc to Quit, Up or Dn arrows Scrolling

```

Case 3 :

Shortcut to assemf3

```

01054 3B2FB2      jlt    outlp
01057 3F2FFD exit   j      *
0105A 522012 swap   ldch  @p1
0105D 572015       stch  temp
01060 52200F       ldch  @p2
01063 562009       stch  @p1
01066 53200C       ldch  temp
01069 562006       stch  @p2
0106C 4F0000 rsub
0106F             p1    resw 1
01072             p2    resw 1
01075             temp  resw 1
01078 000001 inc   word  1
0107B             m    resw 1
0107E 000007 n     word  7
01081 616263 arr   byte  c'abcdefg'h
64656b
6768
01089           end

successful assembly
end
of program
symbol table

```

Shortcut to SICSIM

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1000	75	00	00	01	00	00	6D	00	00	05	00	00	01	20	72	0F
1010	20	5D	1B	20	63	0F	20	5A	B4	00	52	20	52	AC	05	52
1020	20	50	A0	50	33	20	06	3B	20	03	4B	20	2D	03	20	4E
1030	94	40	0F	20	46	03	20	37	1B	20	3D	0F	20	31	1B	20
1040	37	0F	20	2E	2F	20	34	3B	2F	CE	01	00	01	90	04	03
1050	20	2C	A0	40	3B	2F	B2	3F	2F	FD	52	20	12	57	20	15
1060	52	20	0F	56	20	09	53	20	0C	56	20	06	4F	00	00	00
1070	10	82	00	10	83	FF	FF	FF	00	00	01	00	00	01	00	00
1080	07	61	62	63	64	65	66	67	68	FF						
1090	FF															
10A0	FF															
10B0	FF															
10C0	FF															
10D0	FF															
10E0	FF															
10F0	FF															

Press Esc to Quit, Up or Dn arrows Scrolling.