

CERTIFICATE

This is to Certify that **Mr. Parmar Parth.** from R.K. University Rajkot having Enrollment Number **20SOECE13022** has completed term work of Sem-5 for subject Microprocessor and Microcontroller (EC509) During the Academic year 2021-22.

Date : 10/10/2021

Signature of Teacher.....

HOD.....

EXPERIMENT-1

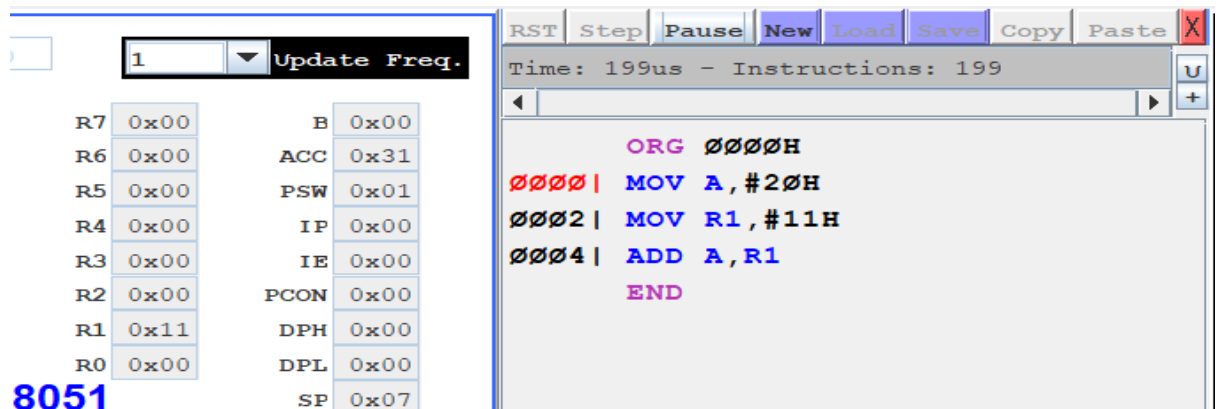
Aim: Write an Assembly Language program to

(A) Add/subtract two 8-bit numbers.

➤ **Code:**

```
ORG 0000H
MOV A,#20H
MOV R1,#11H
ADD A,R1
END
```

➤ **Output:**



(B) Add/subtract two 16-bit numbers

➤ **Code:**

```
ORG 0000H
MOV A,#30H
MOV R1,#11H
SUBB A,R1
END
```

➤ **Output:**

The screenshot shows a microcontroller simulator interface. On the left, a table displays the values of various registers:

R7	0x00	B	0x00
R6	0x00	ACC	0x1F
R5	0x00	PSW	0x41
R4	0x00	IP	0x00
R3	0x00	IE	0x00
R2	0x00	PCON	0x00
R1	0x11	DPH	0x00
R0	0x00	DPL	0x00
		SP	0x07

Below the register table, the value **8051** is displayed in blue. At the top, there is a dropdown menu set to '1' and a button labeled 'Update Freq.'. On the right, a window shows assembly code with the following instructions:

```

ORG 0000H
0000 | MOV A, #30H
0002 | MOV R1, #11H
0004 | SUBB A, R1
      END
  
```

At the top of the right window, there are buttons: RST, Step, Pause, New, Load, Save, Copy, Paste, and a close button (X). Below these buttons, it says 'Time: 92us - Instructions: 92'.

(C) Add/subtract two 32-bit numbers

➤ **Code:**

```

ORG 0H
MOV 40H, #23
MOV 41H, #15
MOV 42H, #60
MOV 43H, #70
MOV 50H, #30
MOV 51H, #40
MOV 52H, #63
MOV 53H, #77
MOV R0, #40H
MOV R1, #60H
MOV R2, #4
SETB RS0
MOV R1, #50H
CLR RS0
CLR C
REPEAT: MOV A, @R0
SETB RS0
ADDC A, @R1
INC R1
CLR RS0
MOV @R1, A
  
```

```

INC R0
INC R1
DJNZ R2, REPEAT
JNC EXIT
INC @R1
EXIT: NOP
SJMP $
END

```

➤ **Output:**

The screenshot displays the Proteus 8051 simulator interface. On the left, the internal state of the 8051 microcontroller is shown, including registers (R0-R7, ACC, PSW, IP, IE, PCON, DPH, DPL, SP), special function registers (TH0, TL0, TMOD, TCON, TH1, TL1, P0-P3), and the Program Counter (PC) set to 0x0035. The system clock is 12.0 MHz. The assembly code is loaded and displayed on the right, starting with 'ORG 0H' and followed by instructions: MOV 40H, #23; MOV 41H, #15; MOV 42H, #60; MOV 43H, #70; MOV 50H, #30; MOV 51H, #40; MOV 52H, #63; MOV 53H, #77; MOV R0, #40H; MOV R1, #60H; MOV R2, #4; SETB RS0; MOV R1, #50H; CLR RS0; CLR C; REPEAT: MOV A, @R0; SETB RS0; ADDC A, @R1; INC R1; CLR RS0. The code is executed, and the instruction counter shows 93 instructions. The data memory is visible at the bottom, showing the contents of RAM addresses 00 to 70.

EXPERIMENT-2

Aim: Write an Assembly Language program to

(A) Divide an 8-bit number by 8-bit number.

➤ **Code:**

MOV R0, #20H;set source address 20H to R0

MOV R1, #30H;set destination address 30H to R1

MOV A, @R0;take the first operand from source to register A

INC R0; Point to the next location

MOV B, @R0; take the second operand from source to register B

DIV AB ; Divide A by B

MOV @R1, A; Store Quotient to 30H

INC R1; Increase R1 to point to the next location

MOV @R1, B; Store Remainder to 31H

HALT: SJMP HALT ;Stop the program

➤ **Output:**

The screenshot displays the 8051 microcontroller simulator interface. On the left, the register window shows the following values:

R7	0x00	B	0x00
R6	0x00	ACC	0x00
R5	0x00	PSW	0x04
R4	0x00	IP	0x00
R3	0x00	IE	0x00
R2	0x00	PCON	0x00
R1	0x31	DPH	0x00
R0	0x21	DPL	0x00
		SP	0x07

The instruction window on the right shows the following assembly code being executed:

```

0000| MOV R0, #20H;set source address
0002| MOV R1, #30H;set destination a
0004| MOV A, @R0;take the first oper
0005| INC R0; Point to the next loca
0006| MOV B, @R0; take the second op
0008| DIV AB ; Divide A by B
  
```

The status bar at the top indicates "Time: 344us - Instructions: 174". The address 8051 is displayed in the bottom left corner.

(B) Multiply two 8-bit numbers.

➤ **Code:**

```
ORG 000H
MOV A,30H
MOV B,31H
MUL AB
MOV 34H,A
MOV 35H,B
```

➤ **Output:**

The screenshot displays the Proteus 8051 simulator interface. The main window shows the register values and the instruction list. The instruction list on the right is as follows:

```

ORG 000H
0000 | MOV A, 30H
0002 | MOV B, 31H
0005 | MUL AB
0006 | MOV 34H, A
0008 | MOV 35H, B

```

The Data Memory window at the bottom shows the values stored at addresses 34H and 35H:

Address	Value
34H	30H
35H	31H

(C) Multiply two 16-bit numbers.

➤ **Code:**

```
ORG 0000H
MOV R0,#0FFH
MOV R1,#0EFH
MOV R2,#0FFH
MOV R3,#9DH
MOV A,R2
MOV B,R3
MUL AB
MOV R4,A
MOV R5,B
MOV A,R2
MOV B,R1
```

```

MUL AB
MOV R6,B
ADDC A,R5
MOV R5,A
MOV A,R1
MOV B,R4
MUL AB
ADDC A,R5
MOV R5,A
MOV A,B
ADDC A,R6
MOV R6,A
MOV A,R1
MOV B,R2
MUL AB
ADDC A,R6
MOV R6,A
MOV A,B
ADDC A,#00H
MOV R7,A
END

```

➤ **Output:**

The screenshot shows the EDISIM5.01 microprocessor simulator. The main window displays the state of the microprocessor, including registers (R0-R7, ACC, PSW, IP, IE, PCON, DPH, DPL, SP) and memory (Data Memory, PC). The assembly code is displayed on the right, showing instructions like MOV, MUL, and ADDC. The PC register is highlighted in blue, indicating the current instruction address.

Registers:

R/o	W/o	TH0	TL0	R7	B
0x00	0x00	0x00	0x00	0xEE	0xEE
RXD	TXD			R6	ACC
1	1			0x5C	0xEE
SCON	0x00	TMOD	0x00	R5	PSW
		TCON	0x00	0x1A	0x00
				R4	IP
				0x63	0x00
				R3	IE
				0x9D	0x00
				R2	PCON
				0xFF	0x00
				R1	DPH
				0xEF	0x00
				R0	DPL
				0xFF	0x00
					SP
					0x07

Memory:

pins	bits	TH1	TL1	PC
0xFF	0xFF	P3	0x00	0x00
0xFF	0xFF	P2		
0xFF	0xFF	P1		
0xFF	0xFF	P0		

Assembly Code:

```

ORG 0000H
0000| MOV R0,#0FFH
0002| MOV R1,#0EFH
0004| MOV R2,#0FFH
0006| MOV R3,#9DH
0008| MOV A,R2
0009| MOV B,R3
000B| MUL AB
000C| MOV R4,A
000D| MOV R5,B
000F| MOV A,R2
0010| MOV B,R1
0012| MUL AB
0013| MOV R6,B
0015| ADDC A,R5
0016| MOV R5,A
0017| MOV A,R1
0018| MOV B,R4
001A| MUL AB
001B| ADDC A,R5
001C| MOV R5,A

```

EXPERIMENT-3

Aim: Write an Assembly Language program to

(A) Add block of data stored in internal/external memory locations.

➤ **Code:**

```
ORG 0000H
MOV R1,#05
MOV R0,#30H
LOOP:ADD A,@R0
INC R0
DJNZ R1,LOOP
END
```

➤ **Output:**

The screenshot displays the 8051 simulator interface. On the left, the register and pin status is shown. The Program Counter (PC) is at 0x0084. The main window shows the assembly code being executed:

```

ORG 0000H
0000| MOV R1,#05
0002| MOV R0,#30H
0004| LOOP:ADD A,@R0
0005| INC R0
0006| DJNZ R1,LOOP
END
  
```

Below the code, the memory dump shows the data being added to the accumulator. The memory address 0x34 contains the value 0x05, which is being added to the accumulator (A) at address 0x00.

addr	0x34	0x05	value
00	35	00	FF
01	9D	63	1A
02	5C	EE	00
03	54	00	00
04	00	00	00
05	00	00	00
06	00	00	00
07	00	00	00
08	00	00	00
09	00	00	00
0A	00	00	00
0B	00	00	00
0C	00	00	00
0D	00	00	00
0E	00	00	00
0F	00	00	00
10	00	00	00
11	00	00	00
12	00	00	00
13	00	00	00
14	00	00	00
15	00	00	00
16	00	00	00
17	00	00	00
18	00	00	00
19	00	00	00
1A	00	00	00
1B	00	00	00
1C	00	00	00
1D	00	00	00
1E	00	00	00
1F	00	00	00
20	00	00	00
21	00	00	00
22	00	00	00
23	00	00	00
24	00	00	00
25	00	00	00
26	00	00	00
27	00	00	00
28	00	00	00
29	00	00	00
2A	00	00	00
2B	00	00	00
2C	00	00	00
2D	00	00	00
2E	00	00	00
2F	00	00	00
30	01	02	03
31	04	05	01
32	00	30	00
33	00	00	00
34	00	00	00
35	00	00	00
36	00	00	00
37	00	00	00
38	00	00	00
39	00	00	00
3A	00	00	00
3B	00	00	00
3C	00	00	00
3D	00	00	00
3E	00	00	00
3F	00	00	00
40	17	0F	3C
41	46	00	00
42	00	00	00
43	00	00	00
44	00	00	00
45	00	00	00
46	00	00	00
47	00	00	00
48	00	00	00
49	00	00	00
4A	00	00	00
4B	00	00	00
4C	00	00	00
4D	00	00	00
4E	00	00	00
4F	00	00	00
50	1E	28	3F
51	4D	00	00
52	00	00	00
53	00	00	00
54	00	00	00
55	00	00	00
56	00	00	00
57	00	00	00
58	00	00	00
59	00	00	00
5A	00	00	00
5B	00	00	00
5C	00	00	00
5D	00	00	00
5E	00	00	00
5F	00	00	00
60	35	37	7B
61	93	00	00
62	00	00	00
63	00	00	00
64	00	00	00
65	00	00	00
66	00	00	00
67	00	00	00
68	00	00	00
69	00	00	00
6A	00	00	00
6B	00	00	00
6C	00	00	00
6D	00	00	00
6E	00	00	00
6F	00	00	00
70	00	00	00
71	00	00	00
72	00	00	00
73	00	00	00
74	00	00	00
75	00	00	00
76	00	00	00
77	00	00	00
78	00	00	00
79	00	00	00
7A	00	00	00
7B	00	00	00
7C	00	00	00
7D	00	00	00
7E	00	00	00
7F	00	00	00

- (B) Transfer block of data from internal memory locations to external memory locations.

➤ **Code:**

```

MOV R0, #30H
MOV A, #00H
MOV R7, #07H
MOV DPTR, #0200H
BACK : MOV A, @R0
MOVX @DPTR, A
INC R0
INC DPTR
DJNZ R7, BACK
END

```

➤ **Output:**

The screenshot displays the ED SIM501 microcontroller simulator. The main window shows the state of the microcontroller during execution. The registers section on the right lists R0 through R7, ACC, PSW, IP, IE, PCON, DPH, DPL, and SP. The code window on the right shows the assembly code being executed, with the current instruction highlighted. The memory window at the bottom shows the data memory layout.

Registers:

R/O	W/O	TH0	TL0	R7	B
0x00	0x00	0x00	0x00	0x07	0x00
RXD	TXD			R6	ACC
1	1			0x5C	0x01
SCON	0x00	TMOD	0x00	R5	PSW
		TCON	0x00	0x1A	0x01
				R4	IP
				0x63	0x00
				R3	IE
				0x9D	0x00
				R2	PCON
				0xFF	0x00
				R1	DPH
				0x00	0x02
				R0	DPL
				0x30	0x00
					SP
					0x07

PC: 8051

PSW: 0000 0001

Memory:

addr	0x34	0x05	value
0	1	2	3
00	30	00	FF 9D 63 1A 5C 07 00 54 00 00 00 00 00 00 00
10	00	00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20	00	00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
30	01	02	03 04 05 01 00 30 00 00 00 00 00 00 00 00 00
40	17	0F	3C 46 00 00 00 00 00 00 00 00 00 00 00 00 00
50	1E	28	3F 4D 00 00 00 00 00 00 00 00 00 00 00 00 00
60	35	37	7B 93 00 00 00 00 00 00 00 00 00 00 00 00 00
70	00	00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Code Window:

```

0000| MOV R0, #30H
0002| MOV A, #00H
0004| MOV R7, #07H
0006| MOV DPTR, #0200H
0009| BACK : MOV A, @R0
000A| MOVX @DPTR, A
000B| INC R0
000C| INC DPTR
000D| DJNZ R7, BACK
END

```

EXPERIMENT-4

Aim: Write an Assembly Language program to

- (a) Find the maximum/minimum number from block of 5 data bytes stored from memory location 20h onwards.

➤ **Code:**

```
MOV R0, #05H
MOV R1, #20H
MOV B, @R1
LOOP:
    INC R1
    MOV A, @R1
    CMP B
    JNC SKIP
    MOV B, A
SKIP: DJNZ R0, LOOP
```

- (b) Sort block of data in ascending or descending order stored from memory location 20h onwards.

➤ **Code:**

```
LXI H,20H    ;Set pointer for array
MOV C,M      ;Load the Count
DCR C        ;Decrement Count
REPEAT: MOV D,C
    LXI H,20H    ;Load starting address of data array
LOOP:  MOV A,M    ;copy content of memory location to Accumulator
    INX H
    CMP M
    JNC SKIP      ;Jump to skip if carry not generated
    MOV B,M      ;copy content of memory location to B - Register
    MOV M,A      ;copy content of A - Register to memory location
    DCX H        ;Decrement content of HL pair of registers
    MOV M,B
    INX H        ;Increment content of HL pair of registers
SKIP:  DCR D
    JNZ LOOP      ;Jump to LOOP if not Zero
    DCR C        ;Decrement Count
    JNZ REPEAT    ;Jump to REPEAT if not Zero
    HLT          ;Terminate Program
```

EXPERIMENT-5

Aim: Write an Assembly Language program to

(A) Blinking LED at One second.

➤ **Code:**

```
START: CPL P1.0
      ACALL WAIT
      SJMP START
WAIT:  MOV R4,#05H
WAIT1: MOV R3,#00H
WAIT2: MOV R2,#00H
WAIT3: DJNZ R2,WAIT3
      DJNZ R3,WAIT2
      DJNZ R4,WAIT1
      RET
```

➤ **Output:**



(B) Ring counter (Shifting) pattern at One second.

➤ **Code:**

```
ORG 0000H
MOV P1,#00H // FOR OUTPUT DEVICE SET THE PORT TO 0
AND FOR INPUT SET THE PORT TO 1
ABC:
      SETB P1.7 // INITIALIZING THE PIN TO 1
      ACALL Delay
      CLR P1.7

      SETB P1.6
      ACALL Delay
      CLR P1.6

      SETB P1.5
      ACALL Delay
      CLR P1.5

      SETB P1.4
      ACALL Delay
      CLR P1.4
```

SJMP ABC // SHORT JUMP

Delay:

MOV R0, #255H

Z:

MOV R1, #255

Y:

MOV R2, #255

X:

DJNZ R2, X

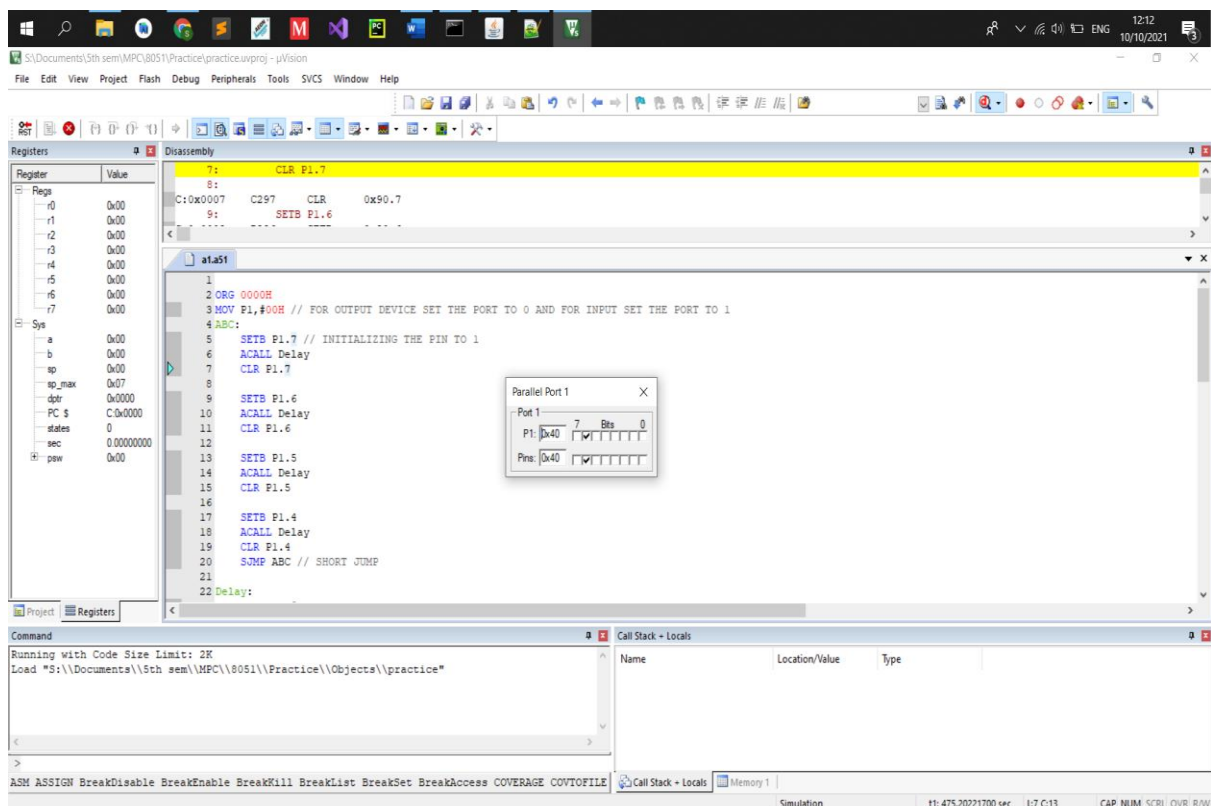
DJNZ R1, Y

DJNZ R0, Z

RET

END

➤ Output:



EXPERIMENT-6

Aim: Write an Assembly Language program to

(A) Monitoring P1.2 until it becomes high; when P1.2 becomes high write value 45H on P0, Sent a high to low pulse to P2.3.

➤ **Code:**

```
CLR P1.2
MOV A,#45H
AGAIN:JNB P1.2,AGAIN
MOV P0,A
SETB P2.3
CLR P2.3
END
```

(B) Check a status of P1.7 Switch and perform the following

1. If switch = 0, send letter "N" to P2.
2. If switch = 1, send letter "Y" to P2.

➤ **Code:**

```
SETB P1.7
AGAIN:
JB P1.2,OVER
MOV P2,#'N'
SJMP AGAIN
OVER:
MOV P2,#'Y'
SJMP AGAIN
```

EXPERIMENT-7

Aim: Write an Assembly Language program to.

(A) Write a program to generate 5 KHz pulse waveform of 50% duty cycle on pin 1.0 using timer 1 in mode 2.

➤ **Code:**

```
MOV TMOD,#01
HERE: MOV TL0,#0F2H
MOV TH0,#0FFH
CPL P1.5
ACALL DELAY
SJMP HERE
DELAY:
SETB TR0
AGAIN: JNB TF0,AGAIN
CLR TR0
CLR TF0
RET
```

(B) Write a program to generate 1 KHz pulse wave form of 70% duty cycle on pin 1.0 using timer.

➤ **Code:**

```
MOV TMOD, #01H
HERE: MOV TL0, #44H
MOV TH0, #0FDH
CPL P1.5
ACALL DELAY
SJMP HERE

DELAY:
SETB TR0
AGAIN: JNB TF0, AGAIN
CLR TR0
CLR TF0
RET
```

EXPERIMENT-8

Aim: Write an Assembly Language program to.

(A) Transfer letter “A” serially, continuously.

➤ **Code:**

```
MOV TMOD,#20H
MOV TH1,#-6
MOV SCON,#50H
SETB TR1
AGAIN:
MOV SBUF,#"A"
HERE:
JNB TI,HERE
CLR TI
SJMP AGAIN
```

(B) The message “YES” serially. Do this continuously.

➤ **Code:**

```
MOV TMOD,#20H
MOV TH1,#-3
MOV SCON,#50H
SETB TR1
AGAIN:
MOV A,#"Y"
ACALL TRANS
MOV A,#"E"
ACALL TRANS
MOV A,#"S"
ACALL TRANS
SJMP AGAIN
TRANS:
MOV SBUF,A
HERE:
JNB TI,HERE
CLR TI
RET
```

(C) Receive bytes of data serially, and put them in P1.

➤ **Code:**

```
MOV TMOD,#20H
MOV TH1,#-6
MOV SCON,#50H
SETB TR1
HERE:
JNB RI,HERE
MOV A,SBUF
MOV P1,A
CLR RI
SJMP HERE
```


EXPERIMENT-9

Aim: Write an Assembly Language program to generate a square wave of 2 Khz and also receive a byte of data serially and send to P0 continuously using interrupt method.

➤ **Code:**

```
MOV TMOD,#01
AGAIN: MOV TL1,#1AH
MOV TH1,#0FFH
SETB TR1
BACK: JNB TF1,BACK
CLR TR1
CLR P0.5
CLR TF1
SJMP AGAIN
```

EXPERIMENT-10

Aim: Write an Assembly Language program to display string “RK University” on 16x2 LCD

➤ **Code:**

```
ORG 0000H
MOV P1, #00H

MOV A,#38H ; Use 2 lines and 5x7 matrix
ACALL CMND
MOV A,#0FH ; LCD ON, cursor ON, cursor blinking ON
ACALL CMND
MOV A,#01H ;Clear screen
ACALL CMND
MOV A,#06H ;Increment cursor
ACALL CMND
MOV A,#87H ;Cursor line one , position 2
ACALL CMND

MOV A,#'R'
ACALL DISP
MOV A,#'K'
```

ACALL DISP

MOV A,#0C3H ;Jump to second line, position 1

ACALL CMND

MOV A,#'U'

ACALL DISP

MOV A,#'N'

ACALL DISP

MOV A,#'I'

ACALL DISP

MOV A,#'V'

ACALL DISP

MOV A,#'E'

ACALL DISP

MOV A,#'R'

ACALL DISP

MOV A,#'S'

ACALL DISP

MOV A,#'I'

ACALL DISP

MOV A,#'T'

ACALL DISP

MOV A,#'Y'

ACALL DISP

HERE: SJMP HERE

CMND: MOV P1,A

CLR P2.0

SETB P2.1

ACALL DELY

CLR P2.1

RET

DISP:MOV P1,A

setb P2.0

SETB P2.1

ACALL DELY

CLR P2.1

RET

DELY:

MOV R1,#255

```

DEF:MOV R2,#255
GHI: DJNZ R2,GHI
      DJNZ R1,DEF
RET

END

```

➤ Output:

