11/12/19

TASK-I

EXP-I

Code:

```
org 0000h
mov a, #05h
mov r0, #05h
add a, r0
H: SJMP H
end
```

```
org 0000h
mov a, #0ah
mov r0, #05h
subb a, r0
H: SJMP H
end
```

```
org 0000h
mov a, #05h
mov b, #05h
mul ab
H: SJMP H
end
```

```
org 0000h
mov a, #020h
mov b, #05h
div ab
H: SJMP H
end.
```

STEPS:-

1) Go to start.
2) 'Kill microvision' ◄ select this option.
3) Go to project and close project.
4) Go to project and select 'new microvision project'.
5) Go to documents.
6) Click 'new folder'.
7) Open the folder.
8) Give a file name and ~~select~~ select 'save'.
9) Search for 'NXP' and enp and 'NXP'
10) out of all the configuration, use 'P89V51RD2'.
11) Click 'Ok'.
12) Click 'yes'.
13) Expand 'target 1' ► source group.
14) Go to 'file' and click 'new'.
15) Write down code.
16) Go to 'file' ► 'save as'.
17) File name must be given with '.ASM' extention.
18) Click 'save'
19) Right click 'source group1'.
20) Click add existing files to group 'source group1'.
21) Folder name must be same and use '.asm' source file ~~and~~ under the 'file of type' drop down menu.
22) Select 'add' and then 'close'.
23) Click on 'translate' to compile program (check for error)
24) Go to debug ► start/stop debug to run the code.
25) After step 23: built ► rebuilt

TASK1 EXPT2

Program - 1

Write and assemble a program to load values into each of registers R0 - R4 and then push each of these registers onto the stack. Single-step the program, and examine the stack and the SP register after the execution of each instruction.

```
ORG 0000H

MOV R0, #25H

MOV R1, #35H

MOV R2, #45H

MOV R3, #55H

MOV R4, #65H

PUSH 0

PUSH 1

PUSH 2

PUSH 3

PUSH 4

END
```

Program - 2

Write an 8051 assemble language program to:

(a) Set SP = 0D,
(b) Put a different value in each of RAM locations 0D, 0C, 0B, 0A, 09 and 08
(c) POP each stack location into registers R0 - R4.

Use the simulator to single-step and examine the registers, the stack, and the stack pointer.

```
ORG 0000H
```

```
MOV SP, #0DH
MOV 08H, #10H
MOV 09H, #11H
MOV 0AH, #12H
MOV 0BH, #13H
MOV 0CH, #14H
MOV 0DH, #16H
POP 0
POP 1
POP 2
POP 3
POP 4
END
```

Program-3

Write and assemble a program to load values into each of registers R0 - R4 and then push each of these registers onto the stack and pop them back. Single-step the program, and examine the stack and the SP register after the execution of each instruction.

```
ORG 0000H
MOV R0, #25H
MOV R1, #35H
MOV R2, #45H
MOV R3, #55H
MOV R4, #65H
PUSH 0
PUSH 1
```

```
PUSH 2

PUSH 3

PUSH 4

POP 0

POP 1

POP 2

POP 3

POP 4

END
```

Program-4

Write and assemble a program to add the following data and then use the simulator to examine the CY flag.

92H, 23H, 66H, 87H, F5H

```
ORG 0000H

MOV A, #92H

MOV R0, #23H

ADD A, R0

JNC L1

INC R7

L1: MOV R1, # 66H

ADD A, R1

JNC L2

INC R7

L2: MOV R2, #87H
```

```
ADD A, R2
JNC L3
INC R7
L3: MOV R3, #0F5H
ADD A, R3
JNC L4
INC R7
L4:
END
```

TASK1-EXPT3

Program 1

Write a program to transfer a string of data from code space starting at address 200H to RAM locations starting at 40H. The data is as shown below:

0200H: DB "VIT UNIVERSITY"

Using the simulator, single-step through the program and examine the data transfer and registers.

```
        ORG 0000H
        MOV A, #00H
        MOV DPTR, #0200H
        MOV R1, #0EH
        MOV R0, #40H
LOOP:   CLR A
        MOVC A,@A+DPTR
        MOV @R0, A
        INC DPTR
        INC R0
        DJNZ R1, LOOP
HERE:   SJMP HERE
        ORG 0200H
        DB "VIT UNIVERSITY"
        END
```

Program 2

Add the following subroutine to the program 1, single-step through the subroutine and examine the RAM locations. After data has been transferred from ROM space into RAM, the subroutine should copy the data from RAM locations starting at 40H to RAM locations starting at 60H.

```
ORG 000H
MOV DPTR, #200H
MOV R0, #40H
MOV R1, #0EH
LOOP: CLR A
MOVC A,@A+DPTR
MOV @R0, A
INC R0
INC DPTR
DJNZ R1, LOOP
MOV R0, #40H
MOV R1, #60H
MOV R3, #0EH
LOOP2: CLR A
MOV A,@R0
MOV @R1, A
INC R0
INC R1
DJNZ R3, LOOP2
HERE: SJMP HERE
ORG 200H
DB "VIT UNIVERSITY"
END
```

TASK1 EXPT4

Program 1

Write a program to add 10 bytes of data and store the result in registers R2 and R3. The bytes are stored in the ROM space starting at 200H. The data would look as follows:

MYDATA: DB 92, 34, 84, 129, ... ;

Pick your own data. Notice that you must first bring the data from ROM space into the CPU's RAM, and then add them together. Use a simulator to single-step the program and examine the data.

```
ORG 000H
MOV DPTR, #200H
MOV R0, #10
LOOP: CLR A
MOVC A,@A+DPTR
ADD A, R2
JNC NEXT
INC R3
NEXT: INC DPTR
MOV R2, A
DJNZ R0, LOOP
HERE: SJMP HERE
ORG 200H
DB 22H,43H,23H,34H,31H,77H,91H,33H,43H,7H
END
```

Program 2

Write a program to add 10 bytes of BCD data and store the result in R2 and R3. The bytes are stored in ROM space starting at 300H. The data would look as follows:

MYDATA: DB 92H, 34H, 84H, 29H ,... ; pick your own data.

Notice that you must first bring the data from ROM space into the CPU's RAM, and then add them together. Use a simulator to single-step the program and examine the data.

```
ORG 000H

MOV DPTR, #200H

MOV R0, #10

LOOP: CLR A

MOVC A,@A+DPTR

ADD A, R2

DA A

JNC NEXT

INC R3

NEXT: INC DPTR

MOV R2, A

DJNZ R0, LOOP

HERE: SJMP HERE

ORG 200H

DB 22H,43H,23H,34H,31H,77H,91H,33H,43H,7H

END
```

TASK1 EXPT5

Program1

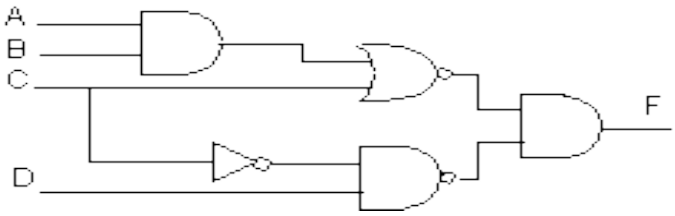
Write a program to get a byte of hex data from P1, convert it to decimal, and then to ASCII. For example, if P1 has FBH, which is equal to 251 in decimal, after conversion we will have 32H, 35H, and 31H. Place the ASCII result in RAM locations starting at 40H. Using a simulator, single-step the program and examine the data.

```
ORG 0000H
MOV P1, #0FBH
MOV R0, #40H
MOV A, P1
LOOP: MOV B,#10
DIV AB
XCH A, B
ADD A, #30H
MOV @R0, A
XCH A, B
INC R0
JNZ LOOP
END
```

TASK 2 EXPT1:

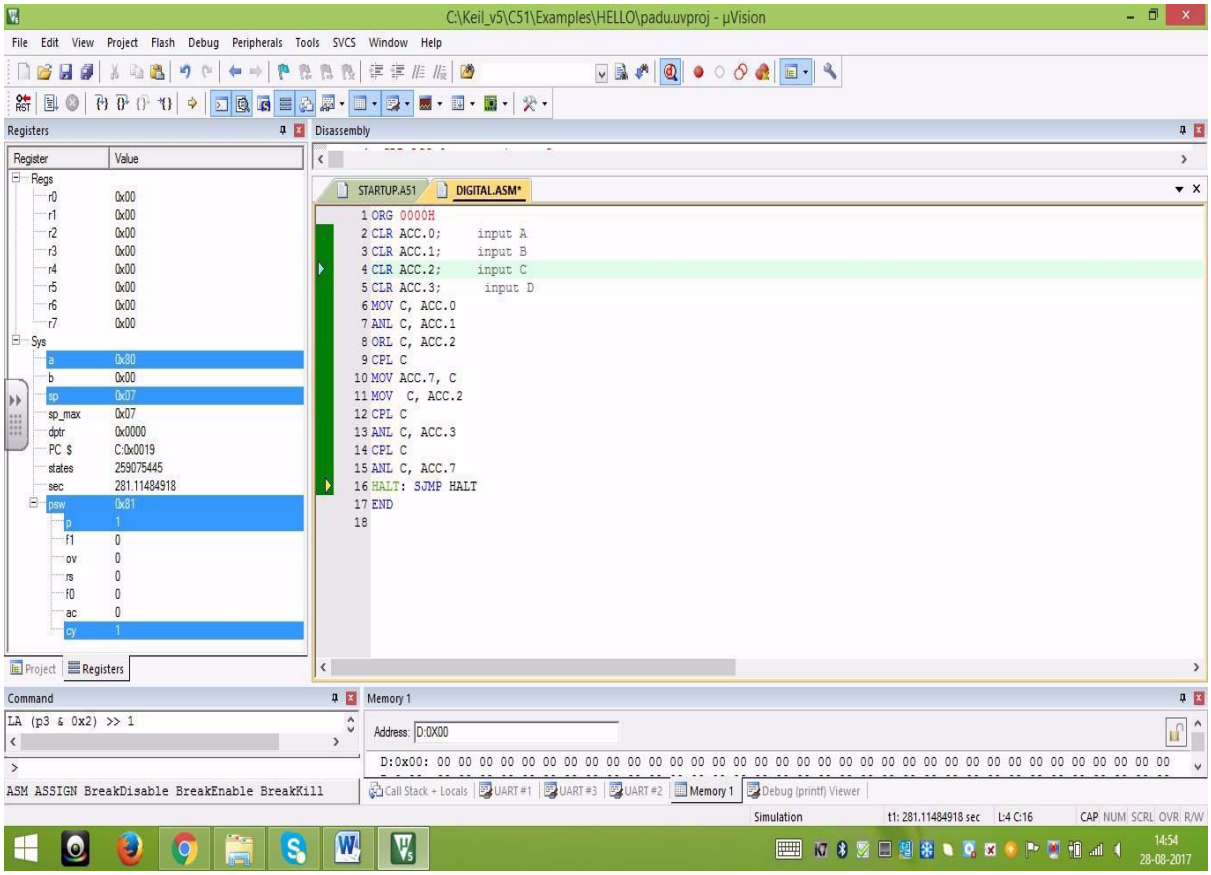
Implementation of digital circuits using KEIL for an 8051 Microcontroller

Objective: To develop an assembly code for 8051 Microcontroller, to implement the given digital circuit using KEIL development tool.




```
ORG 0000H
SETB ACC.0;    input A
SETB ACC.1;    input B
CLR ACC.2;     input C
SETB ACC.3;    input D
MOV C, ACC.0
ANL C, ACC.1
ORL C, ACC.2
CPL C
MOV ACC.7, C
MOV  C, ACC.2
CPL C
ANL C, ACC.3
CPL C
ANL C, ACC.7;  Final output (F)
HALT: SJMP HALT
END
```

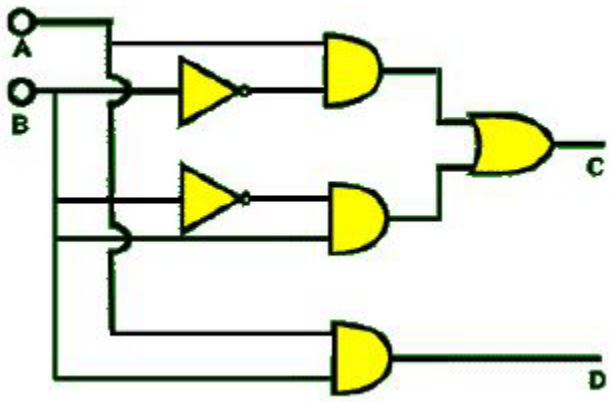
KEIL SIMULATION



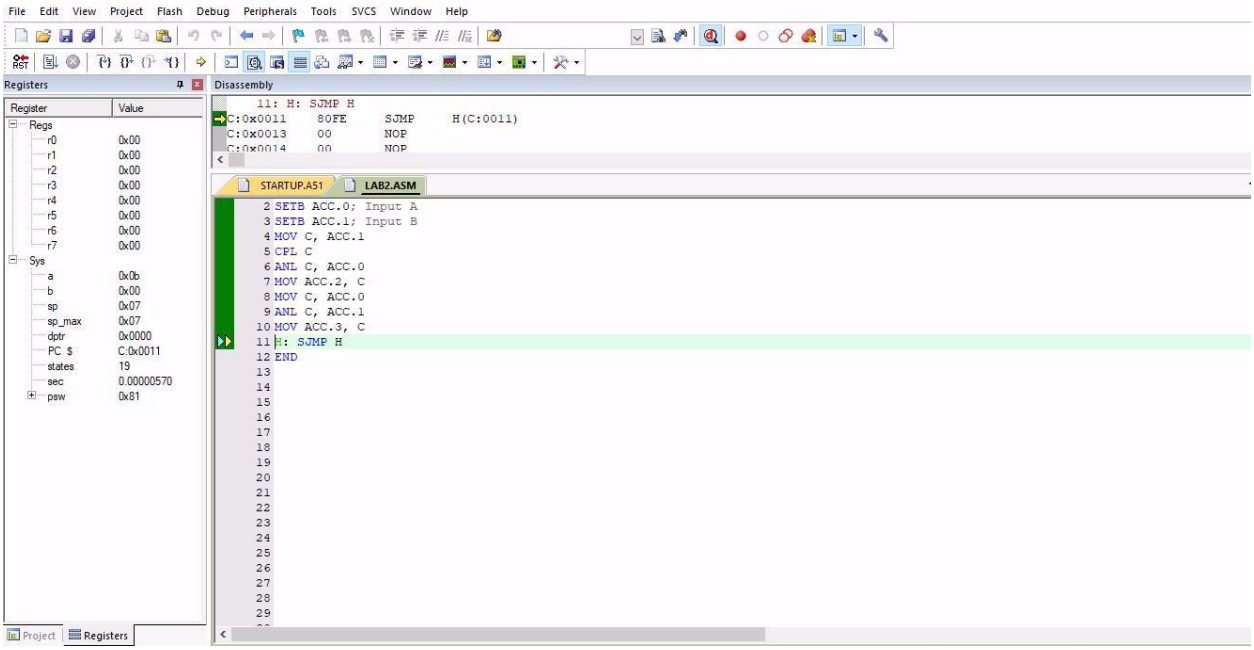
RESULT:

S.NO	A	B	C	D	Output (F)
1	0	0	0	0	
2	0	0	0	1	
3	0	0	1	0	
4	0	0	1	1	
5	0	1	0	0	
6	0	1	0	1	
7	0	1	1	0	
8	0	1	1	1	
9	1	0	0	0	
10	1	0	0	1	
11	1	0	1	0	
12	1	0	1	1	
13	1	1	0	0	
14	1	1	0	1	
15	1	1	1	0	
16	1	1	1	1	

DIGITAL CIRCUIT PROGRAM 2




```
ORG 0000H

SETB ACC.0; Input A

SETB ACC.1; Input B

MOV C, ACC.1

CPL C

ANL C, ACC.0

MOV ACC.2, C; Output C

MOV C, ACC.0

ANL C, ACC.1

MOV ACC.3, C; Output D

H: SJMP H

END
```



RESULT:

S.NO	A	B	Output (C)	Output(D)
1	0	0		
2	0	1		
3	1	0		
4	1	1		

TASK2-EXPT2

PORT PROGRAMMING

PROGRAM 1

Write and assemble a program to toggle all the bits of P0, P1, and P2 continuously by sending 55H and AAH to these ports.
Put a time delay between the "on" and "off" states. Then using the simulator, single-step through the program and examine the ports. Do not single-step through the time delay call.

```
ORG 0000H
HERE: MOV P0, #55H
MOV P1, #55H
MOV P2, #55H
ACALL DELAY
MOV P0, #0AAH
MOV P1, #0AAH
MOV P2, #0AAH
ACALL DELAY
SJMP HERE
DELAY:  MOV R1, #04H
BACK:  MOV R2, #20H
AGAIN: DJNZ R2, AGAIN
DJNZ R1, BACK
RET
END
```

PROGRAM 2

Get the Data from Port P1 and Send it to Port P2, Note: P1 as input Port and P2 as Output Port

```
MOV A, #0FFh        ; A=FFH
MOV P1, A           ; make P1 an input port
HERE: MOV A, P1     ; get data from P1
MOV P2, A           ; send it to P2
SJMP HERE           ; keep doing this
END
```

TASK3

PROGRAM 1-TIMER OPERATION MODE1

Write a program using timer 0 to generate a 500 Hz square wave frequency on one of the pins of P1.0

Then examine the frequency using the KEIL IDE inbuilt Logic Analyzer.

```
ORG 0000H

MOV TMOD, #01H

HERE: MOV TL0, #66H

MOV TH0, #0FCH

CPL P1.0

ACALL DELAY

SJMP HERE

DELAY: SETB TR0

AGAIN: JNB TF0, AGAIN

CLR TR0

CLR TF0

RET

END
```

PROGRAM 2-TIMER OPERATION MODE1

Write a program using timer 1 to generate a 1 kHz square wave frequency on one of the pins of P1.

Then examine the frequency using the oscilloscope.

```
 ORG 0000H
MOV TMOD, #10H
HERE: MOV TL1, #33H
MOV TH1, #0FEH
CPL P1.0
ACALL DELAY
SJMP HERE
DELAY: SETB TR1
AGAIN: JNB TF1, AGAIN
CLR TR1
CLR TF1
RET
END
```

PROGRAM 3-TIMER OPERATION MODE2

Write a program using timer 1 to generate a 2 KHz square wave frequency on one of the pins of P1.0.

Then examine the frequency using the KEIL IDE inbuilt Logic Analyzer.

```
ORG 0000H

MOV TMOD, #20H

HERE: MOV TH1, #19H

BACK: SETB TR1

AGAIN: JNB TF1, AGAIN

CPL P1.0

CLR TF1

SJMP BACK

END
```

PROGRAM 4-COUNTER OPERATION

Assuming that clock pulses are fed into pin T1,write a program for counter 1 in mode 2 to count the pulses and display the state of the TL1 count on P2, which connects to 8 LEDs.

```
MOV TMOD, #01100000B      ; counter 1, mode 2, ;C/T=1 external pulses
MOV TH1, #0               ; clear TH1
SETB P3.5                 ; make T1 input
AGAIN: SETB TR1           ; start the counter
BACK: MOV A, TL1          ; get copy of TL
MOV P2, A                 ; display it on port 2
JNB TF1, BACK             ; keep doing, if TF = 0
CLR TR1                   ; stop the counter 1
CLR TF1                   ; make TF=0
SJMP AGAIN
```

TASK4

PROGRAM1

Write an 8051 assembly program to transfer data serially at baud rate 9600 with 8 bit data, one stop bit and observe the transmitted data in the serial window of the simulator.

```
ORG 0000H

XX: MOV DPTR, #MYDATA

MOV TMOD, #20H

MOV TH1, #-3

MOV SCON, #50H

SETB TR1

MOV R1, #14

AGAIN: CLR A

MOVC A,@A+DPTR

MOV SBUF, A

HERE: JNB TI, HERE

CLR TI

INC DPTR

DJNZ R1, AGAIN

SJMP XX

MYDATA: DB 'VIT UNIVERSITY'

END
```

PROGRAM2

Write an 8051 Assembly Language program to get data from the PC and display it on P1. Assume 8051 is connected to PC and observe the incoming characters. As you press a key on the PC's keyboard, the character is sent to the 8051 serially at 4800 baud rate and is displayed on LEDs. The characters displayed on LEDs are in ASCII (binary).

```
ORG 0000H
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
END
```

TASK5

PROGRAM1

Write an 8051 program to get data from port P0 and send it to port P1continuously while an interrupt will do the following: Timer 0 will toggle the P2.1 bit every 100 microseconds.

```
ORG 0000H
LJMP MAIN                  // Jump to main.
ORG 000BH                  // timer 0 Interrupt vector label
CPL P2.1                   // Toggle P2.1 pin
RETI                       // Return from ISR
ORG 0030H                  // After Vector Table Space
MAIN: MOV TMOD,#02H     //set Timer 0 in mode 2
MOV P0, #0FFH              // Set P0 as I/P port
MOV TH0, #-92              //
MOV IE, #10000010B   // enable Timer 0
SETB TR0                   // Start the Timer
BACK: MOV A, P0      // Get data from P0.
MOV P1, A                  // move data P0 to P1
SJMP BACK             //Loop it unless interrupted
END
```

Program 2

Write an 8051 program to get data from a single bit of P1.2 and send it to P1.7 continuously while an interrupt will do the following: A serial interrupt service routine will receive data from a PC and display it on P2 ports.

```
ORG 0000H

LJMP MAIN

ORG 0023H        ; ----serial interrupt vector table

LJMP SERIAL

ORG 0030H        ;--  after vector table space

MAIN: SETB P1.2 ;    --    P1.2 made as input pin

MOV TMOD, #20H; -- timer 1 mode 2

MOV TH1, #-3; -- set baud rate 9600

MOV SCON, #50H; -- one stop bit

MOV IE, #10010000B; -- serial  int. enabled.

SETB TR1 ;-- Timer 1 stared.

BACK: MOV C, P1.2

MOV P1.7, C

SJMP BACK

SERIAL: JB TI, TRANS

MOV A, SBUF

MOV P2, A

CLR RI

RETI

TRANS: CLR TI

RETI

END
```