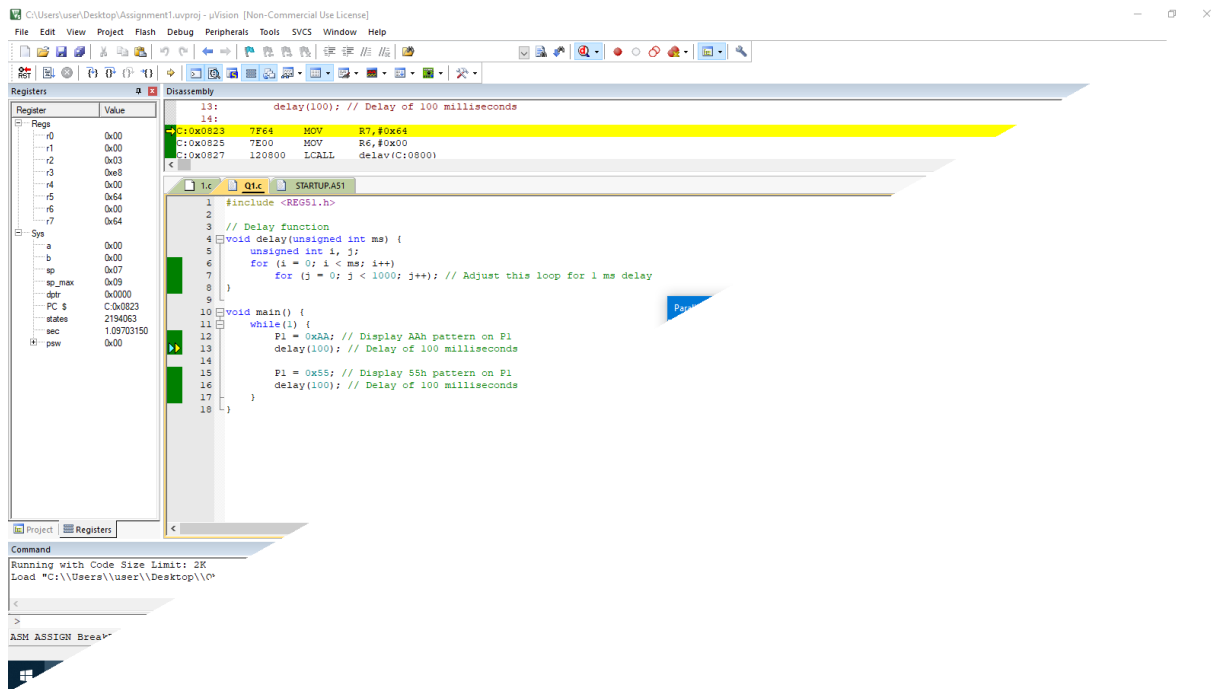


Assignment 6

➤ Write Program in KEIL Embedded C:

1. Write a C program for the 8051 to display a pattern of AA and 55h on port P1 with the delay of 100 ms.

```
#include <REG51.h>
// Delay function
void delay(unsigned int ms) {
    unsigned int i, j;
    for (i = 0; i < ms; i++)
        for (j = 0; j < 1000; j++); // Adjust this loop for 1 ms delay
}
void main() {
    while(1) {
        P1 = 0xAA; // Display AAh pattern on P1
        delay(100); // Delay of 100 milliseconds
        P1 = 0x55; // Display 55h pattern on P1
        delay(100); // Delay of 100 milliseconds
    }
}
```

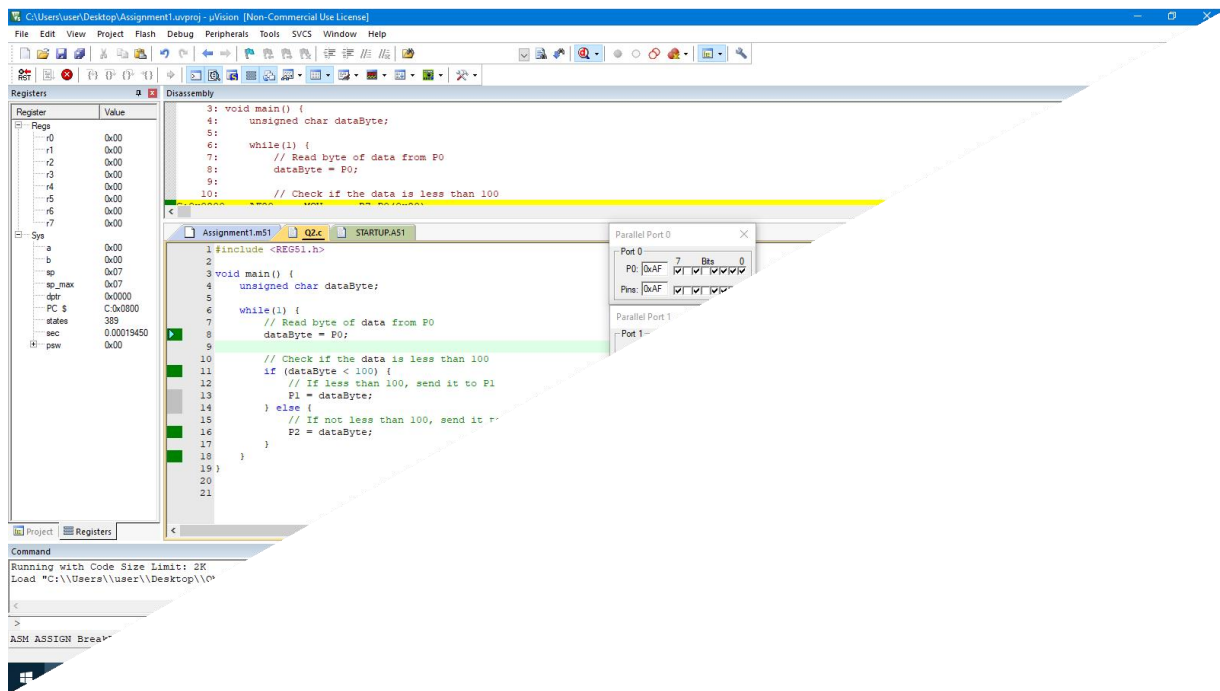


2. Write an 8051 C program to get a byte of data from P0. If it is less than 100, send it to P1; otherwise, send it to P2.

```

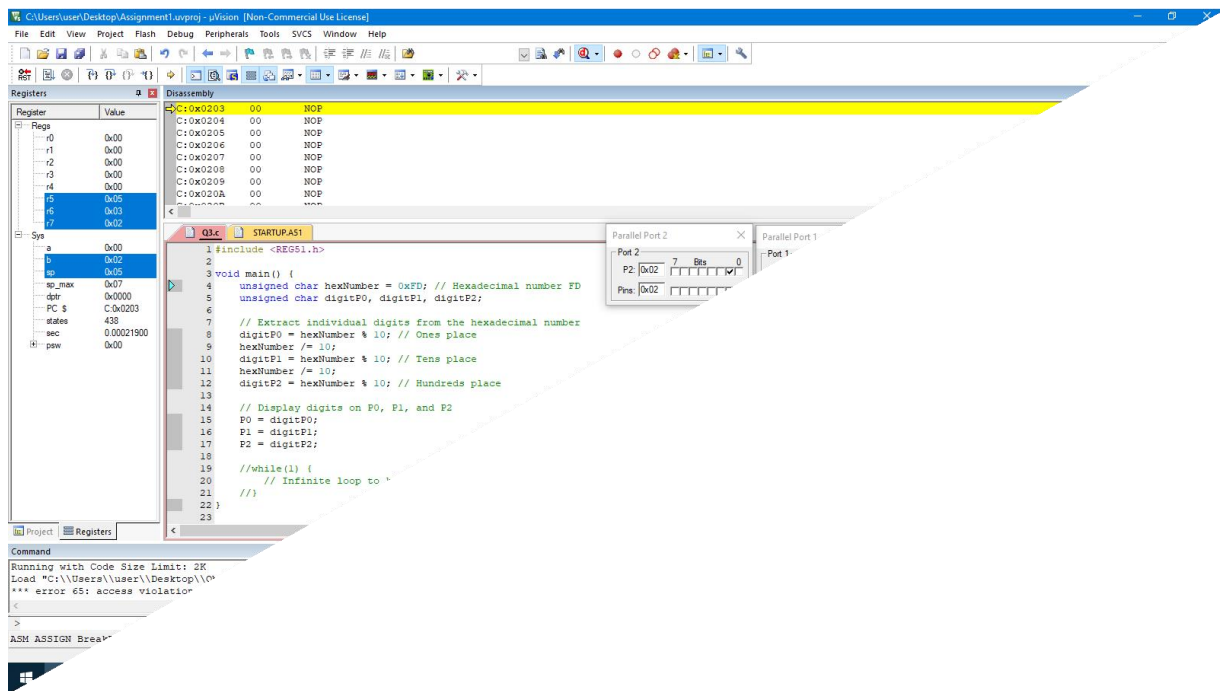
#include <REG51.h>
void main() {
    unsigned char dataByte;
    while(1) {
        // Read byte of data from P0
        dataByte = P0;
        // Check if the data is less than 100
        if (dataByte < 100) {
            // If less than 100, send it to P1
            P1 = dataByte;
        } else {
            // If not less than 100, send it to P2
            P2 = dataByte;
        }
    }
}

```



3. Write an 8051 C program to convert 11111101 (FD hex) to decimal and display the digits on P0, P1 and P2.

```
#include <REG51.h>
void main() {
    unsigned char hexNumber = 0xFD; // Hexadecimal number FD
    unsigned char digitP0, digitP1, digitP2;
    // Extract individual digits from the hexadecimal number
    digitP0 = hexNumber % 10; // Ones place
    hexNumber /= 10;
    digitP1 = hexNumber % 10; // Tens place
    hexNumber /= 10;
    digitP2 = hexNumber % 10; // Hundreds place
    // Display digits on P0, P1, and P2
    P0 = digitP0;
    P1 = digitP1;
    P2 = digitP2;
}
```



➤ To be done using EdSim51 simulator in 8051:

1. Write a Program to check whether a number is palindrome or not. If palindrome store FFh in accumulator.

```

ORG 0000H
MOV DPTR,#8000H
MOVBX A,@DPTR
MOV R0, A
MOV R2, #00H
BACK: MOV B, #0AH
DIV AB
MOV B, A
MUL AB
SUBB A, R0
JNZ NOT_PALINDROME
INC R2
JMP NEXT
NOT_PALINDROME: CLR A
JMP STORE_RESULT
NEXT: INC DPTR
MOVBX A, @DPTR
CJNE A, #0FFH, BACK
JMP STORE_RESULT

```

```
STORE_RESULT: MOV A, #0FFH
MOV DPTR, #8100H
MOVX @DPTR, A
END
```

2. Write an assembly language program to compute prime factors of a number.

```
mov r2,#3Ch
mov r1,#30h
mov r0,#02h
loop1: clr a
clr c
mov a,r2
mov b,r0
div ab
mov r3,a
clr c
mov a,b
subb a,#00h
jz join
clr a
clr c
inc r0
mov a,r3
```

EdSim51DI - Version 2.1.23 | primefactors.asm

System Clock (MHz) 12.0

SBUF

R/O W/O TH0 TL0 R7 0x00 B 0x00

0x00 0x00 0x00 0x00 R6 0x00 ACC 0x00

RXD TXD R5 0x00 PSW 0x00

1 1 TMOD 0x00 R4 0x00 IP 0x00

SCON 0x00 TCON 0x00 R3 0x00 IE 0x00

R2 0x01 PCON 0x00

pins bits TH1 TL1 R1 0x34 DPH 0x00

0xFF 0xFF P3 0x00 0x00 R0 0x06 DPL 0x00

0xFF 0xFF P2 PC 8051 SP 0x07

0xFF 0xFF P1 0x0000 PSW 0 0 0 0 0 0 0 0

0xFF 0xFF P0

Data Memory

addr	0x00	0x01	value
0	0	1	2
1	2	3	4
2	4	5	6
3	6	7	8
4	8	9	A
5	A	B	C
6	B	C	D
7	C	D	E
8	D	E	F
9	E	F	
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			
32			
33			
34			
35			
36			
37			
38			
39			
40			
41			
42			
43			
44			
45			
46			
47			
48			
49			
50			
51			
52			
53			
54			
55			
56			
57			
58			
59			
60			
61			
62			
63			
64			
65			
66			
67			
68			
69			
70			

Copyright ©2005-2021 James Rogers

Remove All

DI / LD

7 6 5 4 3

0.0 V

RST Assm Run New Load Save Copy Paste

File primefactors.asm saved.

```

mov r2, #3Ch
mov r1, #30h
mov r0, #02h

loop1: clr a
      clr c
      mov a, r2
      mov b, r0
      div ab
      mov r3, a
      clr c
      mov a, b
      subb a, b
      jz

```

P0.7 1 Display-select Decoder CS1

P0.6 1 Keypad Column 2

P0.5 1 Keypad Column 1

P0.4 1 Keypad Column 0

P0.3 1 Keypad Row 3

P0.2 1 Keypad Row 2

P0.1 1 Keypad Row 1

P0.0 1 Keypad Row 0

P1.7 1 Keypad Row 7

P1.6 1 Keypad Row 6

```
ORG 0000H
START: MOV P1, #00000000B
LOOP: MOV A, P1
      CPL A
      MOV P1, A
      CALL DELAY
```

```
SJMP LOOP
DELAY: MOV R2, #0FFH
L1: DJNZ R2, L1
RET
END
```