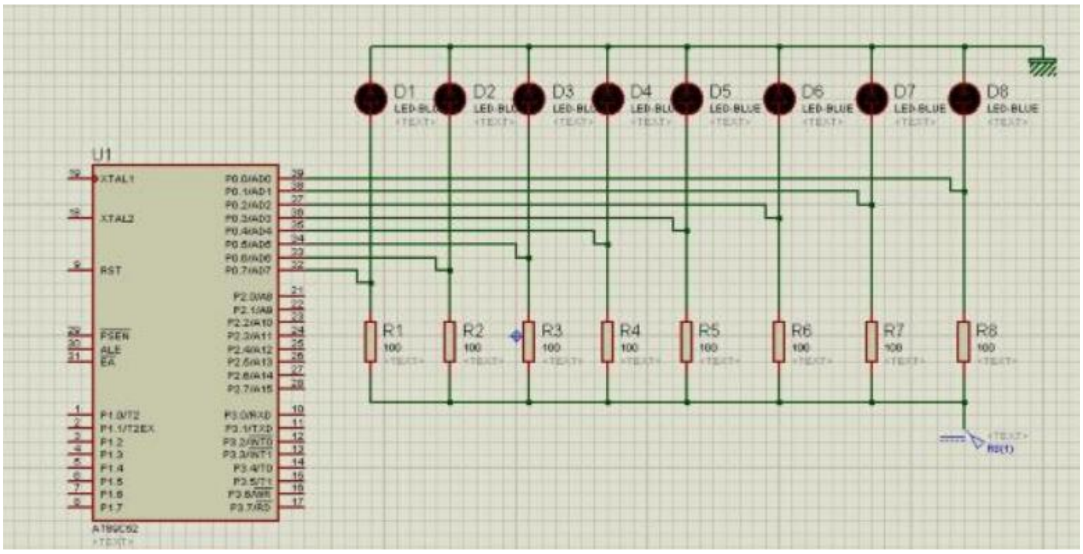
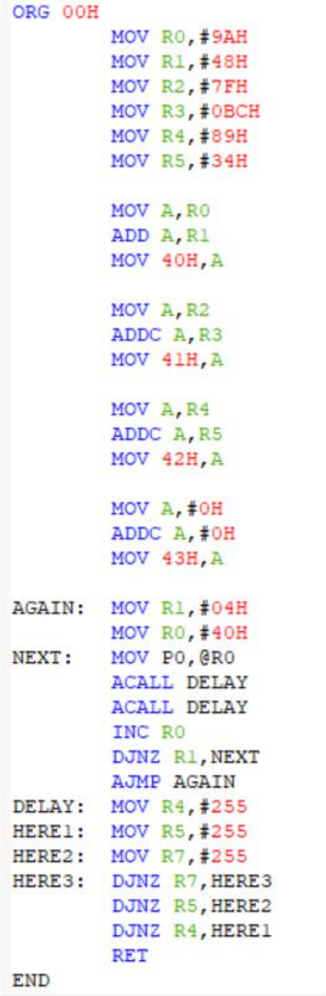


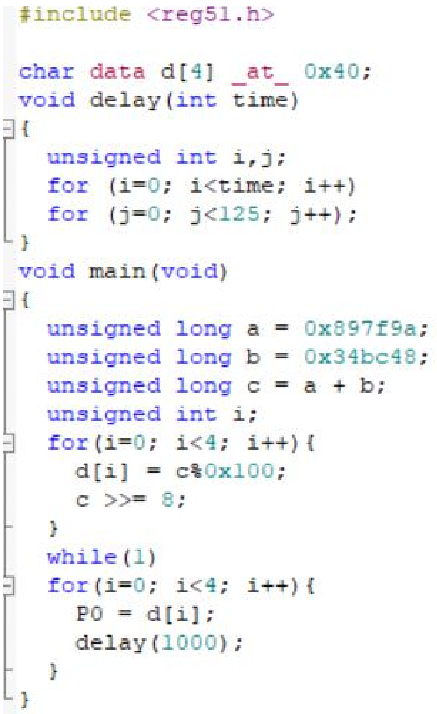
1. Write code to add the numbers 897F9AH and 34BC48H and save the result in internal RAM starting at 40H. The result should be displayed continuously on the LEDs of the development board starting from least significant byte with an appropriate timing interval between each byte. Use port zero (P0) of the micro-controller to interface with LEDs.

Assembly code

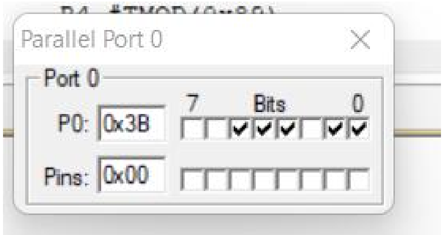


```
ORG 00H
        MOV R0,#9AH
        MOV R1,#48H
        MOV R2,#7FH
        MOV R3,#0BCH
        MOV R4,#89H
        MOV R5,#34H

        MOV A,R0
        ADD A,R1
        MOV 40H,A

        MOV A,R2
        ADDC A,R3
        MOV 41H,A

        MOV A,R4
        ADDC A,R5
        MOV 42H,A

        MOV A,#0H
        ADDC A,#0H
        MOV 43H,A

AGAIN:  MOV R1,#04H
        MOV R0,#40H
NEXT:   MOV P0,@R0
        ACALL DELAY
        ACALL DELAY
        INC R0
        DJNZ R1,NEXT
        AJMP AGAIN
DELAY:  MOV R4,#255
HERE1:  MOV R5,#255
HERE2:  MOV R7,#255
HERE3:  DJNZ R7,HERE3
        DJNZ R5,HERE2
        DJNZ R4,HERE1
        RET
END
```

C code



```
#include <reg51.h>

char data d[4] _at_ 0x40;
void delay(int time)
{
  unsigned int i,j;
  for (i=0; i<time; i++)
  for (j=0; j<125; j++);
}
void main(void)
{
  unsigned long a = 0x897f9a;
  unsigned long b = 0x34bc48;
  unsigned long c = a + b;
  unsigned int i;
  for(i=0; i<4; i++){
    d[i] = c%0x100;
    c >>= 8;
  }
  while(1)
  for(i=0; i<4; i++){
    P0 = d[i];
    delay(1000);
  }
}
```

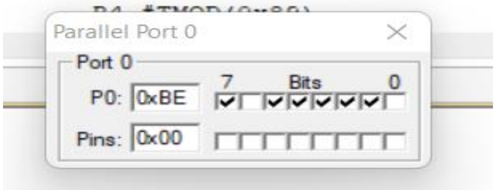


Parallel Port 0

Port 0

P0: 0x3B □□☑☑☑□☑☑

Pins: 0x00 □□□□□□□□

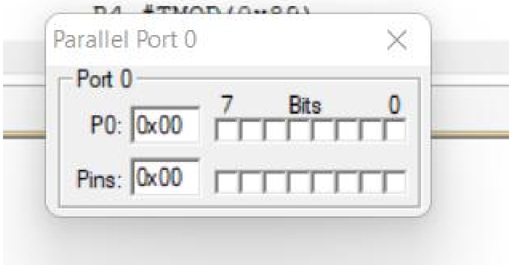


Parallel Port 0

Port 0

P0: 0xBE ☑□☑☑☑☑☑□

Pins: 0x00 □□□□□□□□

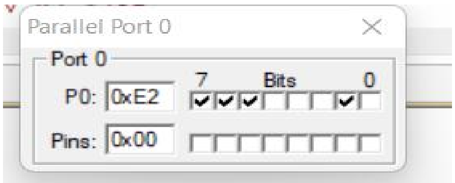


Parallel Port 0

Port 0

P0: 0x00 □□□□□□□□

Pins: 0x00 □□□□□□□□



Parallel Port 0

Port 0

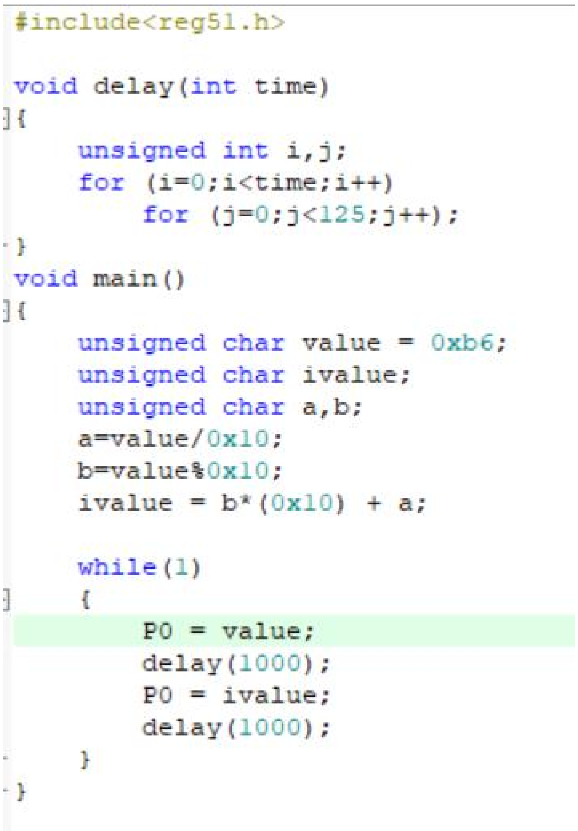
P0: 0xE2 ☑☑☑□□□☑□

Pins: 0x00 □□□□□□□□

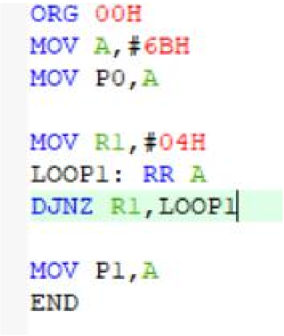
2. Implement a subroutine that replaces the SWAP instruction using rotate right instructions. Test your program on the contents of the accumulator when it contains the number 6BH.

Assembly code

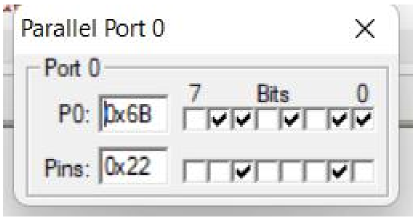
C code



```
#include<reg51.h>

void delay(int time)
{
     unsigned int i,j;
     for (i=0;i<time;i++)
          for (j=0;j<125;j++);
}
void main()
{
     unsigned char value = 0xb6;
     unsigned char ivalue;
     unsigned char a,b;
     a=value/0x10;
     b=value%0x10;
     ivalue = b*(0x10) + a;

     while(1)
     {
          P0 = value;
          delay(1000);
          P0 = ivalue;
          delay(1000);
     }
}
```



```
ORG 00H
MOV A,#6BH
MOV P0,A

MOV R1,#04H
LOOP1: RR A
DJNZ R1,LOOP1

MOV P1,A
END
```

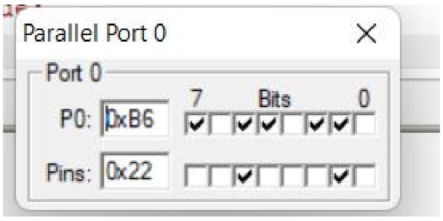


Parallel Port 0

Port 0

P0: 0x6B □☑☑□☑□☑☑

Pins: 0x22 □□☑□□□☑□



Parallel Port 0

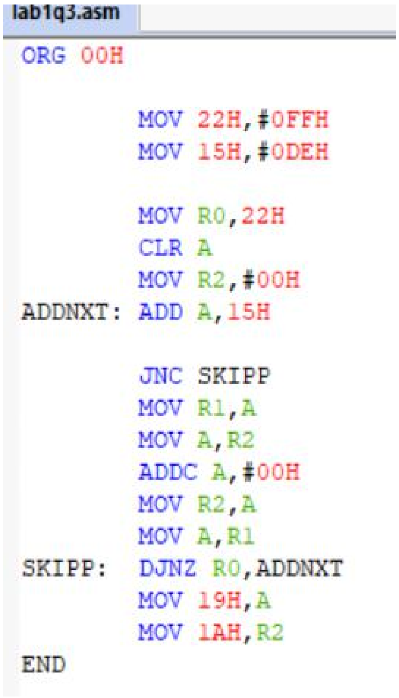
Port 0

P0: 0xB6 ☑□☑☑□☑☑□

Pins: 0x22 □□☑□□□☑□

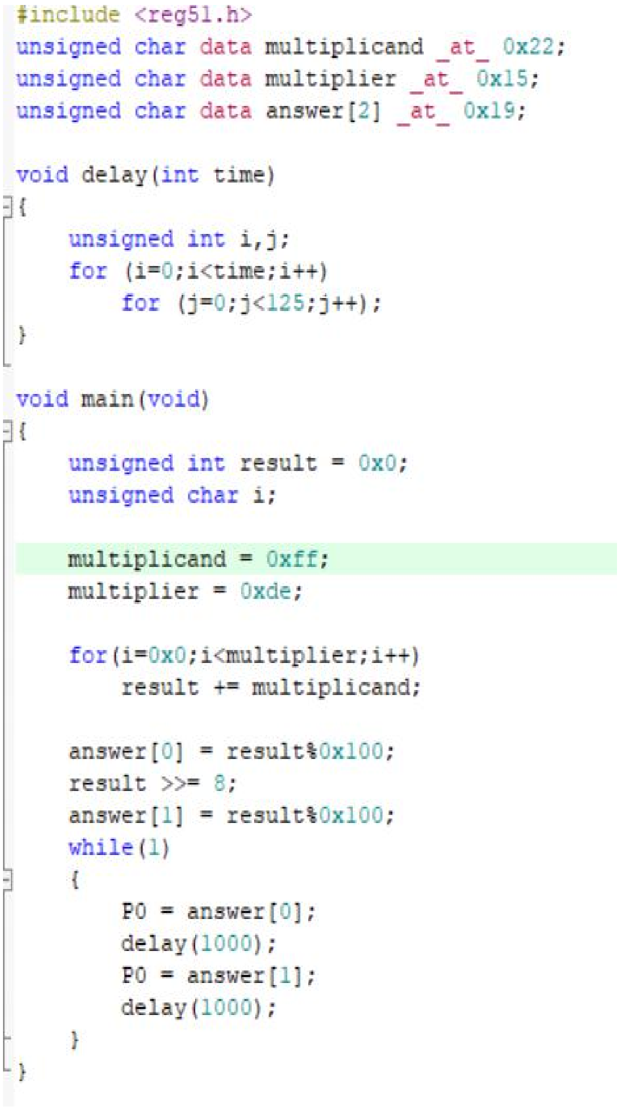
3. Multiply, by using looping and successive addition technique, the data in RAM location 22H by the data in RAM location 15H and put the result in RAM locations 19H (low byte) and 1AH (high byte). Data in 22H should be FFH and data in 15H should be DEH.

Assembly Code

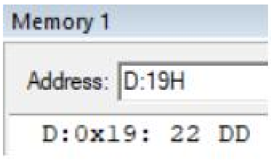


```
ORG 00H

        MOV 22H,#0FFH
        MOV 15H,#0DEH

        MOV R0,22H
        CLR A
        MOV R2,#00H
ADDNXT: ADD A,15H

        JNC SKIPP
        MOV R1,A
        MOV A,R2
        ADDC A,#00H
        MOV R2,A
        MOV A,R1
SKIPP:  DJNZ R0,ADDNXT
        MOV 19H,A
        MOV 1AH,R2
END
```

C code



```
#include <reg51.h>
unsigned char data multiplicand _at_ 0x22;
unsigned char data multiplier _at_ 0x15;
unsigned char data answer[2] _at_ 0x19;

void delay(int time)
{
    unsigned int i,j;
    for (i=0;i<time;i++)
        for (j=0;j<125;j++);
}

void main(void)
{
    unsigned int result = 0x0;
    unsigned char i;

    multiplicand = 0xff;
    multiplier = 0xde;

    for(i=0x0;i<multiplier;i++)
        result += multiplicand;

    answer[0] = result%0x100;
    result >>= 8;
    answer[1] = result%0x100;
    while(1)
    {
        P0 = answer[0];
        delay(1000);
        P0 = answer[1];
        delay(1000);
    }
}
```



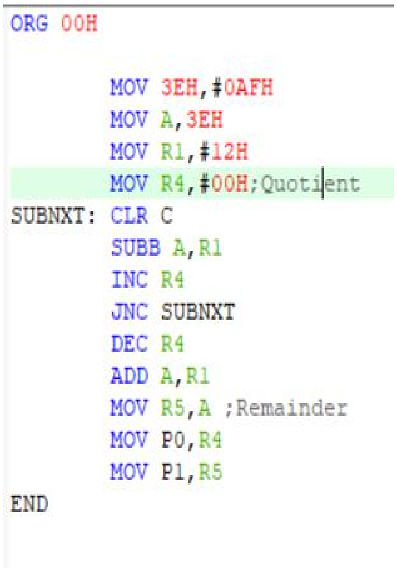
Address: D:19H

```
D:0x19: 22 DD
```

Product of FFH and DEH is DD22H. 19H has lower byte(22H) and 0AH has higher byte(DDH).

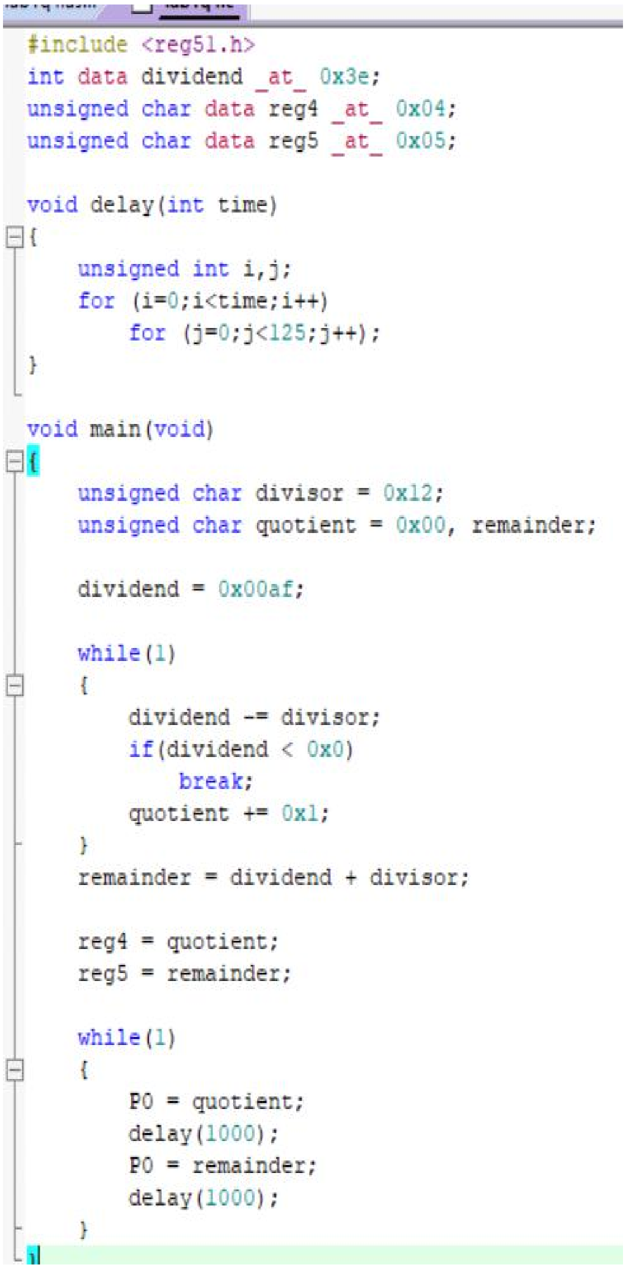
4. Divide, by using looping and successive subtraction technique, the data in RAM location 3EH by the number 12H; put the quotient in R4 and remainder in R5. Data in 3EH should be AFH.

Assembly Code

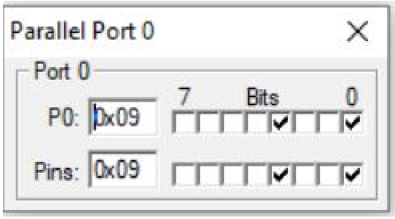


```
ORG 00H

        MOV 3EH,#0AFH
        MOV A,3EH
        MOV R1,#12H
        MOV R4,#00H;Quotient
SUBNXT: CLR C
        SUBB A,R1
        INC R4
        JNC SUBNXT
        DEC R4
        ADD A,R1
        MOV R5,A ;Remainder
        MOV P0,R4
        MOV P1,R5
END
```

C code



```
#include <reg51.h>
int data dividend _at_ 0x3e;
unsigned char data reg4 _at_ 0x04;
unsigned char data reg5 _at_ 0x05;

void delay(int time)
{
    unsigned int i,j;
    for (i=0;i<time;i++)
        for (j=0;j<125;j++);
}

void main(void)
{
    unsigned char divisor = 0x12;
    unsigned char quotient = 0x00, remainder;

    dividend = 0x00af;

    while(1)
    {
        dividend -= divisor;
        if(dividend < 0x0)
            break;
        quotient += 0x1;
    }
    remainder = dividend + divisor;

    reg4 = quotient;
    reg5 = remainder;

    while(1)
    {
        P0 = quotient;
        delay(1000);
        P0 = remainder;
        delay(1000);
    }
}
```

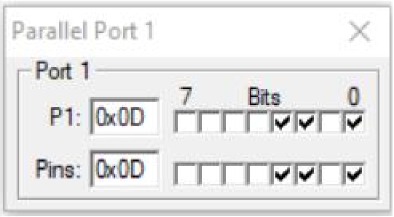


Parallel Port 0

Port 0

P0: 0x09 7 Bits 0 □□□□☑□□☑

Pins: 0x09 □□□□☑□□☑



Parallel Port 1

Port 1

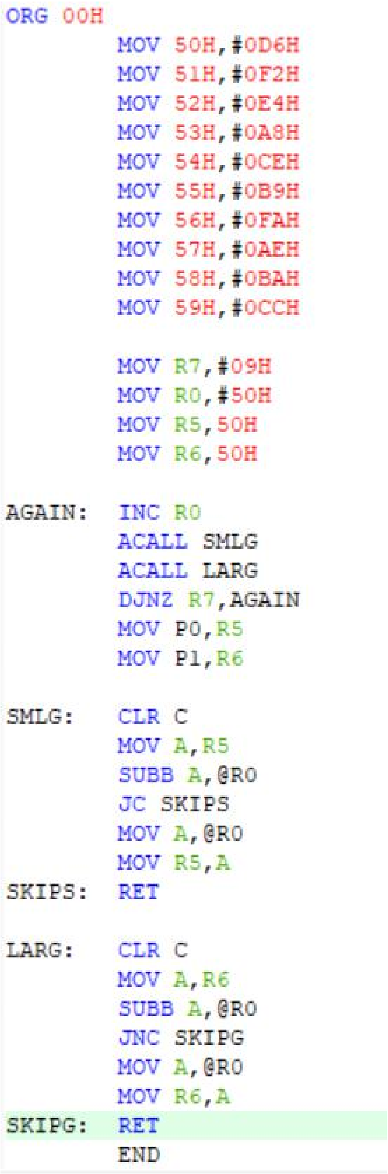
P1: 0x0D 7 Bits 0 □□□□☑☑□☑

Pins: 0x0D □□□□☑☑□☑

When AFH divided by 12H, the Quotient 09H is and the remainder is 0DH.

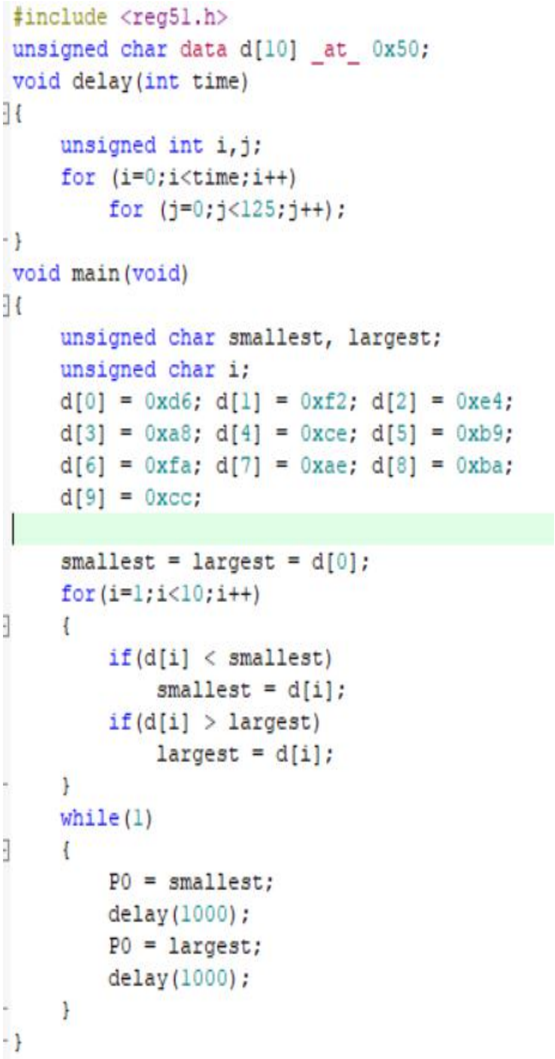
5. Store ten hexadecimal numbers in internal RAM starting from memory location 50H. The list of numbers to be used is: D6H, F2H, E4H, A8H, CEH, B9H, FAH, AEH, BAH, CCH. Implement a subroutine that extracts both the smallest and largest numbers from the stored numbers.

Assembly Code

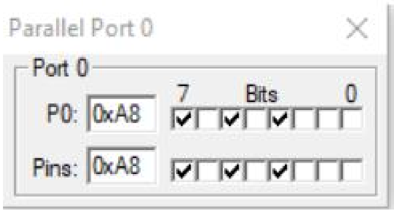


```
ORG 00H
        MOV 50H,#0D6H
        MOV 51H,#0F2H
        MOV 52H,#0E4H
        MOV 53H,#0A8H
        MOV 54H,#0CEH
        MOV 55H,#0B9H
        MOV 56H,#0FAH
        MOV 57H,#0AEH
        MOV 58H,#0BAH
        MOV 59H,#0CCH

        MOV R7,#09H
        MOV R0,#50H
        MOV R5,50H
        MOV R6,50H

AGAIN:  INC R0
        ACALL SMLG
        ACALL LARG
        DJNZ R7,AGAIN
        MOV P0,R5
        MOV P1,R6

SMLG:   CLR C
        MOV A,R5
        SUBB A,@R0
        JC SKIPS
        MOV A,@R0
        MOV R5,A
SKIPS:  RET

LARG:   CLR C
        MOV A,R6
        SUBB A,@R0
        JNC SKIPG
        MOV A,@R0
        MOV R6,A
SKIPG:  RET
        END
```

C code

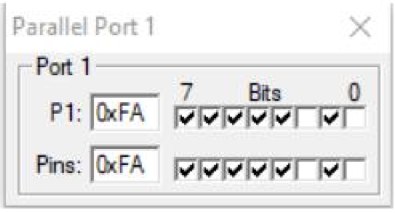


```
#include <reg51.h>
unsigned char data d[10] _at_ 0x50;
void delay(int time)
{
    unsigned int i,j;
    for (i=0;i<time;i++)
        for (j=0;j<125;j++);
}
void main(void)
{
    unsigned char smallest, largest;
    unsigned char i;
    d[0] = 0xd6; d[1] = 0xf2; d[2] = 0xe4;
    d[3] = 0xa8; d[4] = 0xce; d[5] = 0xb9;
    d[6] = 0xfa; d[7] = 0xae; d[8] = 0xba;
    d[9] = 0xcc;

    smallest = largest = d[0];
    for(i=1;i<10;i++)
    {
        if(d[i] < smallest)
            smallest = d[i];
        if(d[i] > largest)
            largest = d[i];
    }
    while(1)
    {
        P0 = smallest;
        delay(1000);
        P0 = largest;
        delay(1000);
    }
}
```



P0: 0xA8 ☑☐☑☐☑☐☐☐

Pins: 0xA8 ☑☐☑☐☑☐☐☐



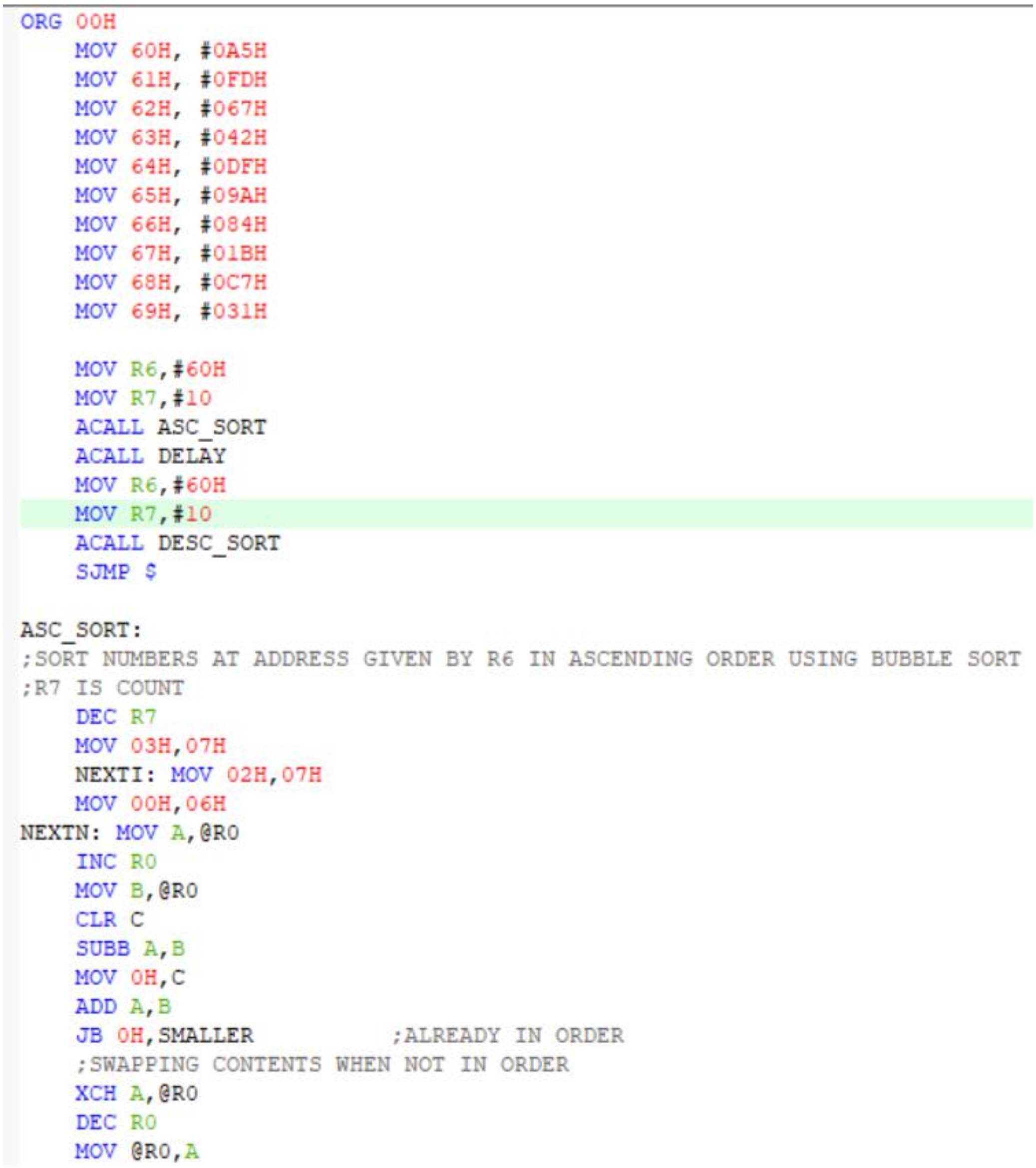
P1: 0xFA ☑☑☑☑☑☐☑☐

Pins: 0xFA ☑☑☑☑☑☐☑☐

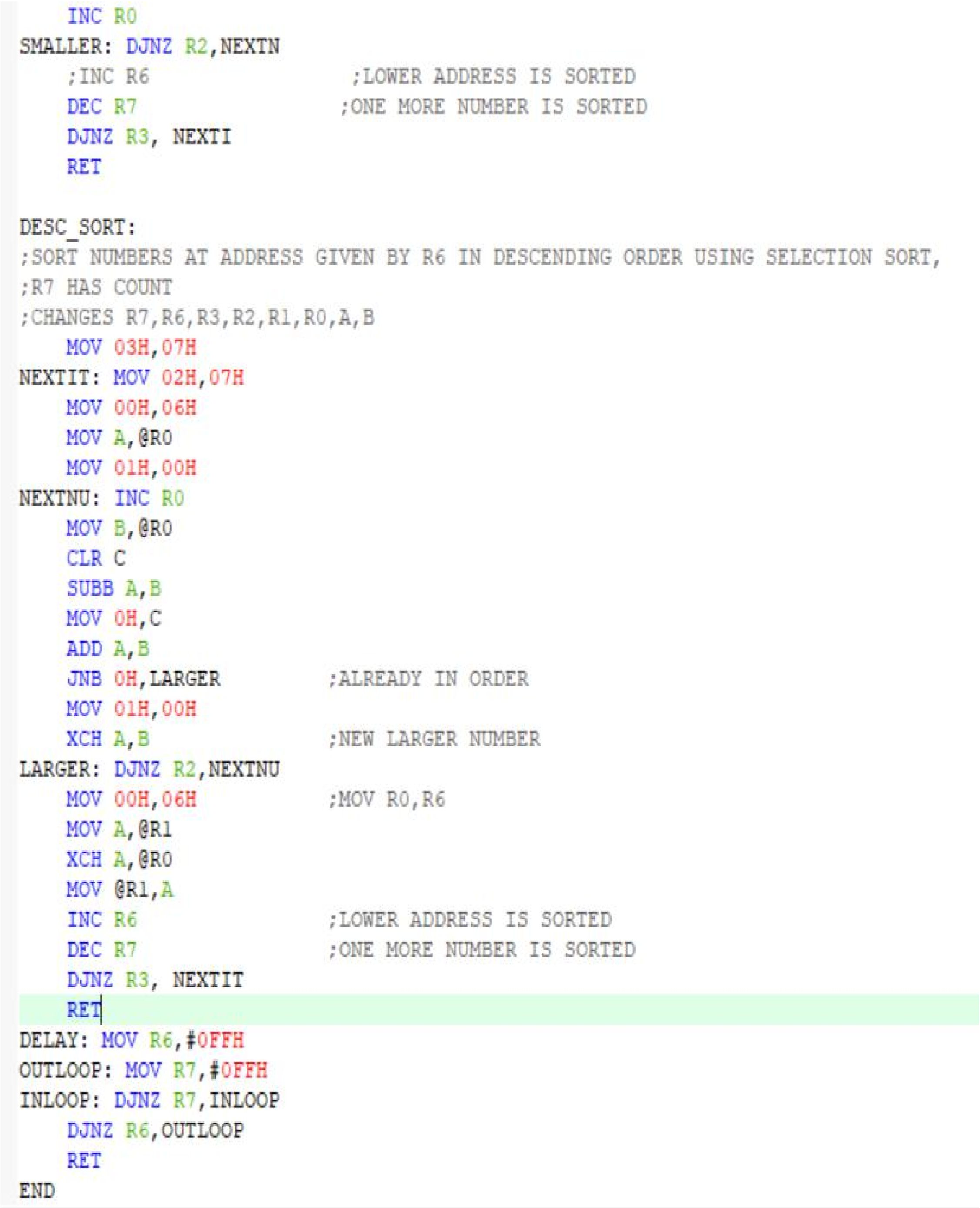
The smallest number is A8H(Port 0) and the largest number is FAH(Port 1).

6.Store ten hexadecimal numbers in internal RAM starting from memory location 60H. The list of numbers to be used is: A5H, FDH, 67H, 42H, DFH, 9AH, 84H, 1BH, C7H, 31H. Implement a subroutine that orders the numbers in ascending order using bubble or any other sort algorithm and implement s subroutine that order the numbers in descending order using selection sort algorithm.

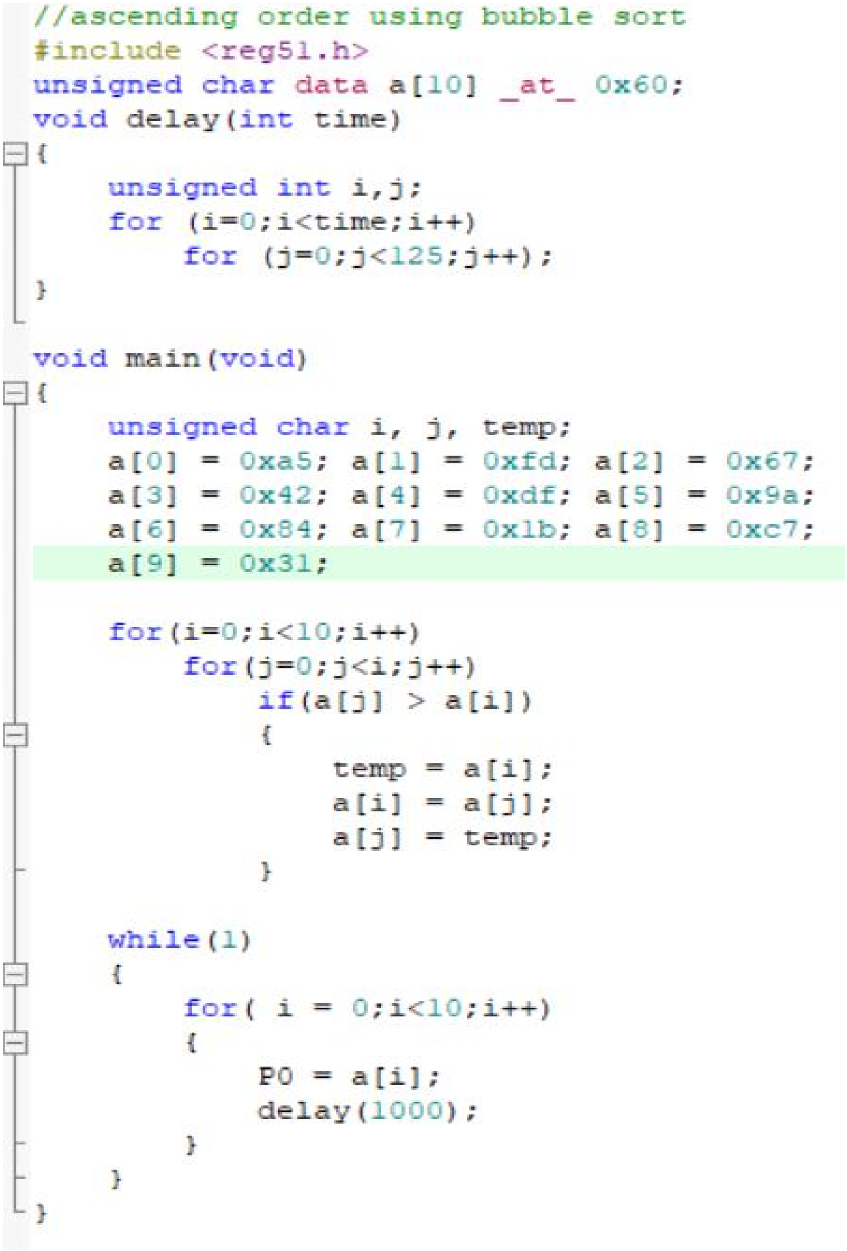
Assembly code



```
ORG 00H
    MOV 60H, #0A5H
    MOV 61H, #0FDH
    MOV 62H, #067H
    MOV 63H, #042H
    MOV 64H, #0DFH
    MOV 65H, #09AH
    MOV 66H, #084H
    MOV 67H, #01BH
    MOV 68H, #0C7H
    MOV 69H, #031H

    MOV R6,#60H
    MOV R7,#10
    ACALL ASC_SORT
    ACALL DELAY
    MOV R6,#60H
    MOV R7,#10
    ACALL DESC_SORT
    SJMP $

ASC_SORT:
;SORT NUMBERS AT ADDRESS GIVEN BY R6 IN ASCENDING ORDER USING BUBBLE SORT
;R7 IS COUNT
    DEC R7
    MOV 03H,07H
    NEXTI: MOV 02H,07H
    MOV 00H,06H
NEXTN: MOV A,@R0
    INC R0
    MOV B,@R0
    CLR C
    SUBB A,B
    MOV 0H,C
    ADD A,B
    JB 0H,SMALLER           ;ALREADY IN ORDER
    ;SWAPPING CONTENTS WHEN NOT IN ORDER
    XCH A,@R0
    DEC R0
    MOV @R0,A
```



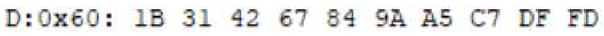
```
    INC R0
SMALLER: DJNZ R2,NEXTN
    ;INC R6                 ;LOWER ADDRESS IS SORTED
    DEC R7                 ;ONE MORE NUMBER IS SORTED
    DJNZ R3, NEXTI
    RET

DESC_SORT:
;SORT NUMBERS AT ADDRESS GIVEN BY R6 IN DESCENDING ORDER USING SELECTION SORT,
;R7 HAS COUNT
;CHANGES R7,R6,R3,R2,R1,R0,A,B
    MOV 03H,07H
NEXTIT: MOV 02H,07H
    MOV 00H,06H
    MOV A,@R0
    MOV 01H,00H
NEXTNU: INC R0
    MOV B,@R0
    CLR C
    SUBB A,B
    MOV 0H,C
    ADD A,B
    JNB 0H,LARGER         ;ALREADY IN ORDER
    MOV 01H,00H
    XCH A,B               ;NEW LARGER NUMBER
LARGER: DJNZ R2,NEXTNU
    MOV 00H,06H           ;MOV R0,R6
    MOV A,@R1
    XCH A,@R0
    MOV @R1,A
    INC R6                ;LOWER ADDRESS IS SORTED
    DEC R7                ;ONE MORE NUMBER IS SORTED
    DJNZ R3, NEXTIT
    RET
DELAY: MOV R6,#0FFH
OUTLOOP: MOV R7,#0FFH
INLOOP: DJNZ R7,INLOOP
    DJNZ R6,OUTLOOP
    RET
END
```



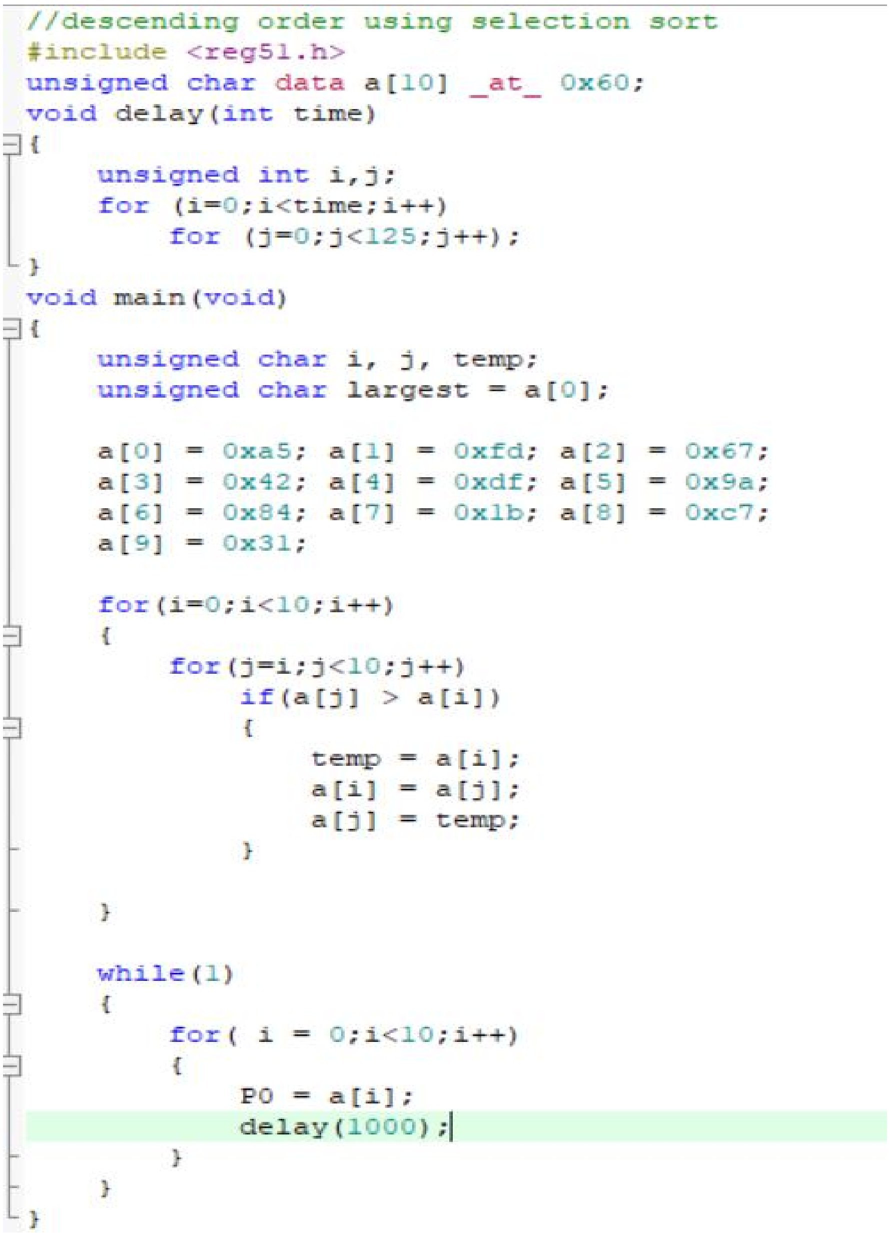
```c
//ascending order using bubble sort
#include <reg51.h>
unsigned char data a[10] _at_ 0x60;
void delay(int time)
{
    unsigned int i,j;
    for (i=0;i<time;i++)
        for (j=0;j<125;j++);
}

void main(void)
{
    unsigned char i, j, temp;
    a[0] = 0xa5; a[1] = 0xfd; a[2] = 0x67;
    a[3] = 0x42; a[4] = 0xdf; a[5] = 0x9a;
    a[6] = 0x84; a[7] = 0x1b; a[8] = 0xc7;
    a[9] = 0x31;

    for(i=0;i<10;i++)
        for(j=0;j<i;j++)
            if(a[j] > a[i])
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }

    while(1)
    {
        for( i = 0;i<10;i++)
        {
            P0 = a[i];
            delay(1000);
        }
    }
}
```

Output

Ascending Order:



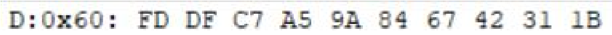
```
D:0x60: 1B 31 42 67 84 9A A5 C7 DF FD
```



```
//descending order using selection sort
#include <reg51.h>
unsigned char data a[10] _at_ 0x60;
void delay(int time)
{
    unsigned int i,j;
    for (i=0;i<time;i++)
        for (j=0;j<125;j++);
}
void main(void)
{
    unsigned char i, j, temp;
    unsigned char largest = a[0];

    a[0] = 0xa5; a[1] = 0xfd; a[2] = 0x67;
    a[3] = 0x42; a[4] = 0xdf; a[5] = 0x9a;
    a[6] = 0x84; a[7] = 0x1b; a[8] = 0xc7;
    a[9] = 0x31;

    for(i=0;i<10;i++)
    {
        for(j=i;j<10;j++)
            if(a[j] > a[i])
            {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }

    }

    while(1)
    {
        for( i = 0;i<10;i++)
        {
            P0 = a[i];
            delay(1000);
        }
    }
}
```

Output

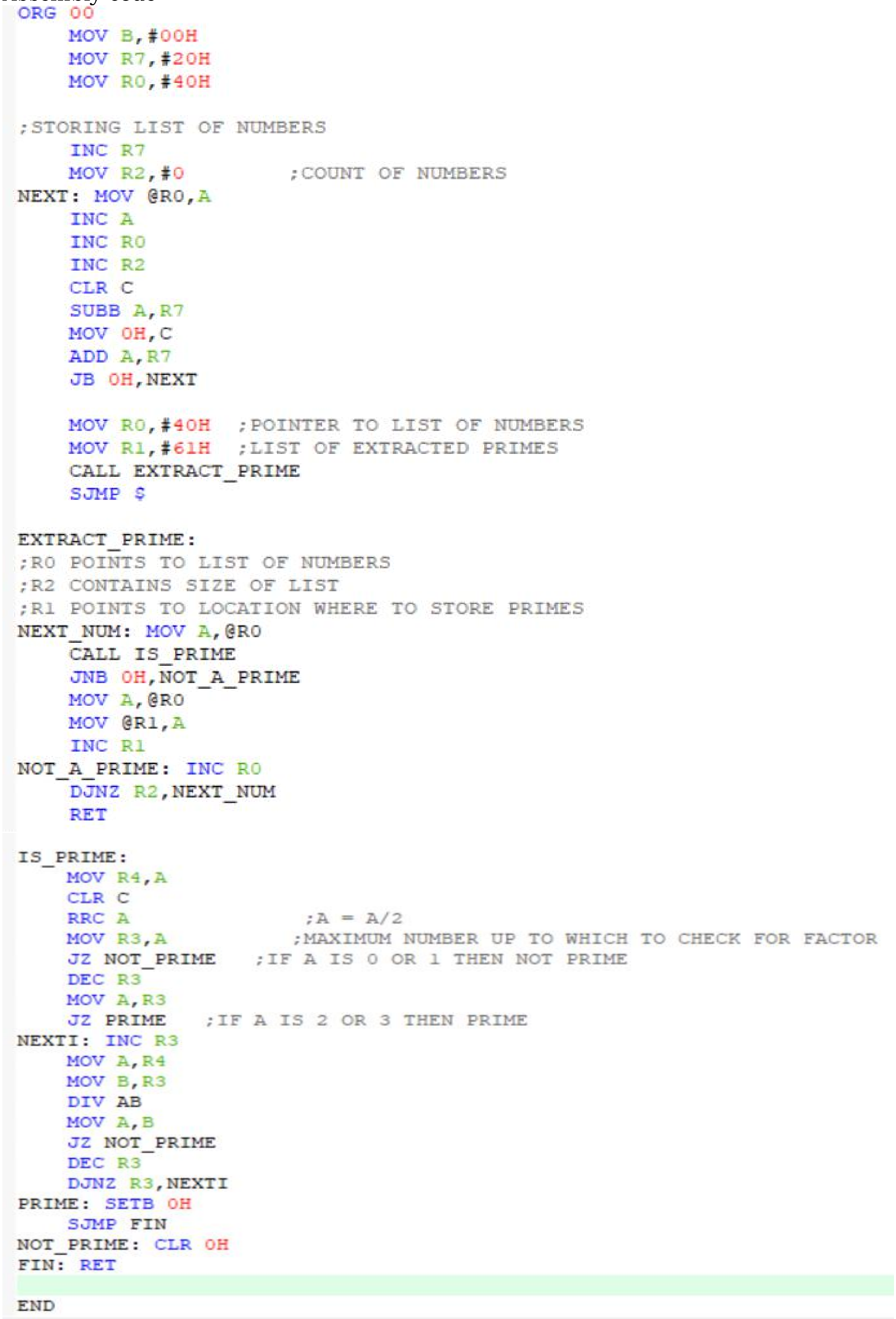
Descending Order:



```
D:0x60: FD DF C7 A5 9A 84 67 42 31 1B
```

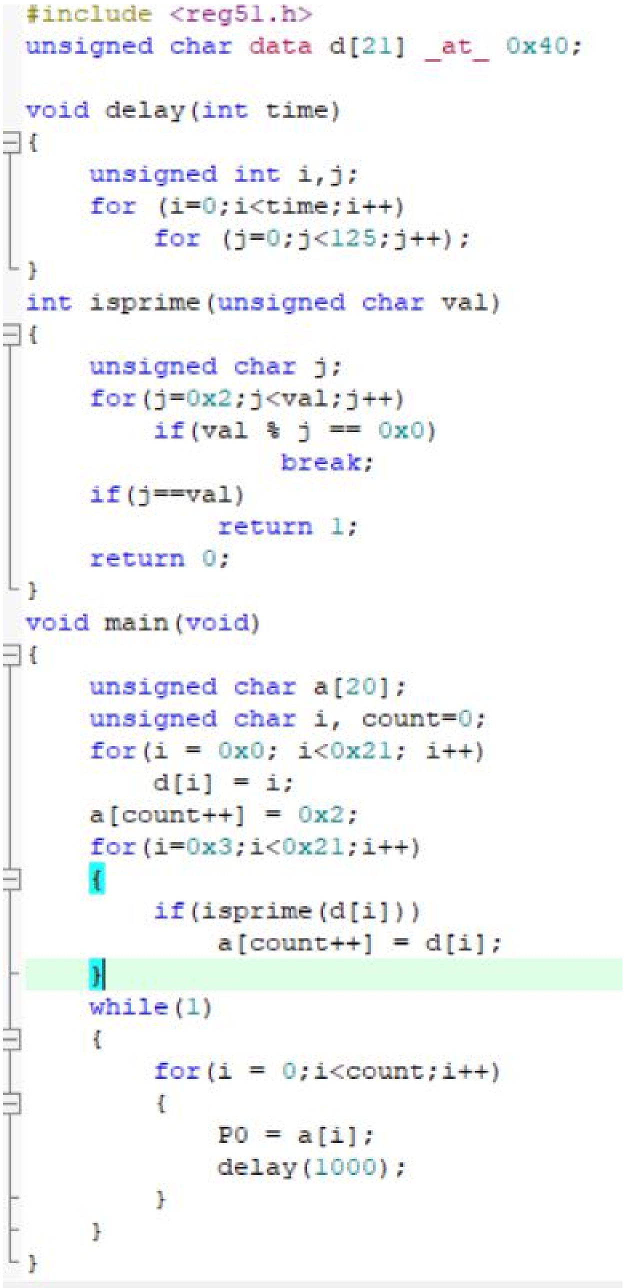
7.Store numbers from 00H to 20H in internal RAM starting from memory location 40H. Implement a subroutine that extracts only the prime numbers.

Assembly code

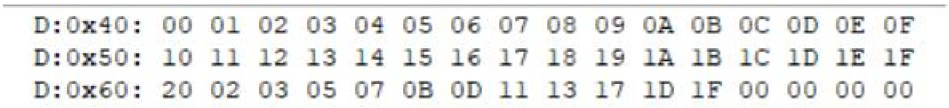


```
ORG 00
    MOV B,#00H
    MOV R7,#20H
    MOV R0,#40H

;STORING LIST OF NUMBERS
    INC R7
    MOV R2,#0          ;COUNT OF NUMBERS
NEXT: MOV @R0,A
    INC A
    INC R0
    INC R2
    CLR C
    SUBB A,R7
    MOV 0H,C
    ADD A,R7
    JB 0H,NEXT

    MOV R0,#40H   ;POINTER TO LIST OF NUMBERS
    MOV R1,#61H   ;LIST OF EXTRACTED PRIMES
    CALL EXTRACT_PRIME
    SJMP $

EXTRACT_PRIME:
;R0 POINTS TO LIST OF NUMBERS
;R2 CONTAINS SIZE OF LIST
;R1 POINTS TO LOCATION WHERE TO STORE PRIMES
NEXT_NUM: MOV A,@R0
    CALL IS_PRIME
    JNB 0H,NOT_A_PRIME
    MOV A,@R0
    MOV @R1,A
    INC R1
NOT_A_PRIME: INC R0
    DJNZ R2,NEXT_NUM
    RET

IS_PRIME:
    MOV R4,A
    CLR C
    RRC A                ;A = A/2
    MOV R3,A            ;MAXIMUM NUMBER UP TO WHICH TO CHECK FOR FACTOR
    JZ NOT_PRIME    ;IF A IS 0 OR 1 THEN NOT PRIME
    DEC R3
    MOV A,R3
    JZ PRIME    ;IF A IS 2 OR 3 THEN PRIME
NEXTI: INC R3
    MOV A,R4
    MOV B,R3
    DIV AB
    MOV A,B
    JZ NOT_PRIME
    DEC R3
    DJNZ R3,NEXTI
PRIME: SETB 0H
    SJMP FIN
NOT_PRIME: CLR 0H
FIN: RET

END
```

C code



```c
#include <reg51.h>
unsigned char data d[21] _at_ 0x40;

void delay(int time)
{
    unsigned int i,j;
    for (i=0;i<time;i++)
        for (j=0;j<125;j++);
}
int isprime(unsigned char val)
{
    unsigned char j;
    for(j=0x2;j<val;j++)
        if(val % j == 0x0)
                break;
    if(j==val)
            return 1;
    return 0;
}
void main(void)
{
    unsigned char a[20];
    unsigned char i, count=0;
    for(i = 0x0; i<0x21; i++)
        d[i] = i;
    a[count++] = 0x2;
    for(i=0x3;i<0x21;i++)
    {
        if(isprime(d[i]))
            a[count++] = d[i];
    }
    while(1)
    {
        for(i = 0;i<count;i++)
        {
            P0 = a[i];
            delay(1000);
        }
    }
}
```

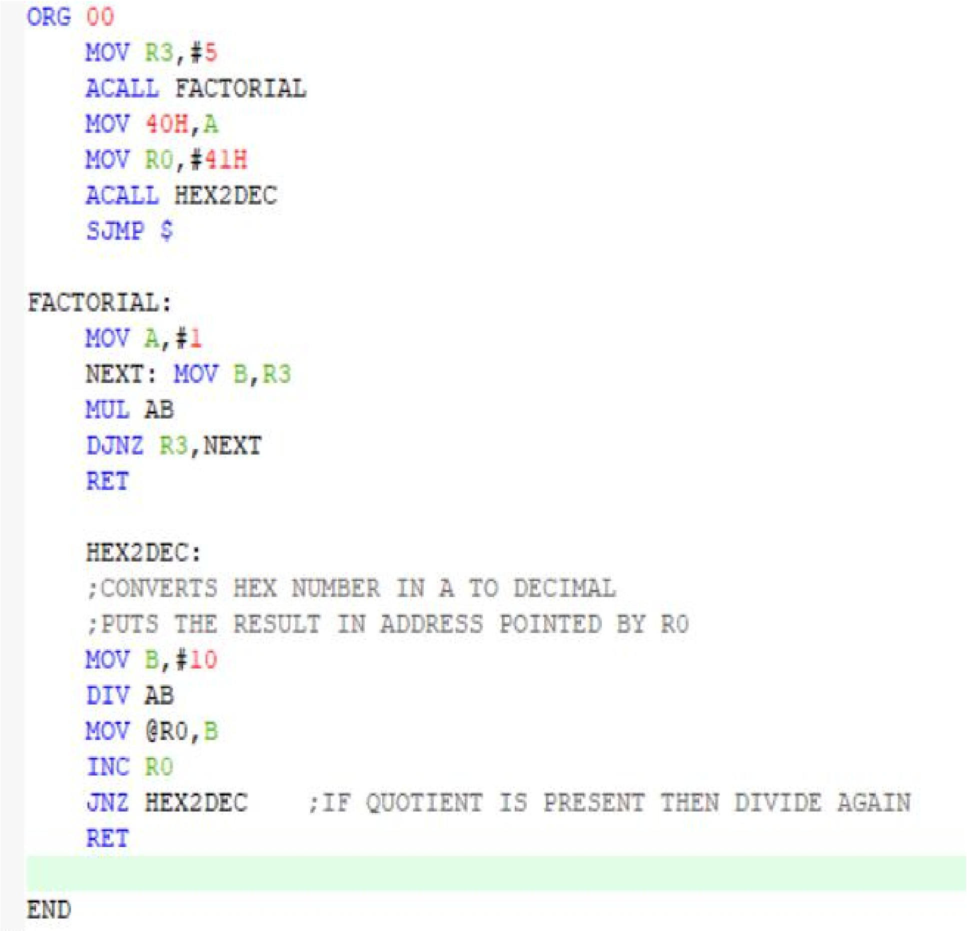


```
D:0x40: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
D:0x50: 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
D:0x60: 20 02 03 05 07 0B 0D 11 13 17 1D 1F 00 00 00 00
```

The Prime numbers are stored in location 60H and onwards.

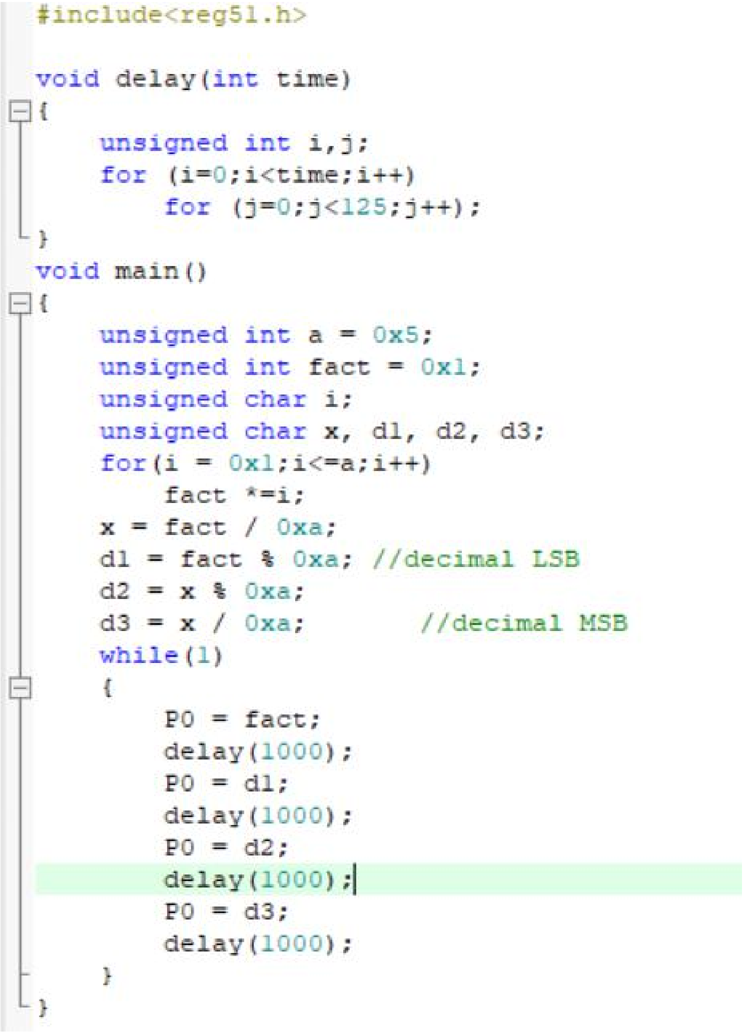
8. Find the factorial of a number stored in R3. The value in R3 could be any number in the range from 00H to 05H. Implement a subroutine that calculates the factorial. The factorial needs to be represented in both hexadecimal and decimal formats.

Assembly Code



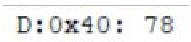
```
ORG 00
    MOV R3,#5
    ACALL FACTORIAL
    MOV 40H,A
    MOV R0,#41H
    ACALL HEX2DEC
    SJMP $

FACTORIAL:
    MOV A,#1
    NEXT: MOV B,R3
    MUL AB
    DJNZ R3,NEXT
    RET

    HEX2DEC:
    ;CONVERTS HEX NUMBER IN A TO DECIMAL
    ;PUTS THE RESULT IN ADDRESS POINTED BY R0
    MOV B,#10
    DIV AB
    MOV @R0,B
    INC R0
    JNZ HEX2DEC    ;IF QUOTIENT IS PRESENT THEN DIVIDE AGAIN
    RET

END
```

C code



```
#include<reg51.h>

void delay(int time)
{
    unsigned int i,j;
    for (i=0;i<time;i++)
        for (j=0;j<125;j++);
}
void main()
{
    unsigned int a = 0x5;
    unsigned int fact = 0x1;
    unsigned char i;
    unsigned char x, d1, d2, d3;
    for(i = 0x1;i<=a;i++)
        fact *=i;
    x = fact / 0xa;
    d1 = fact % 0xa; //decimal LSB
    d2 = x % 0xa;
    d3 = x / 0xa;       //decimal MSB
    while(1)
    {
        P0 = fact;
        delay(1000);
        P0 = d1;
        delay(1000);
        P0 = d2;
        delay(1000);
        P0 = d3;
        delay(1000);
    }
}
```

Output



```
D:0x40: 78
```

The factorial of 05H is 78H.