MIT LAB 10

U18CO021: SAHIL BONDRE

1. Write a program to find the square and cube of a 16-bit number.

Square:

```
model small

.8086
.data
  num dw 0012H
  sqr dw ?

.code
  mov ax, @data
  mov ds, ax
  mov ax, num
  mul ax
  mov sqr, ax
  mov ah, 4ch
  int 21h
end
```

```
-u
076A:0000 B86B07        MOV     AX,076B
076A:0003 8ED8          MOV     DS,AX
076A:0005 A10200        MOV     AX,[0002]
076A:0008 F7E0          MUL     AX
076A:000A A30400        MOV     [0004],AX
076A:000D B44C          MOV     AH,4C
076A:000F CD21          INT     21
076A:0011 0012          ADD     [BP+SI],DL
076A:0013 005034        ADD     [BX+SI+34],DL
076A:0016 E88335        CALL    359C
076A:0019 891E5A04      MOV     [045A],BX
076A:001D 7205          JB      0024
076A:001F 830E7A0402    OR      WORD PTR [047A],+02
```

```
Program terminated normally
-d 076b:0000
076B:0000  21 00 12 00 44 01 E8 83-35 89 1E 5A 04 72 05 83   !...D...5..Z.r..
076B:0010  0E 7A 04 02 BE 36 58 B4-00 8C 0E 50 34 E8 6C 35   .z...6X....P4.l5
076B:0020  89 1E 5C 04 72 05 83 0E-7A 04 04 BE 42 58 B4 00   ..\.r...z...BX..
076B:0030  8C 0E 50 34 E8 55 35 89-1E 5E 04 72 05 83 0E 7A   ..P4.U5..^.r...z
076B:0040  04 08 83 3E 7A 04 00 75-03 E9 AE 00 53 8B 1E 38   ...>z..u....S..8
076B:0050  07 C4 7F 02 83 C7 18 3B-3F 76 09 E8 2B F8 5B 73   .......;?v..+.[s
076B:0060  07 E8 EC F7 87 7F 02 5B-8B D7 A1 56 04 AB A1 74   .......[...V...t
076B:0070  07 AB A1 84 07 AB A1 94-07 AB A1 AC 07 AB B8 01   ................
```

0012H * 0012H = 0144H

```
model small

.8086
.data
  num dw 0003H
  cub dw ?

.code
  mov ax, @data
  mov ds, ax
  mov ax, num
  mov bx, ax
  mul ax
  mul bx
  mov cub, ax
  mov ah, 4ch
  int 21h
end
```

```
C:\TASM>DEBUG.EXE Q1-B.EXE
-u
076A:0000 B86B07        MOV     AX,076B
076A:0003 8ED8          MOV     DS,AX
076A:0005 A10600        MOV     AX,[0006]
076A:0008 8BD8          MOV     BX,AX
076A:000A F7E0          MUL     AX
076A:000C F7E3          MUL     BX
076A:000E A30800        MOV     [0008],AX
076A:0011 B44C          MOV     AH,4C
076A:0013 CD21          INT     21
076A:0015 0003          ADD     [BP+DI],AL
076A:0017 0014          ADD     [SI],DL
076A:0019 0000          ADD     [BX+SI],AL
076A:001B 00EB          ADD     BL,CH
076A:001D 6D            DB      6D
076A:001E 8E06D655      MOV     ES,[55D6]
-
```

```
Program terminated normally
-d 076b:0000
076B:0000  00 B4 4C CD 21 00 03 00-1B 00 00 00 EB 6D 8E 06   ..L.!........m..
076B:0010  D6 55 26 A3 14 00 26 A3-1E 00 8E C0 E8 4A BC 8E   .U&...&......J..
076B:0020  D8 36 8E 06 D2 55 A1 06-00 26 A3 06 00 23 C0 74   .6...U...&...#.t
076B:0030  07 1E 16 1F E8 1C 12 1F-A1 0A 00 26 01 06 0A 00   ...........&....
076B:0040  A1 0C 00 26 11 06 0C 00-A0 24 00 26 10 06 24 00   ...&.....$.&..$.
076B:0050  A0 04 00 24 04 26 08 06-04 00 A0 01 00 26 A2 01   ...$.&.......&..
076B:0060  00 A1 26 00 26 A3 26 00-A1 18 00 26 A3 18 00 A1   ..&.&.&....&....
076B:0070  1A 00 26 A3 1A 00 16 1F-E8 0D 4A A1 D2 55 8E 06   ..&.......J..U..
-
```

003H * 003H * 003H = 01BH

2. Write a program to find LCM of two 8-bit numbers.

```
model small

.8086
.data
  lcm db ?
  lcm1 db ?

.code
  mov ax, @data
  mov ds, ax
  mov ax, 000CH
  mov cx, ax
  mov dx, ax
  mov bl, 10H
  up: div bl
  cmp ah, 00H
  je down
  mov ax, cx
  add ax, dx
  mov cx, ax
  jmp up
  down: mov lcm, ch
  mov lcm1, cl
  mov ah, 4CH
  int 21H
end
```

```
C:\TASM>DEBUG.EXE Q2.EXE
-u
076A:0000 B86C07        MOV     AX,076C
076A:0003 8ED8          MOV     DS,AX
076A:0005 B80C00        MOV     AX,000C
076A:0008 8BC8          MOV     CX,AX
076A:000A 8BD0          MOV     DX,AX
076A:000C B310          MOV     BL,10
076A:000E F6F3          DIV     BL
076A:0010 80FC00        CMP     AH,00
076A:0013 7408          JZ      001D
076A:0015 8BC1          MOV     AX,CX
076A:0017 03C2          ADD     AX,DX
076A:0019 8BC8          MOV     CX,AX
076A:001B EBF1          JMP     000E
076A:001D 882E0A00      MOV     [000A],CH
-_
```

```
Program terminated normally
-d 076c:0000
076C:0000  00 88 0E 0B 00 B4 4C CD-21 8C 00 30 34 E8 6C 35   ......L.!..04.l5
076C:0010  89 1E 5C 04 72 05 83 0E-7A 04 04 BE 42 58 B4 00   ..\.r...z...BX..
076C:0020  8C 0E 50 34 E8 55 35 89-1E 5E 04 72 05 83 0E 7A   ..P4.U5..^.r...z
076C:0030  04 08 83 3E 7A 04 00 75-03 E9 AE 00 53 8B 1E 38   ...>z..u....S..8
076C:0040  07 C4 7F 02 83 C7 18 3B-3F 76 09 E8 2B F8 5B 73   .......;?v..+.[s
076C:0050  07 E8 EC F7 87 7F 02 5B-8B D7 A1 56 04 AB A1 74   .......[...V...t
076C:0060  07 AB A1 84 07 AB A1 94-07 AB A1 AC 07 AB B8 01   ................
076C:0070  00 AB B0 00 AB AB AA 8B-1E 54 04 A1 D8 03 40 A3   .........T....@.
-_
```

LCM of 12(000CH) and 16(0010H) = 48(0030H)

3. Write a program to find GCD of two 8-bit numbers.

```
model small

.8086
.data
  gcd db ?

.code
  mov ax, @data
  mov ds, ax
  mov al, 0CH
  mov bl, 10H
  up: mov cl,al
  cmp cl, bl
  je down
  jnc go
  sub bl, al
```

```
  jmp up
  go: sub al, bl
  jmp up
  down: mov gcd, al
  mov ah, 4ch
  int 21h
end
```

```
C:\TASM>DEBUG.EXE Q3.EXE
-u
076A:0000 B86C07        MOV     AX,076C
076A:0003 8ED8          MOV     DS,AX
076A:0005 B00C          MOV     AL,0C
076A:0007 B310          MOV     BL,10
076A:0009 8AC8          MOV     CL,AL
076A:000B 3ACB          CMP     CL,BL
076A:000D 740A          JZ      0019
076A:000F 7304          JNB     0015
076A:0011 2AD8          SUB     BL,AL
076A:0013 EBF4          JMP     0009
076A:0015 2AC3          SUB     AL,BL
076A:0017 EBF0          JMP     0009
076A:0019 A20000        MOV     [0000],AL
076A:001C B44C          MOV     AH,4C
076A:001E CD21          INT     21
-_
```

```
Program terminated normally
-d 076c:0000
076C:0000  04 7A 04 02 BE 36 58 B4-00 8C 0E 50 34 E8 6C 35   .z...6X....P4.l5
076C:0010  89 1E 5C 04 72 05 83 0E-7A 04 04 BE 42 58 B4 00   ..\.r...z...BX..
076C:0020  8C 0E 50 34 E8 55 35 89-1E 5E 04 72 05 83 0E 7A   ..P4.U5..^.r...z
076C:0030  04 08 83 3E 7A 04 00 75-03 E9 AE 00 53 8B 1E 38   ...>z..u....S..8
076C:0040  07 C4 7F 02 83 C7 18 3B-3F 76 09 E8 2B F8 5B 73   .......;?v..+.[s
076C:0050  07 E8 EC F7 87 7F 02 5B-8B D7 A1 56 04 AB A1 74   .......[...V...t
076C:0060  07 AB A1 84 07 AB A1 94-07 AB A1 AC 07 AB B8 01   ................
076C:0070  00 AB B0 00 AB AB AA 8B-1E 54 04 A1 D8 03 40 A3   .........T....@.
-_
```

GCD of 12(000CH) and 16(0010H) = 4(0004H)

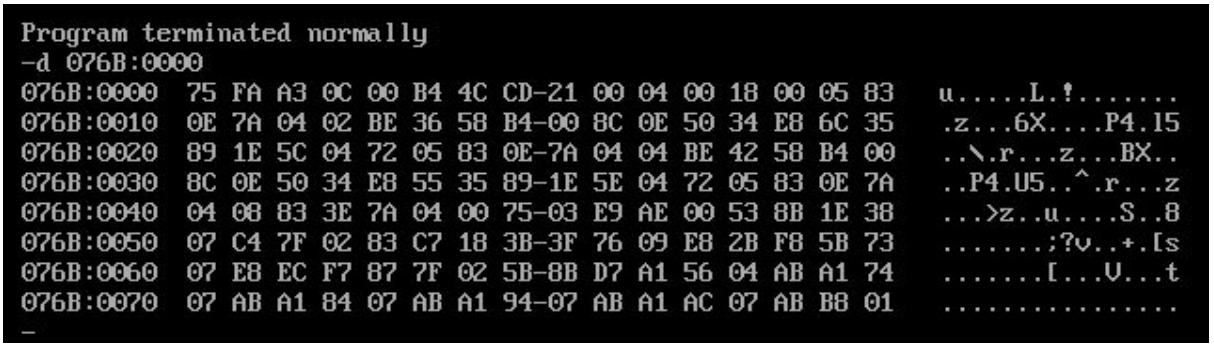
4. Write a program to find factorial of a given number.

```
model small

.8086
.data
  num dw 4
  fact dw ?

.code
mov ax, @data
mov ds, ax
mov cx, num
mov ax, 0001H
up: nop
mul cx
dec cx
jnz up
mov fact, ax
mov ah, 4ch
int 21h
end
```

```
C:\TASM>DEBUG.EXE Q4.EXE
-u
076A:0000 B86B07        MOV     AX,076B
076A:0003 8ED8          MOV     DS,AX
076A:0005 8B0E0A00      MOV     CX,[000A]
076A:0009 B80100        MOV     AX,0001
076A:000C 90            NOP
076A:000D F7E1          MUL     CX
076A:000F 49            DEC     CX
076A:0010 75FA          JNZ     000C
076A:0012 A30C00        MOV     [000C],AX
076A:0015 B44C          MOV     AH,4C
076A:0017 CD21          INT     21
076A:0019 0004          ADD     [SI],AL
076A:001B 0004          ADD     [SI],AL
076A:001D 7205          JB      0024
076A:001F 830E7A0402    OR      WORD PTR [047A],+02
-
```



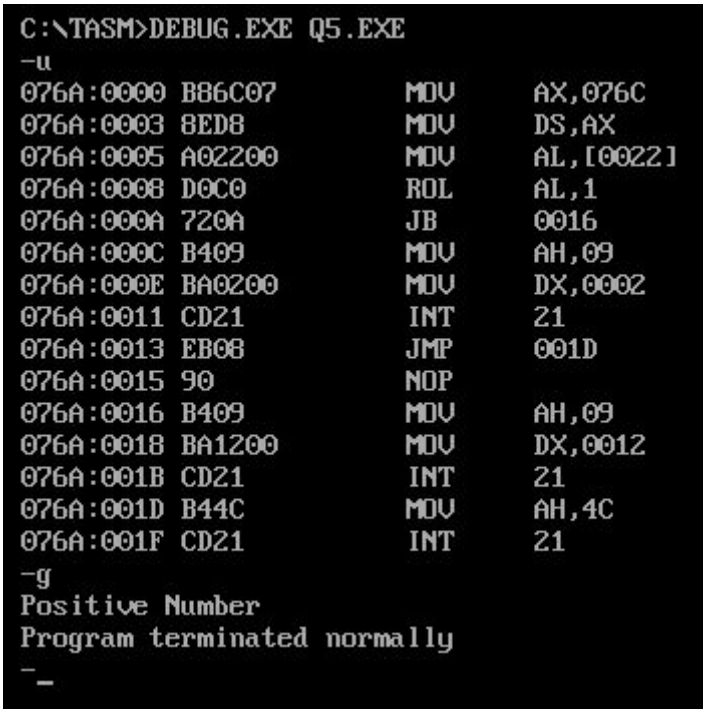
4! = 24 = 0018H

5. Write a program to check whether given data is positive or negative.

```
model small

.8086
.data
  string db "Positive Number$"
  string1 db "Negative Number$"
  num db 0012H

.code
  mov ax, @data
  mov ds, ax
  mov al, num
  rol al, 01
  jc down
  mov ah, 09h
  mov dx, offset string
  int 21h
  jmp fin
  down: mov ah,09h
  mov dx, offset string1
  int 21h
  fin: mov ah, 4ch
  int 21h
end
```



6. Write a program to check whether a given number is odd or even.

```
model small

.8086
.data
  cr equ 0dh
  lf equ 0ah
  m db "Enter a number:$"
  string db cr, lf, "Even Number$"
  string1 db cr, lf, "Odd Number$"
  num db ?

print macro string
  mov ah, 09h
  mov dx, offset string
  int 21h
endm

read macro no
  mov ah, 01h
  int 21h
  sub al, 30h ; ascii
  mov bl, 0ah ; x10
  mul bl
  mov no, al
  mov ah, 01h
```

```
  int 21h
  sub al, 30h ; ascii
  add no, al
endm

.code
  mov ax,@data
  mov ds,ax
  print m
  read num
  mov ah,00h
  mov al,num
  mov bl,02h
  div bl
  cmp ah,00h
  je down
  mov ah,09h
  mov dx, offset string1
  int 21h
  jmp eo
  down: mov ah,09h
  mov dx, offset string
  int 21h
  eo: mov ah,4ch
  int 21h
end
```

```
C:\SOURCE\TASM>DEBUG.EXE Q6.EXE
-u
076A:0000 B86E07        MOV     AX,076E
076A:0003 8ED8          MOV     DS,AX
076A:0005 B409          MOV     AH,09
076A:0007 BA0600        MOV     DX,0006
076A:000A CD21          INT     21
076A:000C B401          MOV     AH,01
076A:000E CD21          INT     21
076A:0010 2C30          SUB     AL,30
076A:0012 B30A          MOV     BL,0A
076A:0014 F6E3          MUL     BL
076A:0016 A23100        MOV     [0031],AL
076A:0019 B401          MOV     AH,01
076A:001B CD21          INT     21
076A:001D 2C30          SUB     AL,30
076A:001F 00063100      ADD     [0031],AL
-g
Enter a number:45
Odd Number
Program terminated normally
-_
```

7. Write a program to count logical 1's and 0's in a given data.

```
model small

.8086
.data
  zero db ?
  one db ?

.code
  mov ax,@data
  mov ds,ax
  mov bx, 0000h
  mov al, 0FEh
  mov cl, 08h
  up: rol al,01
  jnc down
  inc bh
  dec cl
  jnz up
  jmp fin
  down: inc bl
  dec cl
  jnz up
  fin: nop
  mov zero, bh
  mov one, bl
  mov ah, 4ch
  int 21h
end
```

```
C:\SOURCE\TASM>DEBUG.EXE Q7.EXE
-u
076A:0000 B86C07        MOV     AX,076C
076A:0003 8ED8          MOV     DS,AX
076A:0005 BB0000        MOV     BX,0000
076A:0008 B0FE          MOV     AL,FE
076A:000A B108          MOV     CL,08
076A:000C D0C0          ROL     AL,1
076A:000E 7309          JNB     0019
076A:0010 FEC7          INC     BH
076A:0012 FEC9          DEC     CL
076A:0014 75F6          JNZ     000C
076A:0016 EB07          JMP     001F
076A:0018 90            NOP
076A:0019 FEC3          INC     BL
076A:001B FEC9          DEC     CL
076A:001D 75ED          JNZ     000C
076A:001F 90            NOP
-
```

```
-u
076A:0020 883E0C00      MOV     [000C],BH
076A:0024 881E0D00      MOV     [000D],BL
076A:0028 B44C          MOV     AH,4C
076A:002A CD21          INT     21
076A:002C 07            POP     ES
076A:002D 016C35        ADD     [SI+35],BP
076A:0030 891E5C04      MOV     [045C],BX
076A:0034 7205          JB      003B
076A:0036 830E7A0404    OR      WORD PTR [047A],+04
076A:003B BE4258        MOV     SI,5842
076A:003E B400          MOV     AH,00
-
```

```
Program terminated normally
-d 076c:0000
076C:0000  88 3E 0C 00 88 1E 0D 00-B4 4C CD 21 07 01 6C 35   .>.......L.!..l5
076C:0010  89 1E 5C 04 72 05 83 0E-7A 04 04 BE 42 58 B4 00   ..\.r...z...BX..
076C:0020  8C 0E 50 34 E8 55 35 89-1E 5E 04 72 05 83 0E 7A   ..P4.U5..^.r...z
076C:0030  04 08 83 3E 7A 04 00 75-03 E9 AE 00 53 8B 1E 38   ...>z..u....S..8
076C:0040  07 C4 7F 02 83 C7 18 3B-3F 76 09 E8 2B F8 5B 73   .......;?v..+.[s
076C:0050  07 E8 EC F7 87 7F 02 5B-8B D7 A1 56 04 AB A1 74   .......[...V...t
076C:0060  07 AB A1 84 07 AB A1 94-07 AB A1 AC 07 AB B8 01   ................
076C:0070  00 AB B0 00 AB AB AA 8B-1E 54 04 A1 D8 03 40 A3   .........T....@.
-
```

7 zeros and 1 ones

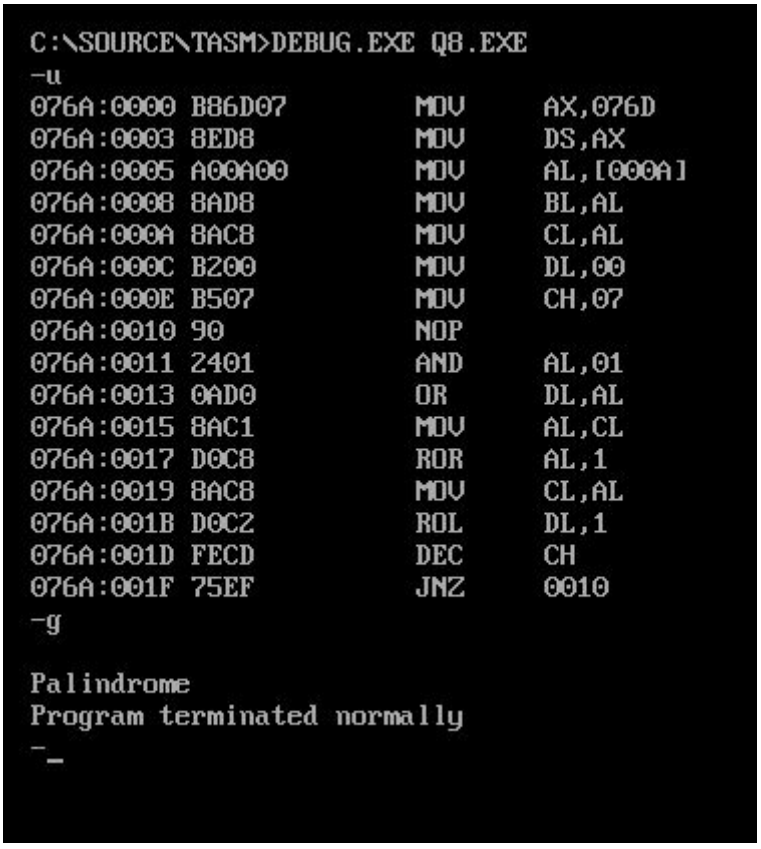
8. Write a program to check the given 8-bit data is bit wise palindrome or not.

```
model small

.8086
.data
  cr equ 0dh
  lf equ 0ah
  num db 3CH
  m1 db cr,lf, "Palindrome$"
  m2 db cr,lf,"Not a palindrome$"

print macro msg
  mov ah,09h
  mov dx,offset msg
  int 21h
endm

.code
  mov ax, @data
  mov ds, ax
  mov al, num
  mov bl, al
  mov cl, al
  mov dl, 00h
  mov ch, 07h
  up: nop
  and al, 01h
  or dl, al
  mov al, cl
  ror al, 01
  mov cl, al
  rol dl, 01
  dec ch
  jnz up
  cmp dl, bl
  je down
  print m2
  jmp eo
  down: print m1
  eo: mov ah,4ch
  int 21h
end
```



9. Write a program to check the given 8-bit data is nibble wise palindrome or not.

```
model small

.8086
.data
  msg db "Nibble palindrome$"
  msg1 db "Not nibble palindrome$"

print macro msg
  mov ah,09h
  mov dx,offset msg
  int 21h
endm

.code
  mov ax, @data
  mov ds, ax
  mov al, 44h
  mov bl, al
  mov cx, 0004h
  rol al, cl
  cmp al, bl
  je down
```

```
  print msg1
  jmp fin
  down: print msg
  fin: mov ah, 4ch
  int 21h
end
```

```
C:\SOURCE\TASM>DEBUG.EXE Q9.EXE
-u
076A:0000 B86C07        MOV     AX,076C
076A:0003 8ED8          MOV     DS,AX
076A:0005 B018          MOV     AL,18
076A:0007 8AD8          MOV     BL,AL
076A:0009 B90400        MOV     CX,0004
076A:000C D2C0          ROL     AL,CL
076A:000E 3AC3          CMP     AL,BL
076A:0010 740A          JZ      001C
076A:0012 B409          MOV     AH,09
076A:0014 BA1A00        MOV     DX,001A
076A:0017 CD21          INT     21
076A:0019 EB08          JMP     0023
076A:001B 90            NOP
076A:001C B409          MOV     AH,09
076A:001E BA0800        MOV     DX,0008
-g
Not nibble palindrome
Program terminated normally
-_
```