MPMC LAB RECORD

Joel Scaria | CSE B | 106118042

8th May 2021

Experiment 1:

Aim:

Assembly program for addition and subtraction of 16 bit numbers

ASM code:

Addition :

```
data segment
    a dw 0202h
    b dw 0901h
    c dw ?
data ends

code segment
assume cs:code, ds:data
start:
    mov ax, data
    mov ds, ax
    mov ax, a
    mov bx, b
    add ax,bx
    mov c,ax
    lea si,c
    int 3
code ends

end start
```

Subtraction :

```
data segment
    a dw 000ah
    b dw 0001h
    c dw ?
data ends

code segment
assume cs:code, ds:data
start:
    mov ax, data
    mov ds, ax
    mov ax, a
    mov bx, b
    sub ax, bx
    mov c, ax
    int 3
```

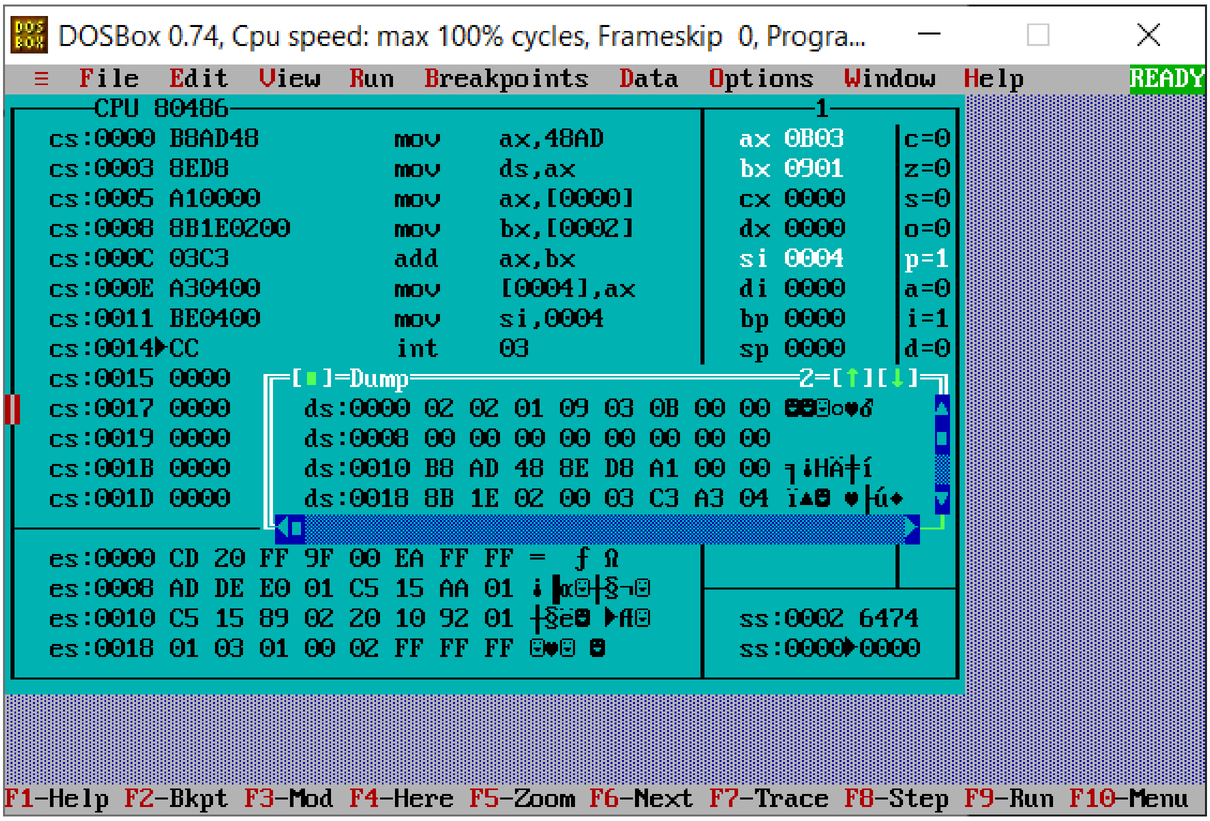
```
code ends
end start
```

Explanation:

- 16 bit is nothing but 2 bytes
- ax and bx registers are used for storing the numbers
- result is stored in c
- program ends with interrupt 3

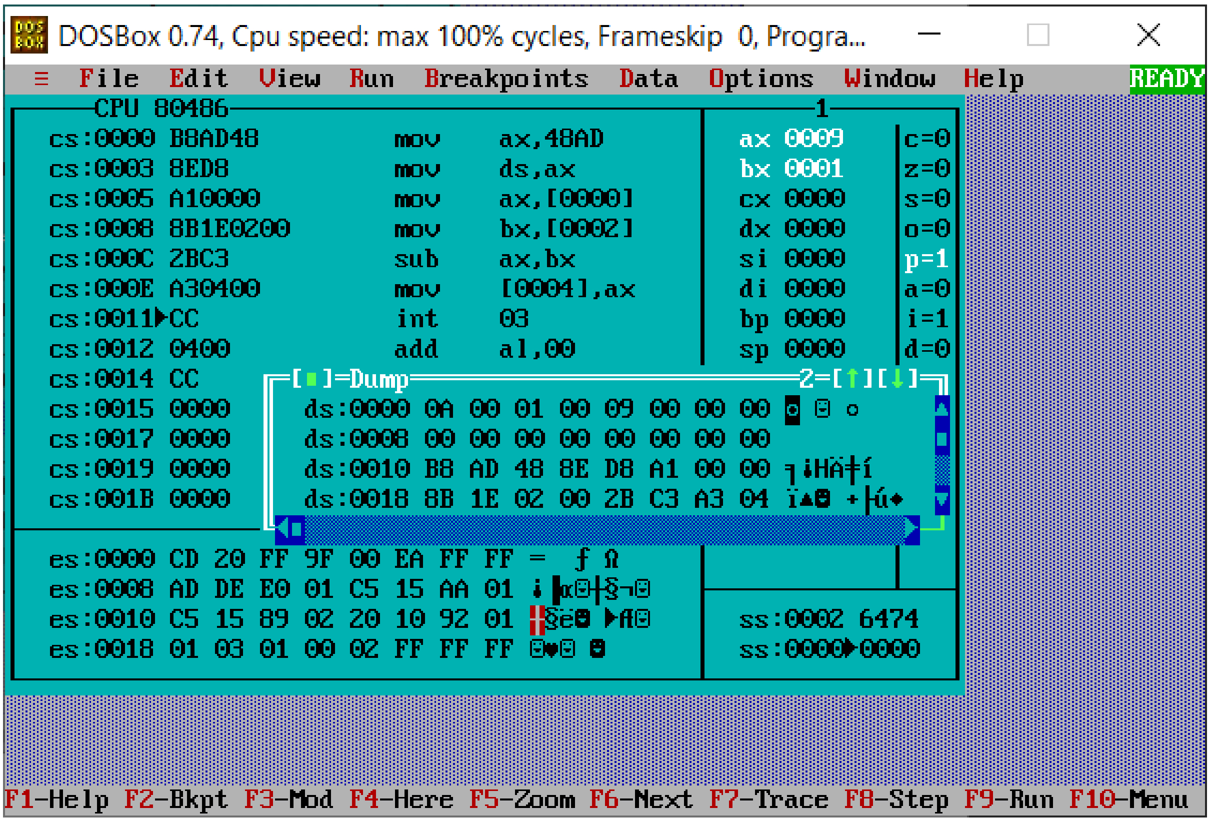
Output:

(a) addition


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
≡ File Edit View Run Breakpoints Data Options Window Help READY
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```
CPU 80486                                            1
cs:0000 B8AD48         mov    ax,48AD      ax 0B03   c=0
cs:0003 8ED8           mov    ds,ax        bx 0901   z=0
cs:0005 A10000         mov    ax,[0000]    cx 0000   s=0
cs:0008 8B1E0200       mov    bx,[0002]    dx 0000   o=0
cs:000C 03C3           add    ax,bx        si 0004   p=1
cs:000E A30400         mov    [0004],ax    di 0000   a=0
cs:0011 BE0400         mov    si,0004      bp 0000   i=1
cs:0014►CC             int    03           sp 0000   d=0
cs:0015 0000    [■]=Dump===========================2=[↑][↓]
cs:0017 0000     ds:0000 02 02 01 09 03 0B 00 00 ☻☻☺○♥♂
cs:0019 0000     ds:0008 00 00 00 00 00 00 00 00
cs:001B 0000     ds:0010 B8 AD 48 8E D8 A1 00 00 ╕¡HÄ╪í
cs:001D 0000     ds:0018 8B 1E 02 00 03 C3 A3 04 ï▲☻ ♥├ú♦

es:0000 CD 20 FF 9F 00 EA FF FF ═  ƒ Ω
es:0008 AD DE E0 01 C5 15 AA 01 ¡▐α☺┼§¬☺
es:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺         ss:0002 6474
es:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻           ss:0000►0000
```

(b) Subtraction


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
≡ File Edit View Run Breakpoints Data Options Window Help READY
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```
CPU 80486                                                1
  cs:0000 B8AD48          mov     ax,48AD          ax 0009   c=0
  cs:0003 8ED8            mov     ds,ax            bx 0001   z=0
  cs:0005 A10000          mov     ax,[0000]        cx 0000   s=0
  cs:0008 8B1E0200        mov     bx,[0002]        dx 0000   o=0
  cs:000C 2BC3            sub     ax,bx            si 0000   p=1
  cs:000E A30400          mov     [0004],ax        di 0000   a=0
  cs:0011▶CC              int     03               bp 0000   i=1
  cs:0012 0400            add     al,00            sp 0000   d=0
  cs:0014 CC         [■]=Dump=========================2=[↑][↓]
  cs:0015 0000          ds:0000 0A 00 01 00 09 00 00 00 ◙ ☺ ○
  cs:0017 0000          ds:0008 00 00 00 00 00 00 00 00
  cs:0019 0000          ds:0010 B8 AD 48 8E D8 A1 00 00 ╕¡HÄ╪í
  cs:001B 0000          ds:0018 8B 1E 02 00 2B C3 A3 04 ï▲☻ +├ú♦

  es:0000 CD 20 FF 9F 00 EA FF FF ═  ƒ Ω
  es:0008 AD DE E0 01 C5 15 AA 01 ¡▐α☺┼§¬☺
  es:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺          ss:0002 6474
  es:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻             ss:0000▶0000
```

Experiment 2:

Aim:

Assembly program for multiplication and division of 16 bit numbers

ASM code:

Multiplication

```
data segment
    a dw 0fffh
    b dw 0ffffh
    c dd ?
data ends

code segment
assume cs:code, ds:data
start:
    mov ax, data
    mov ds, ax
    mov ax, a
    mov bx, b
    ; Multiplying a and b
    ; mul bx is equivalent to ax = ax * bx
```

```
        ; the extra 16bits are saved in dx registed
        ; the whole 32bit number (=a*b) is dx:ax
        mul bx
        ; Copy the values of the ans dx:ax to c
        lea si,c
        mov [si],ax
        mov [si+2],dx
        int 3h
    code ends
    end start
```

Division

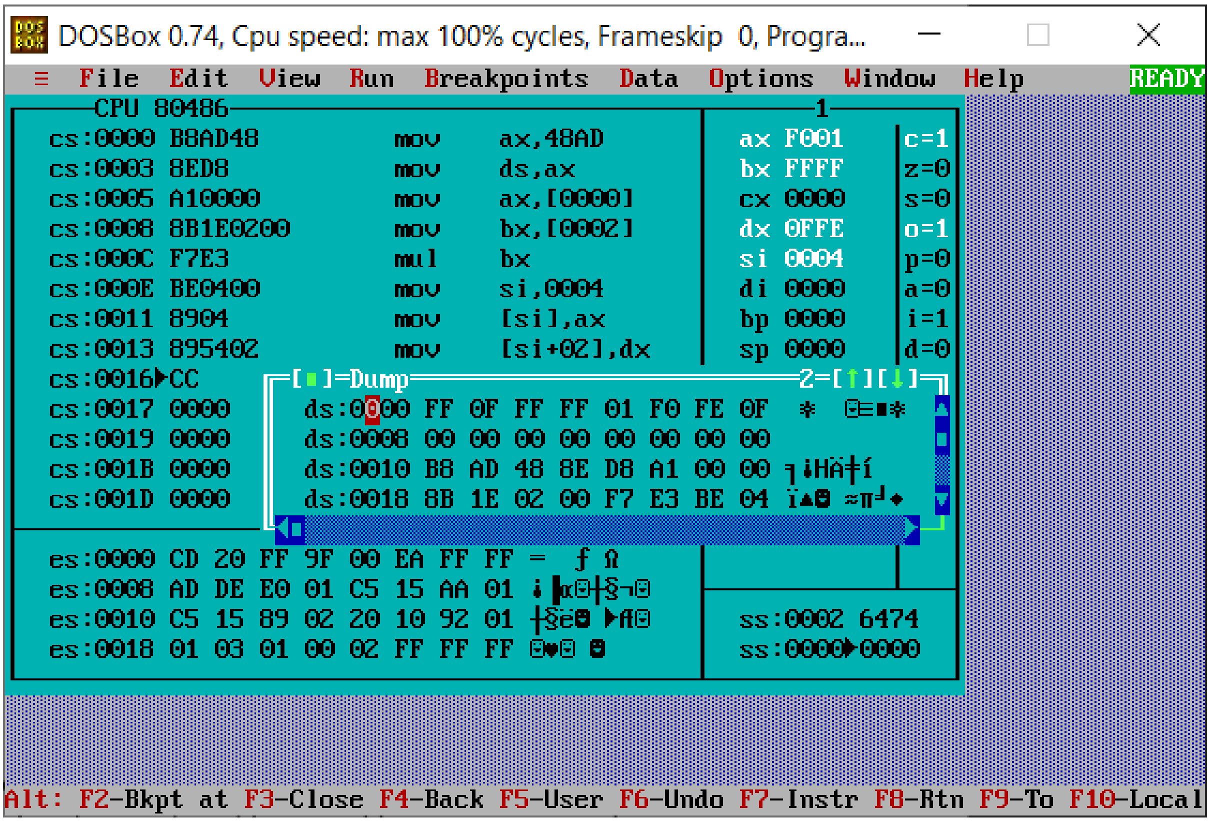
```
    data segment
        a dw 0901h
        b dw 0202h
        c dw ?
    data ends

    code segment
    assume cs:code, ds:data
    start:
        mov ax, data
        mov ds, ax
        mov ax, a
        mov bx, b
        ; Divding a and b
        ; div bx is equivalent to ax = ax / bx
        div bx
        mov c, ax
        int 3h
    code ends
    end start
```

Explanation:

- Process is similar to addition and subtraction
- For multiplication, product is 32 bit number (16 x 2) and is stored in 2 separate registers
- The final answer is dx:ax

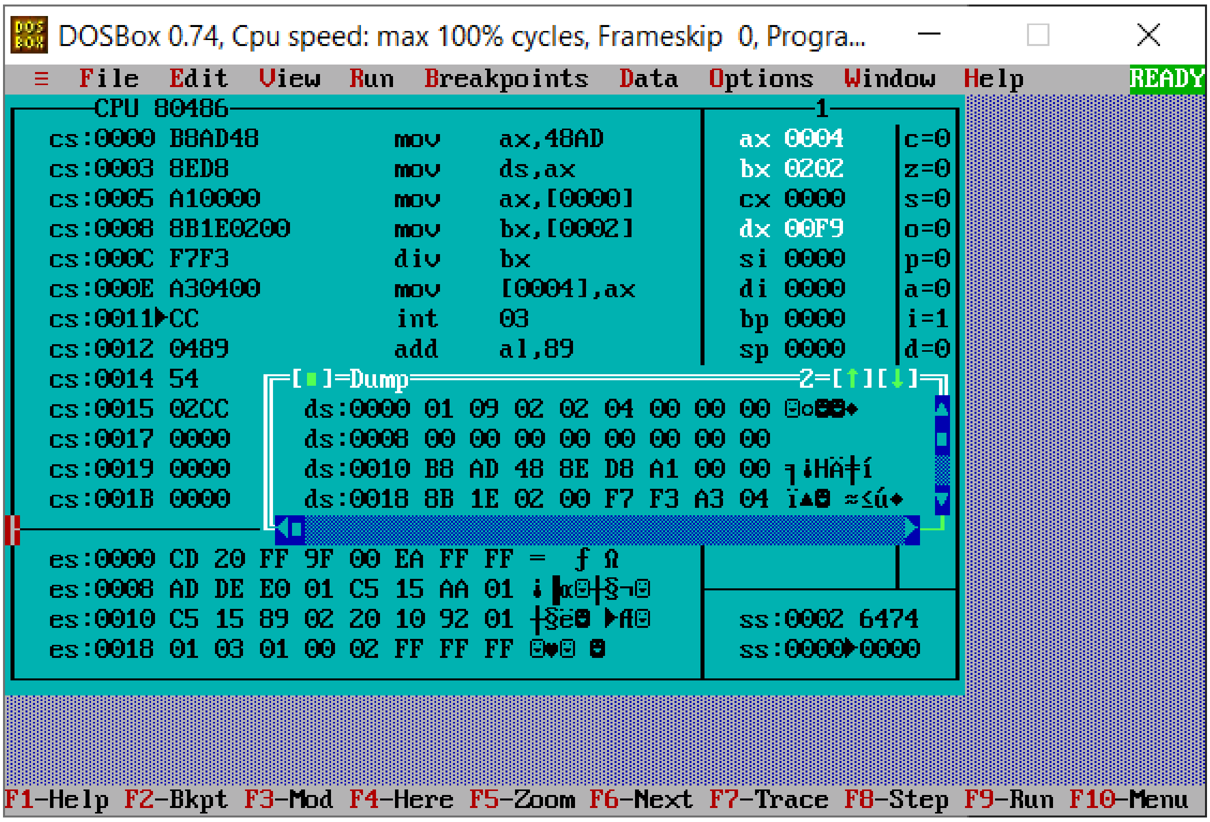
Output:

(a) Multiplication


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
≡ File Edit View Run Breakpoints Data Options Window Help READY
Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-Local

```
CPU 80486                                              1
cs:0000 B8AD48          mov    ax,48AD          ax F001   c=1
cs:0003 8ED8            mov    ds,ax            bx FFFF   z=0
cs:0005 A10000          mov    ax,[0000]        cx 0000   s=0
cs:0008 8B1E0200        mov    bx,[0002]        dx 0FFE   o=1
cs:000C F7E3            mul    bx               si 0004   p=0
cs:000E BE0400          mov    si,0004          di 0000   a=0
cs:0011 8904            mov    [si],ax          bp 0000   i=1
cs:0013 895402          mov    [si+02],dx       sp 0000   d=0
cs:0016►CC          [■]=Dump=============================2=[↑][↓]
cs:0017 0000          ds:0000 FF 0F FF FF 01 F0 FE 0F  ☼  ≡■☼
cs:0019 0000          ds:0008 00 00 00 00 00 00 00 00
cs:001B 0000          ds:0010 B8 AD 48 8E D8 A1 00 00 ╕¡HÄ╪í
cs:001D 0000          ds:0018 8B 1E 02 00 F7 E3 BE 04 ï▲☻ ≈π╛♦

es:0000 CD 20 FF 9F 00 EA FF FF ═  ƒ Ω
es:0008 AD DE E0 01 C5 15 AA 01 ¡▐α☺┼§¬☺
es:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺      ss:0002 6474
es:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻         ss:0000►0000
```

(b) Division


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
≡ File Edit View Run Breakpoints Data Options Window Help READY
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```
CPU 80486                                              1
cs:0000 B8AD48          mov    ax,48AD          ax 0004   c=0
cs:0003 8ED8            mov    ds,ax            bx 0202   z=0
cs:0005 A10000          mov    ax,[0000]        cx 0000   s=0
cs:0008 8B1E0200        mov    bx,[0002]        dx 00F9   o=0
cs:000C F7F3            div    bx               si 0000   p=0
cs:000E A30400          mov    [0004],ax        di 0000   a=0
cs:0011►CC              int    03               bp 0000   i=1
cs:0012 0489            add    al,89            sp 0000   d=0
cs:0014 54          [■]=Dump=============================2=[↑][↓]
cs:0015 02CC          ds:0000 01 09 02 02 04 00 00 00 ☺○☻☻♦
cs:0017 0000          ds:0008 00 00 00 00 00 00 00 00
cs:0019 0000          ds:0010 B8 AD 48 8E D8 A1 00 00 ╕¡HÄ╪í
cs:001B 0000          ds:0018 8B 1E 02 00 F7 F3 A3 04 ï▲☻ ≈≤ú♦

es:0000 CD 20 FF 9F 00 EA FF FF ═  ƒ Ω
es:0008 AD DE E0 01 C5 15 AA 01 ¡▐α☺┼§¬☺
es:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺      ss:0002 6474
es:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻         ss:0000►0000
```

Experiment 3:

Aim:

Sorting numbers in ascending and descending orders (Bubble sort)

ASM code:

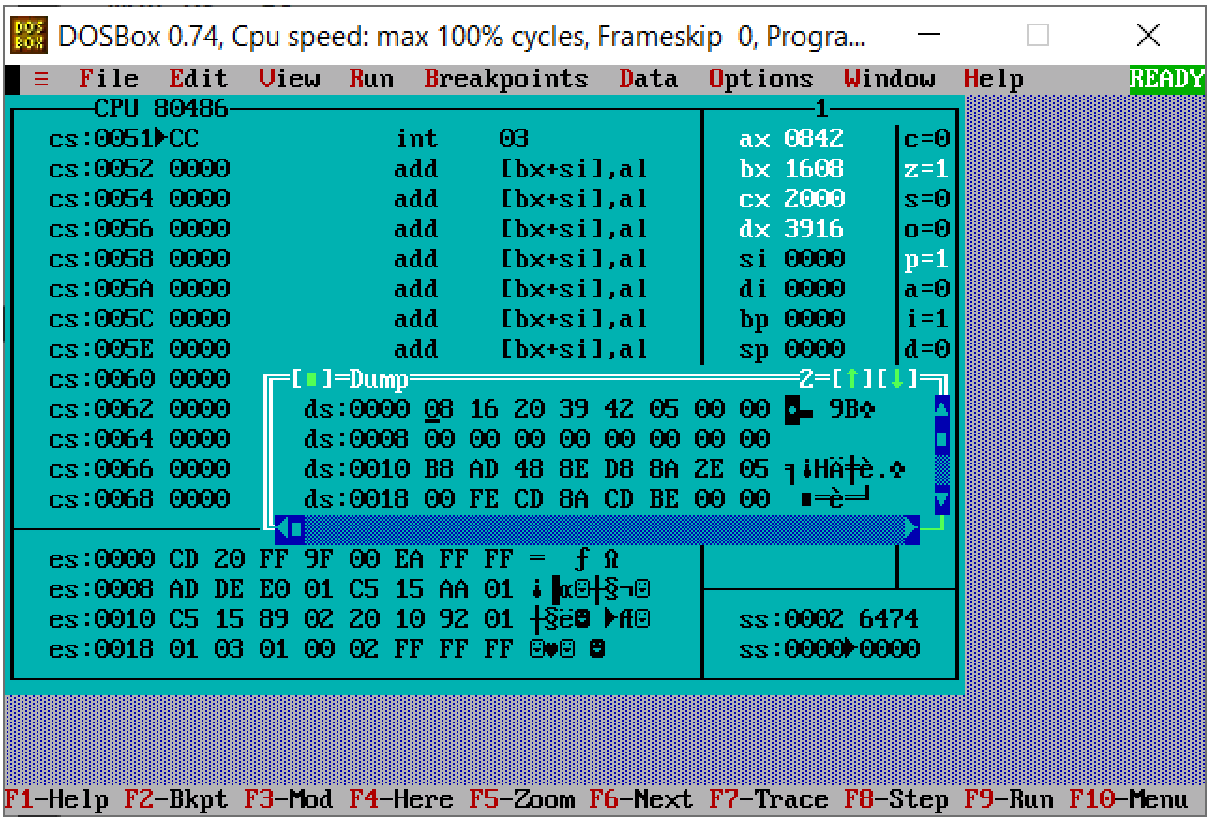
```
;106118042 - joel scaria - bubble sort
data segment
  arr db 20h,16h,42h,39h,08h
  len db len-arr
  desc_bit db 00h
data ends

code segment
assume cs:code, ds:data
start:
  mov ax, data
  mov ds, ax

  mov ch, len
  dec ch
OuterLoop:
  mov cl, ch
  lea si, arr



InnerLoop:
  mov al, [si]
  mov bl, [si + 1]
  cmp desc_bit,00h
  je Ascend
  cmp al,bl
  jnc Continue
  jmp swap

Ascend:
  cmp al,bl
  jc Continue
  jmp swap

Continue:
  inc si
  dec cl
  jnz InnerLoop
  dec ch
  jnz OuterLoop
  jmp stop

swap:
  mov dl, [si + 1]
  xchg [si], dl
  mov [si + 1], dl
  jmp Continue

stop:
  lea si,arr
  ;loading the values in array to view them in data registers
  mov ah,[si]
  mov bh,[si+1]
  mov ch,[si+2]
  mov dh,[si+3]
  mov al,[si+4]
  int 3

code ends
```

```
end start
```

Explanation:

- Input is the 5 numbers , length of array i.e. 5 in this case and a single bit used for determining if sort is in ascending or descending
- 2 loops are used ; innerloop and outerloop (Complexity >> O(n^2))
- Consecutive bits [si] and [si+1] are selected and compared
- If ascending, [si] and [si+1] swapped if [si]>[si+1]
- If descending, [si] and [si+1] swapped if [si]<[si+1]
- Then [si] is incremented by one and process repeated till [si+5]
- Process repeated 5 times

Output:



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
≡ File Edit View Run Breakpoints Data Options Window Help READY

```
CPU 80486                                                1
 cs:0051▶CC             int     03                ax 0842  c=0
 cs:0052 0000           add     [bx+si],al        bx 1608  z=1
 cs:0054 0000           add     [bx+si],al        cx 2000  s=0
 cs:0056 0000           add     [bx+si],al        dx 3916  o=0
 cs:0058 0000           add     [bx+si],al        si 0000  p=1
 cs:005A 0000           add     [bx+si],al        di 0000  a=0
 cs:005C 0000           add     [bx+si],al        bp 0000  i=1
 cs:005E 0000           add     [bx+si],al        sp 0000  d=0
 cs:0060 0000   [■]=Dump                              2=[↑][↓]
 cs:0062 0000     ds:0000 08 16 20 39 42 05 00 00 ◘▬ 9B♠
 cs:0064 0000     ds:0008 00 00 00 00 00 00 00 00
 cs:0066 0000     ds:0010 B8 AD 48 8E D8 8A 2E 05 ╕¡HÄ╪è.♣
 cs:0068 0000     ds:0018 00 FE CD 8A CD BE 00 00  ■═è═╛

 es:0000 CD 20 FF 9F 00 EA FF FF ═  ƒ Ω
 es:0008 AD DE E0 01 C5 15 AA 01 ¡▐α☺┼§¬☺
 es:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺          ss:0002 6474
 es:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻             ss:0000▶0000
```

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

Experiment 4:

Aim:

Factorial of a number greater than 6

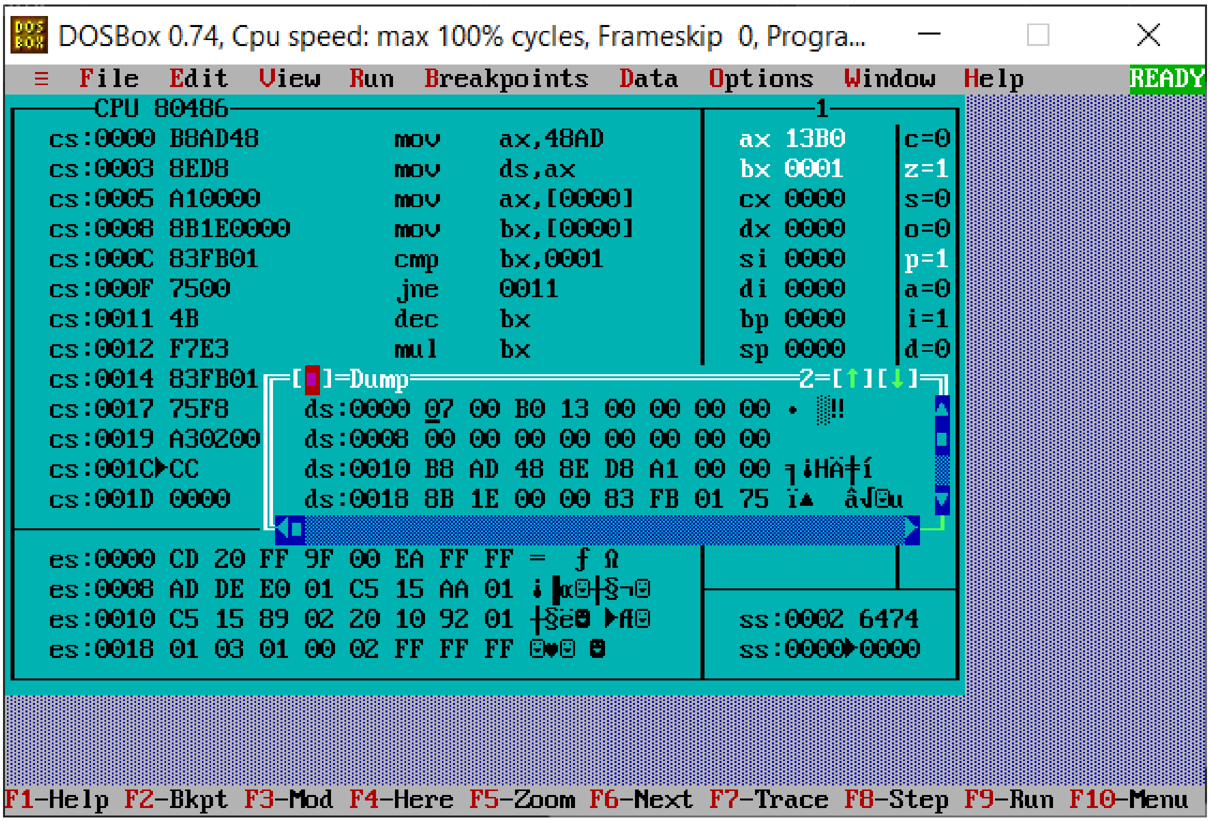
ASM code:

```
; 106118042 Joel Scaria - Factorial
data segment
  a dw 0007h
  b dw ?
data ends

code segment
assume cs:code, ds:data

start:
  mov ax, data
  mov ds, ax

  ; Starting factorial code
  mov ax, a ; Load data
  mov bx, a ; Value of bx decreases by 1 from a to 1

  cmp bx, 01
  jnz factorial
factorial:
  dec bx
  mul bx
  cmp bx, 01
  jnz factorial

  mov b, ax
  int 3

code ends
end start
```

Explanation:

- bx=7
- ax=bx
- bx=bx-1
- ax=ax*bx
- if bx is not 1, go to third step
- else move ax (answer) to b (dump)

Output:


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
≡ File Edit View Run Breakpoints Data Options Window Help READY
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```
CPU 80486                                                  1
 cs:0000 B8AD48       mov     ax,48AD              ax 13B0   c=0
 cs:0003 8ED8         mov     ds,ax                bx 0001   z=1
 cs:0005 A10000       mov     ax,[0000]            cx 0000   s=0
 cs:0008 8B1E0000     mov     bx,[0000]            dx 0000   o=0
 cs:000C 83FB01       cmp     bx,0001              si 0000   p=1
 cs:000F 7500         jne     0011                 di 0000   a=0
 cs:0011 4B           dec     bx                   bp 0000   i=1
 cs:0012 F7E3         mul     bx                   sp 0000   d=0
 cs:0014 83FB01 [■]=Dump                          2=[↑][↓]
 cs:0017 75F8    ds:0000 07 00 B0 13 00 00 00 00 •  ░!!
 cs:0019 A30200  ds:0008 00 00 00 00 00 00 00 00
 cs:001C▶CC      ds:0010 B8 AD 48 8E D8 A1 00 00 ╕¡HÄ╪í
 cs:001D 0000    ds:0018 8B 1E 00 00 83 FB 01 75 ï▲  â√☺u

 es:0000 CD 20 FF 9F 00 EA FF FF = ƒ Ω
 es:0008 AD DE E0 01 C5 15 AA 01 ¡▐α☺┼§¬☺
 es:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺     ss:0002 6474
 es:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻        ss:0000▶0000
```

NB: factorial of 7 is 13B0

Experiment 5:

Aim:

Matrix multiplication of two matrices m and n where m!=n

ASM code:

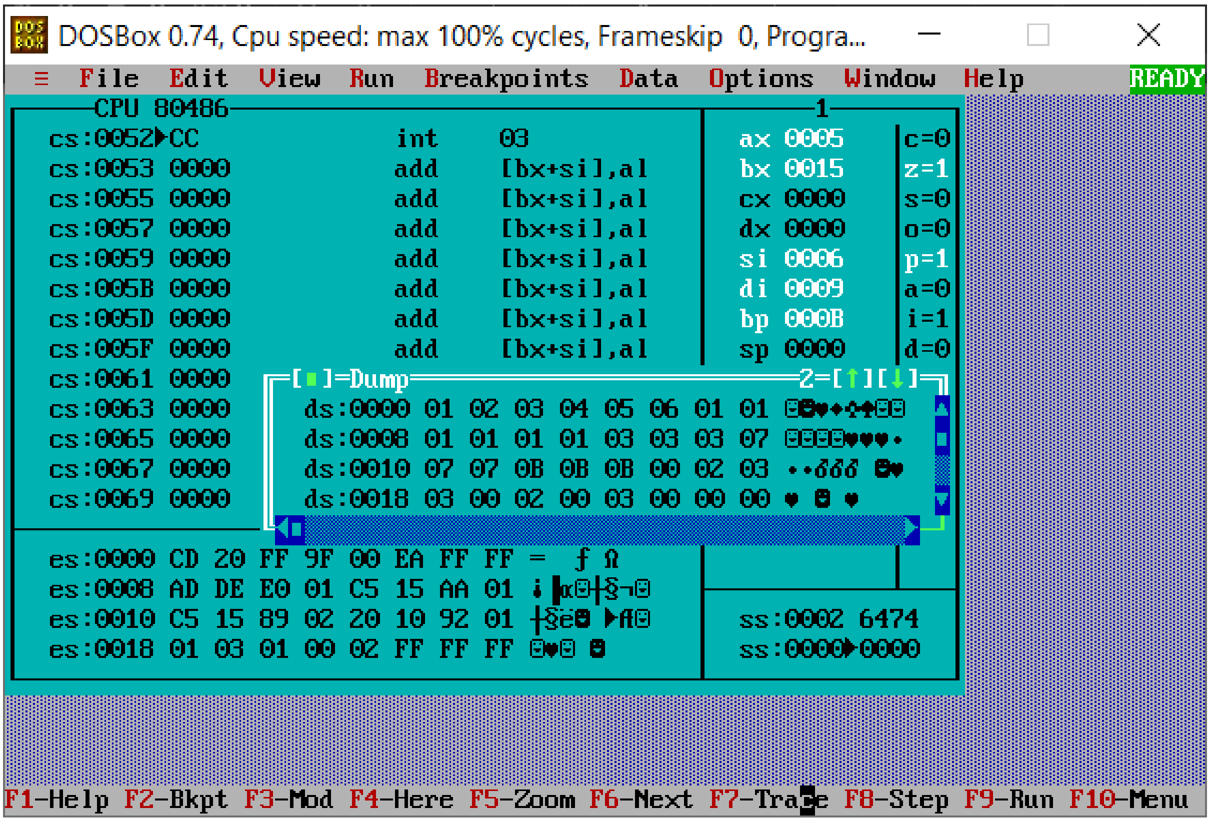
```
; 106118042 - Joel Scaria - Matrix Multiplication
data segment
  ; Declare 3x2 size array
  mat1 db 1h, 2h, 3h, 4h, 5h, 6h


  ; Declare 2x3 size array
  mat2 db 1h, 1h, 1h, 1h, 1h, 1h

  ; Declare an empty 3x3 size array
  mat3 db 09h dup(?)

  small_a db 3h
  small_b db 2h
  small_c db 3h

  a dw 3h
  b dw 2h
  c dw 3h

data ends

code segment
assume cs:code, ds:data
start:
  mov ax, data
  mov ds, ax

  ; ch register store value of index i
  mov ch, small_a ; = i

  ; offset is the address from the beginning of memory segment where the variable is stored
  mov bx, offset mat3
  mov si, offset mat1

NextRow:
  ; di represents the index at mat2
  mov di, offset mat2
  ; Store the value of iterator j in cl
  mov cl, small_c ; j = c

NextCol:
  ; Store the value of iterator k in dl
  mov dl, small_b

  ; Store value of mat[i][j] in bp
  mov bp, 0h

NextElement:
  mov ax, 0h
  mov al, [si]
  mul byte ptr[di]
  add bp, ax
  inc si ; Go to next element in same row for mat1
  add di, c ; Go to next element in same col for mat2

  dec dl ; --k
  ; Continue iterating k from (b to 1)
  jnz NextElement

  ; Here k = 0
  ; Copy calculated value to mat3
  mov [bx], bp

  ; start again from beginning of current row
  sub si, b

  ; Go to beginning of next column
  mov ax, a
  mul b
  dec ax
  sub di, ax

  ; increment mat3
  inc bx

  ; one less column to traverse
  dec cl ; --j

  cmp cl, 0
  jnz NextCol

  ; End of loop having iterator j
  ; Go to beginning of next row
  add si, b

  ; one less row to traverse
  dec ch; --i

  cmp ch, 0
  jnz NextRow

  ; End of iterator i
  int 3 ; breakpoint

code ends
end start
```

Explanation:

- mat1
  1 2
  3 4

```
   5 6
➢  mat2
   1 1 1
   1 1 1
➢  [si]=mat1
➢  [bx]=new empty 3x3 matrix
   a=3,b=2,c=3
   ch=3 (no of rows in mat1)
➢  Outer loop:
           [di]=mat2
           cl=3 (no of columns in mat2)
➢  Inner loop:
           dl=2
           bp=0(initial value of  cell in new matrix)
➢  Innermost loop:
           ax=0
           al=si
           ax=ax*[di]
           bp=bp+ax
           si=si+1(goes to next element in same row in mat1)
           [di]=[di]+c(goes to next element in same column in mat2.here c=3)
           dl=dl-1
           Go to Innermost loop again if dl is not 0 else continue
➢  [bx]=bp(save value to new mat variable)
   [si]=[si]-b (restart current row of mat1 for next element)
   [di]=[di]-((a*b)-1) (move to next column of mat2 for next element)
   [bx]=[bx+1] (new element is to be stored in [bx+1]
   cl=cl-1 (one less column to be traversed)
   Go to Inner loop again if cl is not zero else continue
➢  [si]=[si]+b (move to first element of next row for mat1)
   ch=ch-1 (one less row to be traversed)
   Go to Outer loop again if ch is not zero else continue
➢  program ends
```

Output:


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
≡ File Edit View Run Breakpoints Data Options Window Help READY
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```
CPU 80486                                                    1
  cs:0052►CC          int    03             ax 0005   c=0
  cs:0053 0000        add    [bx+si],al     bx 0015   z=1
  cs:0055 0000        add    [bx+si],al     cx 0000   s=0
  cs:0057 0000        add    [bx+si],al     dx 0000   o=0
  cs:0059 0000        add    [bx+si],al     si 0006   p=1
  cs:005B 0000        add    [bx+si],al     di 0009   a=0
  cs:005D 0000        add    [bx+si],al     bp 000B   i=1
  cs:005F 0000        add    [bx+si],al     sp 0000   d=0
  cs:0061 0000   [■]=Dump                      2=[↑][↓]
  cs:0063 0000     ds:0000 01 02 03 04 05 06 01 01 ☺☻♥♦♣♠☺☺
  cs:0065 0000     ds:0008 01 01 01 01 03 03 03 07 ☺☺☺☺♥♥♥•
  cs:0067 0000     ds:0010 07 07 0B 0B 0B 00 02 03 ••♂♂♂ ☻♥
  cs:0069 0000     ds:0018 03 00 02 00 03 00 00 00 ♥ ☻ ♥

  es:0000 CD 20 FF 9F 00 EA FF FF ═  ƒ Ω
  es:0008 AD DE E0 01 C5 15 AA 01 ¡▐α☺┼§¬☺
  es:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺     ss:0002 6474
  es:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻        ss:0000►0000
```

Experiment 6:

Aim:

Matrix subtraction of 2 matrices m and n where m!=n

ASM code:

```
;106118042 - joel scaria - matrix subtraction
data segment

  ; 3x2 matrices mat1 and mat2

  mat1 dw 1h, 2h, 3h, 4h, 5h, 6h

  mat2 dw 1h, 1h, 1h, 1h, 1h, 1h

  a dw 3h
  b dw 2h
```

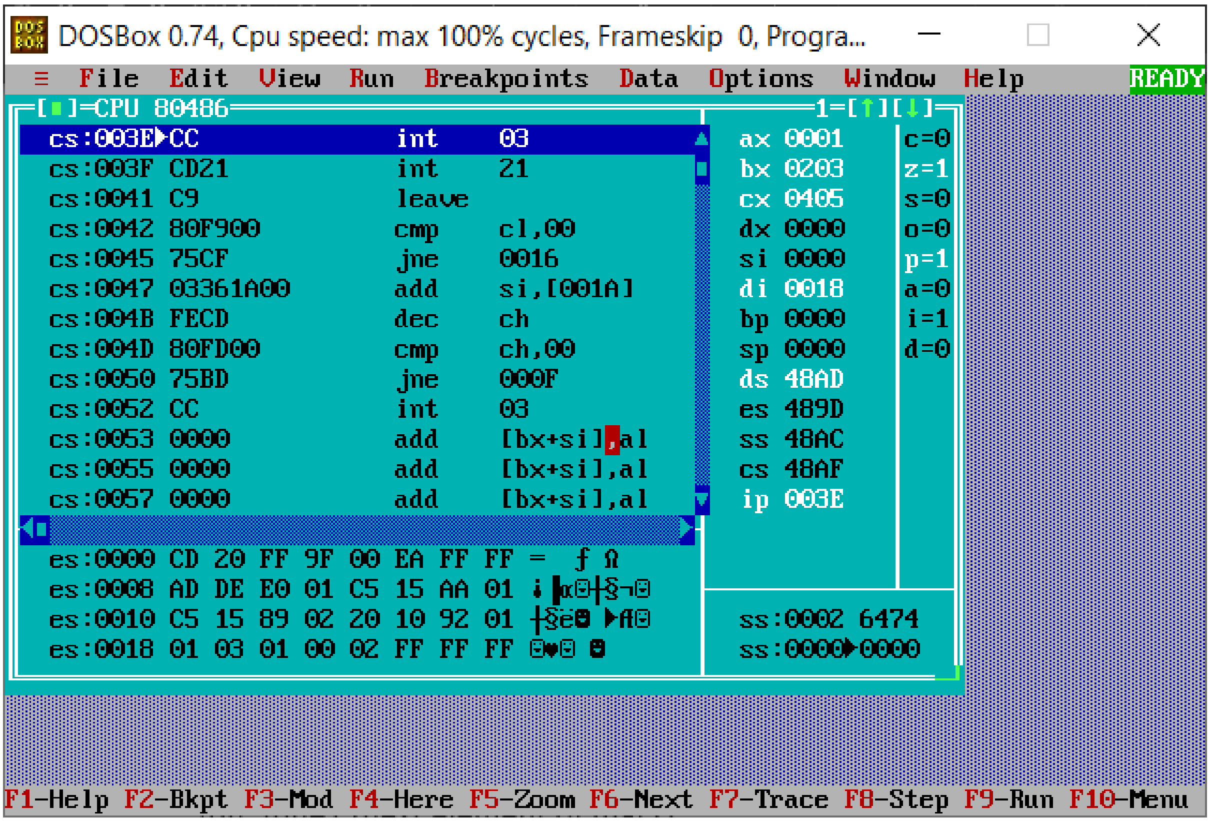
```
data ends

code segment
assume cs:code, ds:data
start:
    mov ax, data
    mov ds, ax

    mov ax, a
    mul b

    mov cx, ax
    lea si, mat1
    lea di, mat2

Loop_i:
    mov ax, [si]
    mov bx, [di]
    sub ax, bx

    mov [si], ax
    add si, 2h
    add di, 2h
    sub cx, 0001h

    cmp cx, 0
    jnz Loop_i

    lea si,mat1
    mov ah,[si]
    mov al,[si+2]
    mov bh,[si+4]
    mov bl,[si+6]
    mov ch,[si+8]
    mov cl,[si+10]

    int 3

    int 21h
code ends
end start
```

Explanation:

- mat1= 1 2
  3 4
  5 6
- mat2= 1 1
  1 1
  1 1
- required output =mat1-mat2= 0 1
  2 3
  4 5
- a=3 (no of rows) b=2 (no of columns)
- cx=3*2=6 (total no of subtractions)
- [si]=mat1, [di]=mat2
- Loop:
  ax=[si]
  bx=[di]
  ax=ax-bx
  [si]=ax (output will be in [si])
  [si]=[si]+1 (next element of mat1)
  [di]=[di]+1 (next element of mat2)
  cx=cx-1 (one less subtraction to go)
  Go to Loop if cx is not zero else continue
- program ends

Output:

NB: Output matrix is displayed using ax,bx and cx registers



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
≡ File Edit View Run Breakpoints Data Options Window Help READY
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```
[■]=CPU 80486=====================================1=[↑][↓]=
 cs:003E▸CC          int    03               ax 0001  c=0
 cs:003F CD21        int    21               bx 0203  z=1
 cs:0041 C9          leave                   cx 0405  s=0
 cs:0042 80F900      cmp    cl,00            dx 0000  o=0
 cs:0045 75CF        jne    0016             si 0000  p=1
 cs:0047 03361A00    add    si,[001A]        di 0018  a=0
 cs:004B FECD        dec    ch               bp 0000  i=1
 cs:004D 80FD00      cmp    ch,00            sp 0000  d=0
 cs:0050 75BD        jne    000F             ds 48AD
 cs:0052 CC          int    03               es 489D
 cs:0053 0000        add    [bx+si],al       ss 48AC
 cs:0055 0000        add    [bx+si],al       cs 48AF
 cs:0057 0000        add    [bx+si],al       ip 003E

 es:0000 CD 20 FF 9F 00 EA FF FF =  ƒ Ω
 es:0008 AD DE E0 01 C5 15 AA 01 ¡▐α☺┼§¬☺
 es:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺      ss:0002 6474
 es:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻         ss:0000▸0000
```

Experiment 7:

Aim:

Generate Fibonacci series

ASM code:

```
; Program for calculating n'th fibonacci number
; 106118042 - Joel Scaria

data segment
    n dw 8h
data ends

code segment
assume cs:code, ds:data
start:
  mov ax, data
  mov ds, ax

  mov bx, 0
  mov cx, 1

  ; Initialize iterator
  mov dx, n
  mov ax, 0
  cmp dx, 1
  jz end_i
  mov ax, 1
  cmp dx, 2
  jz end_i
  sub dx, 2

Loop_i:
  ; This loop runs untill dx is 0
  mov ax, 0
  add ax, bx
  add ax, cx

  ; Update values
  mov bx, cx
  mov cx, ax

  dec dx
  cmp dx, 0
  jnz Loop_i

end_i:
  int 3
```

```
code ends

end start
```

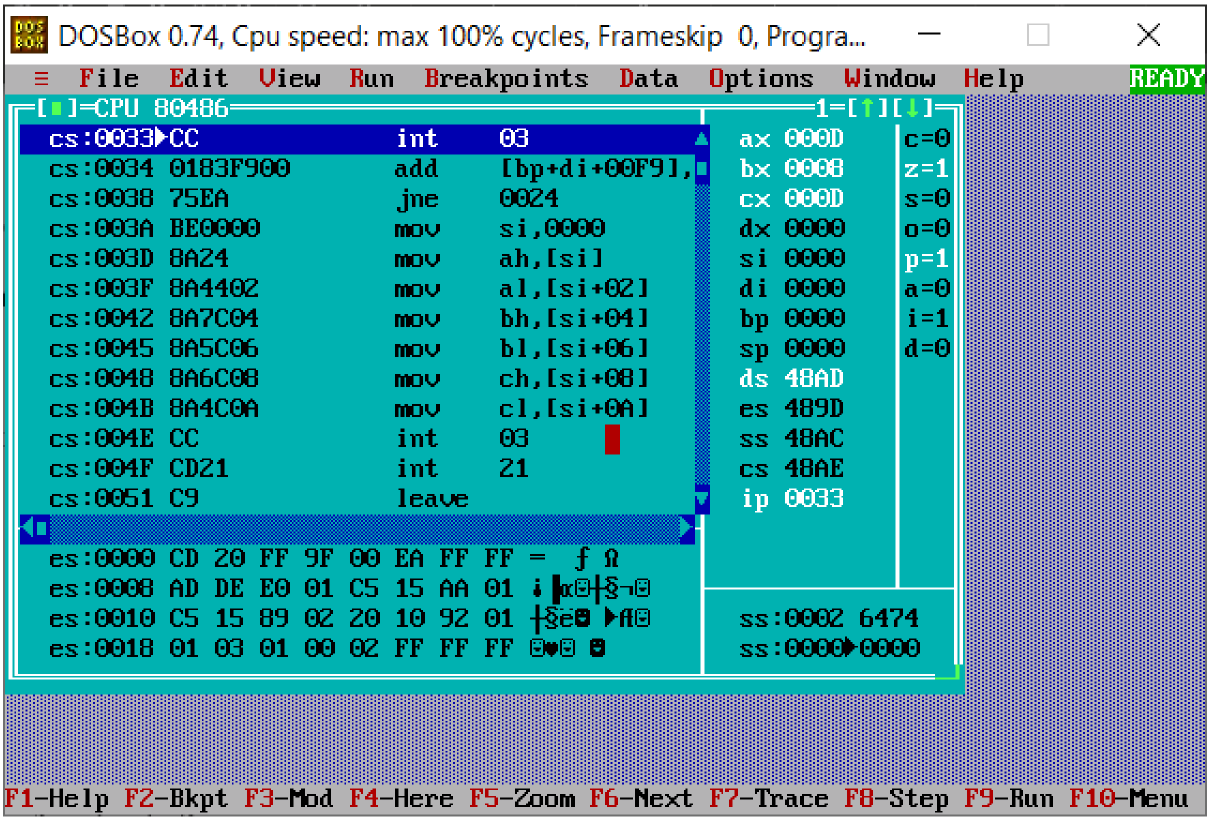
Explanation:

- input = 8
- required output = 8th Fibonacci number = D (13 in decimal)
- let bx=0, cx=1, dx=8
- ax=0
- if dx is 1, end program else continue
- ax=1
- if dx is 2, end program else continue
- dx=dx-2
- Loop:

  ```
  ax=0
  ax=ax+bx
  ax=ax+cx (so finally, ax=ax+(bx+cx))
  bx=cx(swapping 2nd last element)
  cx=ax(swapping last element)
  dx=dx-1 (loop runs till dx=0 i.e. 8 times in total)
  Go to Loop again if dx is not 0 else continue
  ```
- program ends

Output:

NB: Output is in ax register



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
≡ File Edit View Run Breakpoints Data Options Window Help READY
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```
[■]=CPU 80486=======================================1=[↑][↓]=
 cs:0033►CC            int    03                  ax 000D  c=0
 cs:0034 0183F900      add    [bp+di+00F9],       bx 0008  z=1
 cs:0038 75EA          jne    0024                cx 000D  s=0
 cs:003A BE0000        mov    si,0000             dx 0000  o=0
 cs:003D 8A24          mov    ah,[si]             si 0000  p=1
 cs:003F 8A4402        mov    al,[si+02]          di 0000  a=0
 cs:0042 8A7C04        mov    bh,[si+04]          bp 0000  i=1
 cs:0045 8A5C06        mov    bl,[si+06]          sp 0000  d=0
 cs:0048 8A6C08        mov    ch,[si+08]          ds 48AD
 cs:004B 8A4C0A        mov    cl,[si+0A]          es 489D
 cs:004E CC            int    03                  ss 48AC
 cs:004F CD21          int    21                  cs 48AE
 cs:0051 C9            leave                      ip 0033

 es:0000 CD 20 FF 9F 00 EA FF FF = ƒ Ω
 es:0008 AD DE E0 01 C5 15 AA 01 ¡▐α☺┼§¬☺        ss:0002 6474
 es:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺        ss:0000►0000
 es:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻
```

Experiment 8:

Aim:

(a) Binary to grey conversion

(b) decimal to hexadecimal conversion

ASM code:

Binary to grey

```
; 106118042 JOEL SCARIA binary to grey
data segment
      opr db 2Ch
      ; 0010 1100 -> 2C
      ; 0001 0110 (after right shift)
      ; 0011 1010 -> 3A (output)
      res db ?
data ends

code segment
assume cs:code,ds:data
```

```
start:

      mov ax,data
      mov ds,ax
      mov al,opr
      mov bl,al
      shr al,01h
      xor al,bl
      mov res,al
      int 3
code ends
end start
```

decimal to hexadecimal

```
; 106118042 JOEL SCARIA decimal to hexadecimal
data segment
      a db 19h
      ; 19 -> 25
      ; 0001 1001
      ; 13
      b db ?
data ends
code segment
assume cs:code,ds:data
start:
      mov ax,data
      mov ds,ax
      mov bl,a
      and bl,0fh
      mov al,a
      mov dx,00f0h
      and al,dl
      mov cl,04h
      ror al,cl
      mov dl,0ah
      mul dl
      add al,bl
      mov b,al
      int 3
code ends
end start
```

Explanation:

(a) binary to grey

- input opr= 2C

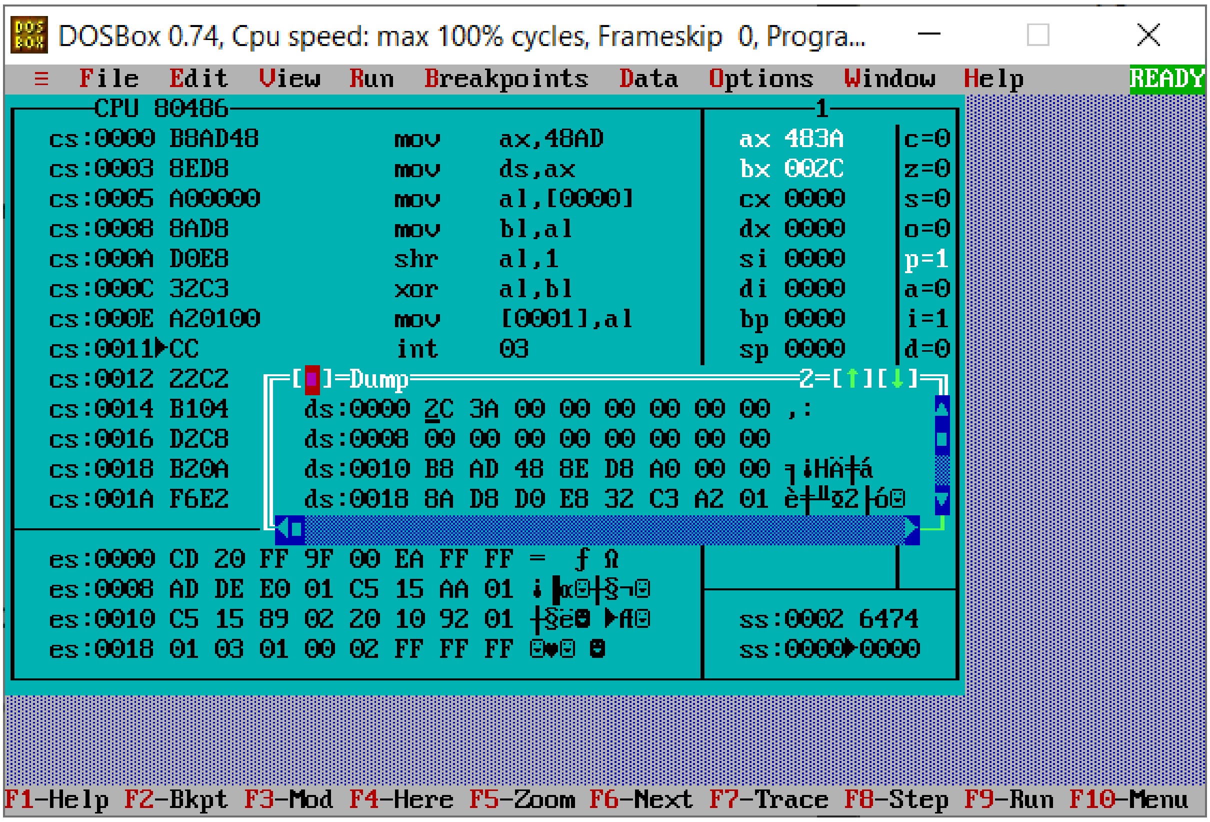
- required output = 3A
- al=bl=opr
- shift al to the right by 1 bit
- xor bl and new al which is obtained after shifting
- store result i.e. grey code in res
- program ends

(b) decimal to hexadecimal

- input= a = 19
- required output = 13
- bl=a
- bl= bl & 0fh=0001 1001 & 0000 1111= 0000 1001 = 9
- al=a
- al=al & 0f0h=0001 1001 & 1111 0000= 0001 0000= 10
- shift al to the right by 4 bits -> al = 0000 0001=1
- al= al*Ah=1*Ah=A
- al=al+bl=Ah + 9h=13h
- store output al into b (dump)
- program ends

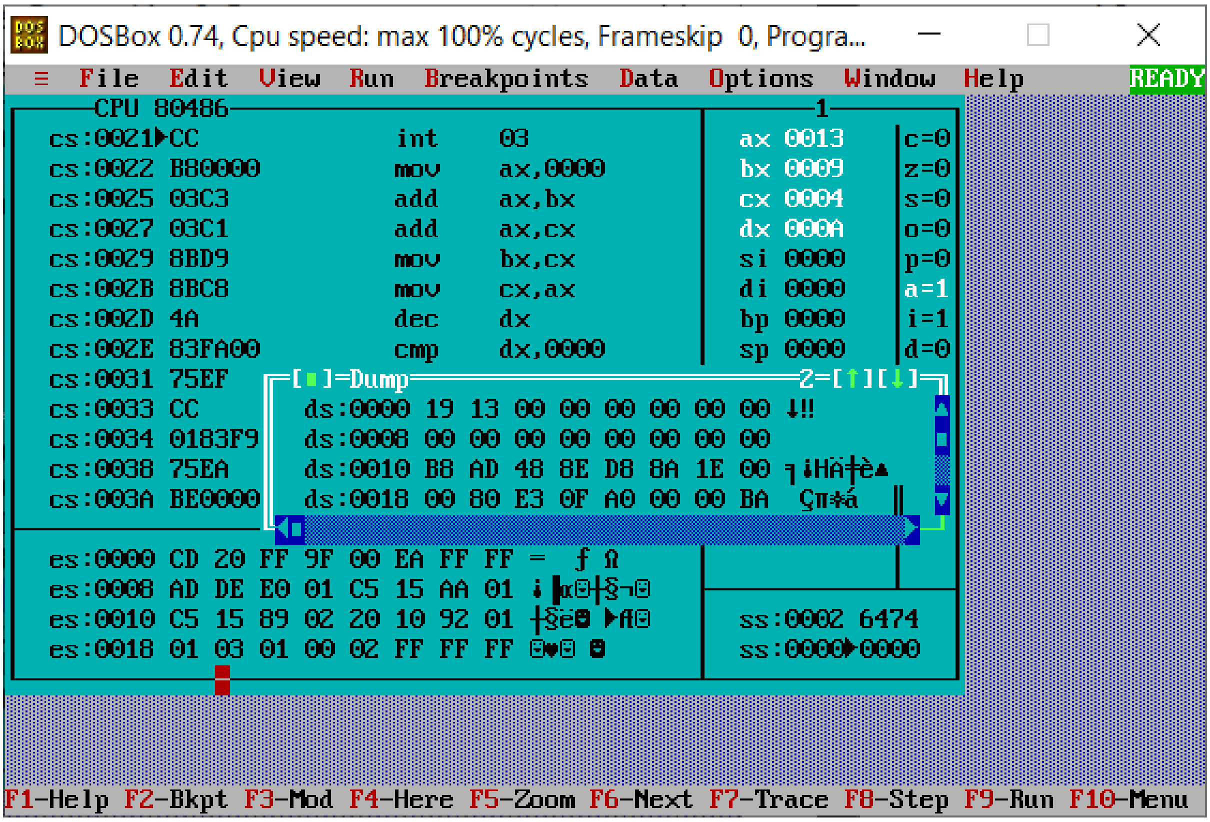
Output:

(a) Binary to grey


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...
≡ File Edit View Run Breakpoints Data Options Window Help READY
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

```
CPU 80486                                               1
 cs:0000 B8AD48        mov    ax,48AD          ax 483A  c=0
 cs:0003 8ED8          mov    ds,ax            bx 002C  z=0
 cs:0005 A00000        mov    al,[0000]        cx 0000  s=0
 cs:0008 8AD8          mov    bl,al            dx 0000  o=0
 cs:000A D0E8          shr    al,1             si 0000  p=1
 cs:000C 32C3          xor    al,bl            di 0000  a=0
 cs:000E A20100        mov    [0001],al        bp 0000  i=1
 cs:0011▶CC            int    03               sp 0000  d=0
 cs:0012 22C2    [■]=Dump                         2=[↑][↓]
 cs:0014 B104      ds:0000 2C 3A 00 00 00 00 00 00 ,:
 cs:0016 D2C8      ds:0008 00 00 00 00 00 00 00 00
 cs:0018 B20A      ds:0010 B8 AD 48 8E D8 A0 00 00 ╕¡HÄ╪á
 cs:001A F6E2      ds:0018 8A D8 D0 E8 32 C3 A2 01 è╪╨Φ2├ó☺

 es:0000 CD 20 FF 9F 00 EA FF FF =  ƒ Ω
 es:0008 AD DE E0 01 C5 15 AA 01 ¡▐α☺┼§¬☺
 es:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺          ss:0002 6474
 es:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻             ss:0000▶0000
```

(b) decimal to hexadecimal



```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip  0, Progra...
 ≡  File  Edit  View  Run  Breakpoints  Data  Options  Window  Help      READY
 ┌─CPU 80486──────────────────────────────────────1──────────────┐
   cs:0021►CC             int     03            ax 0013  │c=0
   cs:0022 B80000         mov     ax,0000       bx 0009  │z=0
   cs:0025 03C3           add     ax,bx         cx 0004  │s=0
   cs:0027 03C1           add     ax,cx         dx 000A  │o=0
   cs:0029 8BD9           mov     bx,cx         si 0000  │p=0
   cs:002B 8BC8           mov     cx,ax         di 0000  │a=1
   cs:002D 4A             dec     dx            bp 0000  │i=1
   cs:002E 83FA00         cmp     dx,0000       sp 0000  │d=0
   cs:0031 75EF     ╔[■]=Dump══════════════════════2=[↑][↓]╗
   cs:0033 CC         ds:0000 19 13 00 00 00 00 00 00 ↓‼
   cs:0034 0183F9     ds:0008 00 00 00 00 00 00 00 00
   cs:0038 75EA       ds:0010 B8 AD 48 8E D8 8A 1E 00 ╕¡HÄ╪è▲
   cs:003A BE0000     ds:0018 00 80 E3 0F A0 00 00 BA  Çπ☼á   ║

   es:0000 CD 20 FF 9F 00 EA FF FF  ═  ƒ Ω
   es:0008 AD DE E0 01 C5 15 AA 01 ¡▐α☺┼§¬☺
   es:0010 C5 15 89 02 20 10 92 01 ┼§ë☻ ►Æ☺      ss:0002 6474
   es:0018 01 03 01 00 02 FF FF FF ☺♥☺ ☻         ss:0000►0000

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```

Experiment 9:

Aim:

Compute nCr using recursive procedure

ASM code:

```
; 106118042 | Joel Scaria | tasm program for finding nCr (n<=8)


data segment

  n dw ?

  r dw ?

  str_n db 'Enter n: $'

  str_r db 'Enter r: $'

  str_ans db 'Answer: $'

data ends
```

```
code segment
assume cs:code, ds:data

start:
  mov ax, data
  mov ds, ax

  call PrintNextLine
  mov ah, 09h
  lea dx, str_n
  int 21h
  call ReadNumber
  mov n, bx

  call PrintNextLine
  mov ah, 09h
  lea dx, str_r
  int 21h
  call ReadNumber
  mov r, bx

  mov bx, n
  call Factorial
  push bx ; Save n! to stack

  mov bx, r
  call Factorial
  push bx ; Save r! to stack

  mov bx, n
  sub bx, r
  call Factorial
  push bx ; Save (n-r)! to stack

  pop cx
  pop bx
  pop ax
  div bx
  div cx
  aam
  mov bx, ax
  call PrintNextLine
  mov ah, 09h
  lea dx, str_ans
  int 21h

  ; Print the number

  call PrintBx



  ;to terminate the program
  mov ah, 1
  int 21h
  mov ah, 4ch
  int 21h

Factorial proc
  push ax
  mov ax, 1
  call FactorialHelper
  mov bx, ax
  pop ax
  ret
```

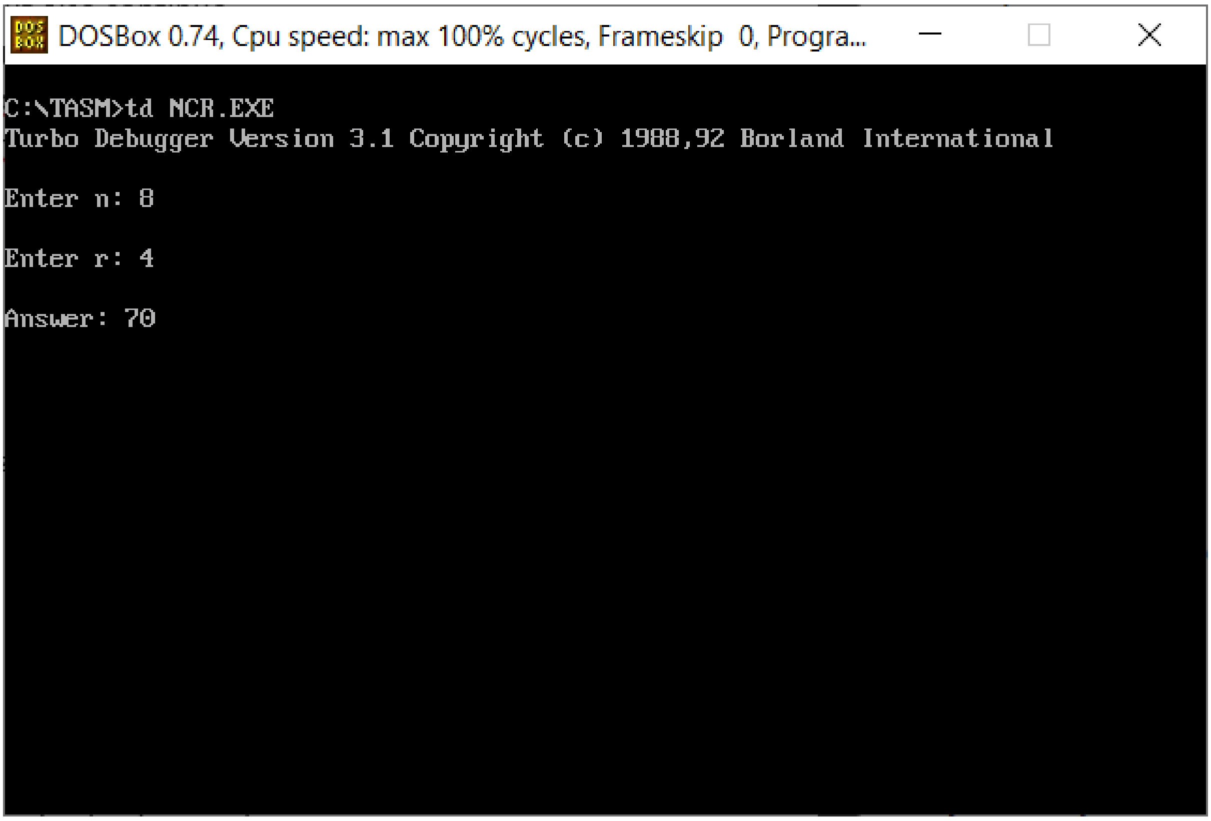
```
Factorial endp



FactorialHelper proc
  cmp bx, 1
  jle FactorialHelperResult
  mul bx
  sub bx, 1
  call FactorialHelper
FactorialHelperResult:
  ret
endp

; Procedure to input a number and save to bx register
ReadNumber proc
    ; Save values of ax, cx & dx registers
    push ax
    push cx
    push dx

    ; Make bx = 0
    mov bx, 0h
LoopReadNumber:
    xor ax, ax ; ax = 0

    ; Read a character in al register
    mov ah, 1
    int 21h ; Using DOS API to take input

    ; Check if the input digit is [0-9]
    cmp al, '0'
    jb BreakReadNumber
    cmp al, '9'
    ja BreakReadNumber


    sub al, 30h
    cbw ; byte to word
    ;cwd  word to dword

    ; Make bx = (bx * 10) + ax
    push ax
    mov ax, bx
    mov cx, 10
    mul cx ; ax = ax * 10
    mov bx, ax
    pop ax
    add bx, ax
    jmp LoopReadNumber

BreakReadNumber:
    pop dx
    pop cx
    pop ax
    ret
ReadNumber endp

; Prints newline
PrintNextLine proc
    push ax
    push dx

    mov dl, 10
    mov ah, 02h
    int 21h
    mov dl, 13
    mov ah, 02h
    int 21h

    pop dx
    pop ax
    ret
PrintNextLine endp

; Print number
PrintBx proc
    add bx, 3030h
    mov dl, bh
    ;add dl, 30h
    mov ah, 02h
    int 21h
    mov dl, bl
    ;add dl, 30h
    mov ah, 02h
    int 21h
    ret
PrintBx endp

code ends
end start
```

Explanation:

- Input and output done using interrupts
- Input:

```
        bx=0
        LoopReadNumber:
                ax=0
                accept single character from user using interrupt and store in al
                if al is not digit, stop Input else continue
                al=al-30h( convert ascii to number )
                bx=bx*10 + ax ( where bx= 10s place and ax= unit place)
                Go to LoopReadNumber ( Loop continues till user enters non-digit )
➢ Using Input, n and r obtained from user
➢ Factorial:
        assume we need n!
        bx=n
        ax=1
        Factorial helper:
                if bx=1, stop factorial else continue
                ax=ax*bx
                bx=bx-1
                Go to Factorial helper
  NB : n! present in ax
➢ ax=factorial(n)/factorial(r)*factorial(n-r)
➢ bx=ax
➢ Print:
        bx=bx+3030h( converting number back into ascii )
        output bl and bh separately using interrupts
➢ Print(bx)
➢ program ends
```

Output:



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

```
C:\TASM>td NCR.EXE
Turbo Debugger Version 3.1 Copyright (c) 1988,92 Borland International

Enter n: 8

Enter r: 4

Answer: 70
```

Experiment 10:

Aim:

check if string is palindrome or not

ASM code:

```
; 106118042 - JOEL SCARIA - Tasm pgm for palindrome check


data segment

    str_inp db 'Enter string: $'

    str_success db 'OK - Input is palindrome$'

    str_fail db 'FAIL - Input is not palindrome$'

    new db '$'

    s db 20 dup(0)

data ends
```

```
code segment
assume cs:code,ds:data

start:
    mov ax,data
    mov ds,ax

    lea dx, str_inp
    mov ah, 09h
    int 21h

    call ReadString

    call CheckPalidrome

    mov ah, 01h
    int 21h

    mov ah, 4ch
    int 21h



ReadString proc
    push ax
    mov bx, 00

ReadStringTakeMore:
    mov ah, 01h
    int 21h
    cmp al, 0dh
    je ReadStringDone
    mov [s+bx], al
    inc bx
    loop ReadStringTakeMore

ReadStringDone:
    pop ax
    ret

ReadString endp


CheckPalidrome proc
    push di
    mov di, 0
    dec bx

CheckPalidromeChar:
    mov al, [s+bx]
    cmp al, [s+di]
    jne CheckPalidromeFail
    inc di
    dec bx
    jnz CheckPalidromeChar

    lea dx, str_success
    mov ah, 09h
    int 21h
    jmp CheckPalidromeEnd

CheckPalidromeFail:
    lea dx, str_fail
    mov ah,09h
    int 21h
```

```
CheckPalidromeEnd:
    pop di
    ret

CheckPalidrome endp

code ends
end start
```

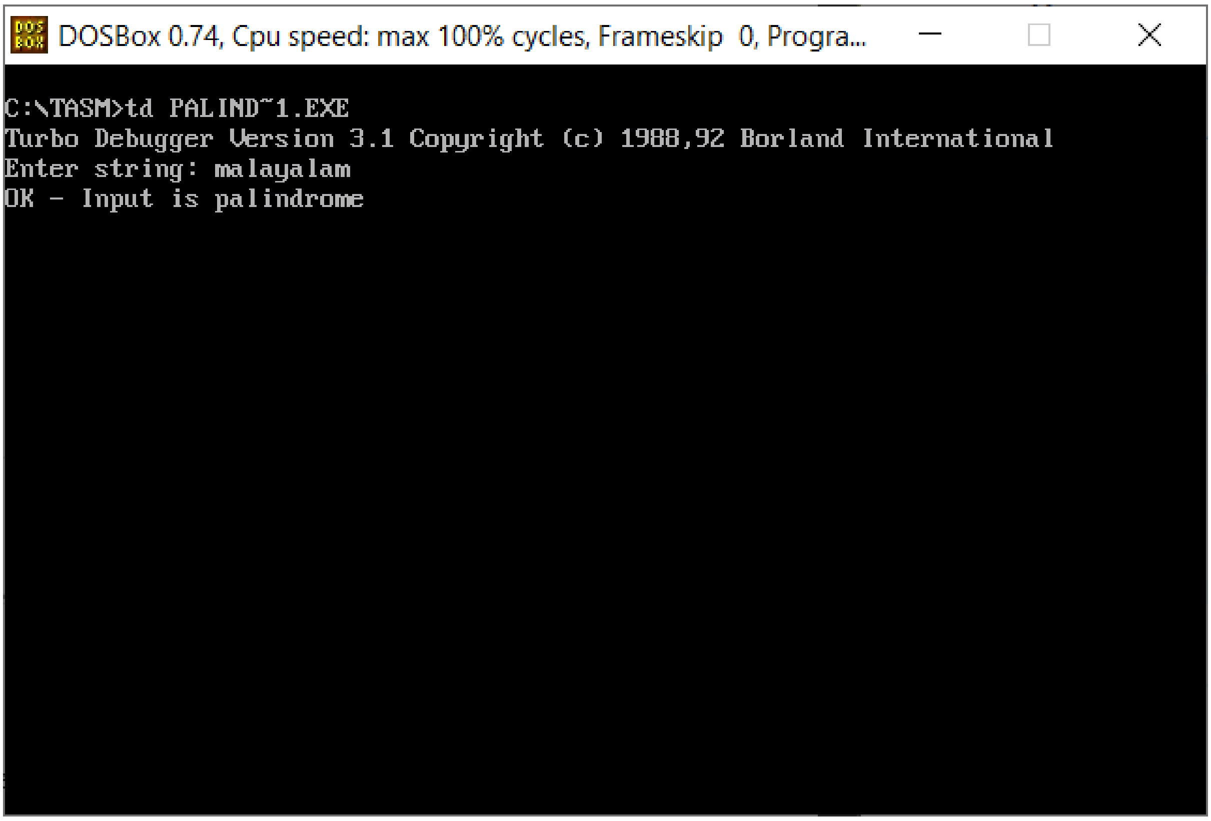
Explanation:

- bx=0
- We will be storing the input string in [s]
- Input:

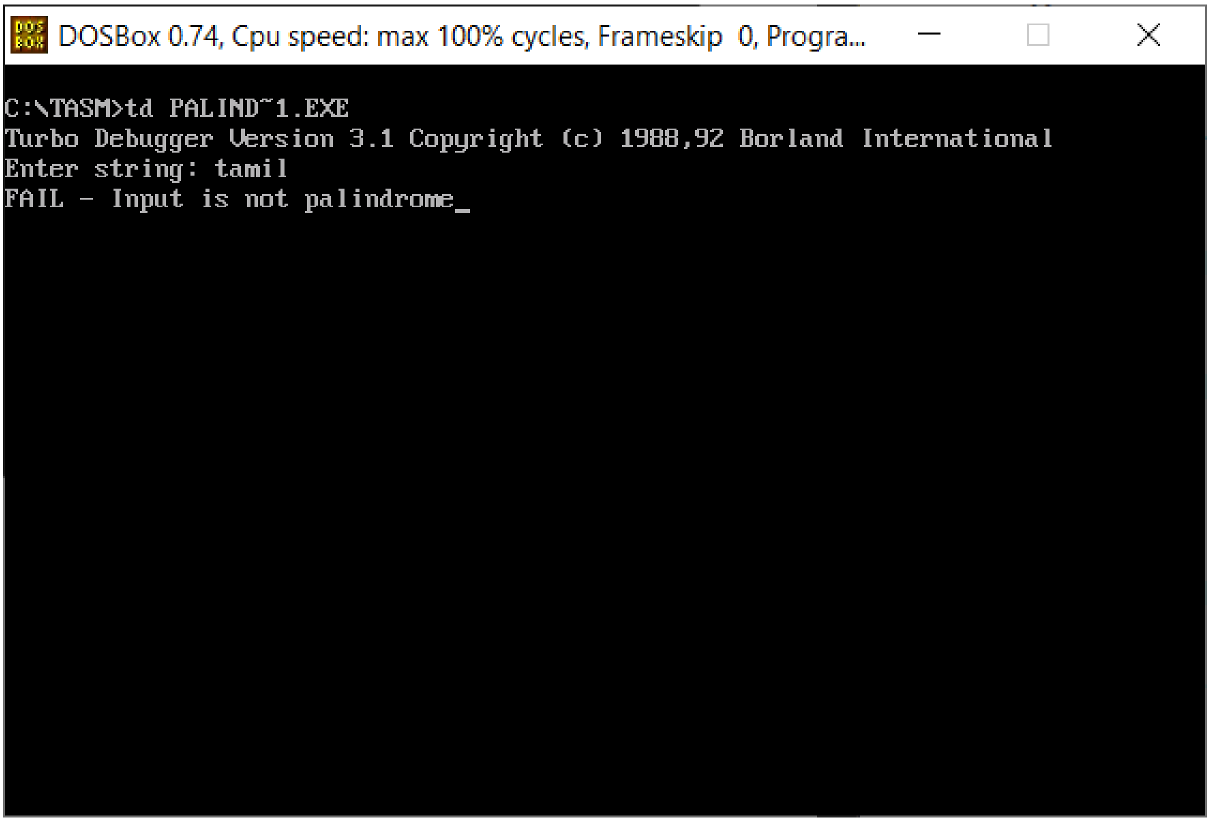
  Input single character from user using interrupt and store in al
  [s+bx]=al
  bx=bx+1(finally bx will be equal to length of string)
  Go to input if al!=0dh (user enters new line) else continue
- di=0
- Palindrome:

  al=[s+bx]
  if al!=[s+di], not palindrome else continue
  di=di+1
  bx=bx-1
  Repeat Palindrome till di=length of string and bx=0
- Output corresponding string using interrupt
- program ends

Output:


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

```
C:\TASM>td PALIND~1.EXE
Turbo Debugger Version 3.1 Copyright (c) 1988,92 Borland International
Enter string: malayalam
OK - Input is palindrome
```


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Progra...

```
C:\TASM>td PALIND~1.EXE
Turbo Debugger Version 3.1 Copyright (c) 1988,92 Borland International
Enter string: tamil
FAIL - Input is not palindrome_
```

Experiment 11:

Aim:

Proteus experiment for Unipolar stepper motor running in full drive mode

ASM code for 8051 micro controller:

```
ORG 00H
      MAIN: MOV A, #09H
      MOV P2, A
      ACALL DELAY
      MOV A,#0CH
      MOV P2, A
      ACALL DELAY
      MOV A,#06H
      MOV P2, A
      ACALL DELAY
      MOV A, #03H
      MOV P2, A
      ACALL DELAY
      SJMP MAIN
      DELAY:  MOV R3,   #08H
      DELAY1: MOV R2,   #0FFH
      DELAY2: MOV R1,   #0FFH
      BASE: DJNZ R1, BASE
      DJNZ R2, DELAY2
      DJNZ R3, DELAY1
      RET
      END
```

Explanation:

- Stepper motor is a synchronous DC motor in which rotation is divided into steps
- The angle covered in each step is called stepper angle
- Stepper motor is divided into mainly 2 types depending on the type of winding:

  Unipolar
  Bipolar

- Unipolar stepper motors are again divided into mainly 3 execution modes:

  Wave drive
  Full drive
  Half drive
- I have selected Full drive for my experiment
- In this mode two electromagnets are energized at a time, so the torque generated will be larger when compared to Wave Drive. This drive is commonly used than others. Power consumption will be higher than other modes.
- In my asm code, 1st I have given 09h which is nothing but 1001 in binary form
- The 1st and last bits are 1 which represents consecutive energised electromagnets of the stepper motor
- In next step, 0Ch is sent which is nothing but 1100.Here also we can see that another pair of consecutive electromagnets have been energised.
- Similarly, 0Ch is followed by 06h(0110) and 03h(0011)
- The 1s represent energised electromagnets and 0s represent non-energised ones
- When 2 consecutive magnets are energised, polarity will be in 45 degrees in between them
- For given setup, stepper angle = +90 degrees
- Delay is created by executing large number of empty loops(8*256*256)

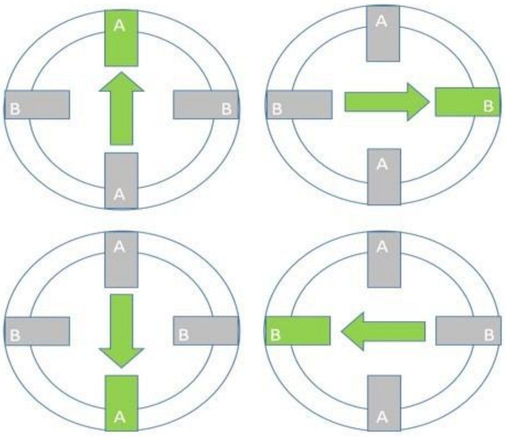


Fig 1 – One phase on – full step

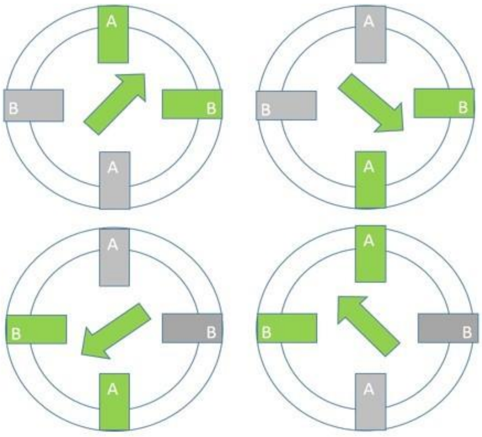
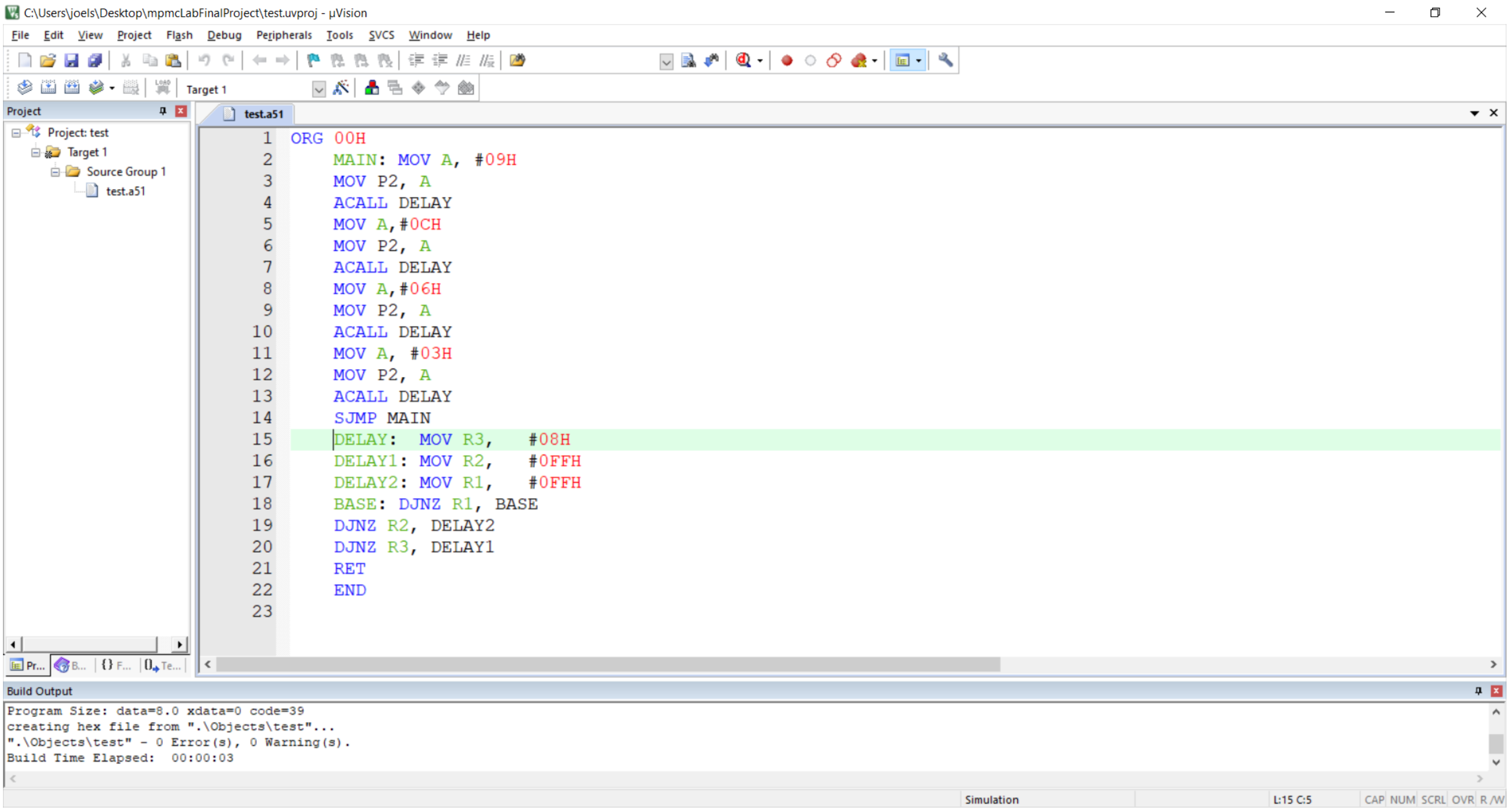


Fig2 – Two phase on – full step

The above image shows accurate diagram of wave drive (left) and full drive (right) execution modes.

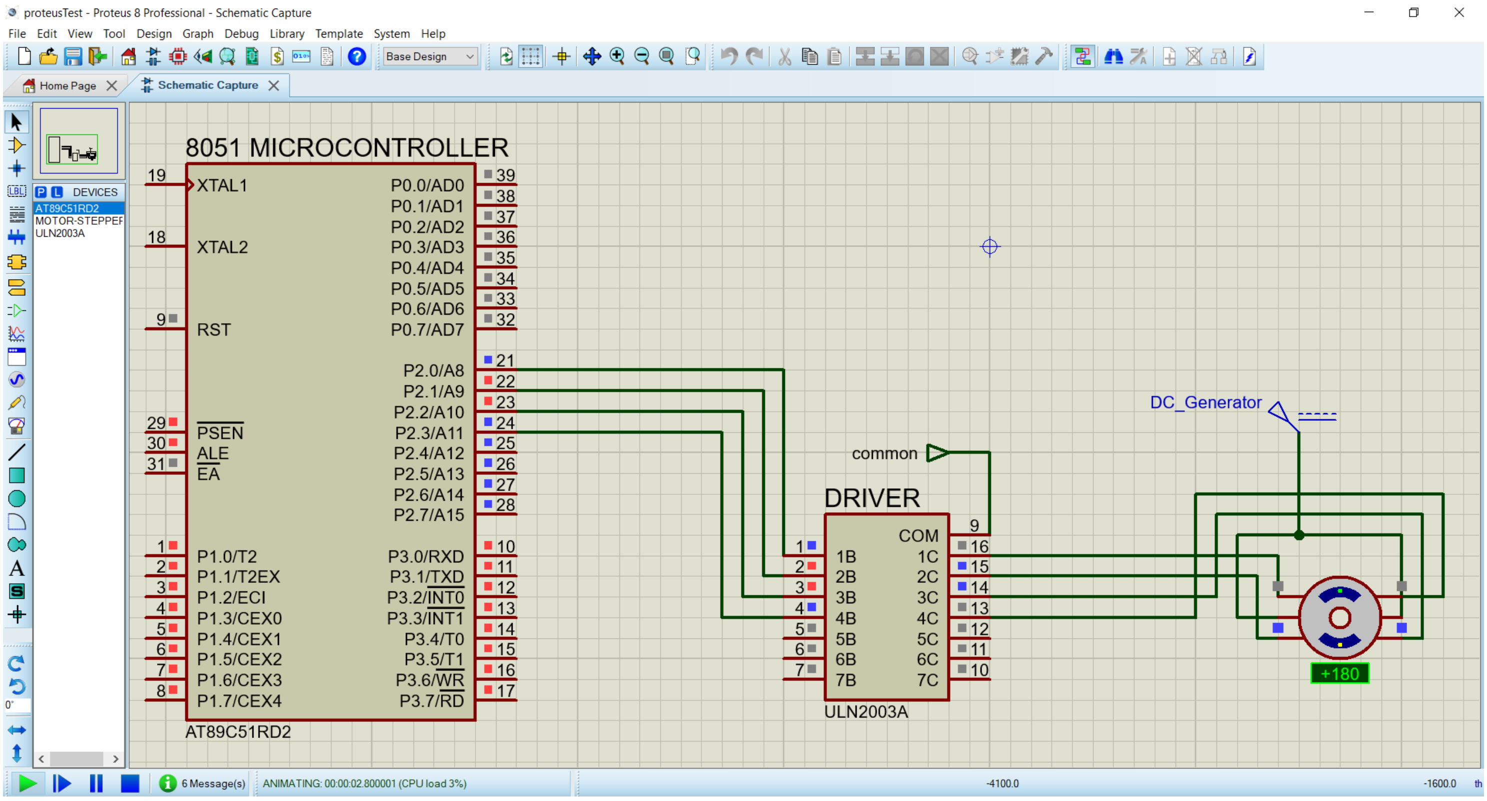
Output:


C:\Users\joels\Desktop\mpmcLabFinalProject\test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Project: test
Target 1
Source Group 1
test.a51
Simulation
L:15 C:5
CAP NUM SCRL OVR R/W

```
ORG 00H
    MAIN: MOV A, #09H
    MOV P2, A
    ACALL DELAY
    MOV A,#0CH
    MOV P2, A
    ACALL DELAY
    MOV A,#06H
    MOV P2, A
    ACALL DELAY
    MOV A, #03H
    MOV P2, A
    ACALL DELAY
    SJMP MAIN
    DELAY:  MOV R3,    #08H
    DELAY1: MOV R2,    #0FFH
    DELAY2: MOV R1,    #0FFH
    BASE: DJNZ R1, BASE
    DJNZ R2, DELAY2
    DJNZ R3, DELAY1
    RET
    END
```

Build Output

```
Program Size: data=8.0 xdata=0 code=39
creating hex file from ".\Objects\test"...
".\Objects\test" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:03
```



6 Message(s)

ANIMATING: 00:00:02.800001 (CPU load 3%)

THANK YOU !!