DIGITAL ASSIGNMENT 3
FIBONACCI,PERMUTATIONS(ASSEMBLY LANGUAGE)

MICROPROCESSOR AND INTERFACING(L51-L52)[MRS SHOBHA REKH]

SEPTEMBER 14, 2022

ANIRUDH VADERA
20BCE2940

QUESTION:

The program should be simulated and verified using DOS BOX/EMU 86 before submission

1) **Generate Fibonacci series using assembly language programming of 8086 The program should take the number of terms from the location mentioned in the program**

The submission should be a pdf file consisting of

a. The algorithm/ flowchart implemented in the code

b. The Inputs/outputs given for testing along with their memory location

c. The screen shots of the code / code written in any editor

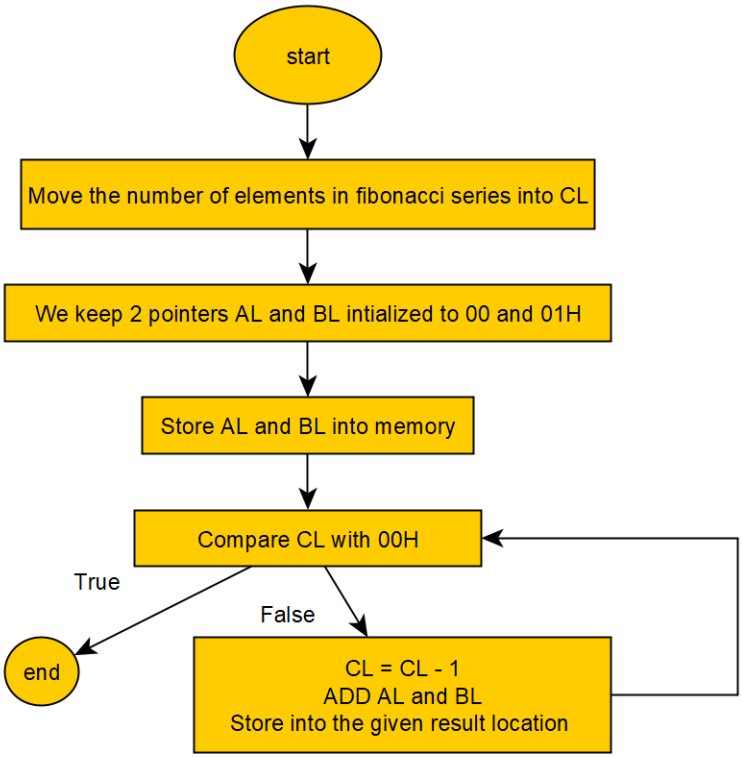
ALGORITHM:

The Fibonacci sequence is generated by adding the (i)th element and the (i-1)th element, and storing it into the (i+1)th position. This holds good given that the 1st and 2nd positions are initialized with 0 and 1 respectively.

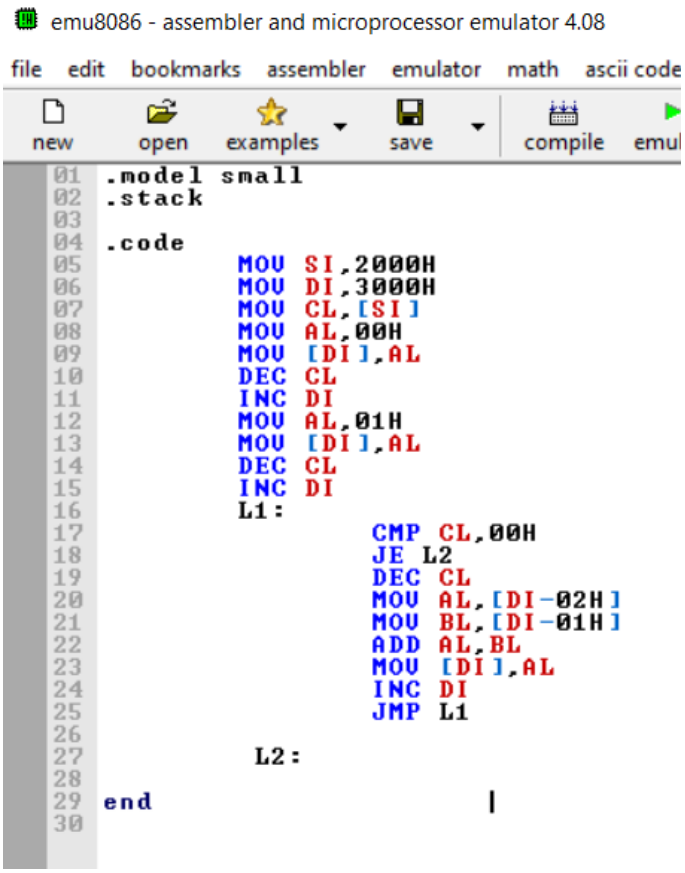
In Assembly Language (Flowchart Explanation):

1. Start
2. First take the number of elements in Fibonacci series in CL from location 2000
3. Then take the 2 pointers AL and BL initialize them by 00H and 01H
4. Store them at 3000H and 30001H (Result memory locaiton)
5. Run a loop until CL == 0
6. Add AL and BL and store into respective result memory location
7. End

FLOWCHART:



CODE SCREENSHOT:




```
.model small
.stack

.code
        MOV SI,2000H
        MOV DI,3000H
        MOV CL,[SI]
        MOV AL,00H
        MOV [DI],AL
        DEC CL
        INC DI
        MOV AL,01H
        MOV [DI],AL
        DEC CL
        INC DI
        L1:
                CMP CL,00H
                JE L2
                DEC CL
                MOV AL,[DI-02H]
                MOV BL,[DI-01H]
                ADD AL,BL
                MOV [DI],AL
                INC DI
                JMP L1

         L2:

end
```

ASSEMBLY LANGUAGE CODE:

```
.model small
.stack

.code
        MOV SI,2000H
        MOV DI,3000H
        MOV CL,[SI]
        MOV AL,00H
        MOV [DI],AL
        DEC CL
        INC DI
        MOV AL,01H
        MOV [DI],AL
        DEC CL
        INC DI
        L1:
                CMP CL,00H
                JE L2
                DEC CL
                MOV AL,[DI-02H]
                MOV BL,[DI-01H]
                ADD AL,BL
                MOV [DI],AL
```

```
        INC DI
        JMP L1

    L2:

end
```

RESULTS:

Command: masm fibonacci.asm

```
C:\>masm fibonacci.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [fibonacci.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51576 + 464968 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>S
```

Command: link calculator.obj

```
C:\>link fibonacci.obj

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [FIBONACCI.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

C:\>S
```

Command: debug calculator.exe

```
C:\>debug fibonacci.exe
-S
```

INPUT:

At Location 2000H: **08H (Fibonacci Series till 8 numbers)**

Giving Inputs from location 2000H

Given Input: 08H

```
C:\>debug calculator.exe
-e ds:2500
075A:2500  FF.01

-e ds:2000
075A:2000  7F.0C   04.DE

-S_
```

Expected Output: 00 01 01 02 03 05 08 0D

OUTPUT:

ScreenShot of process in middle:

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...

AX=FF01  BX=0000  CX=0006  DX=0000  SP=0400  BP=0000  SI=2000  DI=3002
DS=075A  ES=075A  SS=076D  CS=076A  IP=0016   NV UP EI PL NZ NA PO NC
076A:0016 80F900        CMP     CL,00

AX=FF01  BX=0000  CX=0006  DX=0000  SP=0400  BP=0000  SI=2000  DI=3002
DS=075A  ES=075A  SS=076D  CS=076A  IP=0019   NV UP EI PL NZ NA PE NC
076A:0019 740F          JZ      002A

AX=FF01  BX=0000  CX=0006  DX=0000  SP=0400  BP=0000  SI=2000  DI=3002
DS=075A  ES=075A  SS=076D  CS=076A  IP=001B   NV UP EI PL NZ NA PE NC
076A:001B FEC9          DEC     CL

AX=FF01  BX=0000  CX=0005  DX=0000  SP=0400  BP=0000  SI=2000  DI=3002
DS=075A  ES=075A  SS=076D  CS=076A  IP=001D   NV UP EI PL NZ NA PE NC
076A:001D 8A45FE        MOV     AL,[DI-02]                      DS:3000=00

AX=FF00  BX=0000  CX=0005  DX=0000  SP=0400  BP=0000  SI=2000  DI=3002
DS=075A  ES=075A  SS=076D  CS=076A  IP=0020   NV UP EI PL NZ NA PE NC
076A:0020 8A5DFF        MOV     BL,[DI-01]                      DS:3001=01

AX=FF00  BX=0001  CX=0005  DX=0000  SP=0400  BP=0000  SI=2000  DI=3002
DS=075A  ES=075A  SS=076D  CS=076A  IP=0023   NV UP EI PL NZ NA PE NC
076A:0023 02C3          ADD     AL,BL
-S
```

Output Stored at location 3000H

```
-d ds:3000
075A:3000  00 01 01 02 03 05 08 0D-E3 D1 E3 8B 36 7E 21 FF   ............6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
-S_
```

```
-d ds:3000
075A:3000  00 01 01 02 03 05 08 0D-
```

Hence the Output is verified

QUESTION:

The program should be simulated and verified using DOS BOX/EMU 86 before submission

2) Find the factorial of a number using 8086
The number should be taken from a memory location

The submission should be a pdf file consisting of

a. The algorithm/ flowchart implemented in the code

b. The Inputs/outputs given for testing along with their memory location

c. The screen shots of the code / code written in any editor

ALGORITHM:

If the Given Number is a 16-bit number, the AX register is automatically used as the second parameter and the product is stored in the DX:AX register pair. This means that the DX register holds the high part and the AX register holds the low part of a 32-bit number.

In order to get the factorial of the number we follow the simple for loop shown below.

```
Let the number be N

Ans = 1

For(int i=1,i<n+1,i++){

        Ans = Ans*i

}

Print(Ans)
```

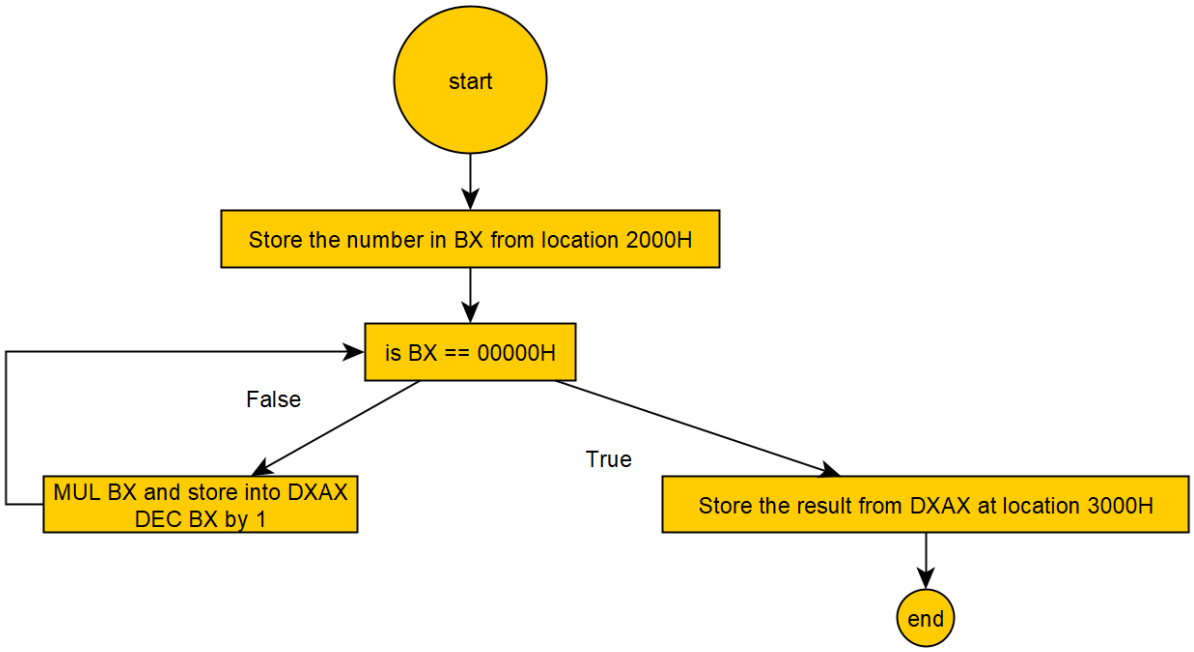
We simply multiply all the numbers from 1 to N .

In Assembly Language (Flowchart Explanation):

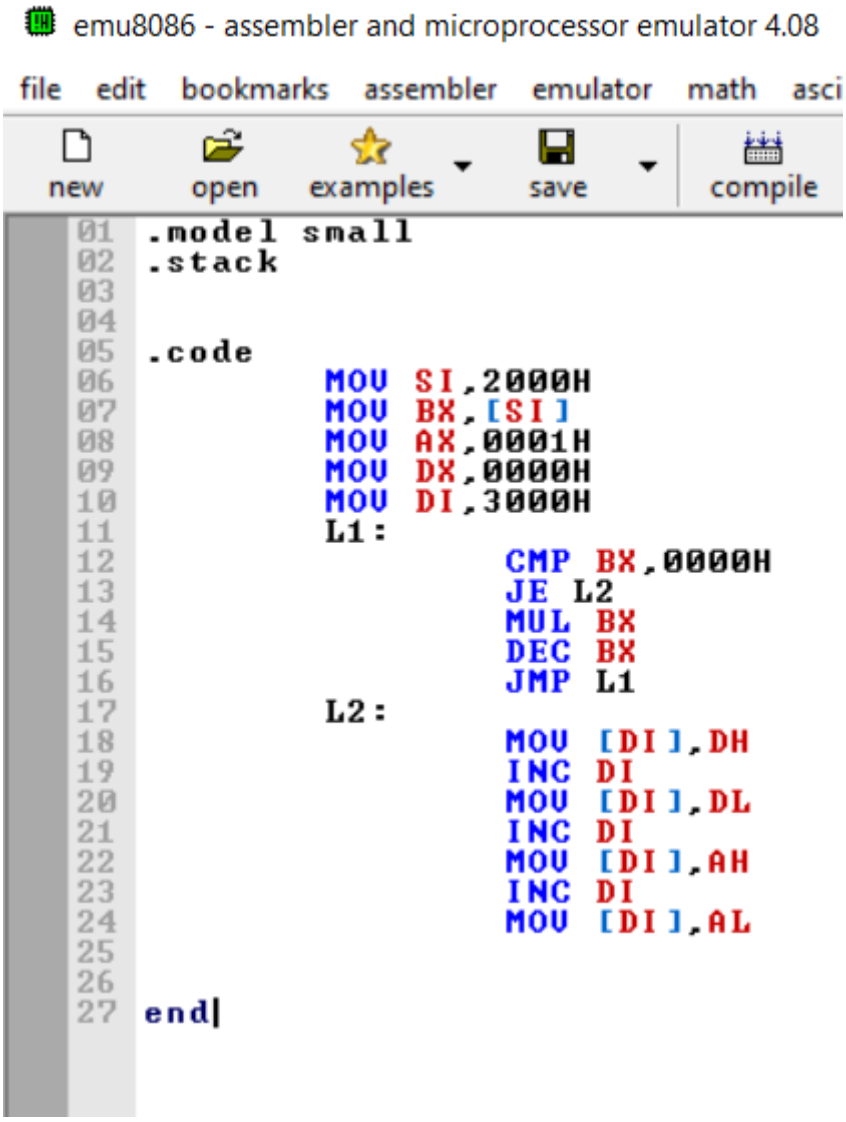
We work with 16 bit inputs here

1. Start
2. Get the number whose factorial is to be found into BX from location 2000
3. We first clear DX variable with 0000H and AX with 0001H because we multiply iteratively in AX
4. We run a loop until BX becomes equal to 0000H
5. If equal to 0000H we store the result at location 3000 stored in DXAX
6. Else we run a loop we multiply BX and store result in AX
7. Then we decrement the value of BX by 1
8. Hence we get ans = n*n-1*n-2.....1
9. End

FLOWCHART:



CODE SCREENSHOT:



ASSEMBLY LANGUAGE CODE:

```
.model small
.stack


.code
        MOV SI,2000H
        MOV BX,[SI]
```

```
        MOV AX,0001H
        MOV DX,0000H
        MOV DI,3000H
        L1:
            CMP BX,0000H
            JE L2
            MUL BX
            DEC BX
            JMP L1
        L2:
            MOV [DI],DH
            INC DI
            MOV [DI],DL
            INC DI
            MOV [DI],AH
            INC DI
            MOV [DI],AL


end
```

RESULTS:

Command: masm factorail.asm

```
C:\>masm factorial.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [factorial.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51652 + 464892 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>S
```

Command: link factorial.obj

```
C:\>link factorial.obj

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [FACTORIAL.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

C:\>S
```

Command: debug factorial.exe

```
C:\>debug factorial.exe
-S
```

INPUT:

At Location 2000H: **0005H (Expected Output 78(120 in decimal))**

Giving Inputs from location 2000H

Given Input: 0005H

```
C:\>debug factorial.exe
-e ds:2000
075A:2000  7F.05   04._
```

Expected Output: 78H at location 3000H

OUTPUT:

ScreenShot of process in middle:

```
DOSBox 0.74-3, Cpu speed:    3000 cycles, Frameskip  0, Progra...

AX=0405  BX=0404  CX=0023  DX=0000  SP=0400  BP=0000  SI=2000  DI=3000
DS=075A  ES=075A  SS=076D  CS=076A  IP=0011   NV UP EI PL NZ NA PO NC
076A:0011 7405          JZ      0018

AX=0405  BX=0404  CX=0023  DX=0000  SP=0400  BP=0000  SI=2000  DI=3000
DS=075A  ES=075A  SS=076D  CS=076A  IP=0013   NV UP EI PL NZ NA PO NC
076A:0013 F7E3          MUL     BX

AX=2414  BX=0404  CX=0023  DX=0010  SP=0400  BP=0000  SI=2000  DI=3000
DS=075A  ES=075A  SS=076D  CS=076A  IP=0015   OV UP EI PL NZ NA PO CY
076A:0015 4B            DEC     BX

AX=2414  BX=0403  CX=0023  DX=0010  SP=0400  BP=0000  SI=2000  DI=3000
DS=075A  ES=075A  SS=076D  CS=076A  IP=0016   NV UP EI PL NZ NA PE CY
076A:0016 EBF6          JMP     000E

AX=2414  BX=0403  CX=0023  DX=0010  SP=0400  BP=0000  SI=2000  DI=3000
DS=075A  ES=075A  SS=076D  CS=076A  IP=000E   NV UP EI PL NZ NA PE CY
076A:000E 83FB00        CMP     BX,+00

AX=2414  BX=0403  CX=0023  DX=0010  SP=0400  BP=0000  SI=2000  DI=3000
DS=075A  ES=075A  SS=076D  CS=076A  IP=0011   NV UP EI PL NZ NA PE NC
076A:0011 7405          JZ      0018
-S
```

Output Stored at location 3000H

```
-d ds:3000
075A:3000  00 00 00 78 8B 5E FE D1-E3 D1 E3 8B 36 7E 21 FF   ...x.^......6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
-S
```

```
-d ds:3000
075A:3000  00 00 00 78
```

Value stored from DXAX

The ans is 0000 078CH

Hence the Output is verified

QUESTION:

The program should be simulated and verified using DOS BOX/EMU 86 before submission

3) Find the permutations for n objects for r selection
The value of n and r are taken from a memory location

The submission should be a pdf file consisting of

a. The algorithm/ flowchart implemented in the code

b. The Inputs/outputs given for testing along with their memory location

c. The screen shots of the code / code written in any editor

ALGORITHM:

For Factorial:

If the Given Number is a 16-bit number, the AX register is automatically used as the second parameter and the product is stored in the DX:AX register pair. This means that the DX register holds the high part and the AX register holds the low part of a 32-bit number.

In order to get the factorial of the number we follow the simple for loop shown below.

```
Let the number be N

Ans = 1

For(int i=1,i<n+1,i++){

        Ans = Ans*i

}

Print(Ans)
```

We simply multiply all the numbers from 1 to N .

For Permutation:

Permutations of n objects for r selection is given by.

P(n,r) = (n!)/(n-r)!

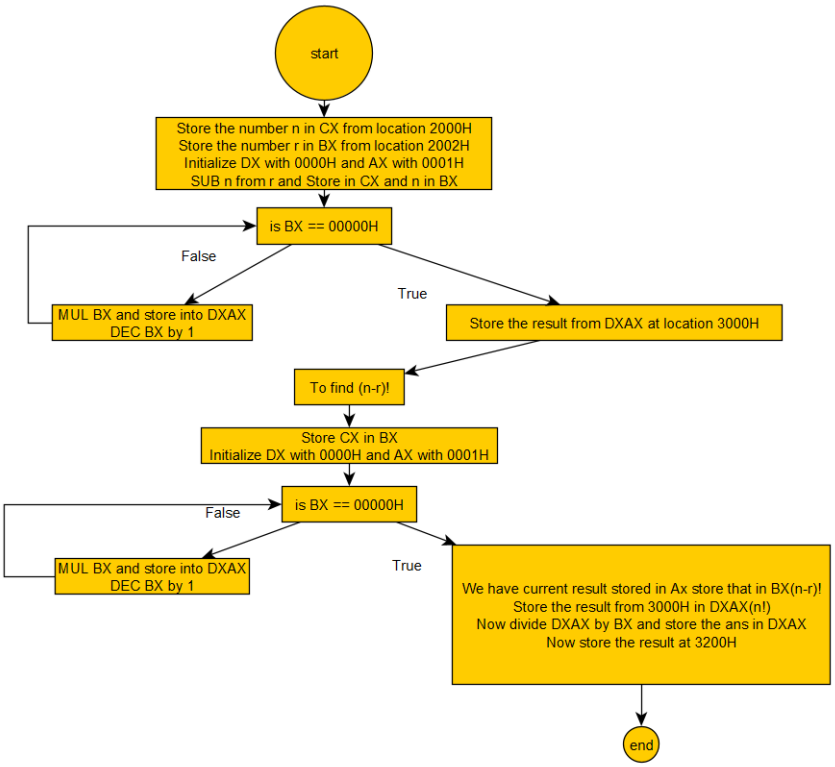
We can first calculate the value of n! and then divide it by (n-r)! to get the required ans.

In Assembly Language (Flowchart Explanation):

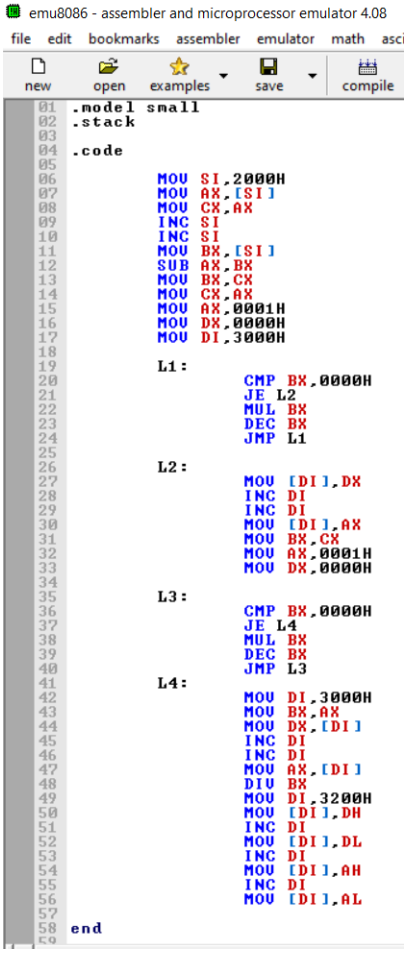
We work with 16 bit inputs here

1. Start
2. We first store the number n from location 2000H in CX
3. We then store the number r from location 2002H in BX
4. We initialize DX with 0000H and AX with 0001H for multiplication and factorial calculation
5. We now store n in BX and calculate n-r and store in CX
6. We then calculate n! using the program above
7. We run a for loop until BX == 0000H
8. If equal to 0000H we store the result at location 3000 stored in DXAX
9. Else we run a loop we multiply BX and store result in AX
10. Then we decrement the value of BX by 1
11. Hence we get ans = n*n-1*n-2.....1
12. We store the value of n! (in DXAX) at location 3000H
13. We then calculate (n-r)! which was stored in CX
14. We again move CX into BX and repeat steps 7 to 11
15. We then now store the ans(n-r)! we get in BX
16. And then take the value of n! from location 3000H into DXAX
17. We then divide BX from DXAX and store result in DX(remaineder)AX(quotient)
18. We finally store the result at location 3200H
19. End

FLOWCHART:



CODE SCREENSHOT:




```
.model small
.stack

.code

        MOV SI,2000H
        MOV AX,[SI]
        MOV CX,AX
        INC SI
        INC SI
        MOV BX,[SI]
        SUB AX,BX
        MOV BX,CX
        MOV CX,AX
        MOV AX,0001H
        MOV DX,0000H
        MOV DI,3000H

        L1:
                CMP BX,0000H
                JE L2
                MUL BX
                DEC BX
                JMP L1

        L2:
                MOV [DI],DX
                INC DI
                INC DI
                MOV [DI],AX
                MOV BX,CX
                MOV AX,0001H
                MOV DX,0000H

        L3:
                CMP BX,0000H
                JE L4
                MUL BX
                DEC BX
                JMP L3
        L4:
                MOV DI,3000H
                MOV BX,AX
                MOV DX,[DI]
                INC DI
                INC DI
                MOV AX,[DI]
                DIV BX
                MOV DI,3200H
                MOV [DI],DH
                INC DI
                MOV [DI],DL
                INC DI
                MOV [DI],AH
                INC DI
                MOV [DI],AL

end
```

ASSEMBLY LANGUAGE CODE:

```
.model small
.stack

.code

        MOV SI,2000H
        MOV AX,[SI]
        MOV CX,AX
        INC SI
        INC SI
        MOV BX,[SI]
        SUB AX,BX
        MOV BX,CX
        MOV CX,AX
        MOV AX,0001H
        MOV DX,0000H
        MOV DI,3000H

        L1:
            CMP BX,0000H
            JE L2
            MUL BX
            DEC BX
```

```
        JMP L1

L2:
        MOV [DI],DX
        INC DI
        INC DI
        MOV [DI],AX
        MOV BX,CX
        MOV AX,0001H
        MOV DX,0000H

L3:
        CMP BX,0000H
        JE L4
        MUL BX
        DEC BX
        JMP L3
L4:
        MOV DI,3000H
        MOV BX,AX
        MOV DX,[DI]
        INC DI
        INC DI
        MOV AX,[DI]
        DIV BX
```

```
MOV DI,3200H
MOV [DI],DH
INC DI
MOV [DI],DL
INC DI
MOV [DI],AH
INC DI
MOV [DI],AL

end
```

RESULTS:

Command: masm nPr.asm

```
C:\>masm nPr.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [nPr.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51682 + 464862 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>S_
```

Command: link nPr.obj

```
C:\>link nPr.obj

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [NPR.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

C:\>S
```

Command: debug nPr.exe

```
C:\>debug nPr.exe
-S_
```

INPUT:

At Location 2000H: **0005H (Taking value of n)**

At Location 2001H: **0003H (Taking value of r)**

Giving Inputs from location 2000H

Given Input: 0005H(n) and 0003H(r)

```
C:\>debug nPr.exe
-e ds:2000
075A:2000  7F.05   04.00   00.03   75.00   42.
```

Expected Output: ($\frac{5!}{(5-3)!}$) 3CH at location 3200H

OUTPUT:

ScreenShot of process in middle:

```
DOSBox 0.74-3, Cpu speed:    3000 cycles, Frameskip  0, Progra...
-t

AX=000E  BX=0002  CX=0002  DX=075A  SP=03FC  BP=0000  SI=067F  DI=3203
DS=075A  ES=075A  SS=0770  CS=076A  IP=006C   NV UP EI PL NZ NA PO NC
076A:006C E8C148        CALL    4930
-t

AX=000E  BX=0002  CX=0002  DX=075A  SP=03FA  BP=0000  SI=067F  DI=3203
DS=075A  ES=075A  SS=0770  CS=076A  IP=4930   NV UP EI PL NZ NA PO NC
076A:4930 55            PUSH    BP
-t

AX=000E  BX=0002  CX=0002  DX=075A  SP=03F8  BP=0000  SI=067F  DI=3203
DS=075A  ES=075A  SS=0770  CS=076A  IP=4931   NV UP EI PL NZ NA PO NC
076A:4931 8BEC          MOV     BP,SP
-d ds:3200
075A:3200  00 00 00 3C 50 2B C0 50-E8 95 11 83 C4 04 E8 E7   ...<P+.P........
075A:3210  2B 3D 61 00 74 08 3D 62-00 74 06 EB 07 90 E8 67   +=a.t.=b.t.....g
075A:3220  E6 E8 64 E6 2B C0 50 B8-03 00 50 8D 86 7C FF 50   ..d.+.P...P..|.P
075A:3230  E8 8D E2 83 C4 06 89 46-FC 89 56 FE 0B C2 75 0A   .......F..V...u.
075A:3240  B8 59 04 50 E8 39 17 83-C4 02 C4 5E FC 26 8A 47   .Y.P.9.....^.&.G
075A:3250  05 8B 1E 22 21 8B 3E FE-26 88 01 8B F3 D1 E6 8B   ..."!.>.&.......
075A:3260  1E 38 21 8B 7E FC 26 8B-45 08 89 00 8B 1E 08 27   .8!.~.&.E......'
075A:3270  26 8B 45 06 89 00 8B 1E-22 21 D1 E3 D1 E3 8B 3E   &.E....."!.....>
-SSS
```

Output Stored at location 3200H

```
-d ds:3200
075A:3200  00 00 00 3C 50 2B C0 50-E8 95 11 83 C4 04 E8 E7   ...<P+.P........
075A:3210  2B 3D 61 00 74 08 3D 62-00 74 06 EB 07 90 E8 67   +=a.t.=b.t.....g
075A:3220  E6 E8 64 E6 2B C0 50 B8-03 00 50 8D 86 7C FF 50   ..d.+.P...P..¦.P
075A:3230  E8 8D E2 83 C4 06 89 46-FC 89 56 FE 0B C2 75 0A   .......F..V...u.
075A:3240  B8 59 04 50 E8 39 17 83-C4 02 C4 5E FC 26 8A 47   .Y.P.9.....^.&.G
075A:3250  05 8B 1E 22 21 8B 3E FE-26 88 01 8B F3 D1 E6 8B   ..."!.>.&.......
075A:3260  1E 38 21 8B 7E FC 26 8B-45 08 89 00 8B 1E 08 27   .8!.~.&.E......'
075A:3270  26 8B 45 06 89 00 8B 1E-22 21 D1 E3 D1 E3 8B 3E   &.E....."!.....>
-S
```

```
-d ds:3200
075A:3200  00 00 00 3C
```

Value stored from DXAX

The ans is (remainder)0000 003CH(quotient)

Hence the Output is verified