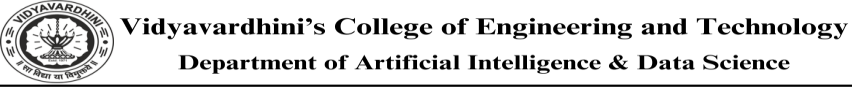


|  |  |
| --- | --- |
| **Name:** | BARI ANKIT VINOD |
| **RollNo:** | 65 |
| **Class/Sem:** | SE/IV |
| **ExperimentNo.:** | 2A |
| **Title:** | ProgramtoperformmultiplicationwithoutusingMULinstruction |
| **DateofPerformance:** | 24/01/24 |
| **DateofSubmission:** | 31/01/24 |
| **Marks:** |  |
| **SignofFaculty:** |  |



**Aim: Program for multiplication without using the multiplication instruction.**

**Theory:**

Inthemultiplicationprogram,wemultiplythetwonumberswithoutusingthedirectinstructions MUL.Herewecansuccessiveadditionmethodstogettheproductoftwonumbers.Forthat,inone registerwewilltakemultiplicandsothatwecanaddmultiplicanditselftillthemultiplierstoredin anotherregisterbecomeszero.

**ORG100H:**

Itisacompilerdirective.Ittellsthecompilerhowtohandlesourcecode.Ittellsthecompiler thattheexecutablefilewillbeloadedattheoffsetof100H(256bytes.)

**INT21H:**

TheinstructionINT21Htransferscontroltotheoperatingsystem,toasubprogramthathandles I/Ooperations.

**MUL:**MULSource.

ThisinstructionmultipliesanunsignedbytefromsomesourcetimesanunsignedbyteintheAL registeroranunsignedwordfromsomesourcetimesanunsignedwordintheAXregister.

Source:Register,MemoryLocation.

WhenabyteismultipliedbythecontentofAL,theresult(product)isputinAX.A16-bitdestination isrequiredbecausetheresultofmultiplyingan8-bitnumberbyan8-bitnumbercanbeaslargeas 16-bits.TheMSBoftheresultisputinAHandtheLSBoftheresultisputinAL.

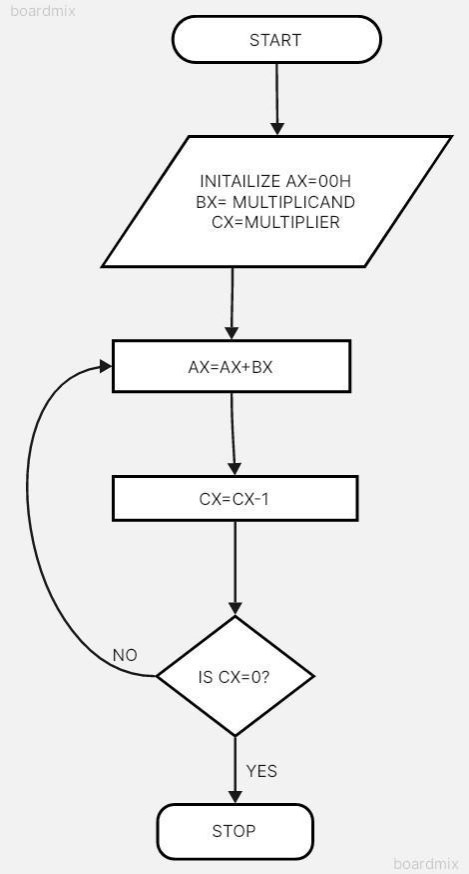
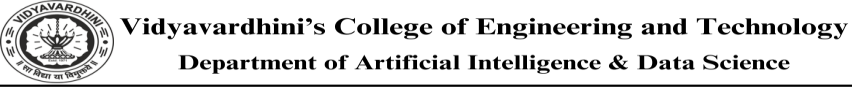
WhenawordismultipliedbythecontentsofAX,theproductcanbeaslargeas32bits.TheMSB oftheresultisputintheDXregisterandtheLSBoftheresultisputintheAXregister.

MULBH;multiplyALwithBH;resultinAX.

Algorithm: 1.Start.

2.SetAX=00H,BX=Multiplicand,CX=Multiplier3AddthecontentofAX andBX.

4.DecrementcontentofCX. 5.Repeatsteps3and4tillCX=0. 6.Stop.



Flowchart:

**Code :**

.model small

.stack 100h

.data

num1 dw 5

num2 dw 3

result dw ?

.code

main proc

mov ax, @data

mov ds, ax

mov ax, num1

mov bx, num2

mov cx, 0

mov result, 0

mul\_loop:

add result, ax

inc cx

cmp cx, bx

jl mul\_loop

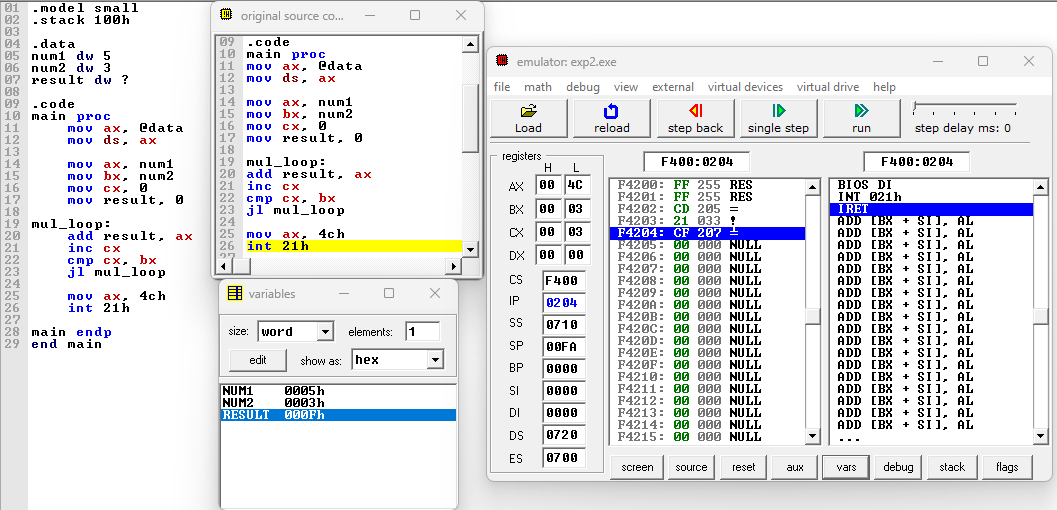
mov ax, 4ch

int 21h

main endp

end main

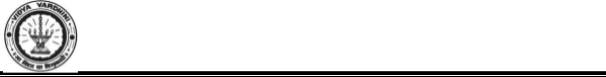
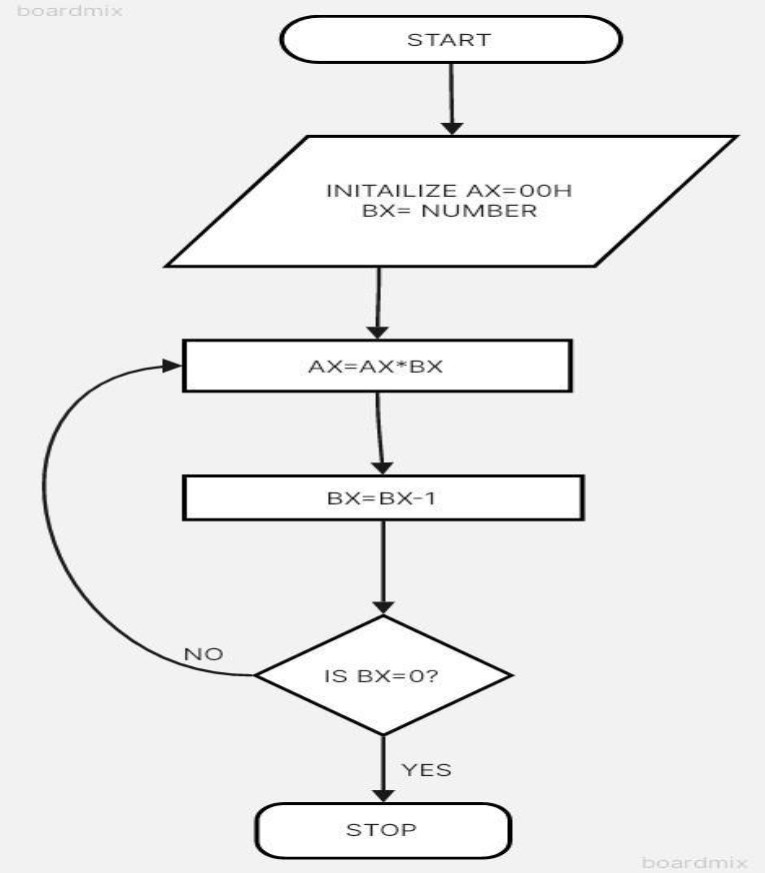
**Output :**



**Conclusion :**

In conclusion, the development and implementation of a program to perform multiplication without relying on the MUL instruction have showcased the ingenuity and versatility of computational techniques. Through a combination of logical operations, bit manipulation, and iterative processes, we have achieved a method to efficiently carry out multiplication tasks. This endeavor underscores the significance of understanding the fundamental principles of computer architecture and algorithm design. By exploring alternative approaches to conventional operations, we not only broaden our understanding of computational mechanisms but also open avenues for innovation in optimizing performance and resource utilization. As we continue to push the boundaries of what is possible within the realm of computing, endeavors like this serve as reminders of the creativity and problem-solving prowess inherent in the field.

|  |  |
| --- | --- |
| **Name:** | BARI ANKIT VINOD |
| **RollNo:** | 65 |
| **Class/Sem:** | SE/IV |
| **ExperimentNo.:** | 2B |
| **Title:** | Programforcalculatingfactorialusingassemblylanguage |
| **DateofPerformance:** | 24/01/24 |
| **DateofSubmission:** | 31/01/24 |
| **Marks:** |  |
| **SignofFaculty:** |  |

Vidyavardhini’sCollegeofEngineering&Technology DepartmentofArtificialIntelligenceandDataScience(AI&DS)

Aim: Program to calculate the Factorial of a number.

**Theory:**

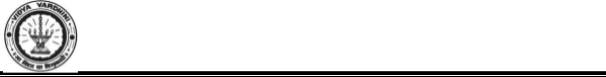
Tocalculatethefactorialofanynumber,weuseMULinstruction.Here,initially,weinitializethe firstregisterbyvalue1.Thesecondregisterisinitializedbythevalueofthesecondregister.After multiplication,decrementthevalueofthesecondregisterandrepeatthemultiplyingsteptillthesecond registervaluebecomeszero.Theresultisstoredinthefirstregister.

Algorithm: 1.Start.

2.SetAX=01H,andBXwiththevaluewhosefactorialwewanttofind. 3.MultiplyAXandBX.

4.DecrementBX=BX-1. 5.Repeatsteps3and4tillBX=0. 6.Stop.

Flowchart:

Vidyavardhini’sCollegeofEngineering&Technology DepartmentofArtificialIntelligenceandDataScience(AI&DS)

**Code :**

.model small

.stack 100h

.data

num dw 5

result dw ?

.code

main proc

mov ax, @data

mov ds, ax

mov ax, num

mov bx, ax

mov cx, 1

mov result, 1

factorial\_loop:

mul bx

dec bx

cmp bx, 0

jnz factorial\_loop

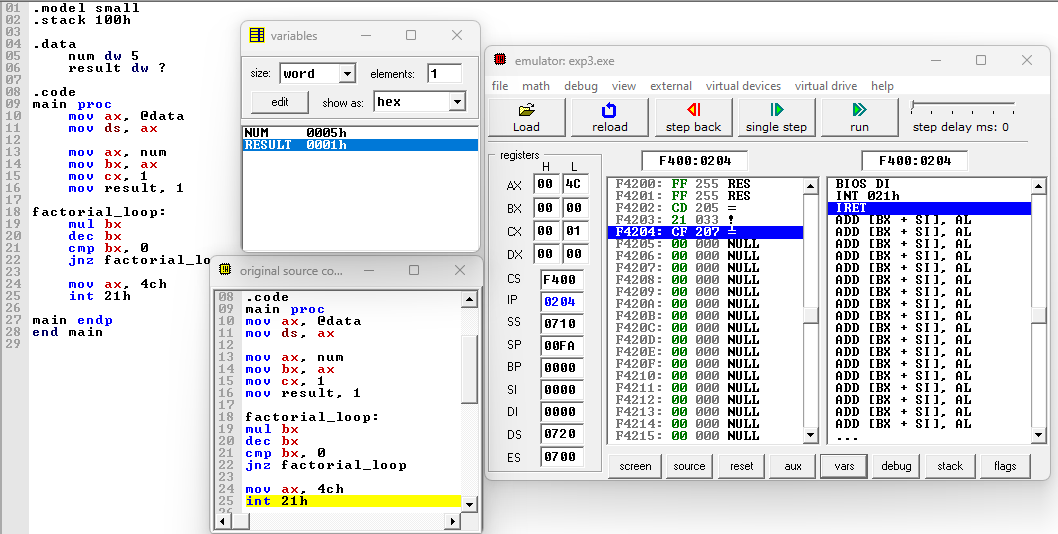
mov ax, 4ch

int 21h

main endp

end main

**Output :**



**Conclusion :**

In conclusion, the program for calculating factorial using assembly language demonstrates the power and efficiency of low-level programming in solving mathematical problems. By delving into the intricacies of processor architecture and instruction sets, we've crafted a solution that efficiently computes factorials, showcasing the direct manipulation of hardware resources to achieve desired outcomes. Through this exercise, we've gained insights into the inner workings of computers, honed our problem-solving skills, and deepened our understanding of assembly language programming. As we conclude this endeavor, let us carry forward the lessons learned and continue exploring the vast landscape of low-level programming, where precision and optimization converge to unlock new realms of possibility in software development.