**EXPERIMENT NO- 1**

**AIM:** WAP to perform 8bit/16 bit arithmetic operations and display the contents of flag register.

**Resource Required**: P-IV and above RAM 128MB, Dot Matrix Printer, Emu 8086, MASM 611/ TASM, Turbo C/C++, Printer, Printout Stationary.

**THEORY:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Assemble language has two types of statements:   1. **Executable**: Instruction that are translated into machine code by the assembler 2. **Assembler Directives**:  * Statements that direct the assembler to do some special task. * No machine language code is produced for these statements. * Their main task is to inform the assembler about the start/ end of a segment, procedure or program, procedure or program, to reserve appropriate space from data storage etc. * Some of the assembler directives are listed below:   .DB (define byte): used to define a byte variable. Ex SUM DB 0. Assembler reserves 1 byte of memory for the variable SUM and initializes it to 0.  .DW (Define word, 16 bit): used to define a word type variable.  .DD (Define double word, 32 bit): used to define a double word type variable.  .DQ (Quad Word): used to define a quad word type variable.  **Instructions:**  **MOV Destination Source**  Move a byte/word from the source to the destination specified in the instruction.  Source: Register, Memory location, immediate number  Destination: Register, Memory location  Both source & destination cannot be memory locations.  MOVE Register, Register  MOV Memory location, Register  MOV Register, Memory location  MOV Register data.  **ADD Destination, Source**  Adds the source to the destination & stores the result back in the destination.  Source: Register, Memory Location, Immediate number  Destination: Register  Both source & destination have to be of the same size.  ADD Register, Register  ADD Memory location, Register  ADD Register, Memory location  ADD Register, data.  **ADC Destination, Source**  Adds the source to the destination & stores the result the with carry back in the destination.  Source: Register, Memory location, immediate number.  Destination: Register  Both source & destination have to be of the same size  ADD Register, Register  ADD Memory location, Register  ADD Register, Memory location  ADD Register, data.  **SUB/SBB Destination, source**  It is similar to ADD/ADC expect that it does subtraction.  **DAA (Decimal adjust for addition)**  It makes the result in BCD from after BCD addition is performed.  It works only on AL register.  If D3-DO (Lower 4-bit) > 9 then AF is set, Add 06h to AL.  If D7-D4 (Upper 4-bit) > 9 then CF is set, Add 60h to AL.  **DAS (Decimal adjust for subtraction)**  It makes the result in packed BCD from after BCD subtraction is performed  It works only on AL register.  If D3-DO (Lower 4-bit) > 9 then AF is set, Add 06h to AL.  If D7-D4 (Upper 4-bit) > 9 then CF is set, Add 60h to AL.  **MUL source (Unsigned 8/16-bit registers)**  Source: Register, Memory Location  If the source is 8-bit it is multiplied with AL & result is stored in AX  (AH- higher byte, AX- lower byte)  If the source is 16-bit, it is multiplied with AX & result is stored in  DX –AX register (DX-higher byte, AX-lower byte)  MUL affects AF, PF, SF, & ZF.  **DIV source (Unsigned 8/16-bit register-divisior)**  This instruction is used for unsigned division.  Divides a word by a byte or a double word by word.  If divisior is 8-bit then the dividend is in AX register.  After division the quotient is in AL & reminder is in DX  If divisior is 16-bit then the dividend is in DX-AX   |  |  |  | | --- | --- | --- | | **Algorithm** | **:** |  | | 1. **An algorithm for addition of two 16-bit numbers.**   Step 1: Start  Step 2: Initialize data segment  Step 3: Declare two variables that hold the actual data.  Step 4: Initialize code segment  Step 5: Initialize DS register to program  Step 6: move first no. in register (bx)  Step 7: move second no. in register (cx)  Step 8: Perform the addition  Step 9: Stop   1. **An algorithm for subtraction of two 16-bit numbers**.   Step 1: Start  Step 2: Initialize data segment  Step 3: Declare two variables that hold the actual data.  Step 4: Initialize code segment  Step 5: Initialize DS register to program  Step 6: move first no. in register (bx)  Step 7: move second no. in register (cx)  Step 8: Perform the subtraction  Step 9: Stop   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Assemble language has two types of statements:   1. **Executable**: Instruction that are translated into machine code by the assembler 2. **Assembler Directives**:  * Statements that direct the assembler to do some special task. * No machine language code is produced for these statements. * Their main task is to inform the assembler about the start/ end of a segment, procedure or program, procedure or program, to reserve appropriate space from data storage etc. * Some of the assembler directives are listed below:   .DB (define byte): used to define a byte variable. Ex SUM DB 0. Assembler reserves 1 byte of memory for the variable SUM and initializes it to 0.  .DW (Define word, 16 bit): used to define a word type variable.  .DD (Define double word, 32 bit): used to define a double word type variable.  .DQ (Quad Word): used to define a quad word type variable.  **Instructions:**  **MOV Destination Source**  Move a byte/word from the source to the destination specified in the instruction.  Source: Register, Memory location, immediate number  Destination: Register, Memory location  Both source & destination cannot be memory locations.  MOV Register, Register  MOV Memory location, Register  MOV Register, Memory location  MOV Register data.  **ADD Destination, Source**  Adds the source to the destination & stores the result back in the destination.  Source: Register, Memory Location, Immediate number  Destination: Register  Both source & destination have to be of the same size.  ADD Register, Register  ADD Memory location, Register  ADD Register, Memory location  ADD Register, data.  **ADC Destination, Source**  Adds the source to the destination & stores the result the with carry back in the destination.  Source: Register, Memory location, immediate number.  Destination: Register  Both source & destination have to be of the same size  ADD Register, Register  ADD Memory location, Register  ADD Register, Memory location  ADD Register, data.  **SUB/SBB Destination, source**  It is similar to ADD/ADC expect that it does subtraction.  **DAA (Decimal adjust for addition)**  It makes the result in BCD from after BCD addition is performed.  It works only on AL register.  If D3-DO (Lower 4-bit) > 9 then AF is set, Add 06h to AL.  If D7-D4 (Upper 4-bit) > 9 then CF is set, Add 60h to AL.  **DAS (Decimal adjust for subtraction)**  It makes the result in packed BCD from after BCD subtraction is performed  It works only on AL register.  If D3-DO (Lower 4-bit) > 9 then AF is set, Add 06h to AL.  If D7-D4 (Upper 4-bit) > 9 then CF is set, Add 60h to AL.  **MUL source (Unsigned 8/16-bit registers)**  Source: Register, Memory Location  If the source is 8-bit it is multiplied with AL & result is stored in AX  (AH- higher byte, AX- lower byte)  If the source is 16-bit, it is multiplied with AX & result is stored in  DX –AX register (DX-higher byte, AX-lower byte)  MUL affects AF, PF, SF, & ZF.  **IMUL source (Signed 8/16-bit registers)**  Sam as MUL expect that the source is a signed number.  **DIV source (Unsigned 8/16-bit register-divisior)**  This instruction is used for unsigned division.  Divides a word by a byte or a double word by word.  If divisior is 8-bit then the dividend is in AX register.  After division the quotient is in AL & reminder is in DX  If divisior is 16-bit then the dividend is in DX-AX  Register.  **IDIV source (Signed 8/16-bit register-divisior)**  Same as DIV except that the source is a signed number.   |  |  |  | | --- | --- | --- | | **Algorithm** | **:** |  | | 1. **An algorithm for addition of two 16-bit numbers.**   Step 1: Start  Step 2: Initialize data segment  Step 3: Declare two variables that hold the actual data.  Step 4: Initialize code segment  Step 5: Initialize DS register to program  Step 6: move first no. in register (bx)  Step 7: move second no. in register (cx)  Step 8: Perform the addition  Step 9: Stop   1. **An algorithm for addition of two 16-bit numbers**.   Step 1: Start  Step 2: Initialize data segment  Step 3: Declare two variables that hold the actual data.  Step 4: Initialize code segment  Step 5: Initialize DS register to program  Step 6: move first no. in register (bx)  Step 7: move second no. in register (cx)  Step 8: Perform the subtraction  Step 9: Stop   1. **An algorithm for multiplication of two 16-bit numbers.**   Step 1 : Start  Step 2: Initialize data segment  Step 3 : Declare two variables that hold the actual data.  Step 4 : Initialize code segment  Step 5 : Initialize DS register to program  Step 6 : move first no. in register (ax)  Step 7 : move second no. in register (cx)  Step 8 : multiply both the numbers by using word pointer & transfer the result in particular variable.  Step 9 : Stop   1. **An algorithm for division of 16-bit number by 8-bit number.**   Step 1 : Start  Step 2: Initialize data segment  Step 3 : Declare two variables that hold the actual data.  Step 4 : Initialize code segment  Step 5 : Initialize DS register to program  Step 6 : move 16-bit no.in register (al)  Step 7 : move 8-bit no.in register (bl)  Step 8 : perform division operation & store the result in particular variable  Step 9 : Stop | | | | | | | |

**CONCLUSION: Hence we have done arithmetic operation on 2 8/16 bits number in emu8086 using assembly and display the output and flag contents**

**1.Addition of two 16bits**:

Data segment

A dw 0032h

B dw 0013h

Result 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

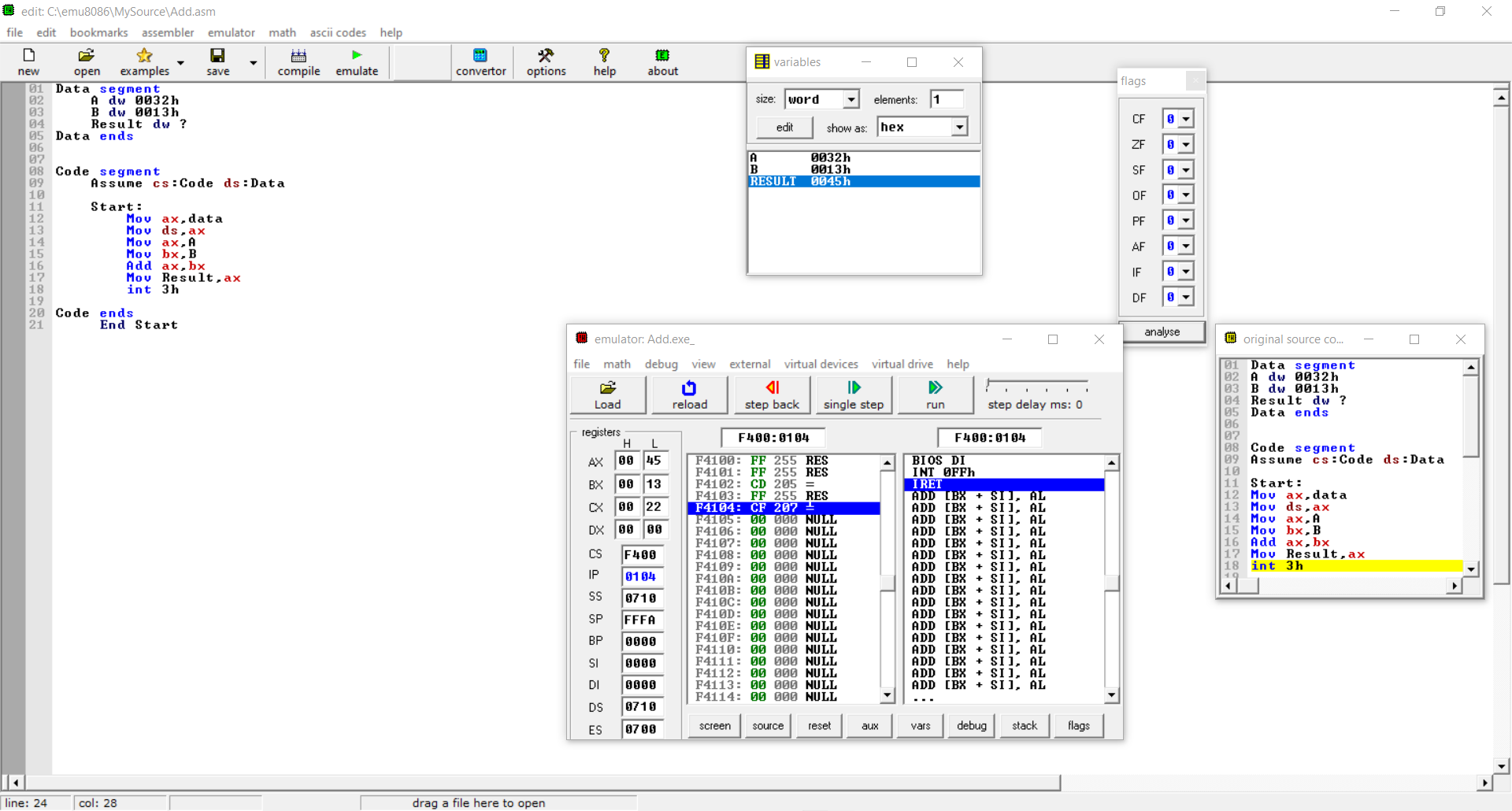
Mov Result,ax

int 3h

Code ends

End Start

**Output:**

****

1. **Subtraction of two 16bits number:**

Data segment

A dw 0032h

B dw 0013h

Result 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

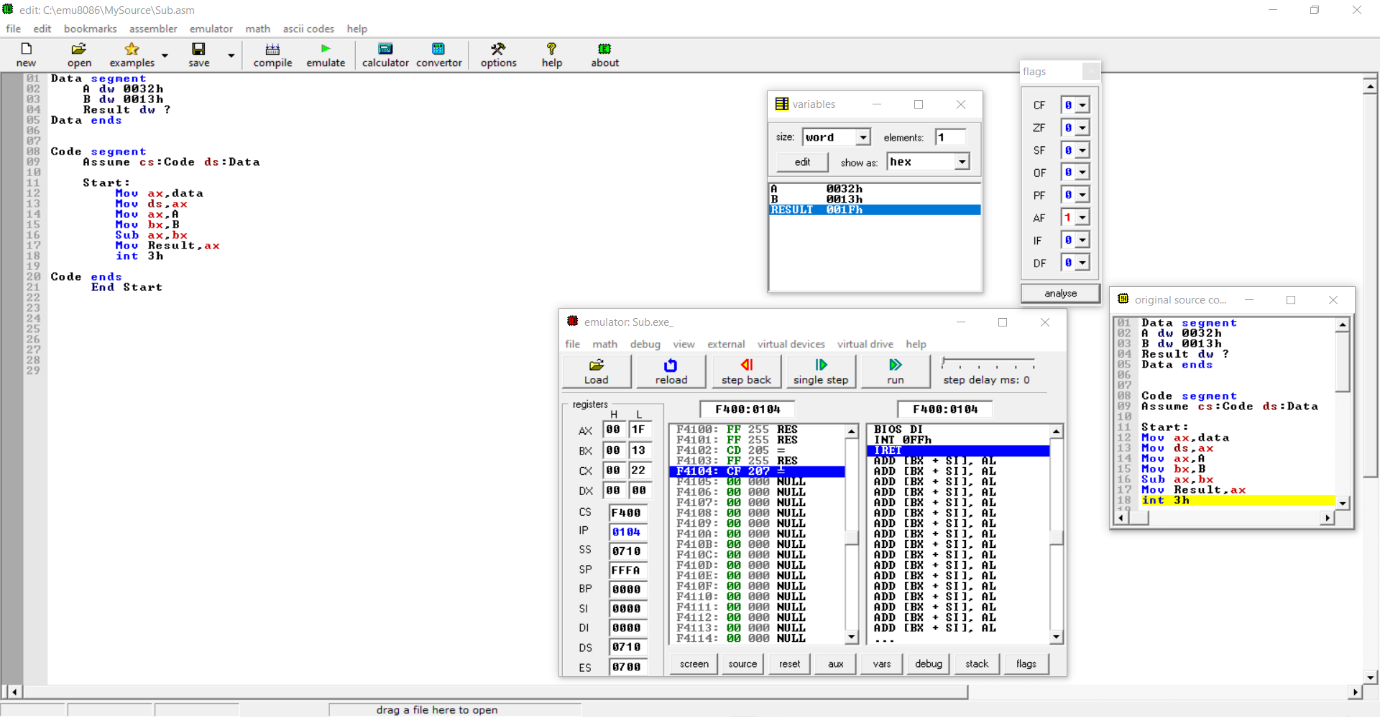
Mov Result,ax

int 3h

Code ends

End Start

**Output** :



1. **Multiplication of two 16bits**:

Data segment

a dw 0013h

b dw 0013h

result dw ?

Data Ends

Code segment

assume cs:Code ds:data

Start:

mov ax,data

mov ds,ax

mov bx,a

mov ax,b

mul bx

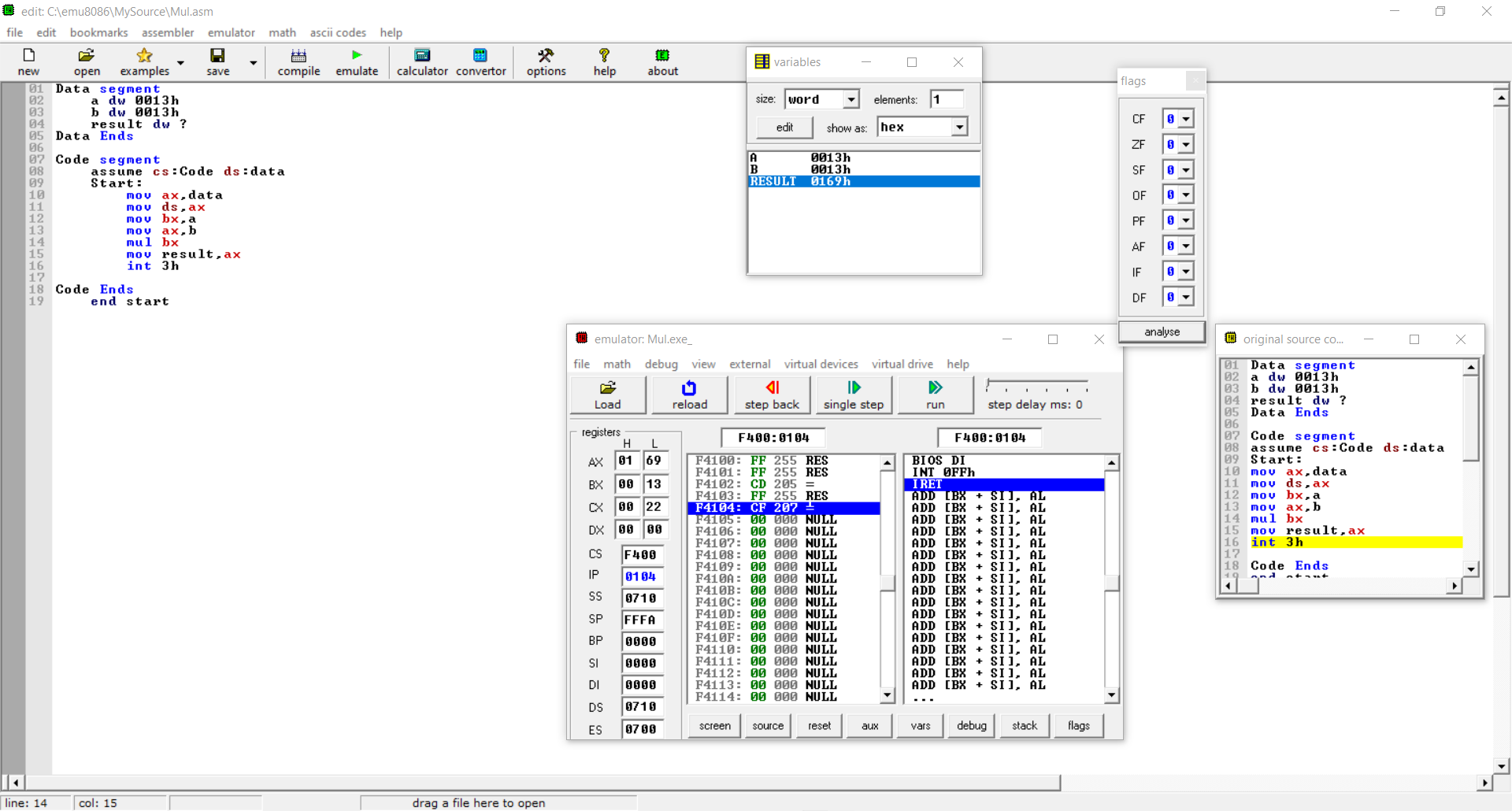
mov result,ax

int 3h

Code Ends

end start

**Output**:



1. **Division of 16bits/8bits**:

Data segment

a dw 0011h

b db 05h

quotient db ?

remainder db ?

Data Ends

Code segment

Assume cs:code ds:data

start:

mov ax,data

mov ds,ax

mov ax,a

mov bl,b

div bl

mov quotient,al

mov remainder,ah

int 3h

Code ends

end start

**Output**:

