# UCS1512 – Microprocessors Lab

# 16 BIT ARITHMETIC OPERATIONS

Exp no : 2 Name: Sreedhar V

Date : 28-08-2020 Reg no: 185001161

# AIM:

To program and execute the 16 bit arithmetic operations like addition, subtraction, multiplication and division in 8086 using an emulator.

# - Bit Addition:

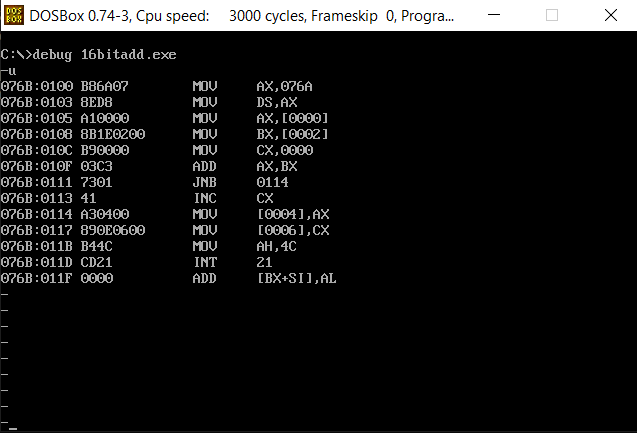
## Algorithm:

* + Program is set to run from any specified memory position.
  + Load data from opr1 to register AX (first number)
  + Load data from opr2 to register BX (second number)
  + Add these two numbers (contents of register AX and register BX)
  + Initialise carry to 0.
  + Jump to final steps if there is no carry.
  + Increment carry.
  + Store additional values to result.
  + Terminate the program.

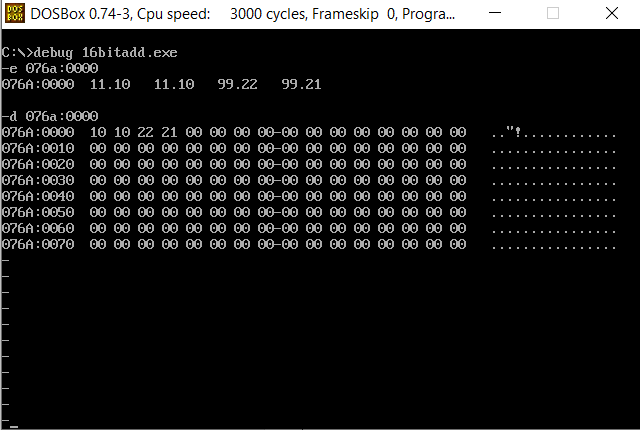
## Program:

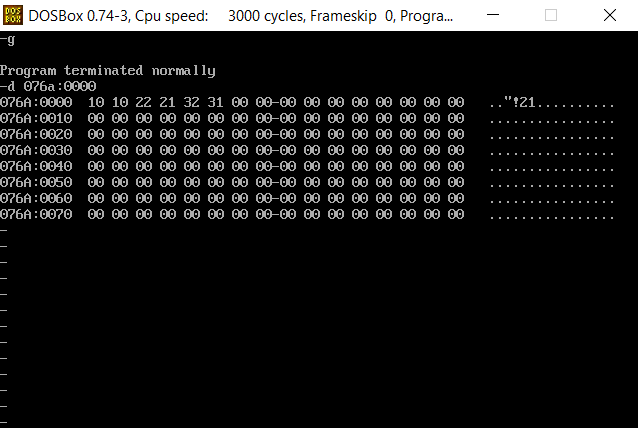
|  |  |
| --- | --- |
| CODE | COMMENT |
| ;Program for adding 2, 16 bit numbers assume cs:code,ds:data  data segment  opr1 dw 1111h  opr2 dw 9999h  result dw 0000H carry dw 0000H  data ends  code segment  org 0100h start:  mov ax,data mov ds,ax  mov ax,opr1 mov bx,opr2 mov cx,0000h add ax,bx  jnc here  inc cx  here:  mov result,ax  mov carry,cx  mov ah,4ch  int 21h  code ends end start | Data segment initialized  opr1 initialised and set to 1111 opr2 initialised and set to 9999  result initialised and set to 0000 carry initialised and set to 0000  Code segment begins  Originating address is set at 0100  Address of data segment moved to ax From ax, transferred to ds  Value of opr1 transferred to ax Value of opr2 transferred to bx  cx is initialised and set to 0 Addition takes palce  Junction created   * If carry is not generated jump to here * Else increment cx   data transferred from ax to result  data transferred from cx to carry  Program terminates |

Unassembled code:



Sample input and output:





Result:

16 bit addition is executed and verified using an emulator.

# - Bit Subtraction:

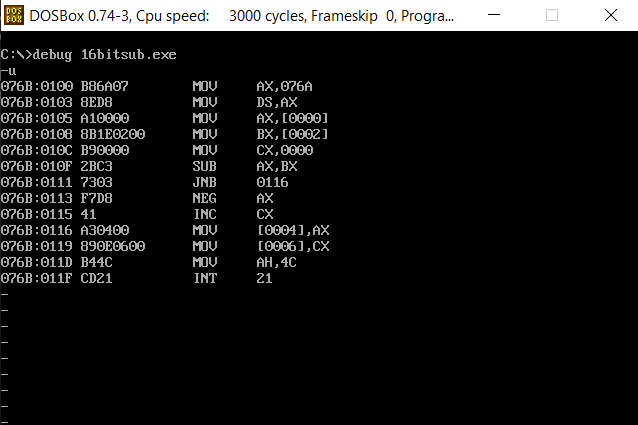
## Algorithm:

* + Program is set to run from any specified memory position.
  + Load data from opr1 to register AX (first number)
  + Load data from opr2 to register BX (second number)
  + Subtract these two numbers (contents of register AX and register BX)
  + Initialise carry to 0.
  + Jump to final steps if there is no carry.
  + Increment carry.
  + And the result is negated.
  + Store answer to result.
  + Terminate the program.

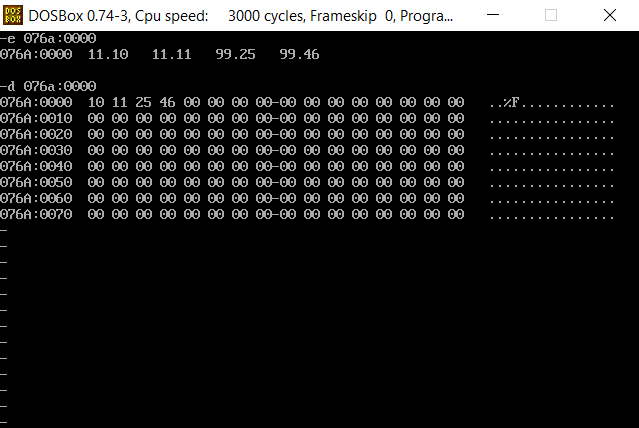
Program:

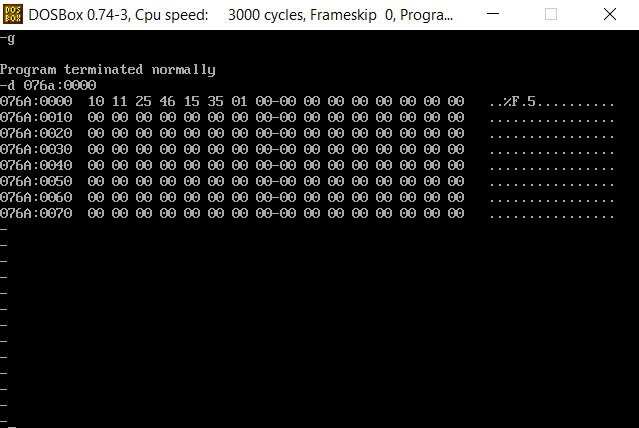
|  |  |
| --- | --- |
| CODE | COMMENT |
| Program for Subtracting 2, 16-bit numbers  assume cs:code,ds:data  data segment  opr1 dw 1111h  opr2 dw 9999h  result dw 0000H  carry dw 0000H  data ends  code segment  org 0100h  start:  mov ax,data  mov ds,ax  mov ax,opr1  mov bx,opr2  mov cx,0000h  sub ax,bx  jnc here  neg ax  inc cx  here:  mov result,ax  mov carry,cx  mov ah,4ch  int 21h  code ends  end start | Data segment initialized  opr1 initialised and set to 1111  opr2 initialised and set to 9999  result initialised and set to 0000  carry initialised and set to 0000  Code segment begins  Originating address is set at 0100  Address of data segment moved to ax  From ax, transferred to ds  Value of opr1 transferred to ax  Value of opr2 transferred to bx  cx is initialised and set to 0  Substarction takes palce  Junction created   * Jump if no carry * Else: negate ah and increment cx   data transferred from ax to result  data transferred from cx to carry  Program terminates |

Unassembled code:



Sample input and output:





Result:

1. bit subtraction is executed and verified using an emulator.

# 16 - Bit Multiplication:

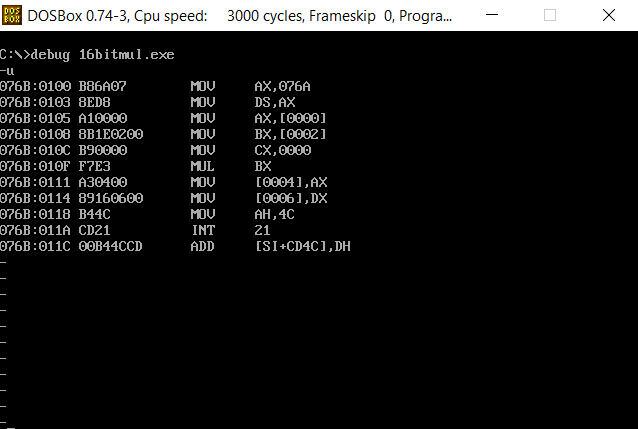
## Algorithm:

* + Program is set to run from any specified memory position.
  + Load data from opr1 to register AX (first number)
  + Load data from opr2 to register BX (second number)
  + Multiply these two numbers (contents of register AX and register BX)
  + Initialise carry to 0.
  + Multiplied values is stored in ax and dx
  + These two values are stored in different locations for better representation.
  + Terminate the program.

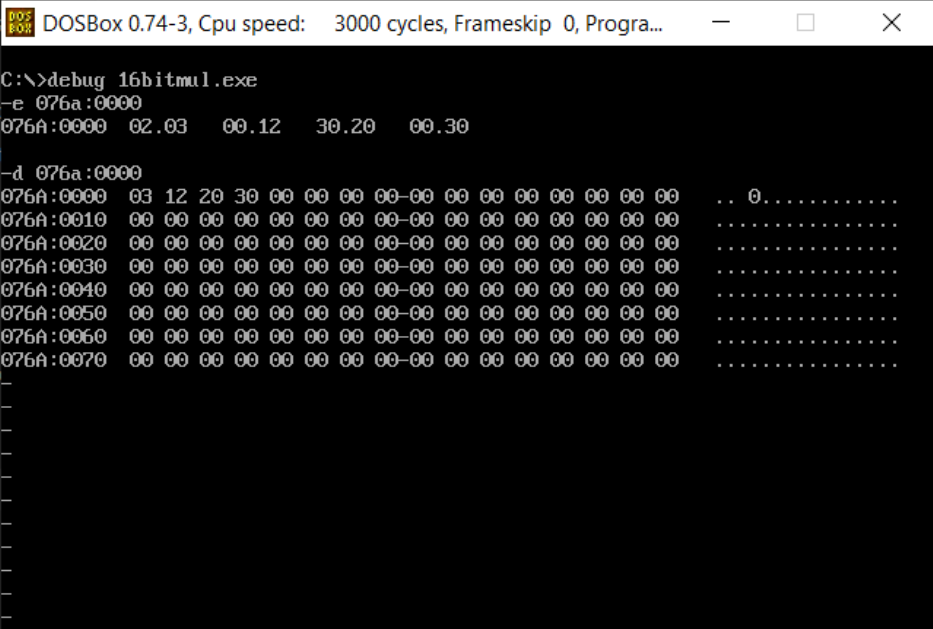
## Program:

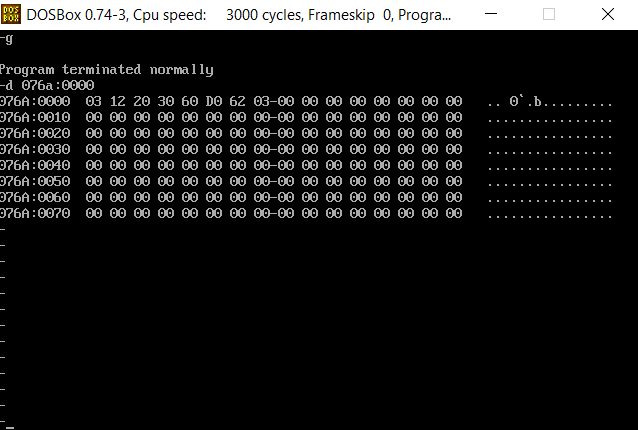
|  |  |
| --- | --- |
| CODE | COMMENT |
| ;Program for Multiplying 2, 16 bit numbers  assume cs:code,ds:data  data segment  opr1 dw 0002h opr2 dw 0030h result dw 0000H res dw 0000H  data ends  code segment  org 0100h  start:  mov ax,data  mov ds,ax  mov ax,opr1  mov bx,opr2  mul bx  mov result,ax  mov res,dx  mov ah,4ch  int 21h  code ends  end start | Data segment initialized  opr1 initialised and set to 0002  opr2 initialised and set to 0030  result initialised and set to 0000  res initialised and set to 0000  Code segment begins  Originating address is set at 0100  Address of data segment moved to ax  From ax, transferred to ds  Value of opr1 is transfers to ax  Value of opr2 is transferred to bx  Multiply ax and bx  data transferred from ax to result  data transferred from dx to res  Program terminates |

Unassembled code:



Sample input and output:





Result:

1. bit multiplication is executed and verified using an emulator.

# 16 -Bit Division:

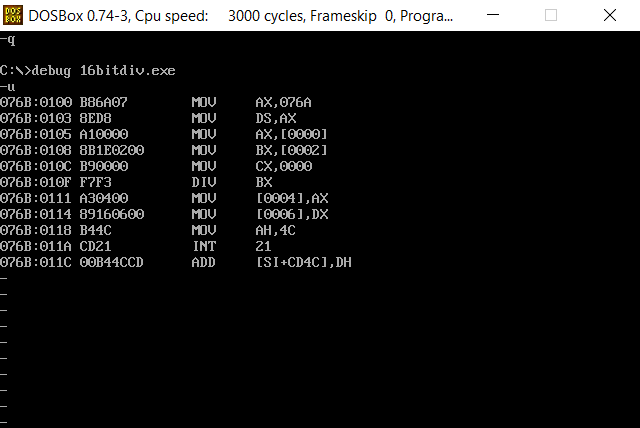
## Algorithm:

* + Load data from opr1 to register AX (first number)
  + Load data from opr2 to register BX (second number)
  + Initialise a variable for remainder to 0.
  + Divide these two numbers (contents of register AX and register BX)
  + Move ax value to quotient variable.
  + Move dx value to remainder variable.
  + Terminate the program.

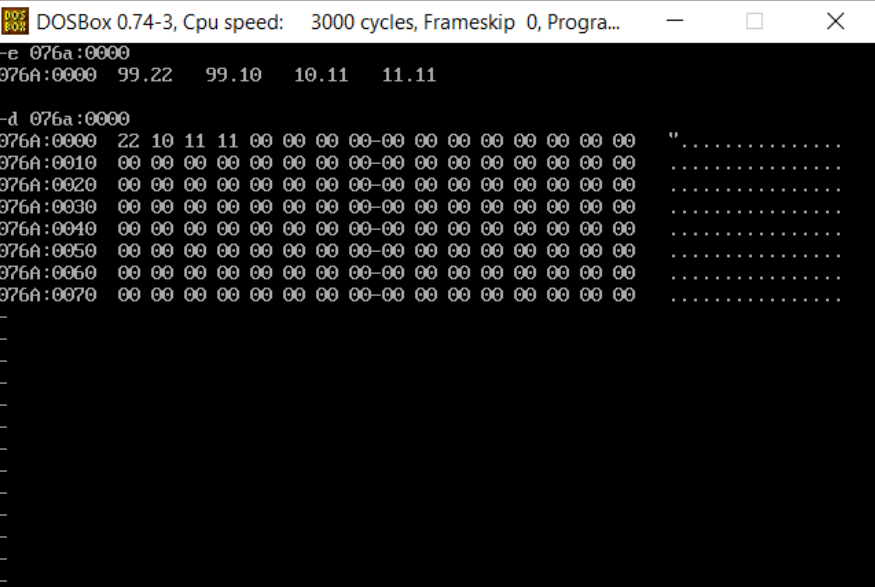
## Program:

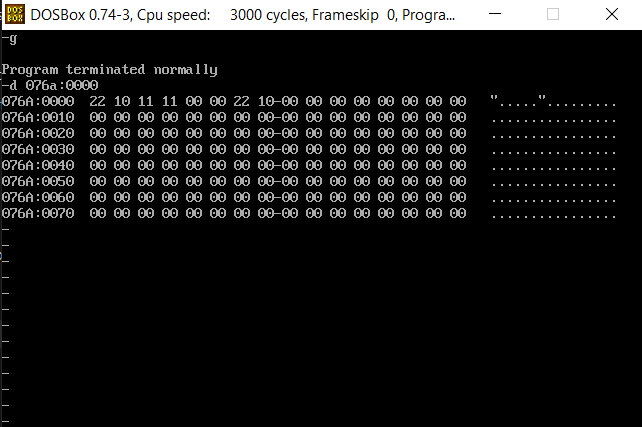
|  |  |
| --- | --- |
| CODE | COMMENT |
| ;Program for Dividing 2, 16 bit numbers  assume cs:code,ds:data data segment  opr1 dw 9999h  opr2 dw 1111h  quotient dw 0000H rem dw 0000H  data ends  code segment  org 0100h  start:  mov ax,data  mov ds,ax  mov ah,0000h  mov ax,opr1  mov bx,opr2  div bx  mov quotient,ax mov rem,dx  mov ah,4ch  int 21h  code ends  end start | Data segment initialized  opr1 initialised and set to 9999 opr2 initialised and set to 1111  quotient initialised and set to 00 rem initialised and set to 0000  Code segment begins  Originating address is set at 0100    Address of data segment moved to ax  From ax, transferred to ds  ax is initialised and set to 0000  Value of opr1 transferred to ax Value of opr2 transferred to bx Division takes place  data transferred from ax to quotient  data transferred from dx to rem  Program terminates |

Unassembled code:



Sample input and output:





Result:

16 bit division is executed and verified using an emulator.