Exp. 2

Date: 26/08/2020

**16 BIT ARITHMETIC OPERATIONS**

Name: Veeraraghavan N

Reg. No: 185001191

**Aim:** To perform 16-bit addition, subtraction, multiplication and division using the MASM assembler.

**Algorithm:**

1. Start
2. Declare names for code and data segments
3. In the data segment, declare variable names to identify memory locations
4. Start the code segment
   1. Move address value of data segment into AX or BX and then to DS
   2. Move operands into registers required to perform the operations
   3. Perform the required arithmetic instructions
   4. Move the result to the destined memory location
   5. Perform the interrupt routine
5. End Code Segment
6. End

**Programs:**

1. 16 bit addition:

|  |  |
| --- | --- |
| **Program** | **Comments** |
| mov bx, 2000h  mov ax, [bx] | Store location 2000h in BX  Move first operand at BX into AX |
| mov cl, 00h | Setting CL as 00h |
| add ax, [bx]+02h | Adding AX and the second operand after AX |
| jnc skip | Jump if no carry to “skip” |
| inc ch | Increment the CH register for carry |
| skip: mov [bx] + 04h, ax  mov [bx] + 06h, cl | Result of addition moved to first 2 registers from bx  Carry is moved to the next register |

2. 16 bit subtraction:

|  |  |
| --- | --- |
| **Program** | **Comments** |
| mov bx, 2000h  mov ax, [bx] | Store location 2000h in BX  Move first operand at BX into AX |
| mov cl, 00h | Setting CL as 00h |
| sub ax, [bx]+02h | Subtracting AX and the second operand after AX |
| jnc skip | Jump if no carry to “skip” |
| Neg AX | If carry is present then the result is negative, so 2s complement is found |
| inc ch | Increment the CH register for carry |
| skip: mov [bx] + 04h, ax  mov [bx] + 06h, cl | Result of subtraction moved to first 2 registers from bx  Carry is moved to the next register and this indicates whether number is negative or not |

3. 16 bit Multiplication:

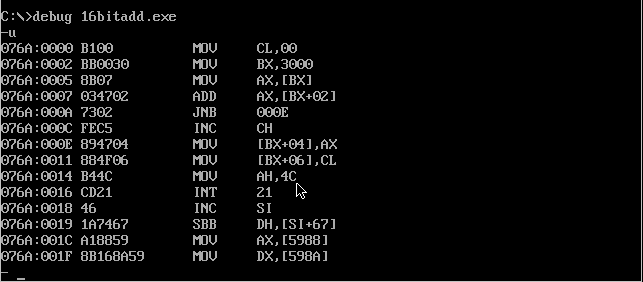
|  |  |
| --- | --- |
| **Program** | **Comments** |
| mov bx, 2000h  mov ax, [bx]  mov cx, [bx] + 02h | Store location 2000h in BX  Move first operand at BX into AX  Move second operand at BX into CX |
| mul cx | Multiplying AX and CX,  AX x CX = DXAX |
| mov [bx] + 04h, dh  mov [bx] + 05h, dl  mov [bx] + 06h, ah  mov [bx] + 07h, al | Result of multiplication is 32 bits. The higher word is moved to the location [BX] + 04h and the lower word is moved to the location [BX] + 06h |

4. 16 bit Division:

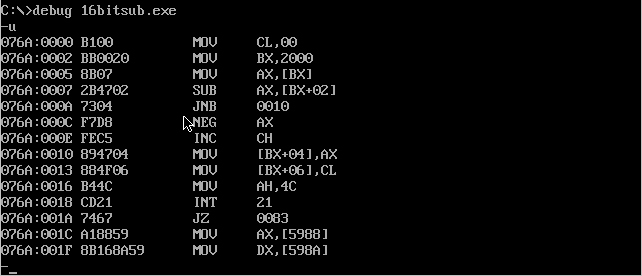
|  |  |
| --- | --- |
| **Program** | **Comments** |
| mov bx, 2000h  mov ax, [bx]  mov cx, [bx] + 02h | Store location 2000h in BX  Move first operand at BX into AX  Move second operand at BX into CX |
| mov dx, 0000h | Setting value of DX as 0000h (zero) |
| div cx | Dividing AX and CX,  DX = AX / CX (Quotient)  AX = AX MOD CX (Reminder) |
| mov [bx] + 04h, dh  mov [bx] + 05h, dl  mov [bx] + 06h, ah  mov [bx] + 07h, al | Quotient is moved to the location [BX] + 04h  Reminder is moved to the location [BX] + 06h |

**Unassembled Code:**

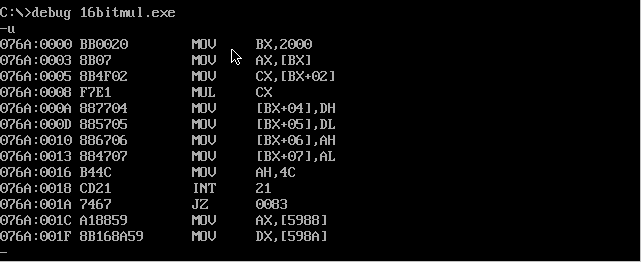
1. 16 bit addition:



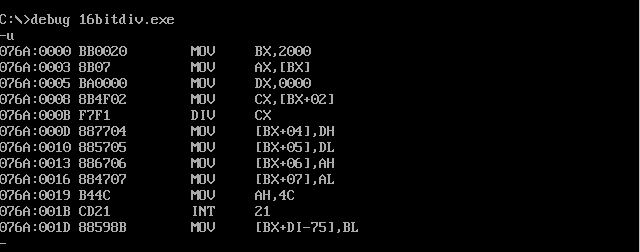
2. 16 bit subtraction:



3. 16 bit multiplication:

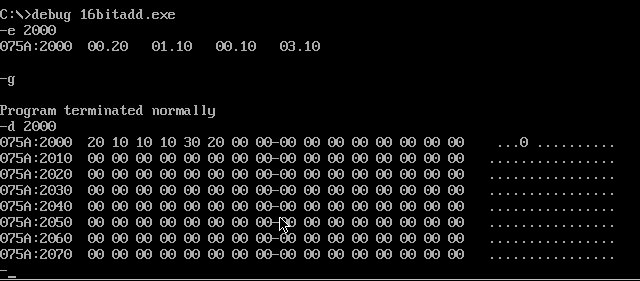


4. 16 bit division:

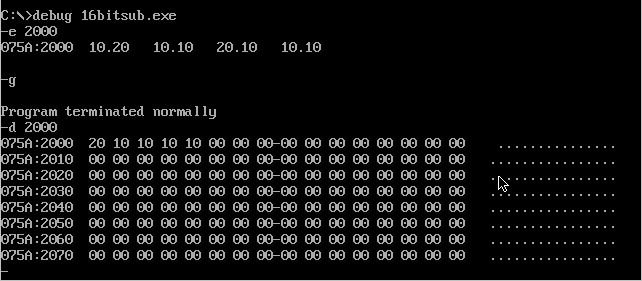


**Sample Input and Output**

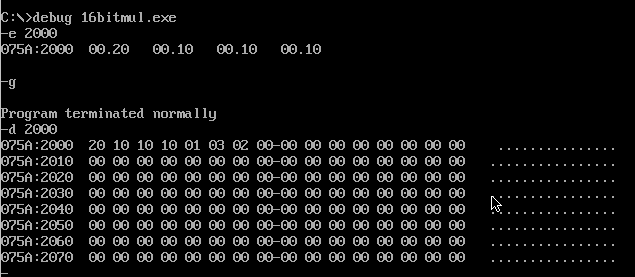
1. 16 bit addition:



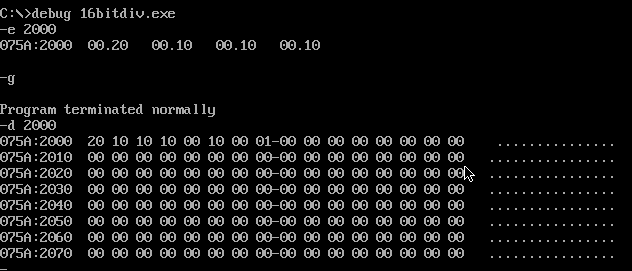
2. 16 bit subtraction:



3. 16 bit multiplication:



4. 16 bit division:



**Result:**

16 bit arithmetic operations such as addition, subtraction, multiplication, division performed successfully