|  |  |  |
| --- | --- | --- |
|  | 16 – BIT ARITHMETIC OPERATIONS |  |
| Exp No.: 2 |  | **Name:** S Vishakan |
| Date: 26-08-2020 |  | **Reg. No:** 18 5001 196 |

**AIM:**

To write assembly language programs to perform 16-bit arithmetic operations and execute them.

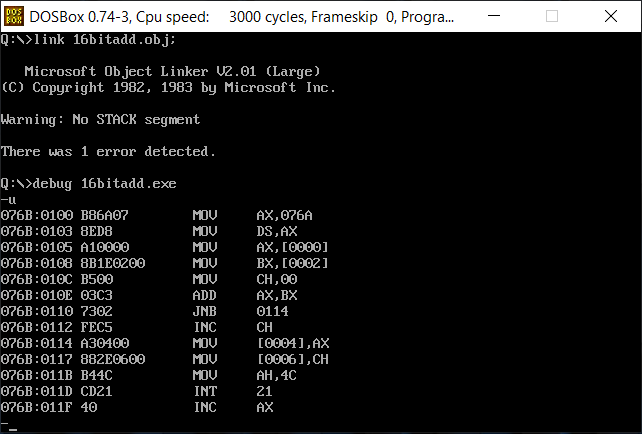
**ALGORITHM:**

* Begin.
* Open data segment.
* Initialize data segment with required operands, data types and values.
* Close the data segment.
* Open code segment.
* Set a preferred offset (preferably 100)
* Load the data segment content into AX register.
* Transfer the contents of AX register to DS register.
* Do the required operation (ADD, SUB, MUL, DIV) on the registers.
  + Jump (whenever ever carry/ overflow is a possibility)
  + Increment carry(add) or negate the value. (2’s complement)
* Introduce an interrupt for safe exit. (INT 21h)
* Close the code segment.
* End.

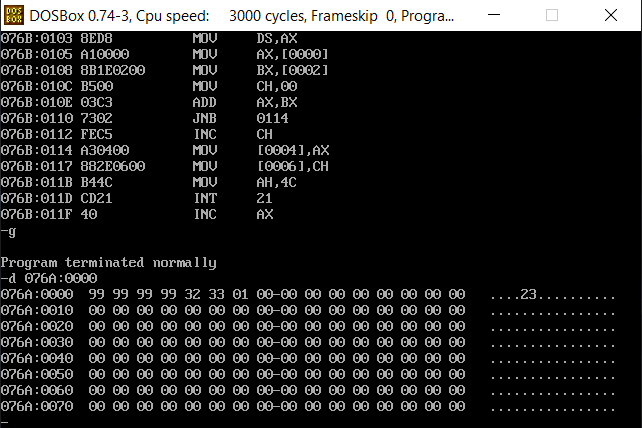
**PROGRAM – 1: 16 – BIT ADDITION:**

|  |  |
| --- | --- |
| **PROGRAM** | **COMMENTS** |
| assume cs:code, ds:data | Declare code and data segment. |
|  |  |
| data segment | Initialize data segment with values. |
| opr1 dw 9999h | Stores operand 1. |
| opr2 dw 9999h | Stores operand 2. |
| result dw 0000h | Stores the result of the operation. |
| carry db 00h | Stores the carry, if any. |
| data ends |  |
|  |  |
| code segment | Start the code segment. |
| org 0100h | Initialize an offset address. |
| start: mov ax, data | Transfer data from memory location [0000] and [0001] to AL AND AH respectively. |
| mov ds, ax | Transfer data from memory location AX to DS. |
| mov ax, opr1 | Transfer value of opr1 to AX. |
| mov bx, opr2 | Transfer value of opr2 to BX. |
| mov ch, 00h | CH = 0. |
| add ax, bx | AX = AX + BX. |
| jnc here | Jump if no carry to “here”. Else, continue. |
| inc ch | CH = CH + 1 |
| here: mov result, ax | Transfer value of AX to result. |
| mov carry, ch | Transfer value of CH to carry. |
| mov ah, 4ch |  |
| int 21h | Interrupt the process with return code and exit. |
| code ends |  |
| end start |  |

**UNASSEMBLED CODE:**



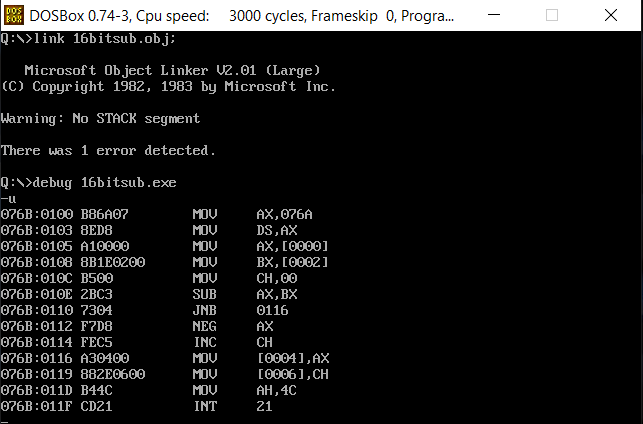
**SAMPLE I/O SNAPSHOT:**

****

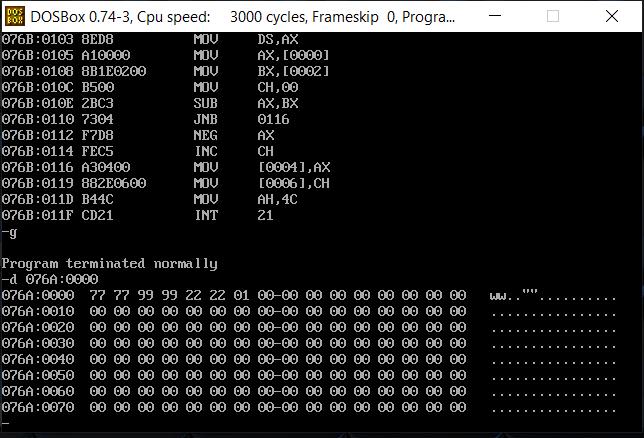
**PROGRAM – 2: 16 – BIT SUBTRACTION:**

|  |  |
| --- | --- |
| **PROGRAM** | **COMMENTS** |
| assume cs:code, ds:data | Declare code and data segment. |
|  |  |
| data segment | Initialize data segment with values. |
| opr1 dw 7777h | Stores operand 1. |
| opr2 dw 9999h | Stores operand 2. |
| diff dw 0000h | Stores the result of the operation. |
| sign db 00h | Stores the sign bit. |
| data ends |  |
|  |  |
| code segment | Start the code segment. |
| org 0100h | Initialize an offset address. |
| start: mov ax, data | Transfer data from memory location [0000] and [0001] to AL AND AH respectively. |
| mov ds, ax | Transfer data from memory location AX to DS. |
| mov ax, opr1 | Transfer value of opr1 to AX. |
| mov bx, opr2 | Transfer value of opr2 to BX. |
| mov ch, 00h | CH = 0. |
| sub ax, bx | AX = AX – BX. |
| jnc here | Jump if no sign change to “here”. Else, continue. |
| neg ax | Take 2’s Complement if negative value. |
| inc ch | CH = CH + 1 |
| here: mov diff, ax | Transfer value of AX to diff. |
| mov sign, ch | Transfer value of CH to sign. |
| mov ah, 4ch |  |
| int 21h | Interrupt the process with return code and exit. |
| code ends |  |
| end start |  |

**UNASSEMBLED CODE:**



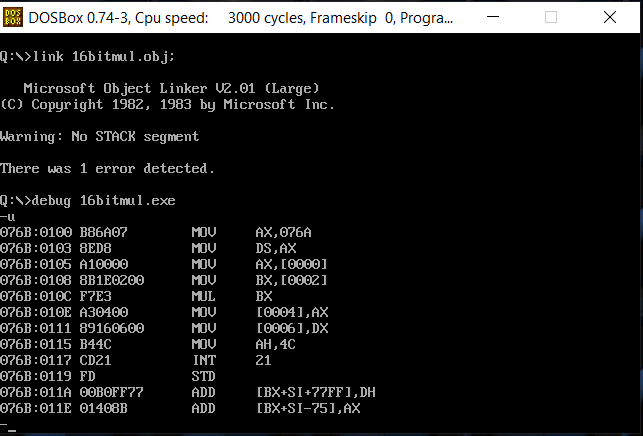
**SAMPLE I/O SNAPSHOT:**

****

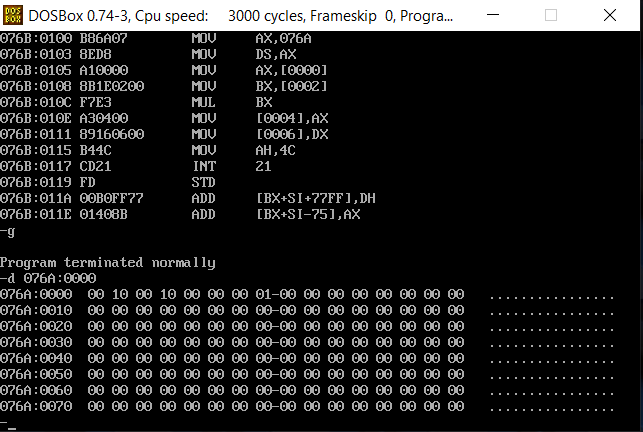
**PROGRAM – 3: 16 – BIT MULTIPLICATION:**

|  |  |
| --- | --- |
| **PROGRAM** | **COMMENTS** |
| assume cs:code, ds:data | Declare code and data segment. |
|  |  |
| data segment | Initialize data segment with values. |
| opr1 dw 1000h | Stores operand 1. |
| opr2 dw 1000h | Stores operand 2. |
| product1 dw 0000h | Stores the lower 16 bits of the operation. |
| product2 dw 0000h | Stores the higher 16 bits of the operation. |
| data ends |  |
|  |  |
| code segment | Start the code segment. |
| org 0100h | Initialize an offset address. |
| start: mov ax, data | Transfer data from memory location [0000] and [0001] to AL AND AH respectively. |
| mov ds, ax | Transfer data from memory location AX to DS. |
| mov ax, opr1 | Transfer value of opr1 to AX. |
| mov bx, opr2 | Transfer value of opr2 to BX. |
| mul bx | DXAX = AX \* BX. |
| mov product1, ax | Transfer value of AX to product1. |
| mov product2, dx | Transfer value of DX to product2. |
| mov ah, 4ch |  |
| int 21h | Interrupt the process with return code and exit. |
| code ends |  |
| end start |  |

**UNASSEMBLED CODE:**



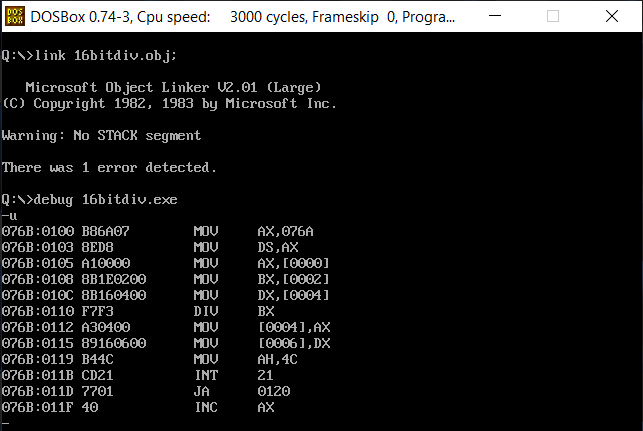
**SAMPLE I/O SNAPSHOT:**



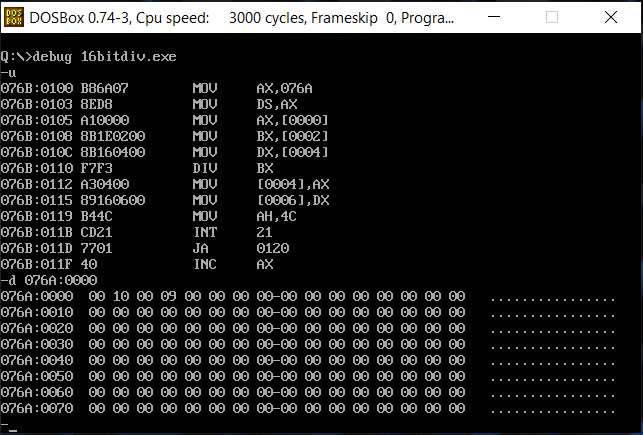
**PROGRAM – 4: 16 – BIT DIVISION:**

|  |  |
| --- | --- |
| **PROGRAM** | **COMMENTS** |
| assume cs:code, ds:data | Declare code and data segment. |
|  |  |
| data segment | Initialize data segment with values. |
| opr1 dw 1000h | Stores the dividend. |
| opr2 dw 0900h | Stores the divisor. |
| quot dw 0000h | Stores the quotient of the division. |
| rem dw 0000h | Stores the remainder of the division. |
| data ends |  |
|  |  |
| code segment | Start the code segment. |
| org 0100h | Initialize an offset address. |
| start: mov ax, data | Transfer data from memory location [0000] and [0001] to AL AND AH respectively. |
| mov ds, ax | Transfer data from memory location AX to DS. |
| mov ax, opr1 | Transfer value of dividend to AX. |
| mov bx, opr2 | Transfer value of divisor to BX. |
| mov dx, quot | Transfer value of quotient (0000h) to DX. |
| div bx | AX = DXAX / BL. (AX has quotient, DX has remainder) |
| mov quot, ax | Transfer value of AX to quot. |
| mov rem, dx | Transfer value of DX to rem. |
| mov ah, 4ch |  |
| int 21h | Interrupt the process with return code and exit. |
| code ends |  |
| end start |  |

**UNASSEMBLED CODE:**



**SAMPLE I/O SNAPSHOT:**



**RESULT:**

The assembly level programs were written to perform the 16 – bit arithmetic operations and compiled. The results were observed and noted down.