National University of Computer and Emerging Sciences 

**Laboratory Manual**

*for*

**Computer Organization and Assembly Language Programming**

| Course Instructor | Aleena Ahmad |
| --- | --- |
| Lab Instructor | Sana Ejaz |
| Semester | Fall 2024 |

Department of Computer Science

FAST-NU, Lahore, Pakistan

Page 1 of 3

**OBJECTIVES:**

∙ How to perform bit operations.

∙ How to perform shift with 64 bit numbers.

∙ How to perform addition and multiplication.

**Instructions:**

| ∙ Run and debug the programs, ensuring that they behave as expected.  ∙ Document your observations and note any issues encountered during implementation in a Word document.  ∙ Submit work in a single Word file with screenshots. No asm, lst , or com. (Do not submit a zip folder) |
| --- |

**Task 1: Calculate the number of one bits in BX and complement an equal number of least significant bits in AX.**

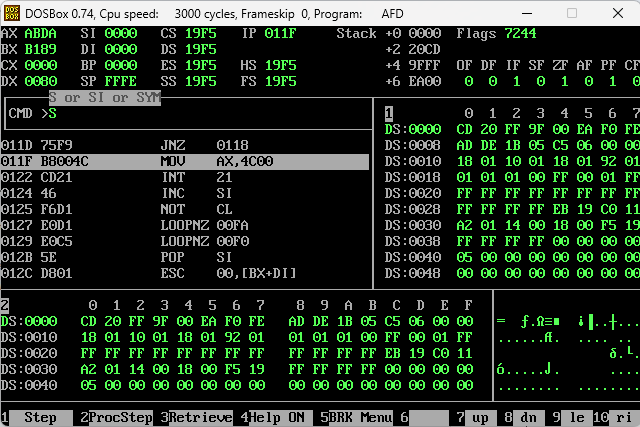
***Hint: Use the XOR instruction and Rotate through carry.***

**Sample Run:**

| **Initial value of BX** | **Total No of 1 Bits in BX** | **Initial value of AX** | **AX after Complementing 7 least significant bits** |
| --- | --- | --- | --- |
| 1011 0001  1000 1001 | 7 | 1010 1011  1**010 0101** | 1010 1 1**101 1010** |

**Code:**

| [org 0x0100]  MOV AX, 0xABA5  MOV BX, 0xB189  MOV CX, 0  MOV DX, 1    checkbit:  TEST BX, DX  JZ sheeft    INC CX  sheeft:  SHL DX, 1  JNZ checkbit  MOV DX, 1  loplop:  XOR AX, DX  SHL DX, 1    DEC CX  JNZ loplop  MOV AX, 0x4C00  INT 0x21 |
| --- |



**Task 2:**

**A) Write a program that shifts a 64-bit number.**

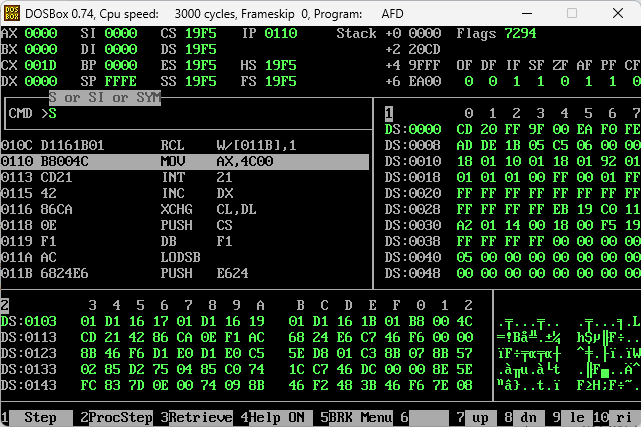
**B) Write a program that adds two 64-bit numbers.**

***Hint: Use dd and word as taught in class.***

Page 2 of 3

**Code:**

| [org 0x0100]  SHL word [num], 1  RCL word [num + 2], 1  RCL word [num + 4], 1  RCL word [num + 6], 1  MOV AX, 0x4C00  INT 0x21  num: dd 0x87654321, 0x12345678 |
| --- |

****

**Task 3: Write a program to multiply two 32-bit numbers and store the answer in a 64-bit location.**

Page 3 of 3

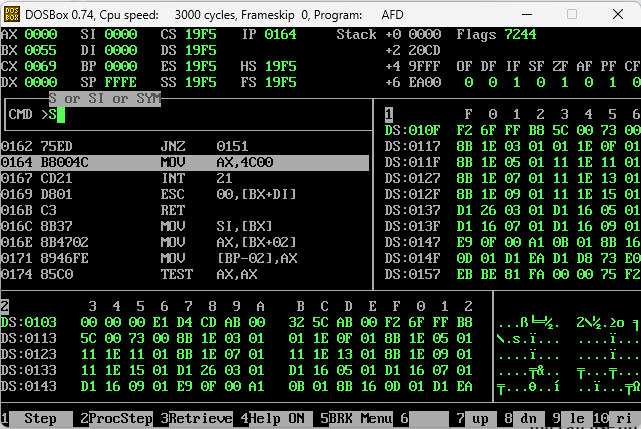
**Sample Run:**

| **a: dq 0xABCDD4E1** ; **dq** allocates 64 bit memory space. a is 32-bit number but it has space allocation of 64 bits  **b: dd 0xAB5C32** ; **32-bit space** for multiplier  **result: dq 0x0** ; result should be 0x73005CB8FF6FF2 verify on calculator programmer’s view |
| --- |

**You can use dd instead of dq.**

**Code:**

| [org 0x0100]  JMP start  multiplicand: dd 0xABCDD4E1, 0  multiplier: dd 0xAB5C32  result: dd 0, 0 ; result = 0x0073005CB8FF6FF2  extendedAdd:  MOV BX, [multiplicand]  ADD [result], BX    MOV BX, [multiplicand + 2]  ADC [result + 2], BX    MOV BX, [multiplicand + 4]  ADC [result + 4], BX    MOV BX, [multiplicand + 6]  ADC [result + 6], BX  extendedShift:  SHL word [multiplicand], 1  RCL word [multiplicand + 2], 1  RCL word [multiplicand + 4], 1  RCL word [multiplicand + 6], 1  JMP skip  start:  MOV AX, [multiplier]  MOV DX, [multiplier + 2]  checkbit:  SHR DX, 1  RCR AX, 1  JNC extendedShift  JMP extendedAdd  skip:  CMP DX, 0  JNZ checkbit    CMP AX, 0  JNZ checkbit  MOV AX, 0x4C00  INT 0x21 |
| --- |

****