**Chapter 4 - Bit Manipulations**

**For all the programs given below, use best instructions studied so far. For bit testing try TEST instruction and shifting both.**

**New Instructions:**

|  |
| --- |
| CLC ; Clear Carry Flag C=0  STC ; Set Carry Flag C=1  CMC ; Complement Carry Flag |

**Lab Manual**

1. **[Bit Manipulation]** Calculate the number of one bits in BX and complement an equal number of least significant bits in AX. HINT: Use the XOR instruction.

**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 1011 1**101 1010** |

1. **[Extended Multiplication]** Write a program to multiply two 32-bit numbers and store the answer in a 64-bit location.

**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 |

**Practice Problems**

1. **[Masking]** Write a program to swap the nibbles (4-bits = 1 nibble) in each byte of the AX register.

**Sample Run:**

|  |  |  |
| --- | --- | --- |
| AX before Swap | **1011 0010 0101 1101** | **0xB25D** |
| AX after Swap | **0010 1011 1101 0101** | **0x2BD5** |

1. Write a program to swap every pair of bits in the AX register i.e. swap bit # 0 with bit # 1, bit # 2 with bit # 3 and so on.

**Sample Run:**

|  |  |
| --- | --- |
| AX before Swap | **10 11 00 10 01 01 11 01** |
| AX after Swap | **01 11 00 01 10 10 11 10** |

1. Write a program that adds two 64-bit numbers.
2. Write a program that subtracts two 64-bit numbers.
3. AX contains a non-zero number. Count the number of ones in it and store the result back in AX. Repeat the process on the result (AX) until AX contains one. Calculate in BX the number of iterations it took to make AX one. For example BX should contain 2 in the following case:

AX = 1100 0101 1010 0011 (input – 8 ones)

AX = 0000 0000 0000 1000 (after first iteration – 1 one)

AX = 0000 0000 0000 0001 (after second iteration – 1 one) STOP

1. Write a program to search a particular element from an array using binary search. If the element is found set AX to one and otherwise to zero. Binary Search searches a number from a sorted array. Shifting a number to right divides it by 2. Do not use division instruction use shifting for division.

Help: In the link given below, search 649 and 650 using Binary Search. It shows all the iteration to explain Binary Search.

<https://www.cs.usfca.edu/~galles/visualization/Search.html>

1. AX contains a number between 0-15. Write code to complement the corresponding bit in BX. For example if AX contains 6; complement the 6th bit of BX.

**Dry Run Practice Problems**

**Exercise 1:** For the code segment given below, write the contents of each register and memory after the execution.

1. Note: Label a is ds:103.

|  |
| --- |
| **; Multiplication with multiplier in memory**  **[org 0x0100]**  **jmp start**  **a: db 13**  **b: db 5**  **res: db 0**  **start: mov cl,4**  **mov al, [a]**  **loop: shr byte[b], 1**  **jnc skipAdd**  **add [res], al**  **skipAdd: shl al,1**  **dec cl**  **jnz loop**  **mov ax, 0x4c00 ;terminate the program**  **int 0x21** |

(Verify your answer with debugger.)

1. Above code runs the loop 4 times even if b=0. Update above code such that it breaks the loop as soon as there is no 1 bit left in b. (Verify your answer with debugger.)

|  |
| --- |
| [org 0x0100]  jmp start  a: dw 0x04E9  start: mov ax, [a]  mov bh, [a]  mov dl, [a]  shl ax, 1  shl ax, 1  shl ax, 1  shl ax, 1  shl word[a], 1  shl word[a], 1  shl word[a], 1  shl word[a], 1  rol bx, 1  rol bx, 1  rol bx, 1  rol bx, 1  rcl dl, 1  rcl dl, 1  rcl dl, 1  rcl dl, 1  rcl byte[a+1], 1  rcl byte[a+1], 1  rcl byte[a+1], 1  rcl byte[a+1], 1  mov ax, 0x4c00 ;terminate the program  int 0x21 |

|  |  |
| --- | --- |
| [org 0x0100]  mov bx, 0xFCA9  ror bl,4  ror bh,4  mov ax, 0x4c00 ;terminate the program  int 0x21 | **BX** |
|  |
|  |
|  |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Code** | **AX** | **BX** | **DX** |
| **[org 0x0100]**  **mov ax, 0xB25D**  **mov bx,0x5555**  **and bx, ax**  **mov dx,0xAAAA**  **and dx, ax**  **shr dx, 1**  **shl bx, 1**  **add bx, dx**  **mov ax, bx**  **mov ax, 0x4c00 ;terminate the program**  **int 0x21** |  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |