**Computer Organization & Assembly Language**

**Lab 08**

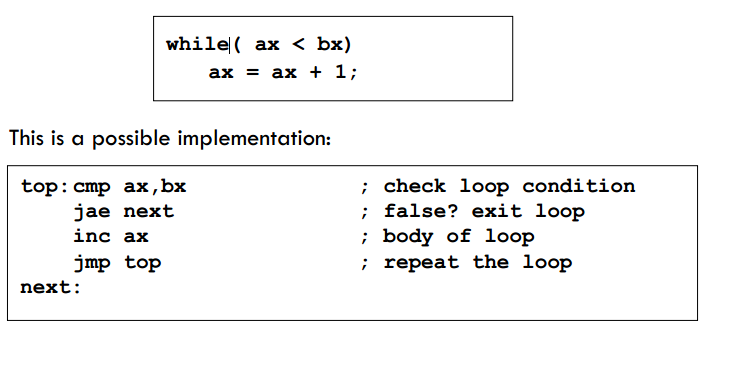
**Topics:**

1. While Loop
2. Conditional Loops
3. Interrupts

**While Loop**

While loop tests a condition before performing a block of statements. As long as the condition remains true, statements are repeated.

**Example:**

****

1. **LOOPZ (Loop if Zero):**

The LOOPZ instruction is an x86 assembly language command that facilitates conditional looping in a program. It repeatedly executes a specific block of code as long as the zero flag (ZF) in the CPU's flags register is set. This instruction is particularly useful for scenarios where you want to iterate through a loop until a specific condition is met, and that condition involves checking if a certain result or variable is zero.

1. **LOOPE (Loop if Equal):**

LOOPE is another x86 assembly instruction that enables conditional looping. It continues to execute a designated code block as long as both the zero flag (ZF) is set and the counter register (typically CX or ECX) is not zero. LOOPE is often employed in scenarios where you need to repeatedly perform an action until a specific equality condition is satisfied, such as searching for a particular element in an array or list.

1. **LOOPNZ (Loop if Not Zero):**

The LOOPNZ instruction, similar to LOOPZ, is used for implementing conditional loops in assembly language programs. It repeatedly executes a specified block of code as long as the zero flag (ZF) is clear and the counter register (CX or ECX) is not zero. LOOPNZ is commonly employed when you want to iterate through a loop until a condition that involves checking if a value is not zero is met.

1. **LOOPNE (Loop if Not Equal):**

LOOPNE is an x86 assembly instruction that creates a conditional loop. It continues to execute a designated code block as long as both the zero flag (ZF) is clear and the counter register (typically CX or ECX) is not zero. LOOPNE is often used when you need to repeatedly perform an action until a specific inequality condition is satisfied, such as searching for a specific element in a data structure while ensuring it is not equal to the desired value.

**Interrupt**

* An interrupt is an event that causes the processor to suspend its present task and transfer control to a new program called the interrupt service routine (ISR).
* Three sources of interrupts
* Software interrupts
* Hardware interrupts
* Processor interrupts

**Common Software Interrupts:**

The Common Interrupts are

**GetStdHandle PROTO**

**WriteConsoleA PROTO, a1:DWORD, a2: PTR BYTE, a3: Dword, a4: ptr dword, a5: dword**

**ReadConsoleA PROTO, a1:DWORD, a2: PTR BYTE, a3: Dword, a4: ptr dword, a5: dword**

**invoke ReadConsoleA, eax, offset buffer, lengthof buffer, offset x, 0**

**Tasks:**

1. Initialize a random array of 10 elements. You are to find the second largest and second smallest element in that array. Store them in separate variables.
2. Write Assembly program to Input upper case letter from user and display lowercase.
3. Write an assembly program that takes an input and tells whether the given number is even or odd
4. Write Assembly program to input letter from the user and display the previous character.
5. Take the marks from user and display the letter grade. (Use the following table)

|  |  |
| --- | --- |
| a < 5 | D |
| 5 ≤ a <7 | C |
| 7 ≤ a < 9 | B |
| =9 | A |

1. Write a program that displays the following output.

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

b) \*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*