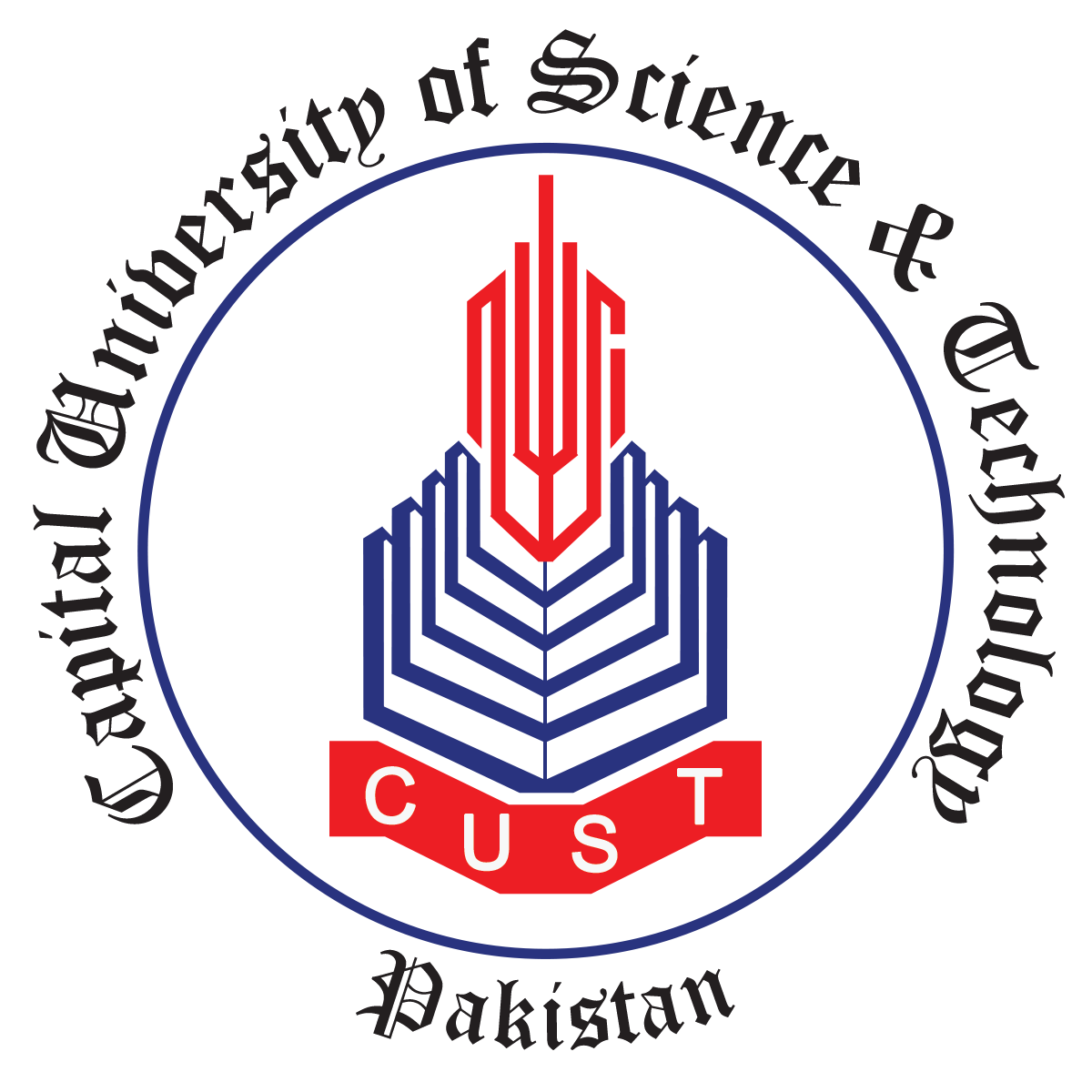
****

**Name: Muhammad Ammar Janjuha**

**Registration No: BCS213104**

**Section: 01**

**Assignment: 03**

**Subject: COAL**

**Submitted to: Sir Adnan Jelani**

**Date: 25-Dec-2023**

**Department of Computer Science**

# Question No.1:

# Solution:

* **The role of flags in decision-making;**

Flags in decision-making serve as crucial indicators that guide the process. They highlight pertinent information, trigger specific actions, or act as reminders, thereby enhancing efficiency and effectiveness. For instance, in software development, flags can indicate the status of a task or signal when a certain condition has been met 1. Similarly, in data analysis, flags can mark data points that meet certain criteria, facilitating easier retrieval of relevant information.

In the broader context, flags can also play a significant role in online communities, such as gaming forums. Here, they can be used to indicate toxic behaviors, which can then be handled by moderators. However, the use of flags is not without its complexities. For example, players may use flags as a rhetorical strategy to influence others' behavior, or as a mechanism to retaliate against perceived unfairness 4. Despite these complexities, the role of flags remains critical in facilitating informed decision-making processes.

* **Significance of Zero flag and Carry flag in conditional jumps**;

The Zero flag and Carry flag play significant roles in conditional jumps in assembly language programming.

The Zero flag is set if the result of an arithmetic operation is zero. This flag is commonly used in comparisons where you want to test if two values are equal without actually changing them. For instance, the CMP instruction performs a subtraction operation between two operands but doesn't store the result, instead setting the Zero flag if the operands are equal. This flag is then used in conditional jump instructions like JZ (Jump if Zero) and JNZ (Jump if Not Zero) to control the flow of the program based on the result of the comparison .

***Code:***

*CMP EAX,EDX ;compare eax and edx*

*JZ L1 ;jump to label L1 if eax equals edx*

On the other hand, the Carry flag is set if the result of an operation exceeds the capacity of the data type. This flag is typically used in arithmetic operations involving unsigned integers. For example, adding 1 to the maximum value that can be stored in a byte (255) would result in a carry, setting the Carry flag. This flag is used in conditional jump instructions like JC (Jump if Carry) and JNC (Jump if No Carry) to control the flow of the program based on whether an overflow has occurred

***Code:***

*ADD AL,4h ;add 4 to al*

*JC L1 ;jump to label L1 if there was a carry*

These flags allow for precise control over program flow, enabling the creation of complex conditional logic within assembly programs.

**Write assembly code snippets implementing conditional jumps based on Zero flag and Carry flag**

(e.g., jump if zero, jump if not zero, jump if carry, jump if not carry).

**Zero flag**

**jump if zero (JZ) operation:**

**Code:**

MOV AX, 10

MOV BX, 10

CMP AX, BX ; Compare AX and BX

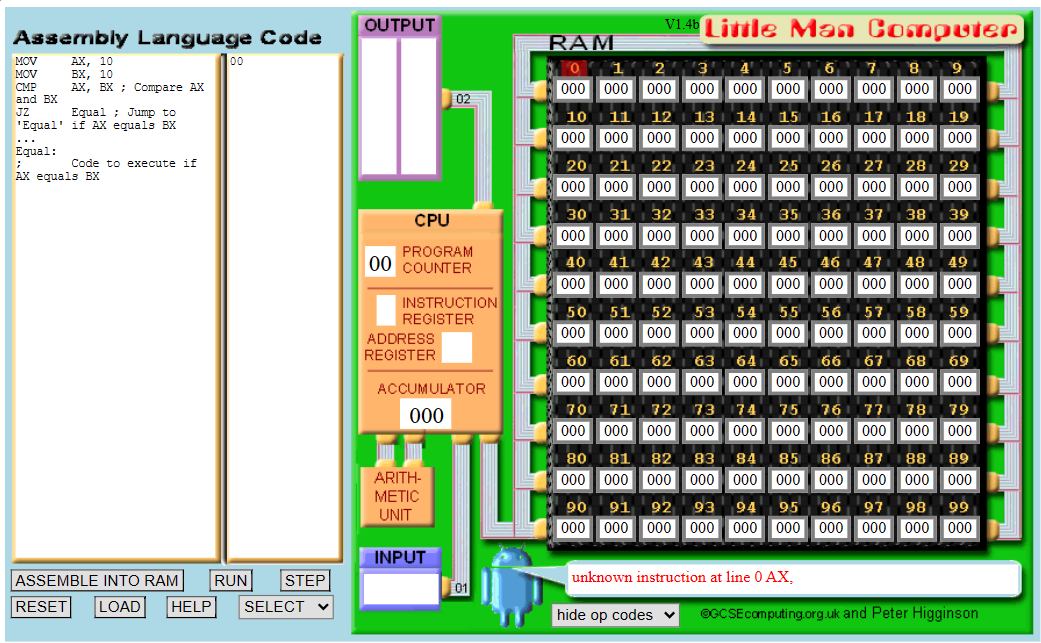
JZ Equal ; Jump to 'Equal' if AX equals BX

...

Equal:

; Code to execute if AX equals BX

**Screenshot:**

****

**Zero flag**

**jump if not zero (JNZ) operation:**

**Code:**

MOV AX, 10

MOV BX, 20

CMP AX, BX ; Compare AX and BX

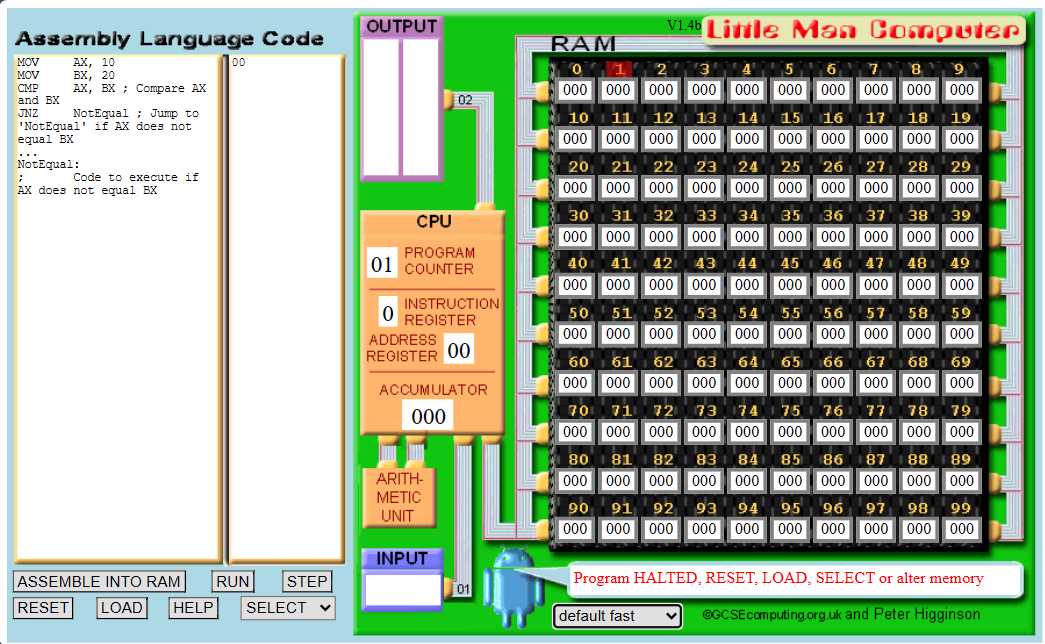
JNZ NotEqual ; Jump to 'NotEqual' if AX does not equal BX

...

NotEqual:

; Code to execute if AX does not equal BX

**Screenshot:**

****

**Carry flag**

**jump if carry (JC) operation:**

**Code:**

MOV AL, 0FFh ; Maximum value for a byte

ADD AL, 1 ; Add 1 to the maximum value, causing a carry

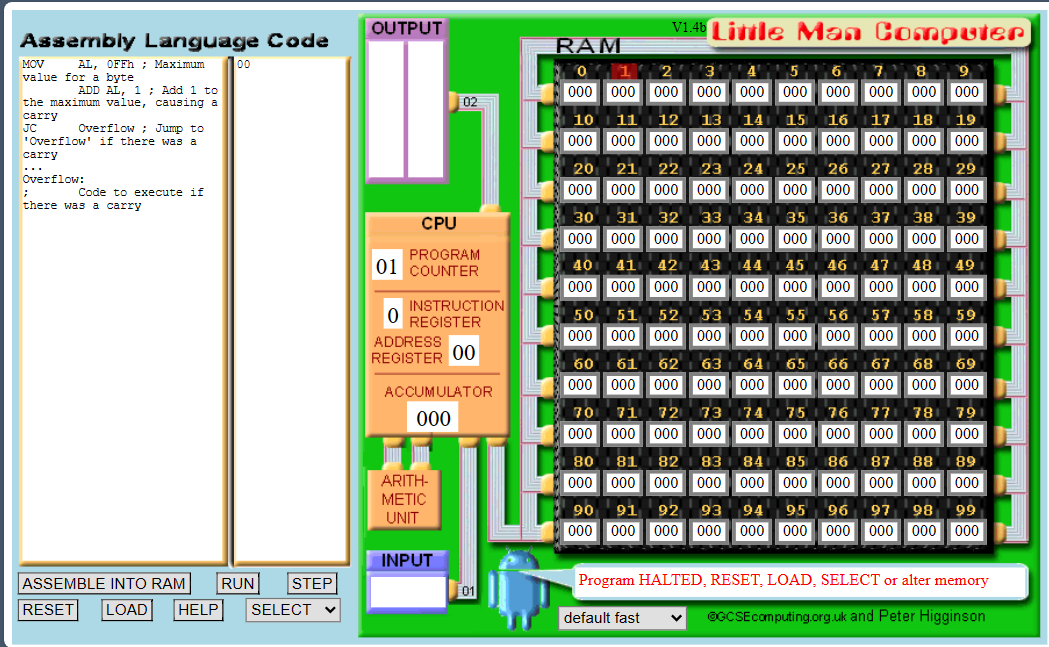
JC Overflow ; Jump to 'Overflow' if there was a carry

...

Overflow:

; Code to execute if there was a carry

**Screenshot:**

****

**Carry flag**

**jump if not carry (JNC) operation:**

**Code:**

MOV AL, 0FFh ; Maximum value for a byte

SUB AL, 1 ; Subtract 1 from the maximum value, no carry

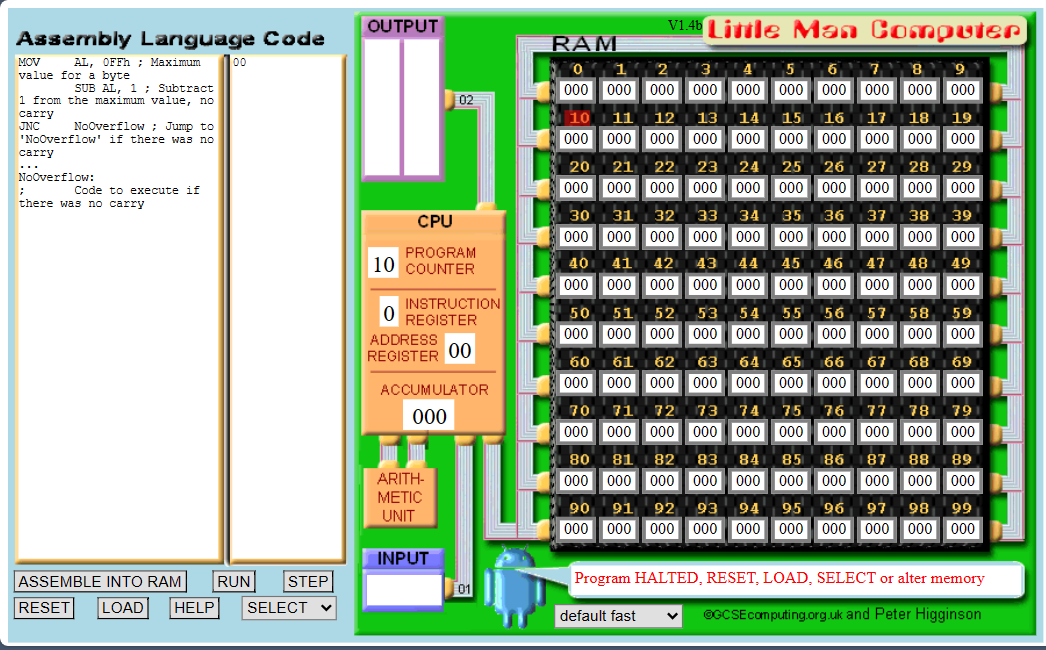
JNC NoOverflow ; Jump to 'NoOverflow' if there was no carry

...

NoOverflow:

; Code to execute if there was no carry

**Screenshot:**



**Question No.2**

An assembly code for the multiplication of two 16-bit numbers.

**CODE**

ORG 0x100

MOV AX, 15 ; First operand

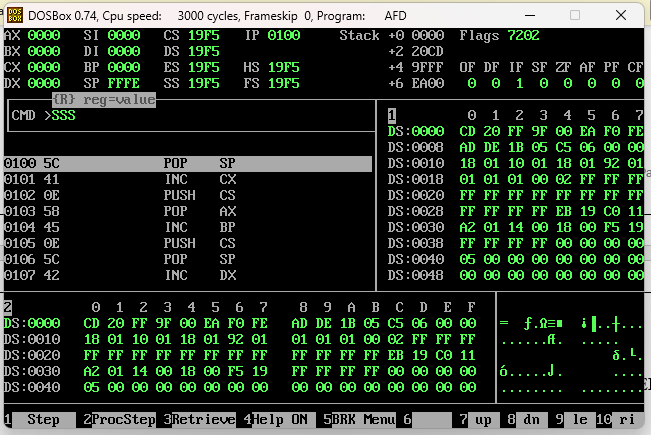
MOV BX, 5 ; Second operand

MUL BX ; Multiply AX by BX, result is in AX

MOV AX, 0x4C00

INT 0x21

**CODE**

****

**Sheet link:**

https://docs.google.com/spreadsheets/d/1HFGjNqC6d4NOyfH1PP3jfLiEeb0EPib9ZFWQ1M kgRbo/edit?usp=sharing

