Assignment 1 for EE2003 (Computer Organisation)

By EE21B017 (Anchal Debnath)

Q21. Write a program that can evaluate the expression (A + B) x (C + D)

The Mano simulator operates as a single-state machine. It loads input data from memory into the Accumulator and performs only arithmetic and logical operations, INC, ADD, and AND. Since there is no multiplication instruction available, we achieve multiplication by repeatedly adding one number (multiplicand) to itself a specific number of times (multiplier).

Logic:

We will add A to B store it in X, Add C and D store in C, then if X !=0 and Y!=0, we will add X to itself for Y times (ie in loop Add X to the accumulator and increment counter value, complement it and And it will Y , and stop when Y=0)

Instruction: -

1. LDA A: Load the value of A into the accumulator.
2. ADD B: Add the value of B to the accumulator.
3. STA X: Store the result in the accumulator (the sum of A and B) into the memory location X.

The same sequence of instructions is repeated for the numbers C and D, with the result being stored in memory location Y.

Next, it enters a loop that performs the multiplication:

4. LDA X: Load the value of X (result of A + B) into the accumulator.
5. SZA: Skip the next instruction if the accumulator is zero (checks if X is zero).
6. BUN Load_Y: Branch unconditionally to the Load_Y label if X is not zero (this starts the multiplication loop).
7. BUN END: Branch unconditionally to the END label if X is zero (indicating the multiplication is complete).

Inside the Load_Y loop:

8. LDA Y: Load the value of Y into the accumulator.

9. SZA: Skip the next instruction if the accumulator is zero (checks if Y is zero).

10. BUN MUL: Branch unconditionally to the MUL label if Y is not zero (continues the multiplication).

11. BUN END: Branch unconditionally to the END label if Y is zero (indicating the multiplication is complete).

In the MUL loop:

12. LDA Result: Load the value of the result into the accumulator.

13. ADD X: Add the value of X to the accumulator.

14. STA Result: Store the result of the addition back into the Result memory location.

15. LDA Mul_Counter: Load the value of the Mul_Counter into the accumulator.

16. INC: Increment the accumulator (increment the counter by 1).

17. STA Mul_Counter: Store the updated counter value back in Mul_Counter.

18. CMA: Complement the accumulator (invert all bits).

19. AND Y: Perform a bitwise AND operation between the accumulator and Y.

20. SZA: Skip the next instruction if the accumulator is zero (checks if Y and the complemented accumulator result in zero).

21. BUN MUL: Branch unconditionally back to the MUL label to continue the multiplication loop.

This loop continues until the counter reaches zero (Mul_Counter becomes zero), which means the multiplication process is complete.

Finally, the program ends with:

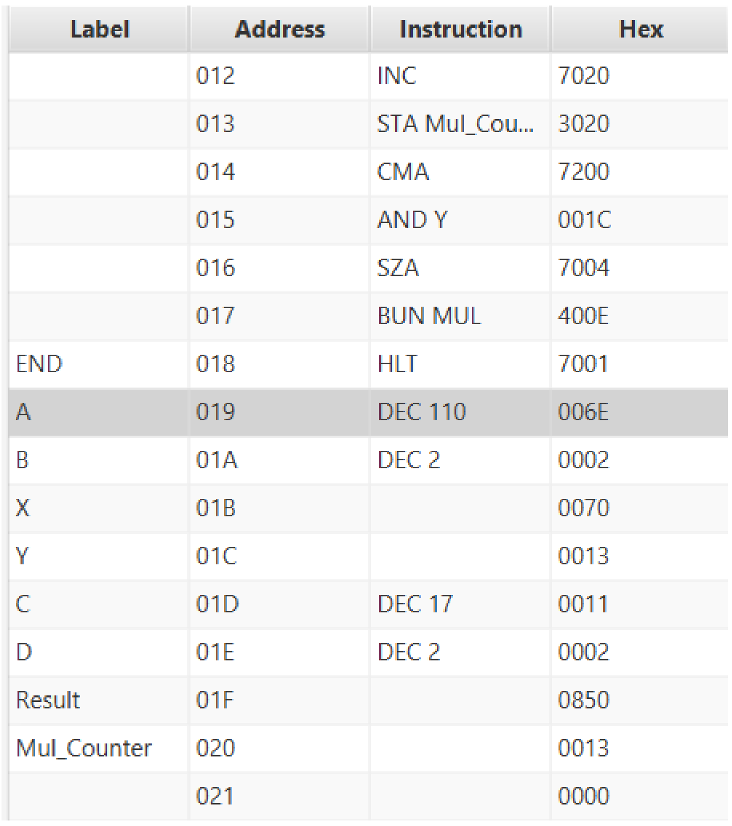
22. END, HLT: The program ends with the HLT (Halt) instruction.

The initial values of A, B, X, Y, C, D, Result, and Mul_Counter are provided as decimal values in the memory locations. The program calculates the result of multiplying A by C and stores it in the Result memory location using a loop-based multiplication algorithm.

Program:

```
LDA A
ADD B
STA X

LDA C
ADD D
STA Y



LDA X
SZA
BUN Load_Y
BUN END

Load_Y, LDA Y
SZA
BUN MUL
BUN END

MUL, LDA Result
ADD X
STA Result
LDA Mul_Counter
INC
STA Mul_Counter
CMA
AND Y
SZA
BUN MUL





END,HLT
A, DEC 110
B, DEC 2
X, DEC 0
Y, DEC 0
C, DEC 17
D, DEC 2
Result, DEC 0
Mul_Counter, DEC 0
```

Simulation:



Label	Address	Instruction	Hex
	012	INC	7020
	013	STA Mul_Cou...	3020
	014	CMA	7200
	015	AND Y	001C
	016	SZA	7004
	017	BUN MUL	400E
END	018	HLT	7001
A	019	DEC 110	006E
B	01A	DEC 2	0002
X	01B		0070
Y	01C		0013
C	01D	DEC 17	0011
D	01E	DEC 2	0002
Result	01F		0850
Mul_Counter	020		0013
	021		0000

Input:

A, DEC 110

B, DEC 2

X, DEC 0

Y, DEC 0

C, DEC 17

D, DEC 2

Result, DEC 0

Mul_Counter, DEC 0

Output:

Result= HEX 0850 =DEC 2128