

Name:

Advanced Programming in C++

Lab Exercise 4/21/2025

A Practice Exercise With MARIE

Machine Architecture that is Really Intuitive and Easy (MARIE)

1. Copy the entire folder named Marie from the \\Ada\data files\C++\ to your desktop.
2. Open the folder and double-click on MarieSim.jar (an executable Java Archive)
3. From the File Menu Select Edit to open up the Marie Assembler Code Editor
4. Type in the following program into the editor:

```

    ORG 100
    Load  Addr
    Store  Next
    Load  Num
    Subt   One
    Store  Ctr
    Clear
Loop, Load  Sum
    AddI   Next
    Store  Sum
    Load  Next
    Add    One
    Store  Next
    Load  Ctr
    Subt   One
    Store  Ctr
    Skipcond 800
    Jump   Loop
    Halt
Addr, Hex  118
Next, Hex  0
Num,  Dec  5
Sum,  Dec  0
Ctr,  Hex  0
One,  Dec  1
      Dec  10
      Dec  15
      Dec  20
      Dec  25
      Dec  30
```

5. Assemble the code by selecting Assemble/Assemble Current File from the Editor menu (Note you must Save your file first before you can Assemble).
6. Select Assemble/Show Assembly Listing and view what the assembler created. You should see an assembly listing that shows what is loaded into memory.
7. Record the range of memory that is used for this program. _____
8. Now load your assembled program into the MARIE simulator.
9. Step through your program until it is completed. If you are in a hurry, just Run the program.
10. Record the contents of the Accumulator, Instruction Register, Memory Address Register, Memory Buffer Register, and Program Counter:

AC _____
IR _____
MAR _____
MBR _____
PC _____

Now try look at some other programs. In the Marie folder, there are three example programs. You should open these in the MARIE Assembler Code Editor, examine them, assemble them and run them in MARIE.

Challenge:

1. Write a program that adds the numbers from 1 to 10.
2. Write a program that will allow the user to enter two numbers and displays the product of those two numbers.

- [illegible]

- | Instruction Number | | | |
|--------------------|-----|-------------|---|
| Binary | Hex | Instruction | Meaning |
| 0001 | 1 | Load X | Load contents of address X into AC. |
| 0010 | 2 | Store X | Store the contents of AC at address X. |
| 0011 | 3 | Add X | Add the contents of address X to AC. |
| 0100 | 4 | Subt X | Subtract the contents of address X from AC. |
| 0101 | 5 | Input | Input a value from the keyboard into AC. |
| 0110 | 6 | Output | Output the value in AC to the display. |
| 0111 | 7 | Halt | Terminate program. |
| 1000 | 8 | Skipcond | Skip next instruction on condition. |
| 1001 | 9 | Jump X | Load the value of X into PC. |

The diagram illustrates the internal components of a CPU and its connection to Main Memory. The CPU is represented by a large box containing several sub-components:

- ALU (Arithmetic Logic Unit):** A large, stylized 'Y' shape at the top left.
- AC (Accumulator):** A small box inside the ALU.
- MBR (Memory Buffer Register):** A box below the AC, connected to it by two vertical arrows.
- PC (Program Counter) and IR (Instruction Register):** Two boxes side-by-side within a larger box labeled **Control Unit** at the bottom.
- OutReg, InReg, and MAR (Memory Address Register):** Three boxes to the right of the ALU and MBR.

 The **Main Memory** is a large vertical rectangle to the right of the CPU. It is connected to the CPU via a horizontal bus. The memory stack is labeled with **Address 0** at the top and **Address 4095** at the bottom. Arrows indicate data flow: from OutReg to the bus, from the bus to InReg, and from the bus to MAR. The MAR is connected to the Main Memory.