Computer Architecture and Assembly Language (60394)

Assignment Report

Group Members:

- Asad Virani, 22787
- Danish Badar Qureshi, 22890

Topic: MARIE Implementation on Logisim

Introduction:

MARIE stands for Machine Architecture that is Really Intuitive and Easy. It is a simple computer architecture consisting of a Memory, Arithmetic and Logic Unit (ALU) and about seven registers.

1. **MAR**: Memory Address Register contains the address of the data that is being used.
2. **MBR**: Memory Buffer Register contains the read data, or data that has to be written in memory
3. **AC**: Accumulator holds that data that needs to be processed by the CPU (i.e. ALU) or the data resulting from a process by the CPU.
4. **PC:** Program Counter holds of address of the instruction to be executed.
5. **IR:** Instruction Register holds the instruction to be executed.
6. **InREG:** Input Register contains data from an input device
7. **OutREG:** Output Register contains data that has to be outputted.

In addition to the above registers, the **ALU** is the component responsible for executing all arithmetic operations like addition, multiplication, subtraction, etc. All the instructions and data are loaded in the **Main Memory** from which the instruction are processed through a Fetch-Decode-Execute cycle. Data is carried to and from components via a common 16-bit bus.

Logisim Simulation - Simplifications:

The submitted Logisim simulation of MARIE architecture takes into account some simplifications and modifications in the architecture. The basic premise is the same, where instruction is fetched from main memory, it is then decoded into opcode (type of instruction) and operands(s) (data or their address on which the instruction is performed), and then it is executed via the ALU (if required). However, this implementation omits the use of MBR and MAR, and the values and instructions are hard-coded onto main memory, which allows the exclusion of the InREG and OutREG as well. In addition, two memories are used to separate data values and instructions

All the instructions are of 16 bits, where the most significant 4 bits are the opcodes and the rest are operands. As instructed, the ISA contains four instructions only: Move, Add, Multiply, End. Because there are only four instructions, from a possible 16 (4 bits for opcode), only the first two least significant bits of the opcodes are mapped to the instructions.

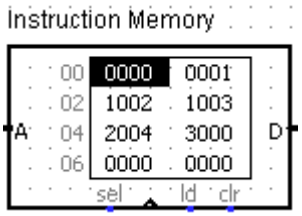
Logisim Simulation - Working:

The following table summarizes the mapping of each of the four MARIE instructions' syntax and explanations:

Opcode (4 bits)	Hex	Instruction (16bits)	Working	Example in Hex
0000	0	Move X	Load the contents of address X into AC	0001
0001	1	Add X	Add the data in address X into the data in AC, and store result in AC	1020
0010	2	Multiply X	Multiply value in address X with the value of AC and store result in AC	2003
0011	3	End	Terminate the program – reset PC and AC	3000

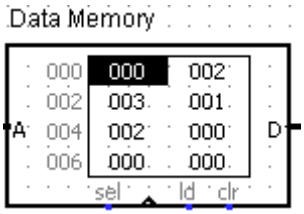
To demonstrate the working, the following instructions in hexadecimal are loaded in the instructions memory:

i. 0x0001
ii. 0x1002
iii. 0x1003
iv. 0x2004
v. 0x3000



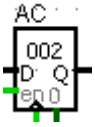
Simultaneously, the data memory is loaded with the following values:

i. 0x002
ii. 0x003
iii. 0x001
iv. 0x002



The following is the fetch-decode-execute cycle:

1. PC = 0 at the start
2. PC = 1, which is also the address of the first instruction in the instruction memory i.e 0001. At location 001 in data memory, the value 0x002 is stored which is MOVEd to the AC



3. PC = 2, so the next instruction is 1002. From location 002 of data memory, the value 0x003 is ADDed into the value in AC resulting in 0x005.

4. PC = 3, so the next instruction is 1003. From location 003 of data memory, the value 0x001 is ADDed into the value in AC resulting in 0x006.

5. PC = 4, so the next instruction is 2004. From location 004 of data memory, the value 0x002 is MULtiplied with the value in AC resulting in 0x00c.

6. PC = 5, so the next instruction is 3000. The program is terminated i.e. PC = 0 and AC = 0.

Screenshot: