MARIE: Architecture Example

CISC 3310 — CUNY Brooklyn College
Lecture Notes

MARIE: Architecture Example

In this chapter, we will introduce an example of a simple CPU architecture called MARIE to illustrate many of the concepts we covered in Topic 5.

MARIE: Intro

What is MARIE?

MARIE stands for **Machine Architecture that is Really Intuitive and Easy**.

It was developed by the computer scientist Linda Null as a simplified computer architecture for educational purposes.

The goal of MARIE is to provide an easy-to-understand model of a computer's inner workings, making it ideal for learning about basic computer architecture.

MARIE consists of basic components, registers, and a simple instruction set that illustrate how a computer processes instructions.

This model is widely used in computer science education to introduce students to core concepts in CPU design and instruction execution. This topic's purpose is to show how MARIE works!

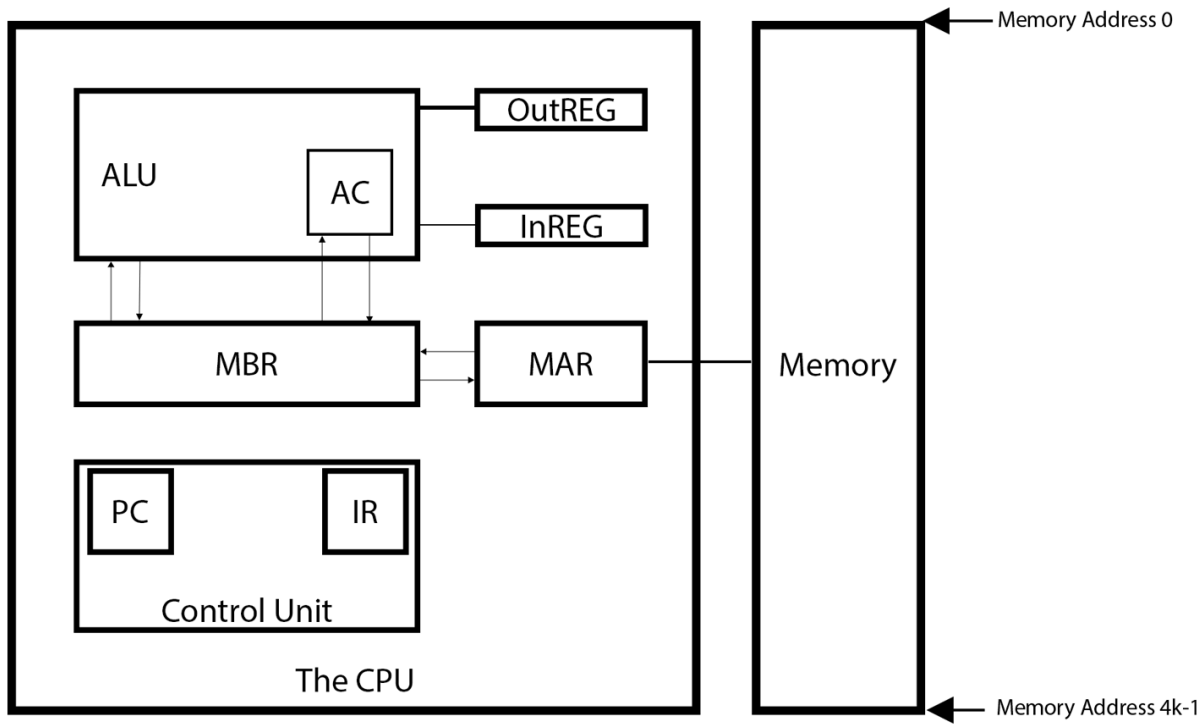
MARIE: Registers

MARIE has seven registers, each serving a unique purpose in instruction processing:

1. **Accumulator (AC)**: An 16-bit register used to store intermediate arithmetic and logic results.
2. **Memory Address Register (MAR)**: A 12-bit register that holds the address in memory where data will be read from or written to.
3. **Memory Buffer Register (MBR)**: A 16-bit register that holds data to be written or that as just read from memory.
4. **Program Counter (PC)**: A 12-bit register that contains the address of the next instruction to be executed.
5. **Instruction Register (IR)**: A 16-bit register that holds the current instruction being executed.
6. **Input Register (InREG)**: An 8-bit register used for receiving input from the user.
7. **Output Register (OutREG)**: An 8-bit register used for displaying output to the user.

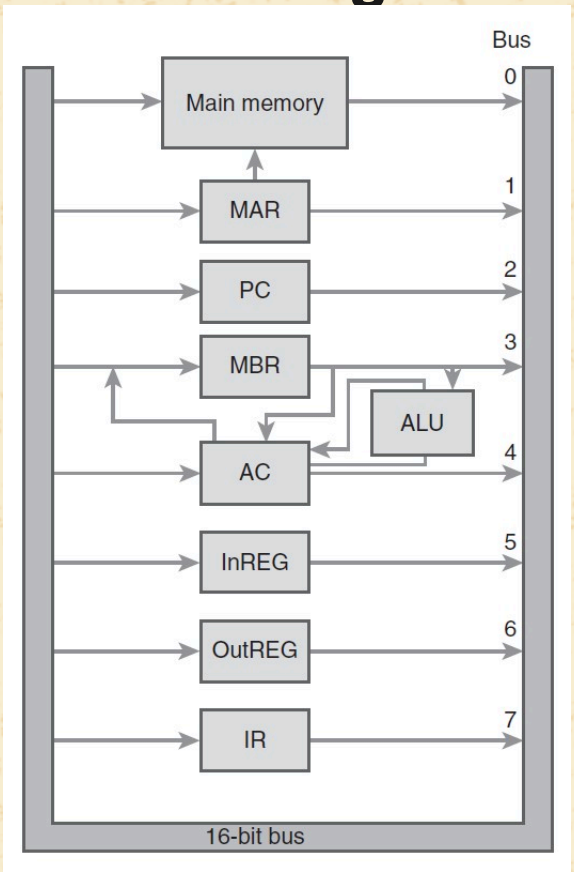
MARIE also has a **Status** register (probably 8-bit long) to keep track of various flags (e.g., zero, sign, etc.) Each register plays an important role in the **fetch–decode–execute** cycle and enables MARIE to carry out its operations efficiently.

MARIE: Registers



The MARIE architecture. Nyugen, Jason, et al. *Introduction to MARIE, A Basic CPU Simulator*. 2nd ed., 2016, *GitHub*.

MARIE: Registers



MARIE's bus. Figure 4.9 on page 244 of *Essentials of Computer Organization and Architecture* by Linda Null, 2023.

MARIE: Registers

Any questions so far?

MARIE: Instructions

Here is MARIE's instruction set (each instruction has an opcode and up to 1 operand:)

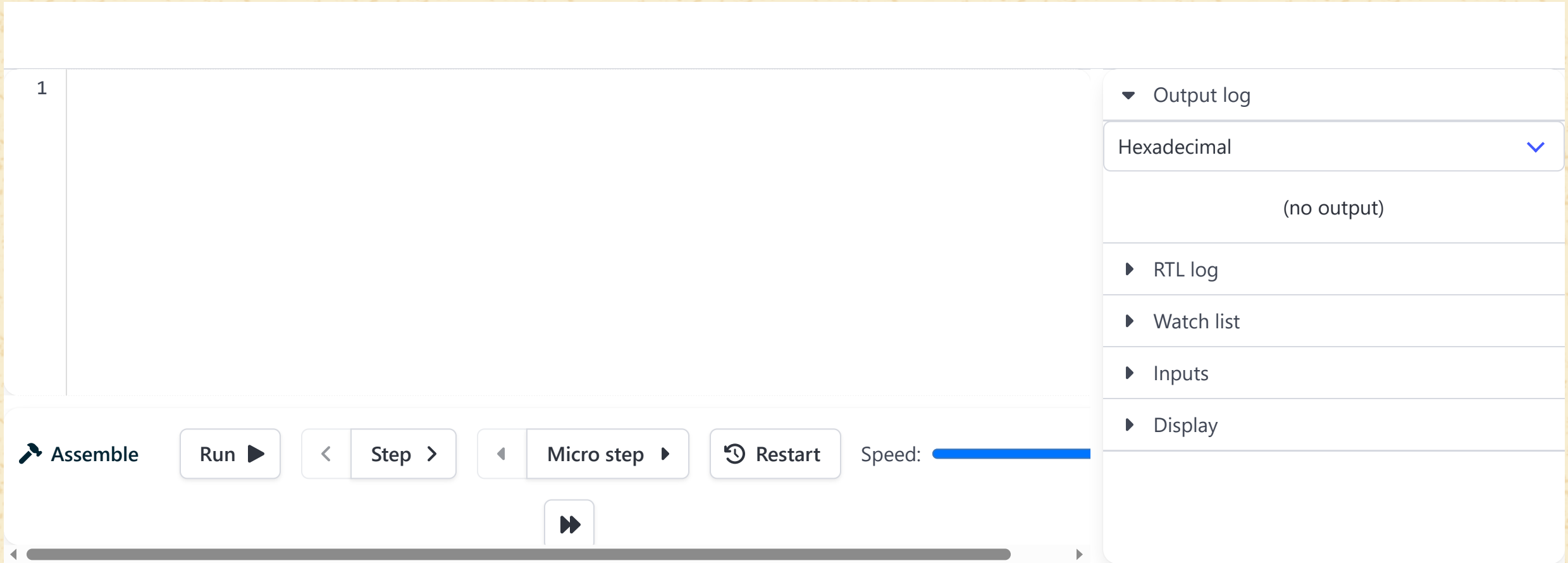
Opcode	Syntax	Description
0001	**LOAD X**	Loads the value at address X into the AC.
0010	**STORE X**	Stores the value in the AC into address X in memory.
0011	**ADD X**	Adds the value at address X to the AC.
0100	**SUBT X**	Subtracts the value at address X from the AC.
0101	**INPUT**	Takes input from the user and places it into the AC.
0110	**OUTPUT**	Outputs the value in the AC to the user.
0111	**HALT**	Stops the execution of the program.
1000	**SKIPCOND**	Skips the next instruction based on a condition in the AC.
1001	**JUMP X**	Sets the PC to X, causing the program to jump to the specified address.
1010	**JNS X**	Jumps to the instruction at address X.
1011	**JUMPI X**	Indirectly jumps to the address stored at address X.

MARIE: Instructions

Most of the instructions are explained pretty well, but we need to talk a bit more about `SKIPCOND`, which we use to create 3 types of `if` conditions: check if a number is zero, positive, or negative. `SKIPCOND` will look at the last 2 bits of the Instruction Register (IR), which we denote as `IR[11-10]`, and will behave according to the following pseudo-code:

```
if (IR[11-10] == 00) // If IR[11] = 0 and IR[10] = 0,
    if (AC < 0) // Check if AC is negative.
        PC = PC + 1;
else if (IR[11-10] == 10) // If IR[11] = 1 and IR[10] = 0,
    if (AC == 0) // Check if AC is zero.
        PC = PC + 1;
else if (IR[11-10] == 01) // // If IR[11] = 0 and IR[10] = 1,
    if (AC > 0) // Check if AC is positive.
        PC = PC + 1;
```

MARIE: Simulator



MARIE: Simulator

Any questions so far?

MARIE: Architecture Example: Sources

Nyugen, Jason, et al. *MARIE.js*. v2.3.0., https://marie.js.org/, Updated April 2026, *GitHub*.
License: The MIT License.