4.8 MARIE

We can now bring together many of the ideas that we have discussed to this point using a very simple model computer.

Our model computer, the Machine Architecture that is Really Intuitive and Easy, MARIE, was designed for the singular purpose of illustrating basic computer system concepts.

While this system is too simple to do anything useful in the real world, a deep understanding of its functions will enable you to comprehend system architectures that are much more complex.

MARIE's seven registers are:

- Accumulator, AC, a 16-bit register that holds a conditional operator (e.g., "less than") or one operand of a two-operand instruction.
- Memory address register, MAR, a 12-bit register that holds the memory address of an instruction or the operand of an instruction.
- Memory buffer register, MBR, a 16-bit register that holds the data after its retrieval from, or before its placement in memory.
- Program counter, PC, a 12-bit register that holds the address of the next program instruction to be executed.
- Instruction register, IR, which holds an instruction immediately preceding its execution.
- Input register, InREG, an 8-bit register that holds data read from an input device.
- Output register, OutREG, an 8-bit register, that holds data that is ready for the output device.
4.8 MARIE

This is the MARIE architecture shown graphically.

The registers are interconnected, and connected with main memory through a common data bus.
Each device on the bus is identified by a unique number set on the control lines whenever that device is required to carry out an operation.
Separate connections are also provided between the accumulator and the memory buffer register, and the ALU and the accumulator and memory buffer register.
This permits data transfer between these devices without use of the main data bus.

This is the MARIE data path shown graphically.

A computer’s instruction set architecture (ISA) specifies the format of its instructions and the primitive operations that the machine can perform.
The ISA is an interface between a computer’s hardware and its software.
Some ISAs include hundreds of different instructions for processing data and controlling program execution.
The MARIE ISA consists of only thirteen instructions.

This is the format of a MARIE instruction:

The fundamental MARIE instructions are:

<table>
<thead>
<tr>
<th>Instruction/Number</th>
<th>Binary</th>
<th>Hex</th>
<th>Instruction</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>0001</td>
<td>1</td>
<td>Load X</td>
<td>Load contents of address X into AC</td>
</tr>
<tr>
<td>0010</td>
<td>0010</td>
<td>2</td>
<td>Store X</td>
<td>Store the contents of AC at address X</td>
</tr>
<tr>
<td>0011</td>
<td>0011</td>
<td>3</td>
<td>Add X</td>
<td>Add the contents of address X to AC</td>
</tr>
<tr>
<td>0100</td>
<td>0100</td>
<td>4</td>
<td>Sub X</td>
<td>Subtract the contents of address X from AC</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
<td>5</td>
<td>Input</td>
<td>Input a value from the keyboard into AC</td>
</tr>
<tr>
<td>1001</td>
<td>1001</td>
<td>6</td>
<td>Output</td>
<td>Output the value of AC to the display</td>
</tr>
<tr>
<td>1100</td>
<td>1100</td>
<td>7</td>
<td>Halt</td>
<td>Halt terminate program</td>
</tr>
<tr>
<td>1101</td>
<td>1101</td>
<td>8</td>
<td>Skip</td>
<td>Skip next instruction on condition</td>
</tr>
<tr>
<td>1110</td>
<td>1110</td>
<td>9</td>
<td>Jump</td>
<td>Jump to the value of X into PC</td>
</tr>
</tbody>
</table>

This is a bit pattern for a LOAD instruction as it would appear in the IR:

We see that the opcode is 1 and the address from which to load the data is 3.
This is a bit pattern for a **SKIPCOND** instruction as it would appear in the IR:

<table>
<thead>
<tr>
<th>opcode</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0</td>
<td></td>
</tr>
</tbody>
</table>

We see that the opcode is 8 and bits 11 and 10 are 1 0, meaning that the next instruction will be skipped if the value in the AC is greater than zero.

What is the hexadecimal representation of this instruction?

Each of our instructions actually consists of a sequence of smaller instructions called **microoperations**.

The exact sequence of microoperations that are carried out by an instruction can be specified using **register transfer language (RTL)**.

In the MARIE RTL, we use the notation M[X] to indicate the actual data value stored in memory location X, and ← to indicate the transfer of bytes to a register or memory location.

The RTL for the **LOAD** instruction is:

- MAR ← X
- MBR ← M[MAR]
- AC ← MBR

Similarly, the RTL for the **ADD** instruction is:

- MAR ← X
- MBR ← M[MAR]
- AC ← AC + MBR

Recall that **SKIPCOND** skips the next instruction according to the value of the AC.

The RTL for this instruction is the most complex in our instruction set:

If IR[11-10] = 00 then
- If AC < 0 then PC ← PC + 1
else if IR[11-10] = 01 then
- If AC = 0 then PC ← PC + 1
else if IR[11-10] = 11 then
- If AC > 0 then PC ← PC + 1