Computer Architecture
(CSC-3501)
Lecture 11
(21 Feb 2008)

Seung-Jong Park (Jay)
http://www.csc.lsu.edu/~sjpark

Announcement

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.

4.8 MARIE

The MARIE architecture has the following characteristics:

- Binary, two's complement data representation.
- Stored program, fixed word length data and instructions.
- 4K words of word-addressable main memory.
- 16-bit data words.
- 16-bit instructions, 4 for the opcode and 12 for the address.
- A 16-bit arithmetic logic unit (ALU).
- Seven registers for control and data movement.

4.8 MARIE

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.

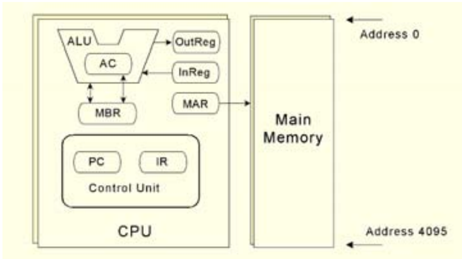
4.8 MARIE

MARIE's seven registers are:

- 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.

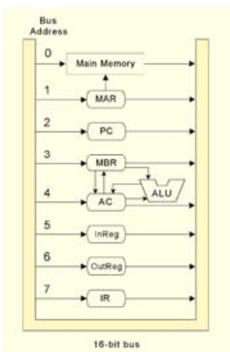


4.8 MARIE

- 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 that is 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.

4.8 MARIE

This is the MARIE data path shown graphically.

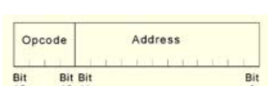


4.8 MARIE

- 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.

4.8 MARIE

- This is the format of a MARIE instruction:

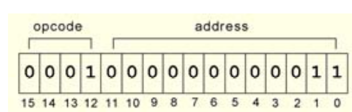


- The fundamental MARIE instructions are:

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.

4.8 MARIE

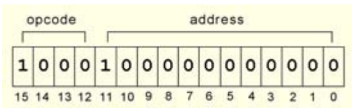
- This is a bit pattern for a **LOAD** instruction as it would appear in the IR:



- We see that the opcode is 1 and the address from which to load the data is 3.

4.8 MARIE

- This is a bit pattern for a **SKIPCOND** instruction as it would appear in the IR:



- We see that the opcode is 8 and bits 11 and 10 are 10, 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?

4.8 MARIE

- 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.

4.8 MARIE

- 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
```


4.8 MARIE

- Recall that **SKIPCOND** skips the next instruction according to the value of the AC.
- The RTL for the 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
```