Topic for today:

More on instructions and
Instruction Set Architecture (ISA)

Recall: Structure of an instruction

An instruction consists of an operation code (*opcode*) and an address field. The opcode specifies the operation to be carried out. The address field typically specifies the address of data that will be used in the operation (although there is much variability in how the address field can be used).

Introducing MARIE

- MARIE = Machine Architecture that is Really Intuitive and Easy
- Very simple model computer.
- Used as a learning tool – too simple to be used for all tasks in the real world

MARIE Architecture

- 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.

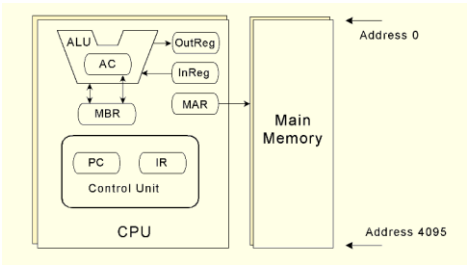
MARIE's seven registers

- 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.

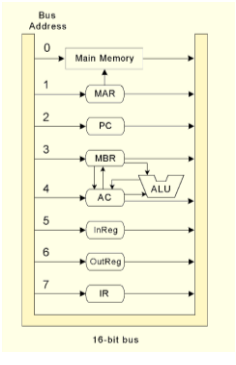
MARIE's seven registers

- 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.

MARIE Architecture



MARIE Datapath



8

Note

The MARIE instruction set has a *1-address* format. That means:

- A typical instruction contains an opcode and the address of an operand
- For binary operations, the location of the second operand and the destination of the result are both assumed to be the *accumulator*

MARIE Instructions

- This is the format of a MARIE instruction:

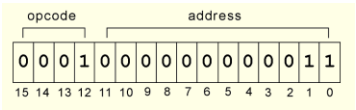
- The fundamental MARIE instructions are:

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.

10

MARIE Instruction Example

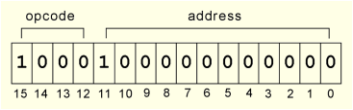
- This is a bit pattern for a **`LOAD`** instruction as it would appear in the IR:



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

MARIE Instruction Example

- 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?

12

Fetch-Decode-Execute cycle

The basic cycle of the CPU is:

- **Fetch** the next instruction from the memory location specified by the Program Counter (PC) into the Instruction Register (IR); increment the PC
- **Decode** the instruction's opcode
- **Execute** the instruction

This process repeats continuously.