جامعة ألأنبار

كلية علوم الحاسوب وتكنولوجيا المعلومات

قسم أنظمة شبكات الحاسوب

المرحلة الثانيه

Computer Architecture

التدريسي: أ.م.د عمر منذر حسين

4.4 The Input / Output Subsystem

- Input and output (I/O) devices allow us to communicate with the computer system.
- I/O is the transfer of data between primary memory and various I/O peripherals.
- Input devices such as keyboards, mice, card readers, scanners, voice recognition systems, and touch screens allow us to enter data into the computer.
- Output devices such as monitors, printers, plotters, and speakers allow us to get information from the computer.
- These devices are not connected directly to the CPU. Instead, there is an interface that handles the data transfers.
- This interface converts the system bus signals to and from a format that is acceptable to the given device.
- The CPU communicates to these external devices via input/output registers. This exchange of data is performed in two ways.
- In memorymapped I/O, the registers in the interface appear in the computer′s memory map and there is no real difference between accessing memory and accessing an I/O device. Clearly, this is advantageous from the perspective of speed, but it uses up memory space in the system.
- With instruction-based I/O, the CPU has specialized instructions that perform the input and output. Although this does not use memory space, it requires specific I/O instructions, which implies it can be used only by CPUs that can execute these specific instructions.
- Interrupts play a very important part in I/O, because they are an efficient way to notify the CPU that input or output is available for use.

4.5 MEMORY ORGANIZATION

- You can envision memory as a matrix of bits. Each row, implemented by a register, has a length typically equivalent to the addressable unit size of the machine.
- Each register (more commonly referred to as a memory location) has a unique address; memory addresses usually start at zero and progress upward. Figure 4.4 illustrates this concept.

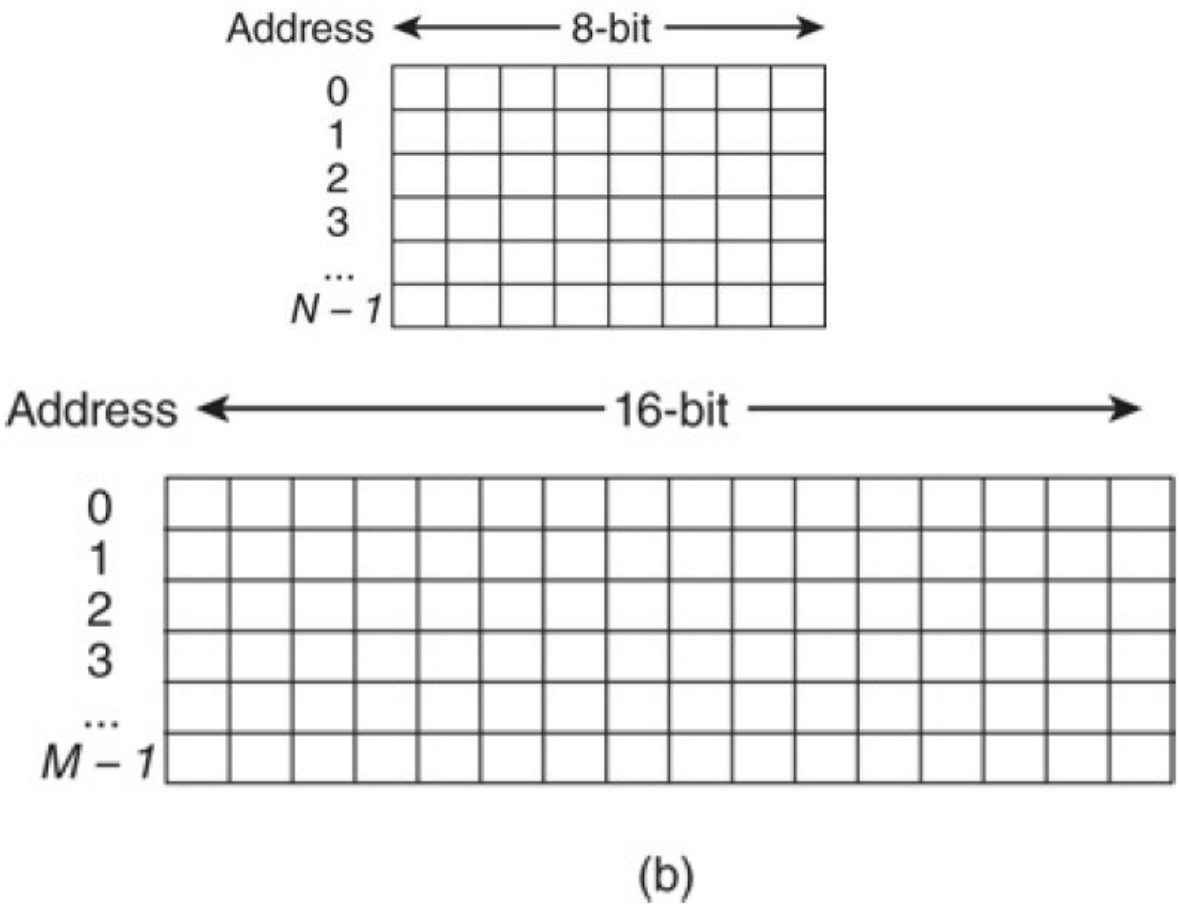


FIGURE 4.4 (a) N 8-Bit Memory Locations
(b) M 16-Bit Memory Locations

- An address is typically represented by an unsigned integer.

- Normally, memory is **byte addressable**, which means that each individual byte has a unique address.
- Some machines may have a word size that is larger than a single byte.
- For example, a computer might handle 32-bit words (which means it can manipulate 32 bits at a time through various instructions and it uses 32-bit registers) but still employ a byte-addressable architecture.
- In this situation, when a word uses multiple bytes, the byte with the lowest address determines the address of the entire word.

4.6 MARIE Architecture

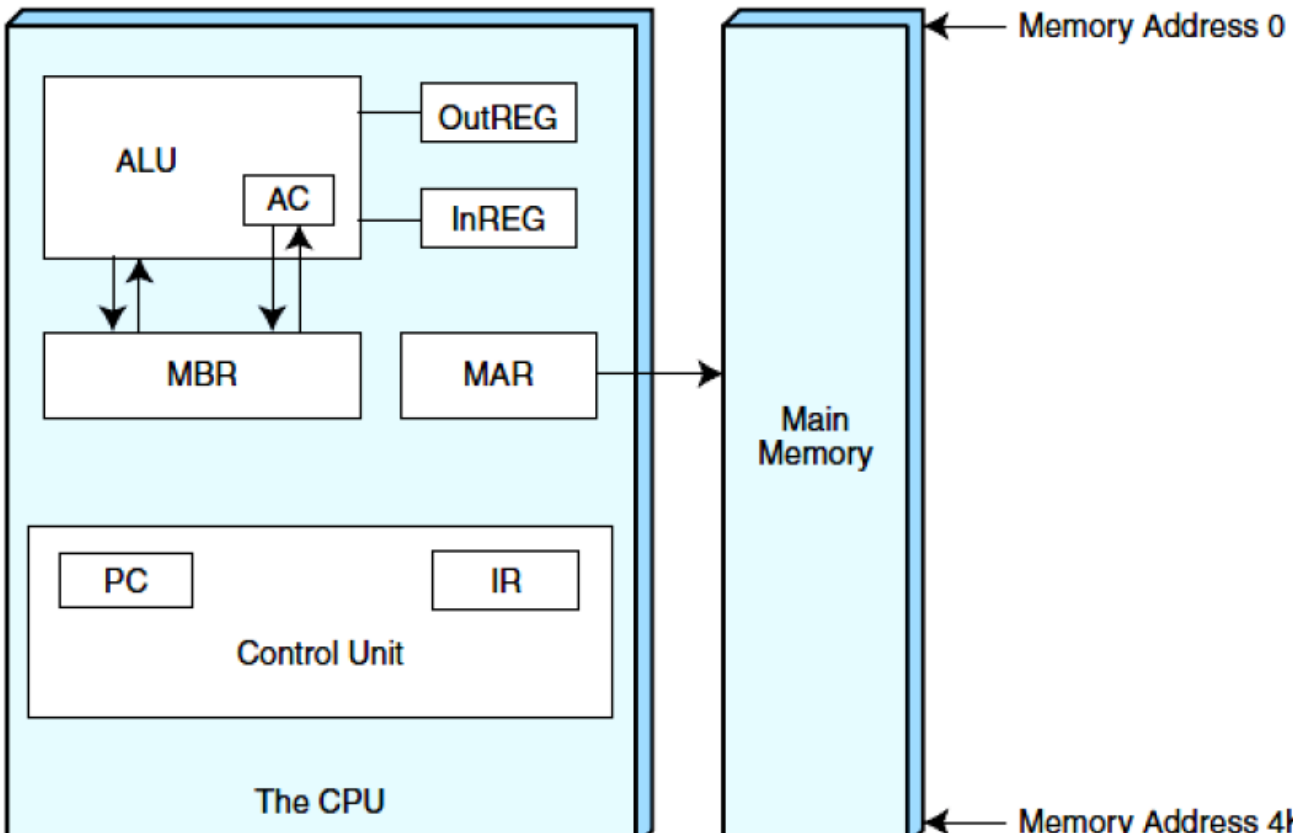
- MARIE, a Machine Architecture that is Really Intuitive and Easy, is a simple architecture consisting of memory (to store programs and data) and a CPU (consisting of an ALU and several registers).
- It has all the functional components necessary to be a real working computer.
- We describe MARIE′s architecture in the following sections.

4.6.1 The Architecture.

- MARIE has the following characteristics:
  • Binary, two’s complement.
  • Stored program, fixed word length.
  • Word (but not byte) addressable.
  • 4K words of main memory (this implies 12 bits per address).
  • 16-bit data (words have 16 bits).
  • 16-bit instructions, 4 for the opcode and 12 for the address.
  • A16-bit accumulator (AC).
  • A16-bit instruction registers (IR).

- A16-bit memory buffer register (MBR).
- A12-bit program counter (PC).
- A12-bit memory address register (MAR).
- An 8-bit input register.
- An 8-bit output register.

- We emphasize again that each location in memory has a unique address (represented in binary) and each location can hold a value.
- These notions of the address versus what is actually stored at that address tend to be confusing.
- To help avoid confusion, just visualize a post office.
- There are post office boxes with various "addresses" or numbers. Inside the post office box, there is mail. To get the mail, the number of the post office box must be known. The same is true for data or instructions that need to be fetched from memory.
- The contents of any memory address are manipulated by specifying the address of that memory location.
- We shall see that there are many different ways to specify this address.



4.6.2 Registers and Buses

- In MARIE, there are seven registers, as follows:
- **AC: The accumulator**, which holds data values. This is a general purpose register and holds data that the CPU needs to process.
- **MAR: The memory address register**, which holds the memory address of the data being referenced.
- **MBR: The memory buffer register**, which holds either the data just read from memory or the data ready to be written to memory.
- **PC: The program counter**, which holds the address of the next instruction to be executed in the program.
- **IR: The instruction register**, which holds the next instruction to be executed.
- **InREG: The input register**, which holds data from the input device.
- **OutREG: The output register**, which holds data for the output device.

4.6.3 The Instruction Set Architecture.

- MARIE has a very simple, yet powerful, instruction set.

- The instruction set architecture (ISA) of a machine specifies the instructions that the computer can perform and the format for each instruction.
- The ISA is essentially an interface between the software and the hardware.
- Some ISAs include hundreds of instructions.
- We mentioned previously that each instruction for MARIE consists of 16 bits.
- The most significant 4 bits, bits 12–15, make up the opcode that specifies the instruction to be executed (which allows for a total of 16 instructions).
- The least significant 12 bits, bits 0–11, form an address, which allows for a maximum memory size of 2^{12}-1.
- The MARIE ISA consists of only thirteen instructions.
- This is the format of a MARIE instruction:

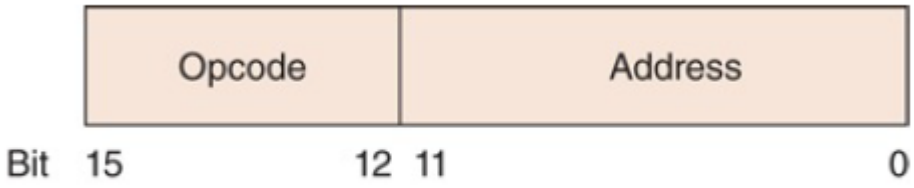


FIGURE 4.10 MARIE's Instruction Format

- The fundamental MARIE instructions are:

Instruction Number		Instruction	Meaning
Bin	Hex		
0001	1	Load *X*	Load the 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 and store the result in AC.
0100	4	Subt *X*	Subtract the contents of address *X* from AC and store the result in 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 the program.
1000	8	Skipcond	Skip the next instruction on condition.
1001	9	Jump *X*	Load the value of *X* into PC.

- We see that the opcode is 1 and the address from which to load the data is 000000000011
- This is a bit pattern for a SKIPCOND instruction as it would appear in the IR: