Lecture 9:

Computer Hardware View – the Stored Program Computer or the von Neumann Architecture

- Most modern computers have the same basic layout, known as the ***stored program computer*** or the ***von Neumann architecture***.

A. The von Neumann Architecture

- **Hardware components:**
  - CPU
  - Memory
  - Input/output devices

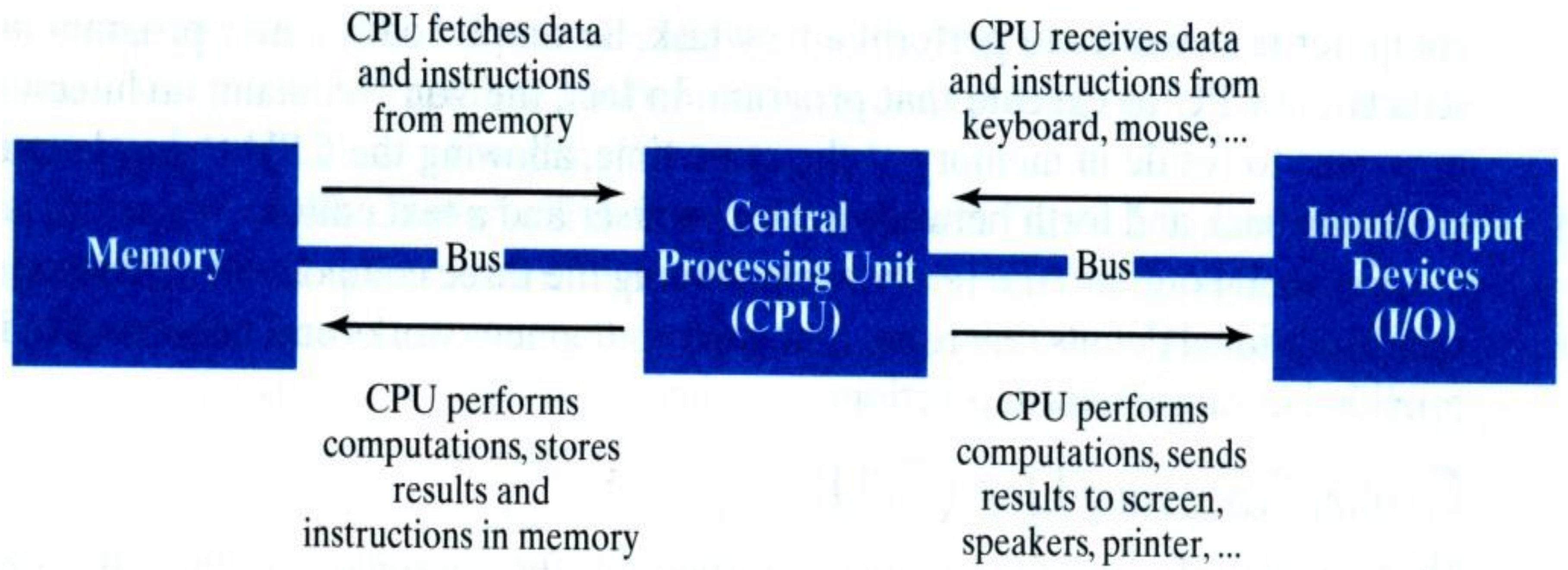
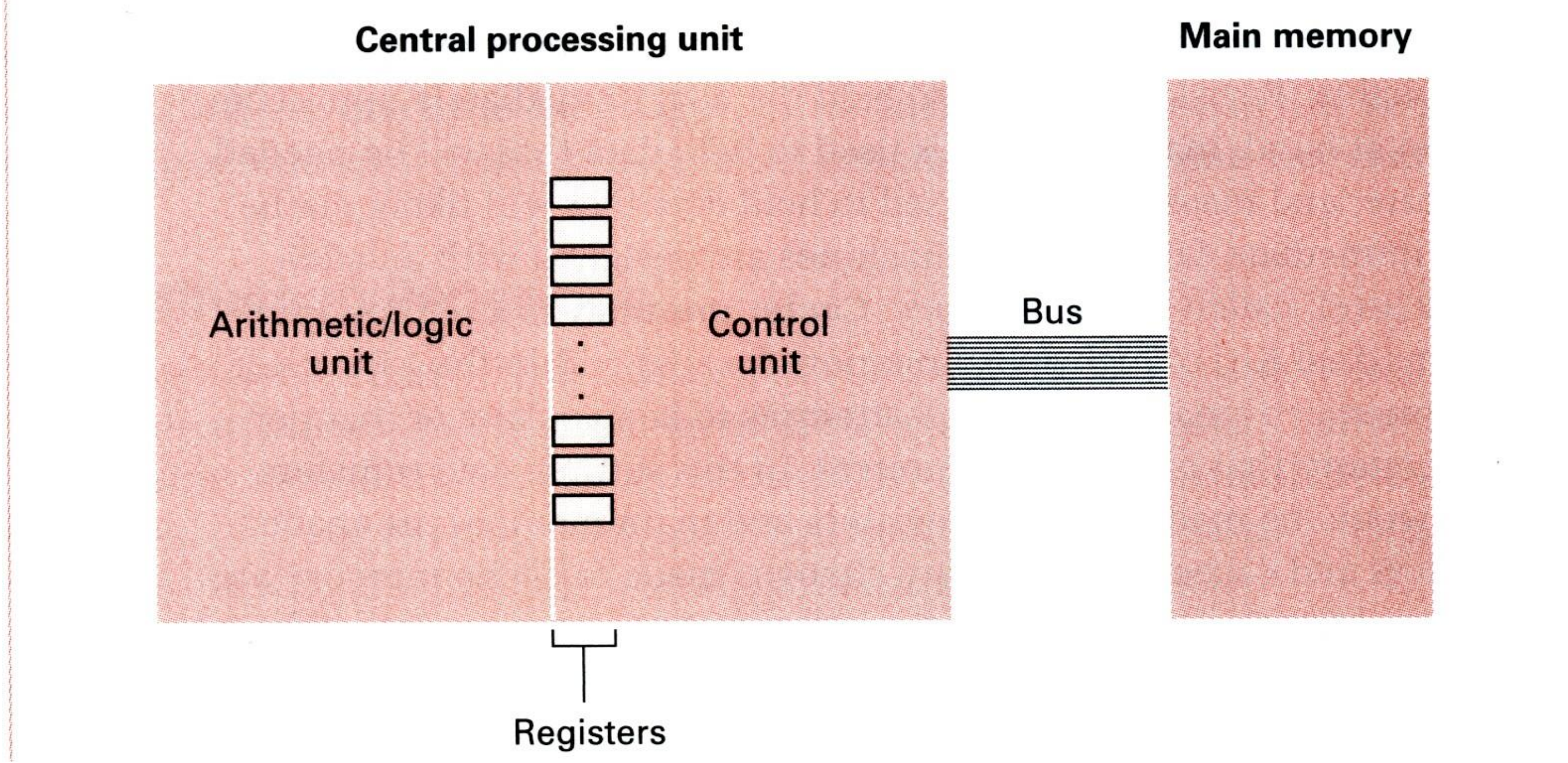

CPU fetches data and instructions from memory
CPU receives data and instructions from keyboard, mouse, ...
Memory
Bus
Central Processing Unit (CPU)
Bus
Input/Output Devices (I/O)
CPU performs computations, stores results and instructions in memory
CPU performs computations, sends results to screen, speakers, printer, ...

Figure 1.4 The von Neumann architecture.

- **CPU with its Subunits and Main Memory**

Figure 2.1 CPU and main memory connected via a bus



(Brookshear Figure 2.1)

- **Arithmetic/logic unit (ALU):** performs all arithmetic and logic operations (or data processing) – i.e. it is responsible for all calculations

- **Control unit (CU)**: controls and coordinates all CPU operations:

  1) reads instructions from memory (to the registers)
  2) decodes (interprets) the instructions
  3) executes the instructions by activating the appropriate circuitry within ALU (ALU does the actual operation).

- **Registers:** temporary storage cells for data and instructions for future use by CPU, including inputs from main memory & results produced by the CPU.

- **Bus:** circuits connecting CPU and main memory, used to transfer data and instructions between the CPU and the main memory.

- **Main memory:** consisting of an array of cells to store the programs (instructions) and the data.

B. Stored Program Concept

- Data that the CPU is processing is stored in the memory.

- Instructions (programs) for the CPU are also stored in the memory just like data.

  One of the most important features of a stored program computer is being able to store the programs as well as the data together in the memory.

  Today's computers are all **stored program computers**.

In the early days (before 1948), computers built were non-stored program computers. With the non-stored program computers, the program was built into the control unit (hardware). Once it was built, it could not be changed easily. Usually, it was built to solve one specific problem (e.g. music boxes, calculators, etc.). To solve another application problem, one needs to rewire the hardware, which is a very difficult task. (Today's computers can run different programs for different applications.)

Turing introduced the idea of stored-program concept. However, von Neumann was the one who published the idea.

C. How CPU Performs Complex Computation Task

CPU breaks all instructions into sequences of very simple executions. For example, adding two numbers (w = x + y) might take CPU 4 steps to finish:

1. Obtain first value (x) from memory and store it in a register.
2. Obtain second value (y) from memory and store it in a second register.
3. Use the addition circuitry (in ALU) to add the two numbers in the registers and place the result in another register.
4. Store the result (w) back to the main memory.

CPU can execute billions of simple operations per second. CPU clock frequency measures its speed.

D. Detailed Discussion of the Execution of a Program

All instructions for a CPU are stored in the memory in the **machine language** format (binary code) that corresponds to the circuitry of that chip.

However, programmers write their code in a **high-level language** that is easy to understand, like HTML, C++ or Javascript.

The program is *translated* into **machine language** by a program called **compiler** or **interpreter**. (This should be a review of topics covered in previous lecture.)

So:

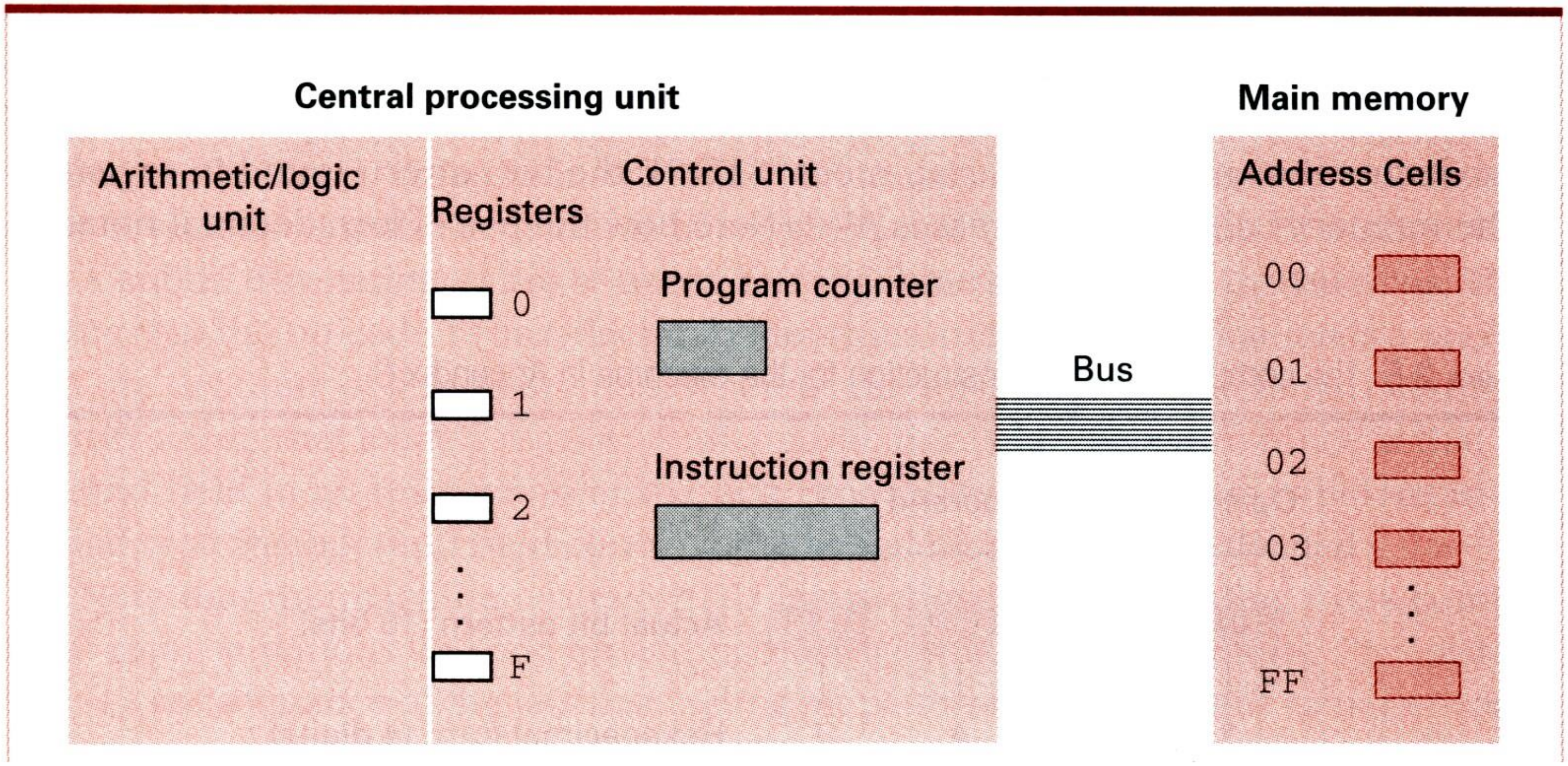
- A user writes a program in high-level computer language.
- The program is translated into machine language.
- The binary code is stored in the main memory with the data and is then executed by the CPU.

E. The Control Unit in the CPU Has Two Important Registers:

- The **program counter (PC)** – contains the address of the next instruction in main memory, i.e. it points to the next instruction in the main memory

- The **instruction register (IR)** – holds the instruction that is currently being executed.

Figure 2.4 The architecture of the machine described in Appendix C



(Brookshear Figure 2.4)

F. The CPU Cycle: Fetch – Decode – Execute

The CPU executes any program by going through the 3 steps of <u>CPU cycle</u> repeatedly:

1. The control unit (CU) **<u>fetches</u>** the instruction, pointed to by the program counter (PC), from main memory. The instruction is stored in the instruction register (IR). The PC is then updated to point to the next instruction in the main memory.

2. The control unit (CU) **decodes** the instruction in IR to find out what operation is to be performed. If operand(s) is/are required, it/they will then be loaded to the register(s) in the CPU.

3. The control unit (CU) then signals to arithmetic/logic unit (ALU) to **execute** the instruction. CU does so by activating the circuits in ALU needed to execute the instruction.

The above three steps (CPU cycle) are repeated for *each instruction* in the program until the end of the program is reached. Usually, the commands are executed one after another. But sometimes, the program contains a **jump** instruction which takes the program to another location in the main memory. This allows the CPU to carry out a **loop** construct where a set of instructions are executed repeatedly.

A computer can execute millions of instructions per second.