CS1101: Lecture 13

Computer Systems Organization: Processors

Dr. Barry O'Sullivan
b.osullivan@cs.ucc.ie

Course Homepage
http://www.cs.ucc.ie/~osullb/cs1101

Lecture Outline

- The CPU
- A Simple Computer
- The Structure of the CPU
- The CPU: Registers
- CPU Organization: The Data Path
- The ALU
- Instruction Execution
- RISC versus CISC
- Design Principles for Modern Computers
- **Reading**: Tanenbaum, Chapter 2.

The CPU

- The **CPU (Central Processing Unit)** is the "brain" of the computer.
- Its function is to execute programs stored in the main memory by fetching their instructions, examining them, and then executing them one after another.
- The components are connected by a **bus**, which is a collection of parallel wires for transmitting address, data, and control signals.
- Buses can be external to the CPU, connecting it to memory and I/O devices, but also internal to the CPU.

A Simple Computer

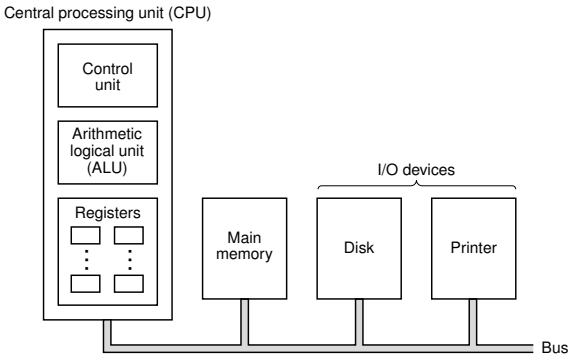

Figure 2-1. The organization of a simple computer with one CPU and two I/O devices.

The Structure of the CPU

- The CPU is composed of several distinct parts.
- The **control unit** is responsible for fetching instructions from main memory and determining their type.
- The **logic unit** performs operations such as addition and Boolean AND to carry out the instructions.
- The CPU also contains a small, high-speed memory used to store temporary certain control information.
- This memory is made up of a number of **registers**, each of which has a certain size and function.
- Usually, all the registers have the same size.

The CPU: Registers

- Each register can hold one number, up to some maximum determined by the size of the register.
- Registers can be read and written at high speed since they are internal to the CPU.
- The most important register is the **Program Counter** (PC), which points to the next instruction to be fetched for execution.
- The name ”program counter” is somewhat misleading because it has nothing to do with counting anything, but the term is universally used.
- Also important is the **Instruction Register** (IR), which holds the instruction currently being executed.

CPU Organization: The Data Path

- The **data path** consists of the registers (typically 1 to 32), the **ALU (Arithmetic Logic Unit)**, and several buses connecting the pieces.
- In the next slide a typical example is presented.
- The registers feed into two ALU input registers, labeled A and B in the figure.
- These registers hold the ALU input while the ALU is computing.
- The data path is very important in all machines

The Data Path

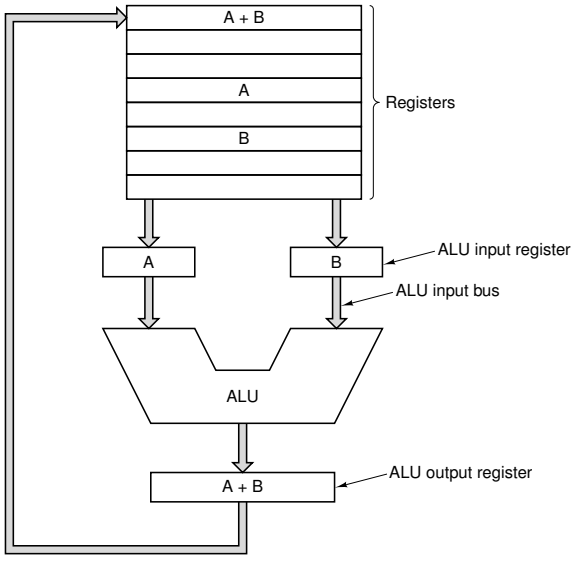

Figure 2-2. The data path of a typical Von Neumann machine.

The ALU

- The ALU performs addition, subtraction, and other simple operations on its inputs, thus yielding a result in the output register.
- This output register can be stored back into a register.
- Later on, the register can be written (i.e., stored) into memory, if desired.
- Not all designs have the A, B, and output registers.
- Most instructions can be divided into one of two categories: register-memory or register-register.
- The process of running two operands through the ALU and storing the result is called the **data path cycle**.
- The faster the data path cycle is, the faster the machine runs.

Instruction Execution

- The CPU executes each instruction in a series of small steps.
- Roughly speaking, the steps are as follows:
  1. Fetch the next instruction from memory into the instruction register.
  2. Change the program counter to point to the following instruction.
  3. Determine the type of instruction just fetched.
  4. If the instruction uses a word in memory, determine where it is.
  5. Fetch the word, if needed, into a CPU register.
  6. Execute the instruction.
  7. Go to step I to begin executing the following instruction.
- This sequence of steps is frequently referred to as the **fetch-decode-execute cycle**.

RISC versus CISC

- **RISC** = **Reduced Instruction Set Computer**
- **CISC** = **Complex Instruction Set Computer**
- The RISC community argue that the best way to design a computer was to have a small number of simple instructions that execute in one cycle of the data path
- Their argument was that even if a RISC machine takes four or five instructions to do what a CISC machine does in one instruction, if the RISC instructions are 10 times as fast (because they are not interpreted), RISC wins.
- Why haven't RISC systems "taken over the world"?
  1. backward compatibility
  2. Intel has been able to combine RISC and CISC architectures.

Design Principles for Modern Computers

There is a set of design principles, sometimes called the RISC design principles, that architects of general-purpose CPUs do their best to follow:

- All Instructions Are Directly Executed by Hardware
  - eliminates a level of interpretation
- Maximise the Rate at Which Instructions are Issued
  - **MIPS** = millions of instructions per second
  - MIPS speed related to the number of instructions issued per second
  - Parallelism can play a role

Design Principles for Modern Computers

- Instructions Should be Easy to Decode
  - a critical limit on the rate of issue of instructions
  - make instructions regular, fixed length, with a small number of fields.
  - the fewer different formats for instructions. the better.
- Only Loads and Stores Should Reference Memory
  - operands for most instructions should come from- and return to- registers.
  - access to memory can take a long time
  - thus, only LOAD and STORE instructions should reference memory.
- Provide Plenty of Registers
  - accessing memory is relatively slow, many registers (at least 32) need to be provided, so that once a word is fetched, it can be kept in a register until it is no longer needed.