CS2303 In Class Exercises

February 25, 2020

Abstract

Today's goal is for you to understand a description of the parts of a computer (at a high level of abstraction), gain more insight into the stack, and introduce assembly language.

Name and WPI email: Kush Shah kshah2@wpi.edu

1 Parts of a Computer

There are parts of a computer that are 9 nanometers in size, so we will not describe all of the parts. We will however, describe chunks of a computer. The central processing unit (CPU), the main memory, the peripherals and the bus are relatively large chunks.

1.1 Bus

The bus interconnects the main parts.

1.1.1 Address Bus

The address bus conveys the address (same use of the word as we have been using with pointers) throughout the system. Nominally, this bus is driven by the CPU and listened to by everything else. There are peripherals (things on the bus), namely direct memory access (DMA) controllers that drive the address bus from time to time.

1.1.2 Data Bus

When data is being sent from one device to another, such as a write to memory by the CPU, or a read from memory by the CPU, it is the data bus that conveys the bits.

1.1.3 Control Bus

The control bus carries signals such as whether a read or a write is occurring, and gives the timing of validity of data and of address values.

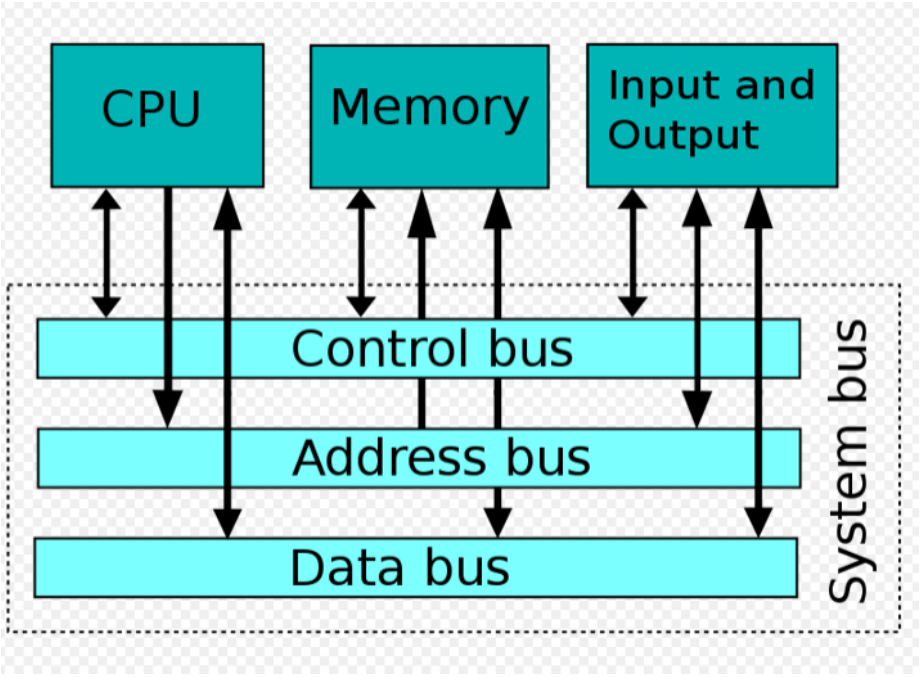

Figure 1: The central processing unit, the main memory and other input/output devices (including disk controllers) are connected to each other by the bus.

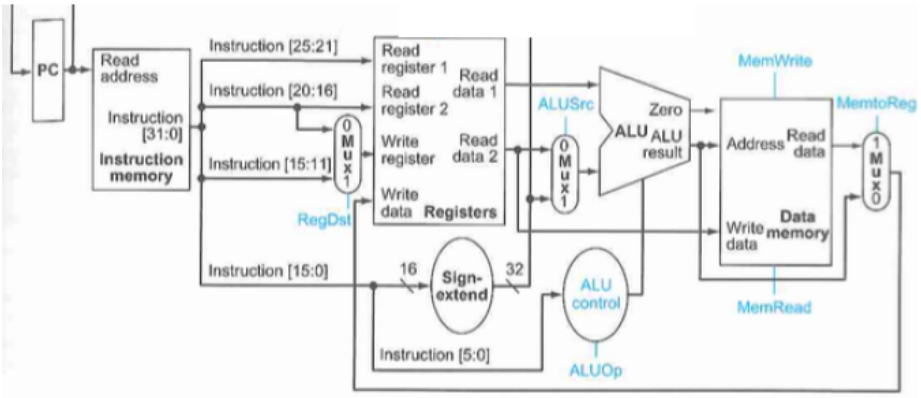

Figure 2: The CPU has a local memory called the register file, and a computational resource called the arithmetic logic unit. In the right hand side of the diagram, a relatively small box represents a relatively large component that is not insie the CPU. It is the main memory.

1.2 CPU

Let's look inside the CPU.

It has various registers, some isolated, some collected in a group. Registers are memory. A RAM chip has many registers, sharing an address bus, activated by a particular distinct value of the address bus, and able to read and write (at separate times) from the databus.

A register file can be thought of as a small RAM chip, but with multiple address buses and multiple databuses. Two registers can be addressed at once, when reading, and each drives one of the databuses. One register can be written at a time.

Which registers of the register file are being used for which function is something that must be deduced from the instruction.

1.3 Instruction Register

One kind of information that is read by the CPU from the main memory is an instruction. Instructions are generated by the compiler, from our code. When an executable is loaded into memory, it becomes available for processing by the CPU. Another program, called the scheduler, decides the order in which executables obtain CPU-time, and how much time. The scheduler enqueues executables. Yet another program, the dispatcher, dequeues the executable and arranges for the executable to be processed by a CPU (or CPUs).

So, we have the instructions in memory, and we have a CPU to use. The CPU reads from the memory, and receives instructions. One by one, instructions go into an instruction register. Once in the instruction register, the contents of the instruction can be used to control what happens in the CPU, as the instruction

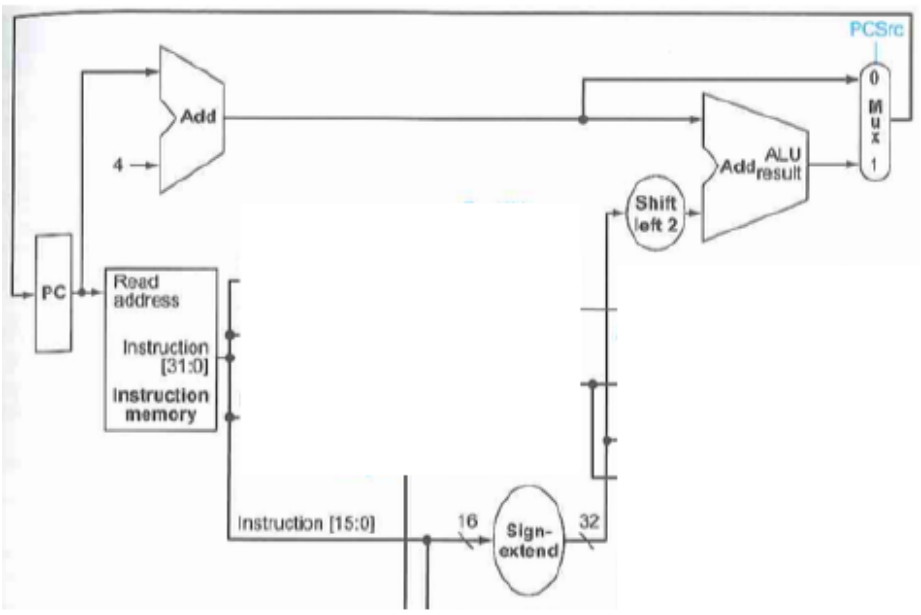

Figure 3: Some circuitry is used in updating the program counter. Control of this circuitry is deduced from the opcode, a field in the present instruction.

is processed.

1.4 Program Counter

This register keeps track of the memory address at which the next instruction can be found. Though we normally advance through code from one instruction to the next, we can also branch (as in conditional statements and iteration statements), and we also can call functions. Thus the program counter is often incremented, and also needs to be loadable. We need a way to know what to load the program counter with, when we are loading it. In particular, returning from a function call implies that the place after the function call must be remembered somewhere.

Whether the program counter is to be incremented or loaded, and if loaded, from where must be deduced from the instruction.

In Figure 3 we can see what's added, so that loading the PC is possible.

- We can see how the value stored in the instruction is obtained.
- We can see how this value is sign extended.
- We can see how this word address is converted into a byte address.
- We can see how this result is selected for going to the PC.

1.4.1 Instructions Have Fields

Some of the bits in the instruction have one purpose, others have other purposes. A group of bits with a common purpose is called a field. For example, one field is named the "opcode". The opcode tells the CPU about the remaining parts of the instruction.

The opcode provides the some information about whether the program counter will be incremented or loaded, and if so from where. Information of this sort is called control. Another example of control is whether the instruction includes immediate data (another field), and how it is to be used.

If the opcode designates that the instruction accesses the register file, then the addresses of the register or registers within the register file come from another part of the instruction.

We can consider instructions in the context of the datapath.

- The datapath is the component of the computer that supports arithmetic operations.
- The control tells the datapath, memory and I/O devices what to do according to the instructions of the program.

We begin with normal, sequential instructions.
We can think about getting sequential instructions from program memory.

- In this picture we see the PC, which is a kind of register, and
- a memory for instructions, and
- an adder.
- The adder has two operands,
- one is the current value of the PC.
- The other is 4. (Why?)

We can think of all the operations on PC.

- We initialize the PC to address (point at) the first instruction of the program.
- We update the PC as instructions are executed.
  - Often the PC is updated to point to the instruction at the next word boundary.
  - For program execution control structures such as if/else and counting loops, small differences from the location of the current instruction are (algebraically) added.

- For procedure calls and returns, the address of the current instruction and next instruction might be far apart.
- When it occurs that if/else or counting loops, require large offsets, clever use of jump instructions can be created.
- A technique is the use of jump tables.

We can think about how the register values specified in instructions, are used.

Bit patterns for register selection are found in instructions.

- It takes 5 bits to specify which of the 32 registers we intend.
- Bits 25 through 21 are one such 5 bit field.
- We see that these are sent to "Read register 1".
- The result of reading that first-specified register is sent to one of the ALU operands.
- There is a 5 bit field register selection field in bits 20 through 16, in some instructions, these can be used.
- There is a 5 bit field register selection field in bits 15 through 11, in some instructions, these can be used.
- There is a way to select between these two previous fields, to be acting as a selection of a register to be written.
- Another source of data to the ALU is data that are in the instruction (found in "immediate" instructions.)
- There is a way to know whether the ALU input comes from a register, or comes from a sign extended version of the immediate data. (What distinguishes these at the assembly language instruction level?)
- When an instruction calls for a register to be written, there are two sources of data: the ALU and data memory.

1.4.2 Instructions are Processed in a Pipeline

For speed (in terms of throughput) and efficiency, instructions are processed several at a time, in a staged fashion.

We can get a feeling for pipelining by considering how several staff share in the processing of lunch orders. In this analogy, instructions are analogous to people ordering lunch. There is a queue of people arriving at the lunch counter, and there are stages of a lunch order being processed. The different parts of the CPU are analogous to the different people working at the lunch counter. In Figure 4 several instructions are queued up. The green instruction arrives first

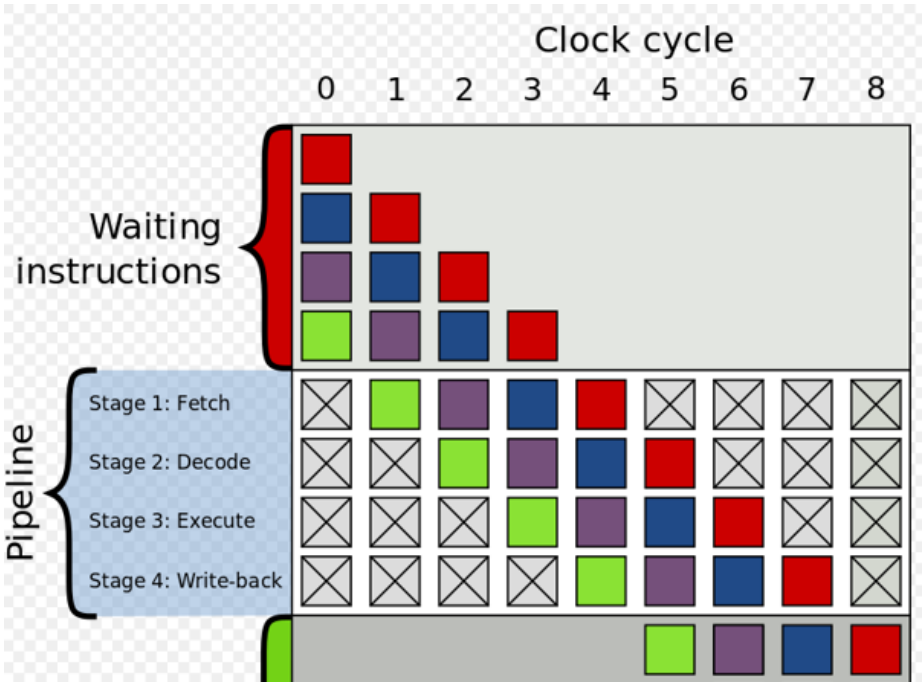

Figure 4: Several instructions are being processed at the same time, but different stages of processing are being carried out on the several instructions.

(having been fetched by a memory read). The opcode of the green instruction is analogous to the order the first customer has. It might reserve the grill, or a different heading element, or none. After the green instruction's order has been placed, it advances to the next stage. If it is an instruction referencing the register file, the particular registers within the register file are identified. In analogy, if a customer's order has options/sides, then the particular sides are identified.

1.5 Control

1.6 Register File

1.7 Arithmetic Logic Unit (ALU)

1.8 Memory Address Register

1.9 Main Memory

1.10 Cache

1.10.1 Levels of Cache

1.11 Disk and other Remote Memory