Introduction to Digital Logic and CPU Design

Nicholas Miehlbradt

August 19, 2022

Quick Recap

- We looked at sequential logic: how do processors store values.
- Worked our way up from SR Latch to D Flip Flop
- Looked at the difference between asynchronous and synchronous circuits.
- We've built two important parts of our CPU:
  - ALU - does the arithmetic and logic operations of the CPU
  - Register file - stores a bank of values that we can use for our operation

If you were not here last week then I've uploaded my work to GitHub.

Finishing the Register File

Specification:

- We want to have 8 registers
- Inputs
  - Clock ✓
  - Read Select 1 (3 bits)
  - Read Select 2 (3 bits)
  - Write Enable (1 bit) ✓
  - Write Select (3 bits) ✓
  - Write Data (16 bits) ✓
- Outputs
  - Read Data 1 (16 bits)
  - Read Data 2 (16 bits)
- Register 0 should always output 0.

Hooking it all up

Lets hook up our register file and execute some instructions.
If you mark the registers as 'measurement values' you can just put a value directly into them.
See if you can do the following:

1. Start with 0x15 in r1 and 0x27 in r2.
2. Put 0x3C into r3.
3. Put 0x14 into r4.
4. Put 0x3F into r5.
5. Put 0xFFFF into r6.

Can you get the value 1 into a register, starting with all zeros?

Control lines

You'll notice that some inputs and outputs of the components feed data through our CPU, other's we've connected to input buttons which we control manually, these are called control lines, they are used to control what the CPU does each clock cycle. Later we'll see how we can set these automatically.

Introducing Memory

So far we have 7 16-bit registers, this isn't a lot of space to store data we want to work on.
Computers have a whole separate memory bank usually called main memory. This memory is much higher capacity compared to registers. The interface is very similar to registers, you put an identifying code (called an address) in and you get the value stored at that address out.

Why do we need two separate memory banks (registers + main memory), why not just have one?

Adding in memory

We can incorporate memory into our CPU. Now instead of using the register values as inputs to the ALU they become inputs into the memory component.

We'll also need a way to choose whether we want the result from memory or the ALU.

Adding instructions

Instructions are stored in main memory. Each cycle an instruction is read from memory and the CPU uses it to decide how to set the control lines.
How do we know what address to use to get the instruction?

The Program Counter

The program counter is a special register that keeps track of where we are up to in the program (ie. the address of the next instruction).
We want the program counter to normally each cycle increment by one. We also want the ability to set it to a specific value (if we have a jump).

Decoding logic

Once we have the instruction from memory we need a circuit that takes the instruction as input and sets the control lines appropriately.

Fortunately this is all just combinational logic. We can go control line by control line and figure out what it should be for each instruction. The truth tables for these can get massive so its sometimes easier to reason through what they should be manually.

Useful Links

- Digital (Logic Simulator)
- Digital Design and Computer Architecture (Textbook)