CSE378

Machine Architecture

Machines

- In this section:
  - Design Perspectives
  - Special purpose machines
  - General purpose machines

Levels in Machine Design

- We can talk about design at a variety of levels (low to high):
  - Circuit design: transistors, resistors, etc. Building gates and flip flops.
  - Logic design: put gates (AND, OR, XOR, etc) and flip-flops together to build blocks such as registers, adders, memory, etc.
  - Register transfer level: describe the execution of instructions by showing how information is transferred/manipulated between adders, registers, etc.
  - Processor description: ISA
  - System description: includes memory hierarchy, I/O devices etc.

Register Transfer Perspective

- We'll use block diagrams or pseudocode to describe our machines in this course.
- We'll start with special purpose machines, then move on to general purpose machines.
- Key component types:
  - Combinational: the output is a function of the inputs (eg. adder)
  - Sequential: the state is remembered

Combination Element: Adder

- An adder computes the sum (output) of two inputs:

Combinational Element: ALU

- ALU computes (combinational) output from two inputs.

Synchronous Design

- Our machines will use a periodic clock that controls when signals can be read and when they can be written. Values in storage elements can only be updated on clock edges (clock down in SMOK).

A Storage Element: Register

- The basic building block is a register.
- Our registers are 32 bits wide.
- A register will only be written on a clock edge AND when the write control line is asserted.
- It can be read and written on the same clock, but the value read will be the old value.

A Counter Machine

- Implement this machine:

```
int i = 0;
while (true) {
   i = i + 1;
}
```

- How do we get the thing to stop?

A Power Machine

- This machine computes X^N

```
int result = 1;
int x = 2;
int n = 10;

while (n != 0) {
   result = result * x;
   n = n - 1;
}
```

- What components do we need to build this machine?

Adding Arrays of Numbers

- To hold a variable amount of data, we need more than a register.
- We use a memory, which can store large amounts of data cheaply, but slowly.

ArraySum Machine

- Here's a description:

```
int i = 0;
int sum = 0;

while (true) {
   sum = sum + memory[i];
   i = i + 1;
}
```

Programs = Data

- We've seen machines that process data from a memory.
- What if the data that a machine processes determines how the machine behaves? We call that kind of data *instructions*.
- A machine that interprets instructions is *general purpose*: it can simulate other kinds of machines!
- What kinds of machines? It depends on the instruction stream...