Introductions

• 3rd year PhD student in Computer Architecture
  – Focus on Data Parallel Architectures, Specializers and Compilers
  – Also spend some time working on RISC-V infrastructure

• colins@eecs
  – Please put [CS152] in the subject
  – Questions about lecture, problem sets, labs, and quizzes should be posted on piazza
Logistics

• Class website
  – http://www-inst.eecs.berkeley.edu/~cs152/sp16/

• Piazza
  – Sign up here:
    http://piazza.com/berkeley/spring2016/cs152

• Sections
  – Thursday 2-4 105 Latimer
  – Thursday 4-6 210 Wheeler

• Office Hours
  – Tuesday 2-4 651 Soda
What are Sections for?

• Answer any questions you have about the material
• Review Labs before they are due
• Review Problem Sets and Quizzes after they are due
• Anything else that seems appropriate
Labs

• Hands on assignments
  – Play with real processors written in Chisel

• Need class account
  – http://inst.eecs.berkeley.edu/webacct

• No specific meeting time
  – Work done on your own time on inst machines
  – http://inst.eecs.berkeley.edu/cgi-bin/clients.cgi?choice=servers
Tools

• RISC-V ISA
  – Entire course uses this (lecture, lab, ps, quizzes)
  – Spec and more available on riscv.org

• Chisel
  – Tutorial and Getting started guide at chisel.eecs.berkeley.edu
  – Processors in labs will be written in this
  – Only need to read
  – If you want open-ended portions allow you to write
Abstraction Layers in Modern Systems

- Application
- Algorithm
- Programming Language
- Operating System/Virtual Machines
- Instruction Set Architecture (ISA)
- Microarchitecture
- Gates/Register-Transfer Level (RTL)
- Circuits
- Devices
- Physics

UCB EECS Courses:
- CS170
- CS164
- CS162
- CS152
- CS150
- EE141
- EE143
Brief introduction of what’s below the Microarchitecture abstraction layer

- Transistors
- INVERTER, NAND, AND, NOR, OR gates
- Tri-state gates
- 2:1, 4:1 Muxes (Multiplexers)
- Latches, Flipflops (Registers)
- Register files
- Adders (Half adders, Full adders)
- RAMs (SRAM, DRAM)
Harvard Mark I

- Built in 1944 in IBM Endicott laboratories
  - Howard Aiken – Professor of Physics at Harvard
  - Essentially mechanical but had some electro-magnetically controlled relays and gears
  - Weighed 5 tons and had 750,000 components
  - A synchronizing clock that beat every 0.015 seconds (66Hz)
  - Inspired by Charles Babbage’s analytic engine

Performance:
- 0.3 seconds for addition
- 6 seconds for multiplication
- 1 minute for a sine calculation

Decimal arithmetic
No Conditional Branch!

Broke down once a week!
Electronic Numerical Integrator and Computer (ENIAC)

- Inspired by Atanasoff and Berry, Eckert and Mauchly designed and built ENIAC (1943-45) at the University of Pennsylvania
- The first, completely electronic, operational, general-purpose analytical calculator!
  - 30 tons, 72 square meters, 200KW
- Performance
  - Read in 120 cards per minute
  - Addition took 200 µs, Division 6 ms
  - 1000 times faster than Mark I
- Not very reliable!

Application: Ballistic calculations

angle = f (location, tail wind, cross wind, air density, temperature, weight of shell, propellant charge, ... )
Manchester SSEM “Baby” (1948)

- Manchester University group build small-scale experimental machine to demonstrate idea of using cathode-ray tubes (CRTs) for computer memory instead of mercury delay lines
- *Williams-Kilburn Tubes were first random access electronic storage devices*
- 32 words of 32-bits, accumulator, and program counter
- Machine ran world’s first stored-program in June 1948
- Led to later Manchester Mark-1 full-scale machine
  - Mark-1 introduced *index* registers
  - Mark-1 commercialized by Ferranti

Williams-Kilburn Tube Store

[Piero71, Creative Commons BY-SA 3.0]
Computers in mid 50’s

- Hardware was expensive
- Store instructions were small (1000 words)
  ⇒ No resident system software!
- Memory access time was 10 to 50 times slower than the processor cycle
  ⇒ Instruction execution time was totally dominated by the memory reference time.
- The ability to design complex control circuits to execute an instruction was the central design concern as opposed to the speed of decoding or an ALU operation
- Programmer’s view of the machine was inseparable from the actual hardware implementation
- MTBF 20 minutes was state of the art
IBM 360: A General-Purpose Register (GPR) Machine

- **Processor State**
  - 16 General-Purpose 32-bit Registers
    - *may be used as index and base register*
    - *Register 0 has some special properties*
  - 4 Floating Point 64-bit Registers
  - A Program Status Word (PSW)
    - *PC, Condition codes, Control flags*

- **A 32-bit machine with 24-bit addresses**
  - But no instruction contains a 24-bit address!

- **Data Formats**
  - 8-bit bytes, 16-bit half-words, 32-bit words, 64-bit double-words

*The IBM 360 is why bytes are 8-bits long today!*
CS152 Executive Summary

The processor you built in CS61C

What you’ll understand and experiment with in CS152

Plus, the technology behind chip-scale multiprocessors (CMPs) and graphics processing units (GPUs)
RISC-V Ecosystem
www.riscv.org

- **Documentation**
  - User-Level ISA Spec v2
  - Privileged ISA draft
  - Compressed ISA draft

- **Software Tools**
  - GCC/glibc/GDB
  - LLVM/Clang
  - Linux
  - Yocto
  - Verification Suite

- **Hardware Tools**
  - Zynq FPGA Infrastructure
  - Chisel

- **Software Implementations**
  - ANGEL, JavaScript ISA Sim.
  - Spike, In-house ISA Sim.
  - QEMU

- **Hardware Implementations**
  - Rocket Chip Generator
    - RV64G single-issue in-order pipe
  - Sodor Processor Collection
  - External implementations
**Documentation**
- User-Level ISA Spec v2
- Privileged ISA draft
- Compressed ISA draft

**Software Tools**
- GCC/glibc/GDB
- LLVM/Clang
- Linux
- Yocto
- Verification Suite

**Hardware Tools**
- Zynq FPGA Infrastructure
- Chisel

**Software Implementations**
- ANGEL, JavaScript ISA Sim.
- Spike, In-house ISA Sim.
- QEMU

**Hardware Implementations**
- Rocket Chip Generator
  - RV64G single-issue in-order pipe
  - Sodor Processor Collection
  - External implementations
Chisel Overview

• Scala-embedded hardware construction language
  – Build HW generators
  – Rapid design-space exploration

• Used in real chips

• Can create arbitrary DSLs on top

```scala
import Chisel._ // importing the Chisel library

/// Creating a Module that computes
///
/// This generator shows how it is possible to program
/// of circuits.

class MaxN(n: Int, w: Int /* parameter */) {
  private def Max2(x: UInt, y: UInt) = {
    val io = new Bundle {
      val in = Vec.fill(n){ UInt(INPUT, w)}
      val out = UInt(OUTPUT, w)
    }
    io.out := io.in.reduceLeft(Max2)
  }

  object MaxNExample {
    // Main Entry Point of the circuit
    def main(args: Array[String]): Unit = {
      // instantiate with 4 ports. Each
      chiselMain(args, () => Module(new MaxN(4, 2)))
    }
  }
```
GCD Example

• Input/Output in a Bundle
• Registers declared with Reg and init values
• Conditional assignment via when and otherwise
• Standard set of operators
• See tutorial for more

class GCD extends Module {
  val io = new Bundle {
    val a = UInt(INPUT, 16)
    val b = UInt(INPUT, 16)
    val z = UInt(OUTPUT, 16)
    val valid = Bool(OUTPUT) }
  val x = Reg(init = io.a)
  val y = Reg(init = io.b)
  when (x > y) {
    x := x - y
  } .otherwise {
    y := y - x
  }
  io.z := x
  io.valid := y === UInt(0)
}
Sodor Snippet
Questions