ECS 154B Discussion 2

January 14th, 2022

Goals

- Assignment 1 FAQ.
- An example of how a program is executed.
- Starting points for assignment 2.

Logistics

- Office Hours
  - Zoom meeting, Thursdays, 1pm - 2pm
- Assignment 2 is released
  - Prompt: https://jlpteaching.github.io/comparch/modules/dino cpu/assignment2/
  - Repo: https://github.com/jlpteaching/dinocpu-wq22
  - Due on January 23rd ;)
  - Need to update your repo (git pull ?) to get the tests for lab 2

Assignment 1: Part V issue

- pc register has a bug that does not allow doing arithmetics directly,

  ```
  pc := pc + some_number
  ```

- Solution: use Adder (in components/helpers.scala)

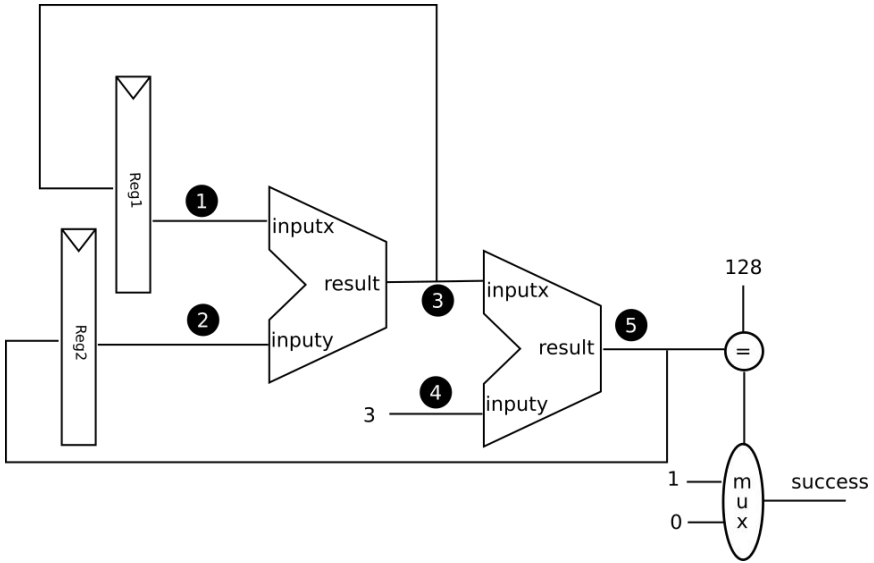
  ```
  val myAdder = Module(new Adder())
  myAdder.inputx := pc
  myAdder.inputy := ???
  pc := myAdder.result
  ```

Assignment 1: Using Singularity

- Singularity works on CSIF machines.
- GitHub codespace: https://campuswire.com/c/G03D9D0A1/feed/81

Assignment 1: Diagram

- Widths
- Additional components (e.g. adders, muxes, comparators)
  - e.g.,



Assignment 1: Reference io is not fully initialized Error

- An output cannot be determined at the end of a CPU cycle.
- Common scenarios,

```
class Example extends Module {
  val io = IO(new Bundle{
    val input1     = Input(UInt(64.W))
    val input2     = Input(UInt(64.W))
    val result1     = Output(UInt(64.W))
    val result2     = Output(Bool())
  })
  io.result1 := io.input1 + io.input2
}
```

```
class Example extends Module {
  val io = IO(new Bundle{
    val input1     = Input(UInt(64.W))
    val input2     = Input(UInt(64.W))
    val result1     = Output(UInt(64.W))
    val result2     = Output(Bool())
  })
  io.result1 := io.input1 + io.input2
  when (io.input1 === io.input2) {
    io.result2 := true.B
  }
}
```

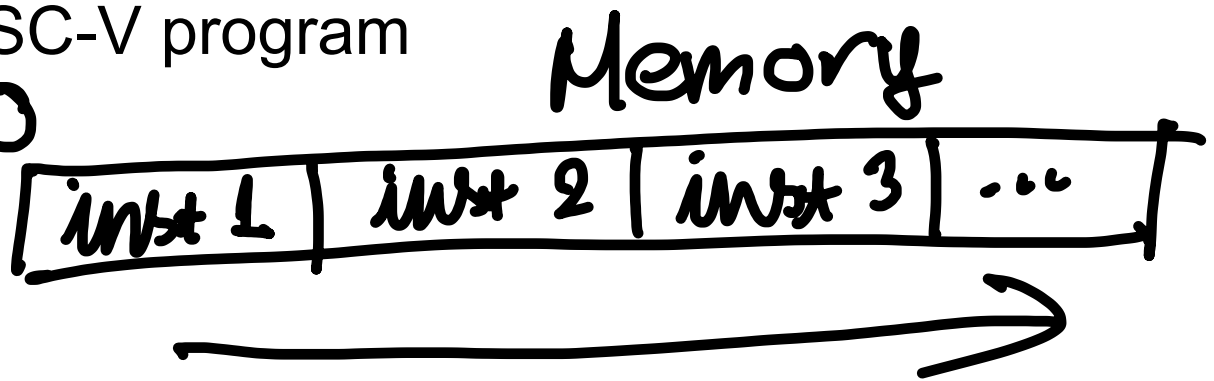
Assignment 1: Run a workload without testers

- Using the single-step debugger,
  - `runMain dinocpu.singlestep <binary> single-cycle`
- Using simulate,
  - `runMain dinocpu.simulate <binary> single-cycle`

Memory in DinoCPU implemented byte addressing

Example RISC-V program

addr = 0 ⇒ first byte
1 ⇒ second byte
2 ⇒ third byte
...



initially: (t0 = b100, t1 = b001) → t2 = 1 if t0 is a power of 2

inst 1: sub t2, t0, t1 // t2 = t0 - t1 = b100 - b1 = b011

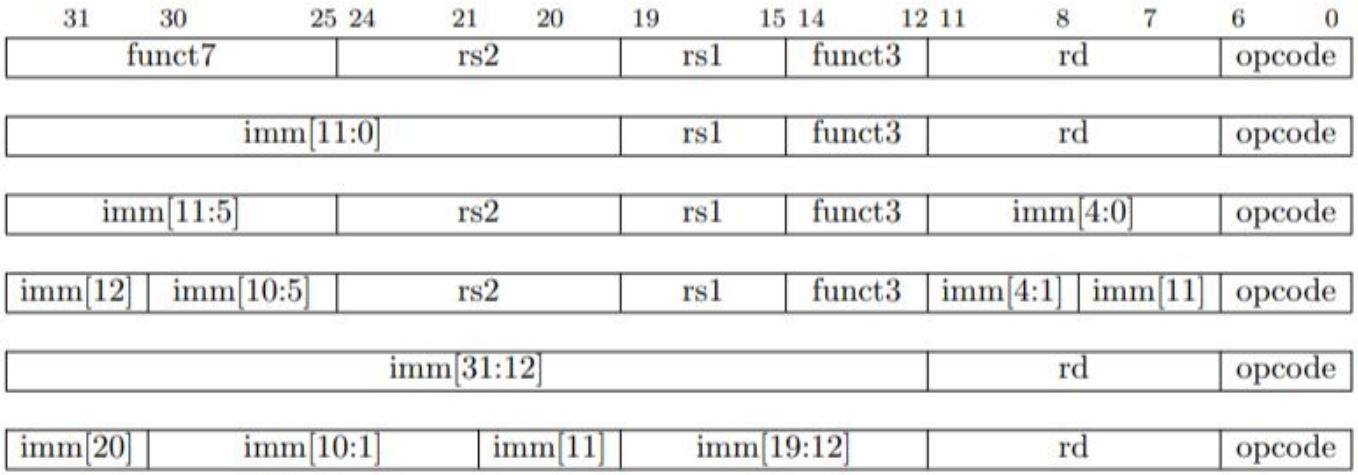
inst 2: and t2, t0, t2 // t2 = t0 & t2 = b011 & b100 = 0

inst 3: sltu t2, t2, t1 // t2 = 1 if (t2 < t1) = 1

Assignment 2: RV64I Instruction Types

Branch: jump if cond == True

imm → immediate



2 ops → R

2 ops → Imm

→ Store

→ Branch

→ Imm

→ Jump

Figure 2.3: RISC-V base instruction formats showing immediate variants.

Jump → jump to the other place

Assignment 2