Lab 3a

CSE141L
Announcements

• Lab 2b “due” today
  – Hardcopy Report
  – Lab Interview

• Lab 3a posted today
Schedule

• 5/21 Lab 3a
  – Datapath Design

• 5/28 Lab 3b
  – Complete the Design
    • Control logic
    • >=One Benchmark runs

• 6/4 Lab 3c
  – Final Debugged Design
Lab 3a

- Design the datapath of the processor
  - Revisit the fetch unit
  - Design the datapath of the backend
  - Implement the datapath of the processor using structural Verilog
Big Picture

- Decoupled Design
  - At least two stage pipeline

- Backend
  - Single-cycle implementation
Revisit your fetch unit

• Instruction Memory Size
  – Size
    • 1K Word -> 8K Word
  – Address Space
    • 10 bits -> 13 bits

• Fetch unit should be largely identical, but you can freely modify the fetch unit.
Major Components of the Backend

• Register File

• Data Memory

• Others
  – ALU
  – Sign extender
  – Mux
  – ......
Register File

- Read: combinational logic
- Write: sequential logic
- Easily implemented by using flip-flops

module regfile#(parameter SEL_WIDTH = 4, D_WIDTH = 34)
{
  input clk,
  input we,
  input [SEL_WIDTH-1 : 0] read_sel,
  input [SEL_WIDTH-1 : 0] write_sel,
  input [D_WIDTH-1 : 0] din,
  output [D_WIDTH-1 : 0] dout
}
Data Memory

• Use ‘Core Generator’
  – optimized implementation
  – easy to generate
  – initialize with your ‘*.coe’ file
  – 34 bit width, 8K word size

• Wrapper
  – provide general interface
  – ‘refused’ signal
The Backend Datapath

![Diagram of the Backend Datapath]

- **we**: write enable signal for the regfile
- **write_sel**: select signal for writing to the regfile
- **din**: data input signal for the regfile
- **read_sel**: select signal for reading from the regfile
- **dout**: data output signal from the regfile
- **read_write_req**: request signal for reading or writing to the memory
- **write_en**: write enable signal for the memory
- **addr**: address signal for the memory
- **din**: data input signal for the memory
- **Wrapper**: generated memory module
- **refused**: signal indicating a refused operation

The diagram visualizes the flow of data and control signals through the regfile and the generated memory module, highlighting the interactions and dependencies within the backend datapath.
I/O Interface

**Backend**

- in_req, out_req
- in_addr [3:0], out_addr [3:0]
- in_data [33:0]
- out_data [33:0]
- in_ack
- out_ack

---

**in**

- in_req <= 1
- in_addr <= channel

- in_ack == 0
- in_ack == 1

- reg[rt] <= din
- ‘in’ completes

---

**out**

- out_req <= 1
- out_addr <= channel
- dout <= data

- out_ack == 0
- out_ack == 1

- ‘out’ completes
Deliverables

• Implemented Leaf Modules
  – register file
  – memory module
  – other leaf modules

• The Datapath Schematic for the Processor
  – with explanations on how each instruction work
  – datapath implementation in structural Verilog
Any Questions?