CSE372
Digital Systems Organization and Design Lab

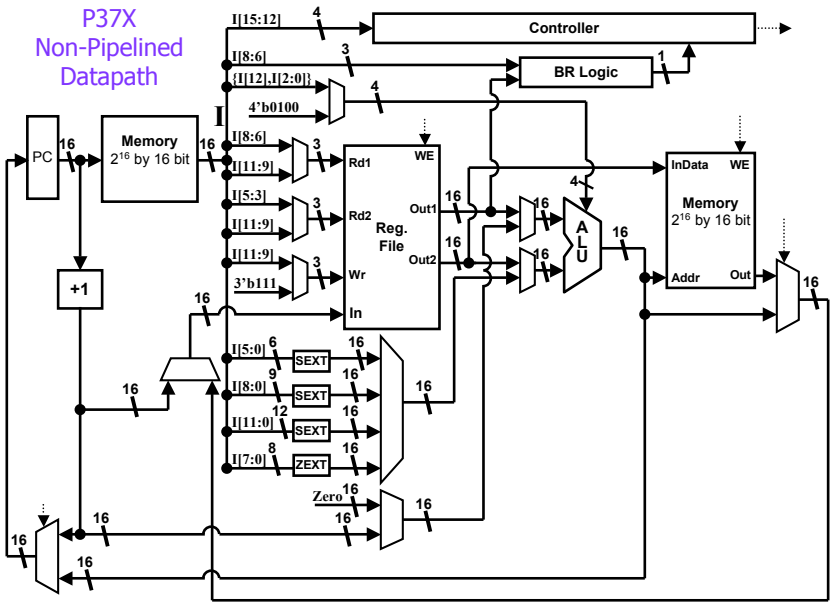
Prof. Milo Martin

Unit 4: The P37X Processor Projects

Final Two Projects

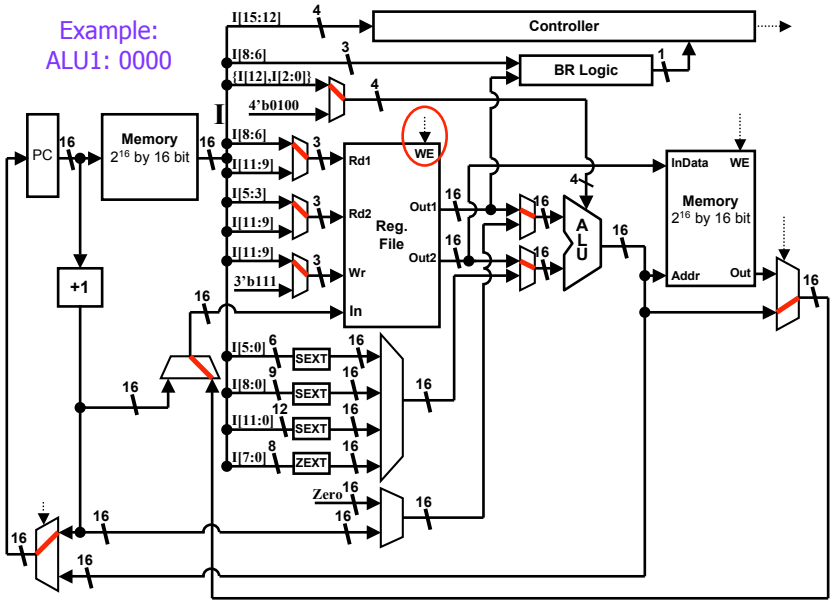
- Lab 3 - Non-pipelined Processor
  - Worksheet - ~1 week
    - Note: use colored pen or pencil
  - Simulation demo - ~1 week
  - Hardware demo - ~1 week
- Lab 4 - Pipelined Processor
  - Much less guidance
  - 3 weeks total
- Both labs done in pairs

P37X Non-Pipelined Datapath



P37X Non-Pipelined Datapath

- Non-pipelined
  - Think of it as a single-cycle design
- Instruction memory and data memory
  - Actually a single two-ported memory
  - One read port, one read/write port
- Register file and ALU as in Lab 1 and Lab 2
- Control
  - 11 muxes, two write-enable signals
  - Fully determined by 4-bit opcode and 1-bit branch outcome



Verilog for the Datapath

- Use fully-structural Verilog
  - **Give all muxes and wires GOOD names**
- Instantiate modules, muxes, wires to connect them
  - Your register file
    - Slightly different "register" module
  - Your ALU
    - You can now use +, * to use Xilinix-optimized blocks
  - Memory
    - Provide by us
  - Controller
  - Branch Logic
  - Other
    - 16-bit PC register
    - Lots of Muxes

Verilog for Controller

- Controller
  - Inputs: 4-bit opcode, 1-bit branch outcome
  - Outputs: all mux and write enable signals
- Part 1: decode instructions
  - `wire is_Arithmetic_Op = (opcode == 4'b0000);`
  - `wire is_ST =  (opcode == 4'b1111);`
- Part 2: set control signals
  - `assign mem_we = (is_STR | is_ST);`
  - Can use negation: `assign sig = ~(… | … | … );`
- Only ~40 lines of code!

Memory Module

- Processor storage
  - 2^{16} location, each 16-bits
  - Used "Block RAM" on the FPGAs
- Memory mapped I/O
  - Memory mapped display (much like LC-3)
    - Only difference: 128x120 (rather than 128x124)
  - Timer registers
  - Keyboard registers
  - **Read switches**
  - **Set LEDs**
  - **Set 7-segment display**
- Like "register", implemented in behavioral Verilog

New "Register" Module

```
module register(out, in, we, gwe, rst, clk);
   parameter n = 1;
   parameter reset_value = 0;

   output [n-1:0] out;
   input [n-1:0]  in;
   input          clk, we, gwe, rst;
   reg [n-1:0] state;
   assign #(1) out = state;
   always @(posedge clk)
     begin
       if (rst)
         state = reset_value;
       else if (gwe & we)
         state = in;
     end
endmodule
```


Single-Cycle or Multi-Cycle?

- The built-in memory only reads on a clock edge
- One "big clock" is four regular clocks
  - Implemented using "global write enable" on registers
- Will work well for the pipelined design…

Next Steps…

- Assignment description available soon
- Find your partner
- Work on the datapath worksheet