CS 61C Spring 2010 — Logic Blocks & Introduction to CPUs

Scott Beamer (cs61c-ta)

Week 10 (3/30)

Warm-Up
- What is the logic equation for a 2-input mux?
- How would you efficiently make a 3-input mux? (draw it)

Adders
- Basic Equations: Sum = A⊕B⊕C C_{out} = MAJ(A, B, C_{in}) = AB + BC_{in} + AC_{in}
- *Tricky Problem:* Remember the Extremely Clever Subtractor from lecture 23? We want to add XOR to its bag of tricks. Add a second signal *XOR*, to the combinational logic block so when XOR is high, it returns A⊕B. If both SUB and XOR are high at the same time, it should return !(A⊕B). How worthwhile is this optimization?

Register Transfer Language (RTL)
- Way to describe what goes where *destination* ← *source*
- Everything on a line is done in parallel, i.e. swap: a←b, b←a
- One 'line' per clock cycle (sometimes it all can't fit on a line, so its a group of lines)
- When describing MIPS, use R[x] to mean register x, and MEM[y] to mean memory at y
- Example: xor $t1, $t2, $t3 is R[9] ← R[10] ^ R[11] PC ← PC + 4

RTL Exercise

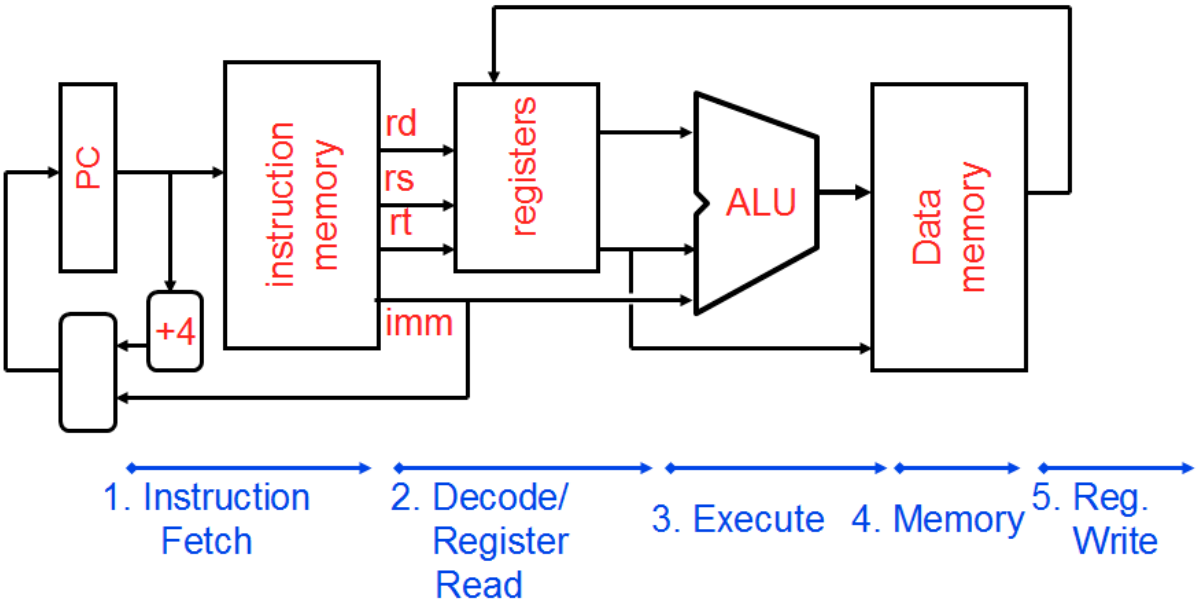
Take the following RTL and design the circuit that will execute it. Assume there is some wire coming in with RST, and when it is high on the rising edge, the RST line is executed. Otherwise the LOOP line is executed.

RST: x←0, y←1
LOOP: x←y, y←x+y

CPU Overview
- Black-box that takes in instructions as input and performs the needed operations
- Memory (both Data and Instruction) are used by the processor but not part of it
- To make design simpler, it is broken into two aspects:
  - *Datapath*: the hardware and logic to perform the operations on the data
  - *Control*: the logic that controls the datapath, such as mux selects, and write enable's
- Design is also simplified by dealing with logic blocks instead of all 2-input gates

CPU Stages

- Break CPU into stages to break it into manageable subproblems
- For now, all 5 stages are done in one cycle, so one instruction is completed per cycle
- *Instruction Fetch (I):* Read current instruction out of instruction memory
- *Decode/Register Read (D):* Read instruction arguments out of register file
- *Execute (E):* Perform needed arithmetic operations
- *Memory (M):* Access data memory
- *Register Writeback (W):* Store results in register file



CPU Exercises

- On the diagram above, try to label the widths of the buses
- For each instruction, fill in what needs to happen at each stage in more detail or if it is different from the summary above

Instruction	I	D	E	M	W
add					
ori					
lw					
sw					
j					
jr					
beq					