

Processor Design in Two Acts

Act I: *A single-cycle CPU*

Foreshadowing

- **Act I: A Single-cycle Processor**
 - Simplest design - Not how many real machines work (maybe some deeply embedded processors)
 - Figure out the basic parts; what it takes to execute instructions
- **Act II: A Pipelined Processor**
 - This is how many real machines work
 - Exploit parallelism by executing multiple instructions at once.

Target ISA

- We will focus on part of MIPS
 - Enough to run into the interesting issues
 - Memory operations
 - A few arithmetic/Logical operations
(Generalizing is straightforward)
 - BEQ and J
- You should be able to extend it to handle other instructions
 - You will do this in 141L.

Basic Steps

- Fetch an instruction from the instruction store
- Decode it
 - What does this instruction do?
- Gather inputs
 - From the register file
 - From memory
- Perform the operation
- Write back the outputs
 - To register file or memory
- Determine the next instruction to execute

The MIPS core subset

- Arithmetic ops

- add rd, rs, rt
- sub, and, or, slt
- "R-Type"

- RTL

- $PC = PC + 4$
- $REG[rd] = REG[rs] \text{ op } REG[rt]$

- Format

bits	31:26	25:21	20:16	15:11	10:6	5:0
name	op	rs	rt	rd	shamt	funct
# bits	6	5	5	5	5	6

The MIPS core subset

- Immediate Arithmetic ops
 - `add rd, rs, imm`
 - `sub, subu, addu, and, or, slt`
 - “I-Type”
- RTL - most ops
 - $PC = PC + 4$
 - $REG[rd] = REG[rs] \text{ op } \text{SignExtImm}$
- RTL - andu (unsigned bitwise and)
 - $PC = PC + 4$
 - $REG[rd] = REG[rs] \& \text{ZeroExtImm}$
- Format

bits	31:26	25:21	20:16	15:0
name	op	rs	rt	imm
# bits	6	5	5	16

The MIPS core subset

- Ld/St

- lw rt, (imm)rs

- sw rt, (imm)rs

- RTL

- PC = PC + 4

- Load: REG[rt] = MEM[signextendImm + REG[rs]]

- PC = PC + 4

- Store: MEM[signextendImm + REG[rs]] = REG[rt]

bits	31:26	25:21	20:16	15:0
name	op	rs	rt	immediate
# bits	6	5	5	16

The MIPS core subset

- Branch
 - Beq rs, rt, imm
 - I-type
- RTL
 - $PC = (REG[rs] == REG[rt]) ? PC + \text{SignExtImmediate} : PC + 4;$
- Format

bits	31:26	25:21	20:16	15:0
name	op	rs	rt	displacement
# bits	6	5	5	16

The Processor Design Algorithm

- Once you have an ISA...
- Design/Draw the datapath
 - Identify and instantiate the hardware for your architectural state
 - Foreach instruction
 - Simulate the instruction
 - Add and connect the datapath elements it requires
 - Is it workable? If not, fix it.
- Design the control
 - Foreach instruction
 - Simulate the instruction
 - What control lines do you need?
 - How will you compute their value?
 - Modify control accordingly
 - Is it workable? If not, fix it.
- We will do this for the core subset now.
- You will do this your ISA in 141L.