CS 61C:
Great Ideas in Computer Architecture
Single Cycle MIPS CPU
Instructor:
David A. Patterson
http://inst.eecs.Berkeley.edu/~cs61c/sp12

3/21/12
Spring 2012 - Lecture #19

• Parallel Requests
  Assigned to computer
e.g., Search “Katz”
• Parallel Threads
  Assigned to core
e.g., Lookup, Ads
• Parallel Instructions
  >1 instruction @ one time
e.g., 5 pipelined instructions
• Parallel Data
  >1 data item @ one time
e.g., Add of 4 pairs of words
• Hardware descriptions
  All gates @ one time
• Programming Languages

Review
• Hardware systems made from Stateless Combinational Logic and Stateful “Memory” Logic (Registers)
  – Setup and Hold times important
• We pipeline long-delay CL for faster clock cycle
  – Split up the critical path
• Finite State Machines extremely useful
• Use muxes to select among input
  – S input bits selects 2^S inputs
  – Each input can be n-bits wide, indep of S
• Can implement muxes hierarchically
• Can implement FSM with register + logic

Agenda
• MIPS-lite Datapath
• Administrivia
• CPU Timing
• MIPS-lite Control
• Datapath Control
• Control Implementation
• (jump implementation bonus slides)

Processor Design Process
• Five steps to design a processor:
  1. Analyze instruction set → datapath requirements
  2. Select set of datapath components & establish clock methodology
  3. Assemble datapath meeting the requirements
  4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.
  5. Assemble the control logic
    • Formulate Logic Equations
    • Design Circuits
The MIPS-lite Subset

- ADDU and SUBU
  - `addu rd,rs,rt`
  - `subu rd,rs,rt`

- OR Immediate:
  - `ori rt,rs,imm16`

- LOAD and STORE Word
  - `lw rt,rs,imm16`
  - `sw rt,rs,imm16`

- BRANCH:
  - `beq rs,rt,imm16`

Register Transfer Language (RTL)

- RTL gives the meaning of the instructions
  - `(op , rs , rt , rd , shamt , funct) ← MEM[ PC ]`
  - `(op , rs , rt , Imm16) ← MEM[ PC ]`

- All start by fetching the instruction

- ADDU Register Transfer
  - `R[rd] ← R[rs] + R[rt]; PC ← PC + 4`

- SUBU Register Transfer
  - `R[rd] ← R[rs] − R[rt]; PC ← PC + 4`

- ORI Register Transfer
  - `R[rt] ← R[rs] | zero_ext(Imm16); PC ← PC + 4`

- LOAD Register Transfer
  - `R[rt] ← MEM[ R[rs] + sign_ext(Imm16)]; PC ← PC + 4`

- STORE Register Transfer
  - `MEM[ R[rs] + sign_ext(Imm16) ] ← R[rt]; PC ← PC + 4`

- BEQ
  - if ( R[rs] == R[rt] ) then PC ← PC + 4 + (sign_ext(Imm16) || 00) else PC ← PC + 4

Step 1: Requirements of the Instruction Set

- Memory (MEM)
  - Instructions & data (will use one for each: really caches)

- Registers (R: 32 x 32)
  - Read rs
  - Read rt
  - Write rt or rd

- PC

- Extender (sign/zero extend)

- Add/Sub/Or unit for operation on register(s) or extended immediate

- Add 4 (+ maybe extended immediate) to PC

- Compare if registers equal?

Step 2: Components of the Datapath

- Combinational Elements

- State Elements + Clocking Methodology

- Building Blocks

ALU Needs for MIPS-lite + Rest of MIPS

- Addition, subtraction, logical OR, ==:
  - ADDU `R[rd] = R[rs] + R[rt]; ...`
  - SUBU `R[rd] = R[rs] − R[rt]; ...`
  - ORI `R[rt] = R[rs] | zero_ext(Imm16); ...`
  - SEQ if ( R[rs] == R[rt] )...

- Test to see if output == 0 for any ALU operation gives == test. How?

- P&H also adds AND, Set Less Than (1 if A < B, 0 otherwise)

- ALU from Appendix C, section C.5
Storage Element: Idealized Memory

- Memory (idealized)
  - One input bus: Data In
  - One output bus: Data Out

- Memory word is found by:
  - Address selects the word to put on Data Out
  - Write Enable = 1: address selects the memory word to be written via the Data In bus

- Clock input (CLK)
  - CLK input is a factor ONLY during write operation
  - During read operation, behaves as a combinational logic block: Address valid ⇒ Data Out valid after “access time”

Step 3: Add & Subtract

- \( R(rd) = R(rs) \) \( \text{op} \) \( R(rt) \) (addu rd, rs, rt)\
  - Ra, RB, and Rn come from instructions Rs, Rt, and Rd fields

- ALUctrl and RegWrite: control logic after decoding the instruction

- ... Already defined the register file & ALU

Storage Element: Register (Building Block)

- Similar to D Flip Flop except
  - N-bit input and output
  - Write Enable input

- Write Enable:
  - Negated (or deasserted) (0): Data Out will not change
  - Asserted (1): Data Out will become Data In on rising edge of clock

Storage Element: Register File

- Register File consists of 32 registers:
  - Two 32-bit output busses: busA and busB
  - One 32-bit input bus: busW

- Register is selected by:
  - RA (number) selects the register to put on busA (data)
  - RB (number) selects the register to put on busB (data)
  - RW (number) selects the register to be written via busW (data) when Write Enable is 1

- Clock input (clk)
  - CLK input is a factor ONLY during write operation
  - During read operation, behaves as a combinational logic block:
    - RA or RB valid ⇒ busA or busB valid after “access time.”

Step 3: Assemble DataPath Meeting Requirements

- Register Transfer Requirements
  -⇒ Datapath Assembly

- Instruction Fetch

- Read Operands and Execute Operation

- Common RTL operations
  - Fetch the Instruction: mem(PC)
  - Update the program counter:
    - Sequential Code: PC ← PC + 4
    - Branch and Jump: PC ← “something else”

Administrivia

- Project 3, Part 2 due Sunday 4/1
  - Thread Level Parallelism and OpenMP

- Last homework due Sunday 4/8

- Project 4, Part 1 due Sunday 4/8; Part 2 4/15
  - Design a 16-bit pipelined computer in Logism
  - Labs 10 and 11 prepare for Project 4

- Lab 12 — More Logism; Lab 13 Malloc/Free in C

- Extra Credit due 4/22 — Fastest Matrix Multiply

- Final Exam Wednesday 5/9 11:30-2:30PM
Project 3, Part 1 Results

Administivia

- What classes should I take (now)?
- Take classes from great teachers! (teacher > class)
  - Distinguished Teaching Award (very hard to get: ~3/year)
  - Dan Klein CS188 Fall 2012, Brian Harvey CS 195 Fall 2012
  - HKN Course evaluations (26.0 is very good)
  - https://hkn.eecs.berkeley.edu/courseevals
  - Vakil/Shprintz, Sinclair CS374
  - EECS web site has plan for year (up in late spring: now)
- If have choice of multiple great teachers
  - CS152 Computer Architecture and Engineering (Spring 13)
  - CS162 Operating Systems and Systems Programming
  - CS169 Software Engineering (for Saas, Fox/Patterson Fall 12)
  - CS194 Engineering Parallel Software (Spring 13)

Agenda

- MIPS-lite Datapath
- Administivia
- CPU Timing
- MIPS-lite Control
- Datapath Control
- Control Implementation

Clocking Methodology

- Storage elements clocked by same edge
- "Critical path" (longest path through logic) determines length of clock period
- Have to allow for Clock-to-Q and Setup Times too
- This lecture (and FIH sections 4.3-4.4) do whole instruction in 1 clock cycle for pedagogic reasons
  - Project 4 will do it in 2 clock cycles via simple pipelining
  - Soon explain pipelining and use 5 clock cycles per instruction

Register-Register Timing: One Complete Cycle
Logical Operations with Immediate

- \( R[rt] = R[rs] \text{ op ZeroExt}[imm16] \)

Example: \( lw \ rt, rs, imm16 \)

Load Operations

- \( R[rt] = \text{Mem}[R[rs] + \text{SignExt}[imm16]] \)

Example: \( lw \ rt, rs, imm16 \)

RTL: The Add Instruction

- \( \text{add \ rd, rs, rt} \)
  - \( \text{MEM}[PC] \) Fetch the instruction from memory
  - \( R[rd] = R[rs] + R[rt] \) The actual operation
  - \( PC = PC + 4 \) Calculate the next instruction’s address
**Instruction Fetch Unit at the Beginning of Add**

- Fetch the instruction from Instruction memory:
  - Instruction = MEM[PC]
  - Same for all instructions

**Single Cycle Datapath during Add**

- \( R[rd] = R[rs] + R[rt] \)

**Instruction Fetch Unit at End of Add**

- PC = PC + 4
  - Same for all instructions except: Branch and Jump

**Single Cycle Datapath during Or Immediate**

- \( R[rt] = R[rs] \text{ OR } \text{ZeroExt[Imm16]} \)

**Single Cycle Datapath during Load**

- \( R[rt] = \text{Data Memory}[R[rs] + \text{SignExt}[imm16]] \)
Single Cycle Datapath during Load

- \( R[rt] = Data\ Memory[R[rs] + \text{SignExt}[\text{imm16}]) \)

Single Cycle Datapath during Store

- Data Memory \( R[rs] + \text{SignExt}[\text{imm16}] \)

Single Cycle Datapath during Branch

- If \( R[rs] - R[rt] == 0 \) then \( \text{Zero} = 1 \); else \( \text{Zero} = 0 \)

Instruction Fetch Unit at the End of Branch

- What is encoding of \( nPC\_sel \)?
  - Direct MUX select?
  - Branch inst. / not branch
  - Let's pick 2nd option

Q: What logic gate?
Summary: Datapath’s Control Signals

- **ExtOp:** "zero", "sign"
- **ALUsrc:** 0 ⇒ regB; 1 ⇒ imm16
- **ALUctr:** "ADD", "SUB", "OR"
- **MemWr:** 1 ⇒ write memory
- **RegttoReg:** 0 ⇒ "rt"; 1 ⇒ "rd"
- **RegWr:** 1 ⇒ write register

Summary: Single-Cycle Processor

- Five steps to design a processor:
  1. Analyze instruction set → datapath requirements
  2. Select set of datapath components & establish clock methodology
  3. Assemble datapath meeting the requirements
  4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.
  5. Assemble the control logic
     - Formulate Logic Equations
     - Design Circuits