inst.eecs.berkeley.edu/~cs61c

CS61C : Machine Structures

Lecture 34 – Single Cycle CPU Control I

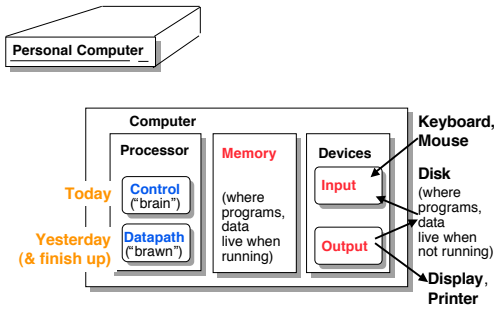
2004-04-16

Lecturer PSOE Dan Garcia

www.cs.berkeley.edu/~ddgarcia

10.5 Quake?! NBC ⇒ movie on May 3rd.

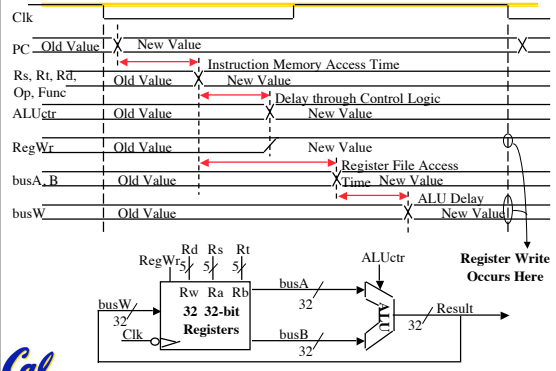
Anatomy Review: 5 components of any Computer



Review: How to Design a Processor: step-by-step

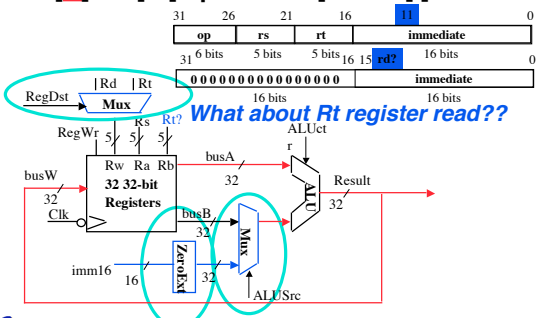
- 1. Analyze instruction set architecture (ISA) => datapath requirements
  - meaning of each instruction is given by the *register transfers*
  - datapath must include storage element for ISA registers
  - datapath must support each register transfer
- 2. Select set of datapath components and establish clocking methodology
- 3. Assemble datapath meeting requirements
- **4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.**
- **5. Assemble the control logic**

Register-Register Timing: One complete cycle



3c: Logical Operations with Immediate

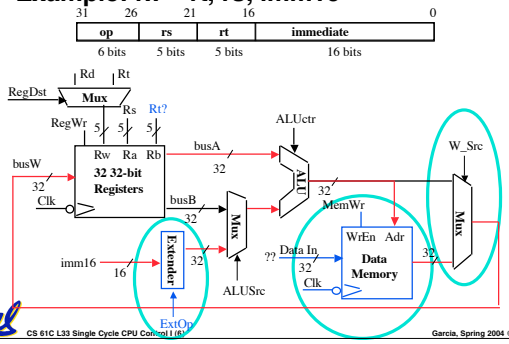
- **R[rt] = R[rs] op ZeroExt[imm16]]**



Already defined 32-bit MUX; Zero Ext?

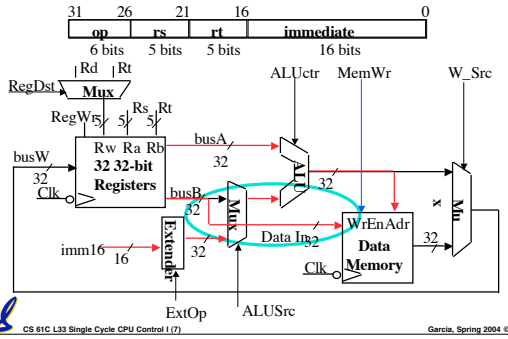
3d: Load Operations

- **R[rt] = Mem[R[rs] + SignExt[imm16]]**
  Example: lw rt, rs, imm16

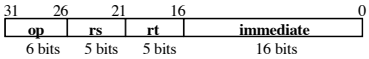


3e: Store Operations

- Mem[R[rs] + SignExt[imm16]] = R[rt]
  Ex.: sw rt, rs, imm16



3f: The Branch Instruction



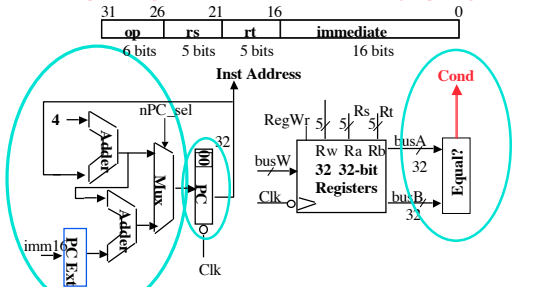
- beq rs, rt, imm16
  - mem[PC] Fetch the instruction from memory
  - Equal = R[rs] == R[rt] Calculate the branch condition
  - if (Equal) Calculate the next instruction's address
    - PC = PC + 4 + (SignExt(imm16) x 4)

  else
    - PC = PC + 4

Datapath for Branch Operations

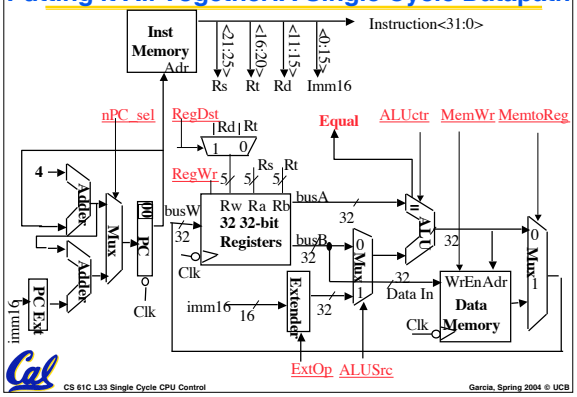
- beq rs, rt, imm16

Datapath generates condition (equal)

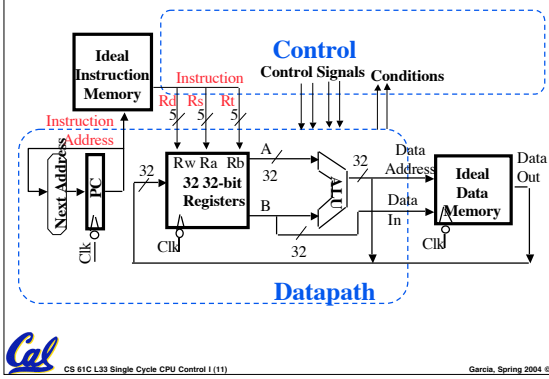


- Already MUX, adder, sign extend, zero

Putting it All Together:A Single Cycle Datapath



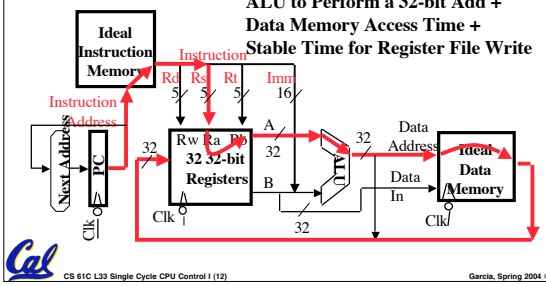
An Abstract View of the Implementation



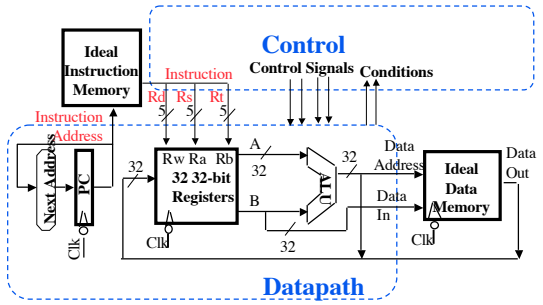
An Abstract View of the Critical Path

- This affects how much you can overclock your PC!

Critical Path (Load Operation) =
Delay clock through PC (FFs) +
Instruction Memory's Access Time +
Register File's Access Time +
ALU to Perform a 32-bit Add +
Data Memory Access Time +
Stable Time for Register File Write

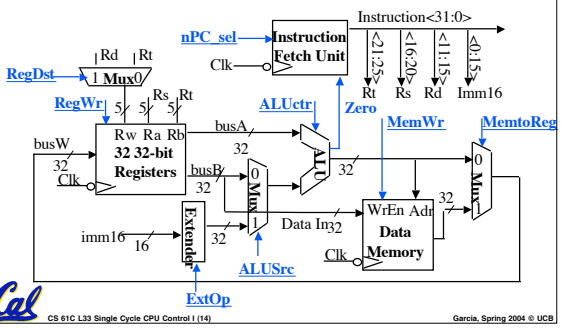


An Abstract View of the Implementation



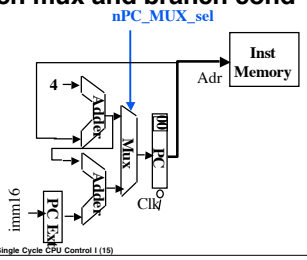
Recap: A Single Cycle Datapath

- Rs, Rt, Rd, Imed16 connected to datapath
- We have everything except control signals



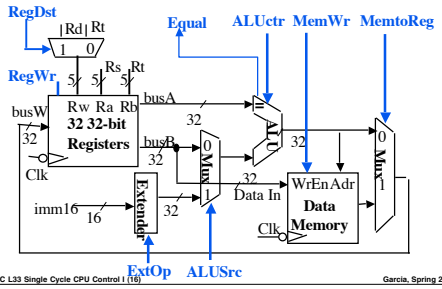
Recap: Meaning of the Control Signals

- **nPC_MUX_sel:** 0 ⇒ PC <– PC + 4
  1 ⇒ PC <– PC + 4 + {SignExt(Im16) , 00 }
  "n"=next
- Later in lecture: higher-level connection between mux and branch cond



Recap: Meaning of the Control Signals

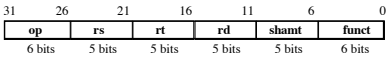
- ExtOp: "zero", "sign"
- ALUsrc: 0 ⇒ regB; 1 ⇒ immed
- ALUctr: "add", "sub", "or"
- MemWr: 1 ⇒ write memory
- MemtoReg: 0 ⇒ ALU; 1 ⇒ Mem
- RegDst: 0 ⇒ "rt"; 1 ⇒ "rd"
- RegWr: 1 ⇒ write register



Administrivia

- Final Exam will be here! (2050 VLSB)
  - Sat, 2004-05-22, 12:30–3:30pm
- Take EECS Survey this week! (results will be presented to Faculty; you CAN make a difference in your education & undergraduate experience)
  - `http://eecssurvey.berkeley.edu`

RTL: The `Add` Instruction

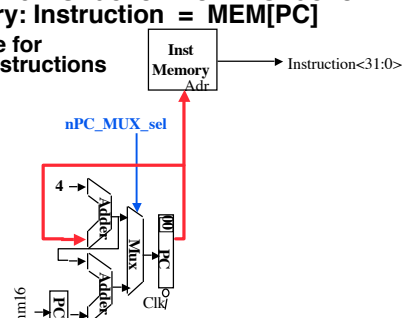


`add rd, rs, rt`

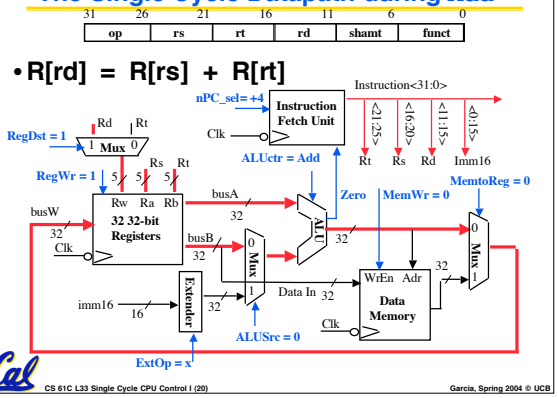
- MEM[PC] Fetch the instruction from memory
- R[rd] = R[rs] + R[rt] The actual operation
- PC = PC + 4 Calculate the next instruction's address

Instruction Fetch Unit at the Beginning of Add

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

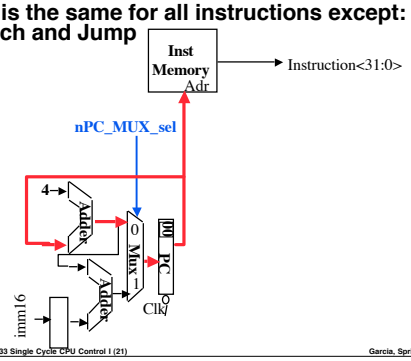


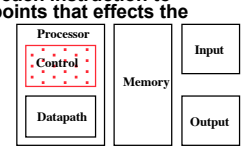
The Single Cycle Datapath during Add



Instruction Fetch Unit at the End of Add

- PC = PC + 4
  - This is the same for all instructions except: Branch and Jump



Summary: Single cycle datapath

- 5 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
- Control is the hard part
- MIPS makes that easier
  - Instructions same size
  - Source registers always in same place
  - Immediates same size, location
  - Operations always on registers/immediates