

Lecture #17: CPU Design II – Control

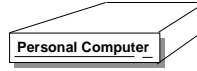


2006-07-26

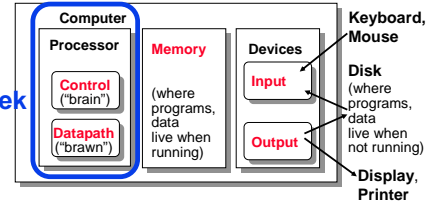
Andy Carle



Anatomy: 5 components of any Computer

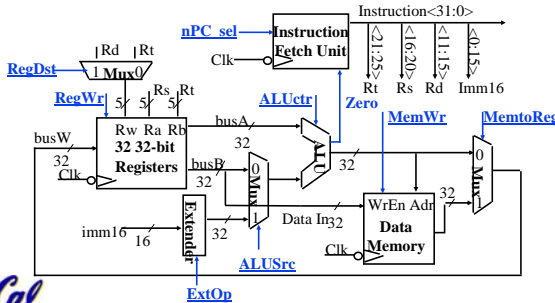


This week



Review: A Single Cycle Datapath

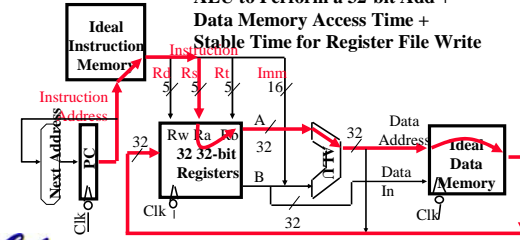
- Rs, Rt, Rd, Imm16 connected to datapath
- We have everything except **control signals**



An Abstract View of the Critical Path

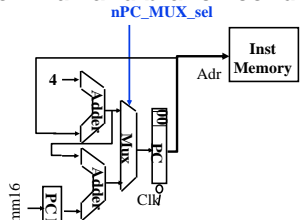
- This affects how fast you can clock your PC

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



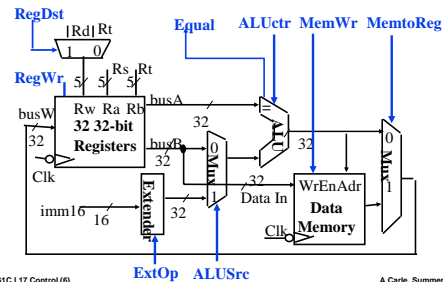
Recap: Meaning of the Control Signals

- nPC_MUX_sel: 0 ⇒ PC ← PC + 4
 "n"=next
 1 ⇒ PC ← PC + 4 + {SignExt(Imm16), 00}
- Later in lecture: higher-level connection between mux and branch cond

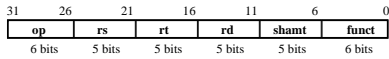


Recap: Meaning of the Control Signals

- ExtOp: "zero", "sign"
- ALUsrc: 0 ⇒ regB; 1 ⇒ imm16
- ALUctr: "add", "sub", "or"
- MemWr: 1 ⇒ write memory
- MemtoReg: 0 ⇒ ALU; 1 ⇒ Mem
- RegDst: 0 ⇒ "rt"; 1 ⇒ "rd"
- RegWr: 1 ⇒ write register



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



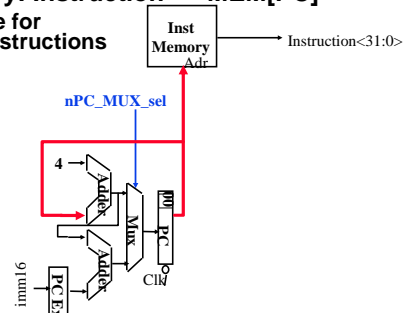
CS 61C L17 Control (7)

A. Carle, Summer 2006 © UCB

Instruction Fetch Unit at the Beginning of Add

- Fetch the instruction from Instruction memory: Instruction = MEM[PC]

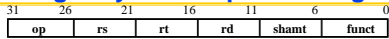
- same for all instructions



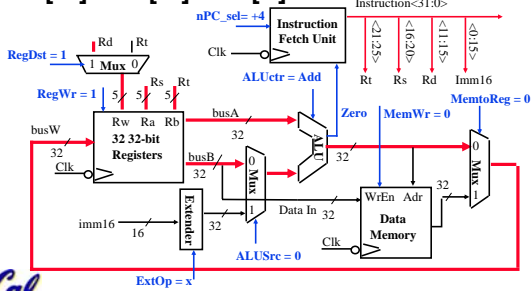
CS 61C L17 Control (8)

A. Carle, Summer 2006 © UCB

The Single Cycle Datapath during Add



- R[rd] = R[rs] + R[rt]



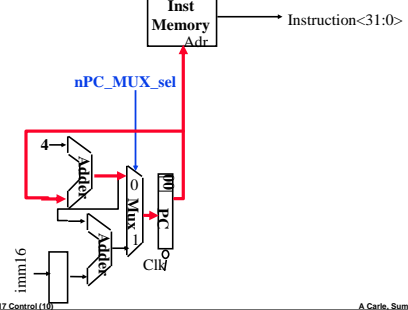
CS 61C L17 Control (9)

A. Carle, Summer 2006 © UCB

Instruction Fetch Unit at the End of Add

- PC = PC + 4

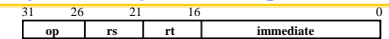
- This is the same for all instructions except: Branch and Jump



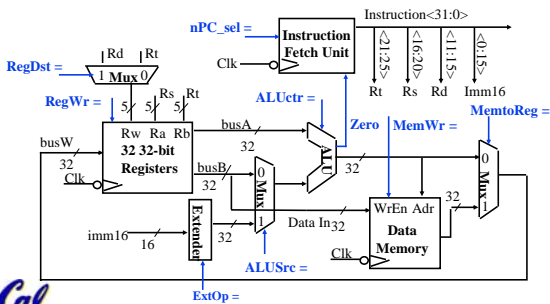
CS 61C L17 Control (10)

A. Carle, Summer 2006 © UCB

Single Cycle Datapath during Or Immediate?



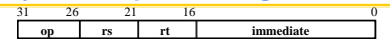
- R[rt] = R[rs] OR ZeroExt[Imm16]



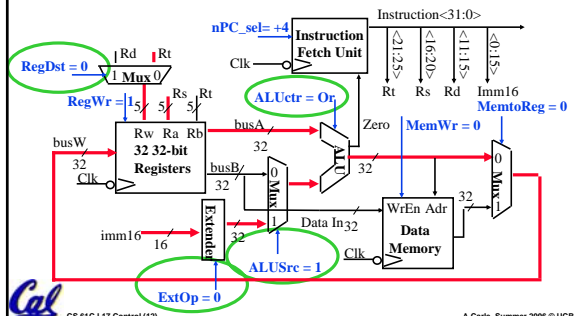
CS 61C L17 Control (11)

A. Carle, Summer 2006 © UCB

Single Cycle Datapath during Or Immediate?

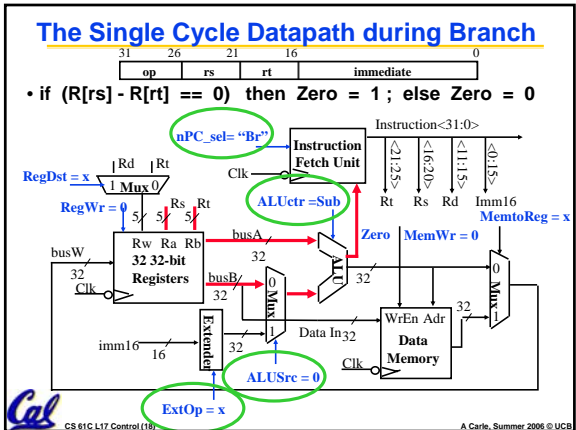
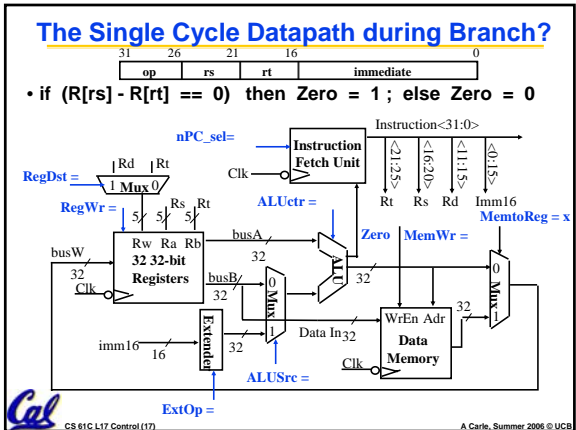
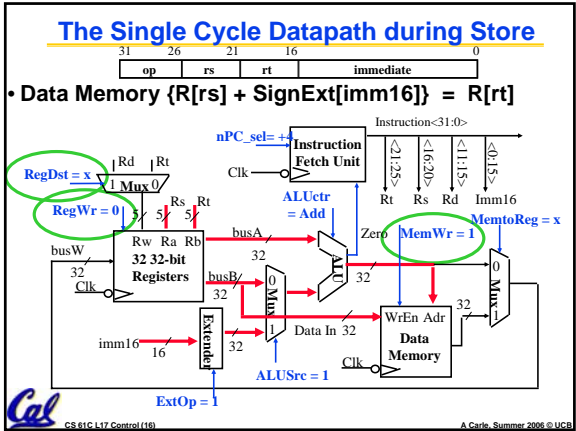
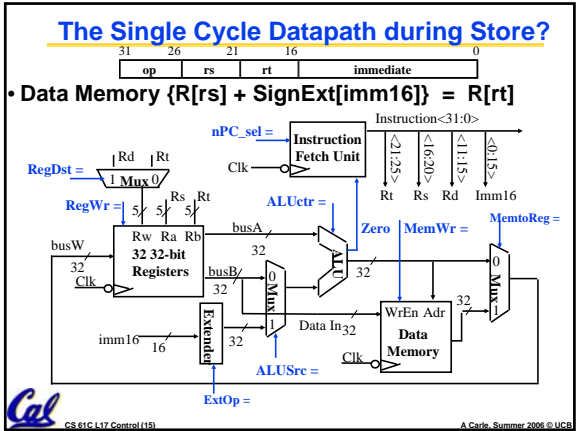
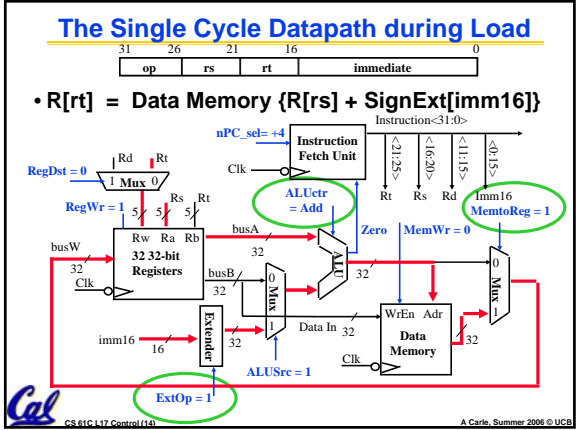
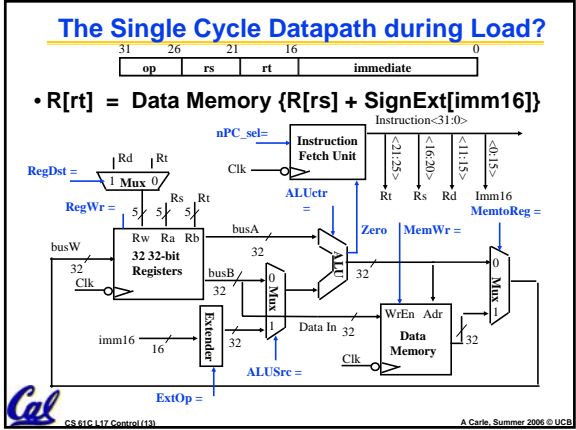


- R[rt] = R[rs] OR ZeroExt[Imm16]



CS 61C L17 Control (12)

A. Carle, Summer 2006 © UCB



Instruction Fetch Unit at the End of Branch

• if (Zero == 1) then PC = PC + 4 + SignExt[imm16]*4 ;
else PC = PC + 4

• What is encoding of nPC_sel?

- Direct MUX select?
- Branch / not branch
- Let's pick 2nd option

Q: What logic gate?

nPC_sel	zero?	MUX
0	x	0
1	0	0
1	1	1

CS 61C L17 Control (19) A Carls, Summer 2006 © UCB

Step 4: Given Datapath: RTL -> Control

CS 61C L17 Control (20) A Carls, Summer 2006 © UCB

A Summary of the Control Signals (1/2)

inst Register Transfer

ADD R[rd] <- R[rs] + R[rt]; PC <- PC + 4
ALUSrc = RegB, ALUctr = "add", RegDst = rd, RegWr, nPC_sel = "+4"

SUB R[rd] <- R[rs] - R[rt]; PC <- PC + 4
ALUSrc = RegB, ALUctr = "sub", RegDst = rd, RegWr, nPC_sel = "+4"

ORI R[rt] <- R[rs] + zero_ext(Imm16); PC <- PC + 4
ALUSrc = Im, Extop = "Z", ALUctr = "or", RegDst = rt, RegWr, nPC_sel = "+4"

LOAD R[rt] <- MEM[R[rs] + sign_ext(Imm16)]; PC <- PC + 4
ALUSrc = Im, Extop = "Sn", ALUctr = "add", MemtoReg, RegDst = rt, RegWr, nPC_sel = "+4"

STORE MEM[R[rs] + sign_ext(Imm16)] <- R[rs]; PC <- PC + 4
ALUSrc = Im, Extop = "Sn", ALUctr = "add", MemWr, nPC_sel = "+4"

BEQ if (R[rs] == R[rt]) then PC <- PC + sign_ext(Imm16) || 00 else PC <- PC + 4
nPC_sel = "Br", ALUctr = "sub"

CS 61C L17 Control (21) A Carls, Summer 2006 © UCB

A Summary of the Control Signals (2/2)

See Appendix A

func	10 0000	10 0010	We Don't Care :-)				
op	00 0000	00 0000	00 1101	10 0011	10 1011	00 0100	00 0010
	add	sub	ori	lw	sw	beq	jump
RegDst	1	1	0	0	x	x	x
ALUSrc	0	0	1	1	1	0	x
MemtoReg	0	0	0	1	x	x	x
RegWrite	1	1	1	1	0	0	0
MemWrite	0	0	0	0	1	0	0
nPCsel	0	0	0	0	0	1	0
Jump	0	0	0	0	0	0	1
ExtOp	x	x	0	1	1	x	x
ALUctr<2:0>	Add	Subtract	Or	Add	Subtract	xxx	xxx

R-type: op rs rt rd shamt funct add, sub

I-type: op rs rt immediate ori, lw, sw, beq

J-type: op target address jump

CS 61C L17 Control (22) A Carls, Summer 2006 © UCB

Administrivia

- Project 2 Due Friday
- HW 5 Now Available

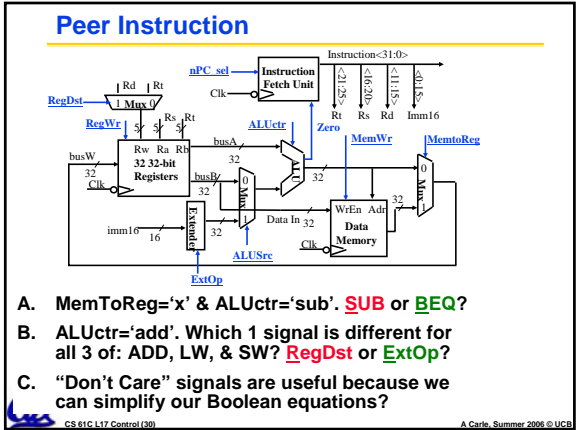
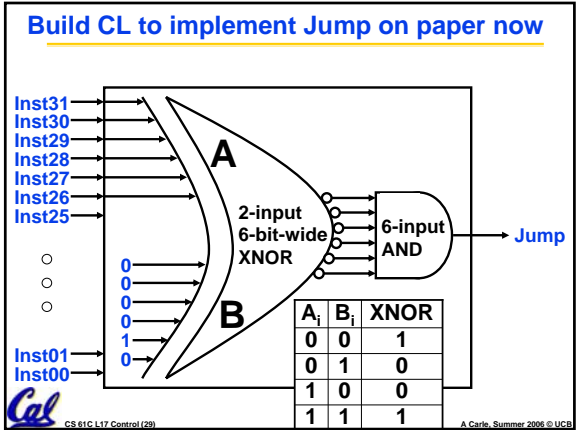
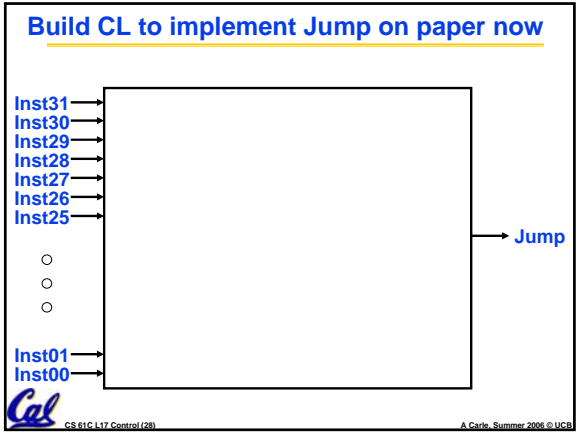
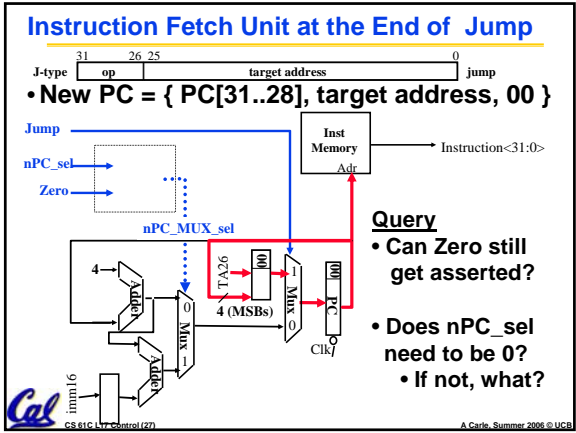
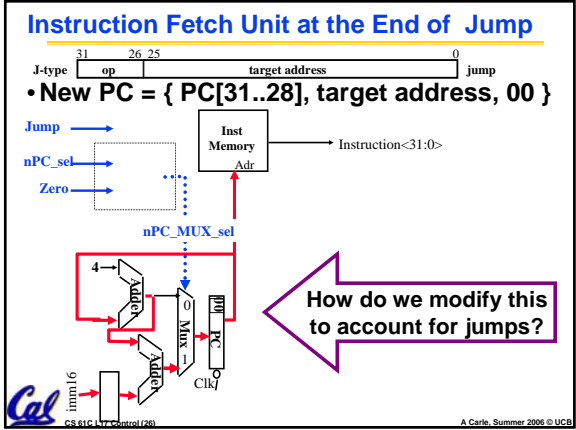
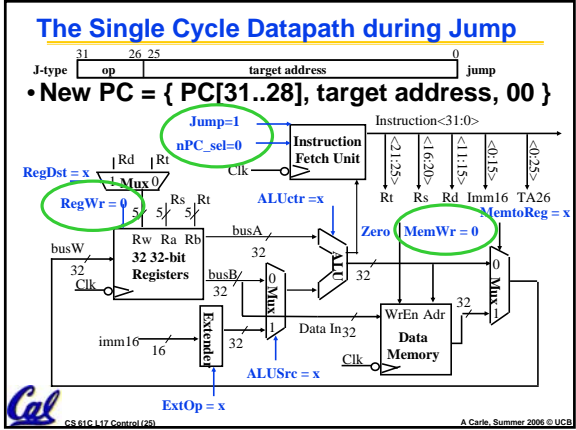
CS 61C L17 Control (23) A Carls, Summer 2006 © UCB

The Single Cycle Datapath during Jump

J-type: op target address jump

• New PC = { PC[31..28], target address, 00 }

CS 61C L17 Control (24) A Carls, Summer 2006 © UCB



And in Conclusion... Single cycle control

◦ 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



CS 61C 147 Control (31)

A. Chien, Summer 2006 © UCB

