inst.eecs.berkeley.edu/~cs61c

UC Berkeley CS61C : Machine Structures

Lecture 27
Single-cycle CPU Control

2010-04-02

Lecturer SOE Dan Garcia

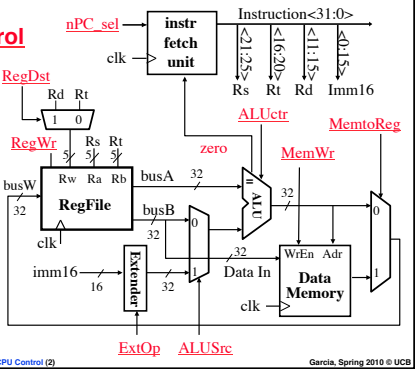
www.cs.berkeley.edu/~ddgarcia

Success! ⇒ The large Hadron collider @ CERN is finally able to get subatomic particles colliding. No Higgs boson particle yet, but stay tuned. 16 years and 10 G$!

www.nytimes.com/2010/03/31/science/31collider.html

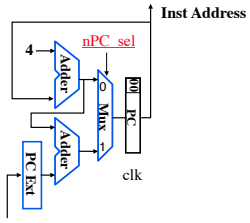
Review: A Single Cycle Datapath

- We have everything except control signals



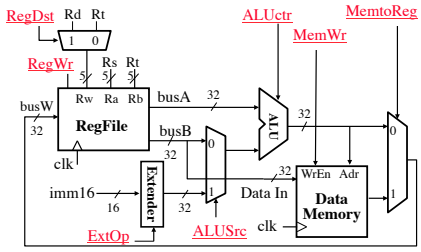
Recap: Meaning of the Control Signals

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

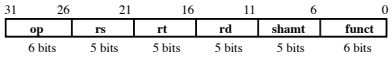


Recap: Meaning of the Control Signals

- 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



RTL: The Add Instruction

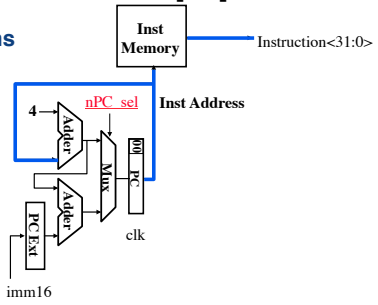


`add rd, rs, rt`

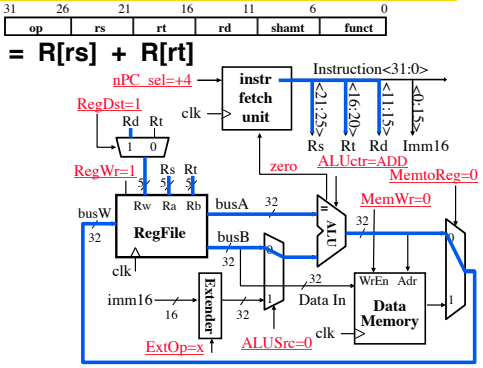
- MEM[PC] Fetch the instruction from memory
- R[rd] = R[rs] + R[rt] The actual operation
- PC = PC + 4 Calculate the next instruction's address

Instruction Fetch Unit at the Beginning of Add

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

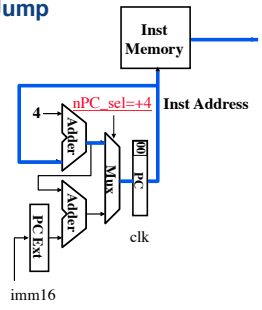


The Single Cycle Datapath during Add

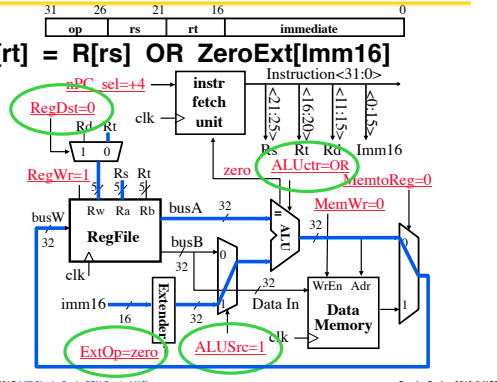


Instruction Fetch Unit at the End of Add

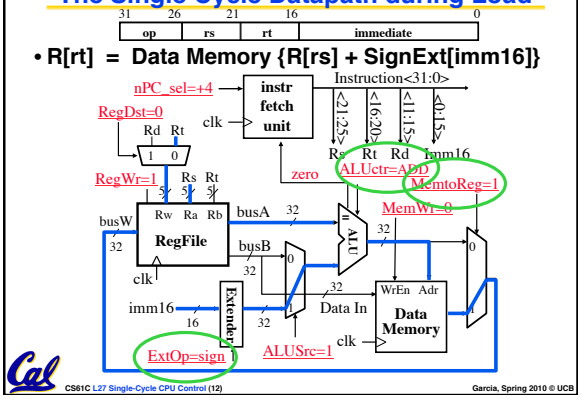
- PC = PC + 4
  - This is the same for all instructions except: Branch and Jump



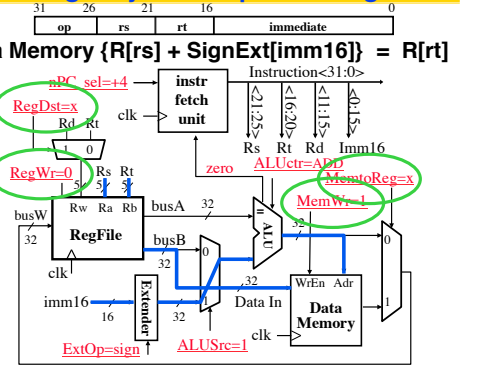
Single Cycle Datapath during Or Immediate?



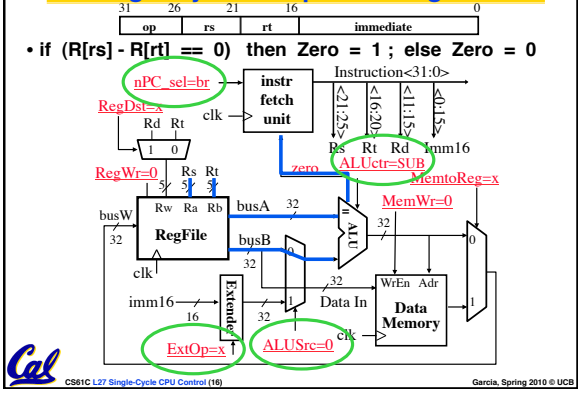
The Single Cycle Datapath during Load



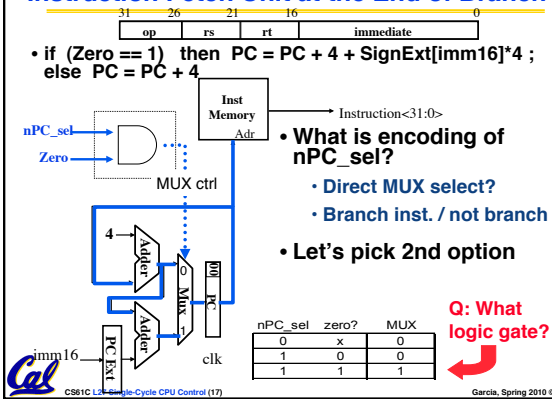
The Single Cycle Datapath during Store



The Single Cycle Datapath during Branch



Instruction Fetch Unit at the End of Branch



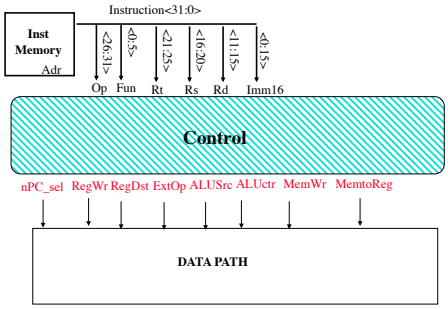
- if (Zero == 1) then PC = PC + 4 + SignExt[imm16]*4 ; else PC = PC + 4

- What is encoding of nPC_sel?
  - Direct MUX select?
  - Branch inst. / not branch
- Let's pick 2nd option

Q: What logic gate?

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

Step 4: Given Datapath: RTL → Control



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"

lw R[rt] ← MEM[R[rs] + sign_ext(Imm16)]; PC ← PC + 4

ALUsrc = Im, Extop = "sn", ALUctr = "ADD", MemtoReg, RegDst = rt, RegWr, nPC_sel = "+4"

sw MEM[R[rs] + sign_ext(Imm16)] ← R[rs]; PC ← PC + 4

ALUsrc = Im, Extop = "sn", ALUctr = "ADD", MemWr, nPC_sel = "+4"

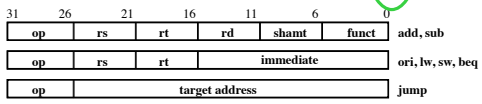
beq if (R[rs] == R[rt]) then PC ← PC + sign_ext(Imm16)] || 00 else PC ← PC + 4

nPC_sel = "br", ALUctr = "SUB"

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	?
Jump		0	0	0	0	0	0	1
ExtOp		x	x	0	1	1	x	x
ALUctr<2:0>		Add	Subtract	Or	Add	Add	Subtract	x



Boolean Expressions for Controller

RegDst = add + sub
ALUSrc = ori + lw + sw
MemtoReg = lw
RegWrite = add + sub + ori + lw
MemWrite = sw
nPCsel = beq
Jump = jump
ExtOp = lw + sw
ALUctr[0] = sub + beq (assume ALUctr is 00 ADD, 01: SUB, 10: OR)
ALUctr[1] = or

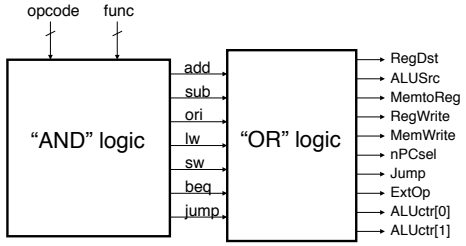
where,

rtype = $\sim op_5 \bullet \sim op_4 \bullet \sim op_3 \bullet \sim op_2 \bullet \sim op_1 \bullet \sim op_0$,
ori = $\sim op_5 \bullet \sim op_4 \bullet op_3 \bullet op_2 \bullet \sim op_1 \bullet op_0$
lw = $op_5 \bullet \sim op_4 \bullet \sim op_3 \bullet \sim op_2 \bullet op_1 \bullet op_0$
sw = $op_5 \bullet \sim op_4 \bullet op_3 \bullet \sim op_2 \bullet op_1 \bullet op_0$
beq = $\sim op_5 \bullet \sim op_4 \bullet \sim op_3 \bullet op_2 \bullet \sim op_1 \bullet \sim op_0$
jump = $\sim op_5 \bullet \sim op_4 \bullet \sim op_3 \bullet \sim op_2 \bullet op_1 \bullet \sim op_0$

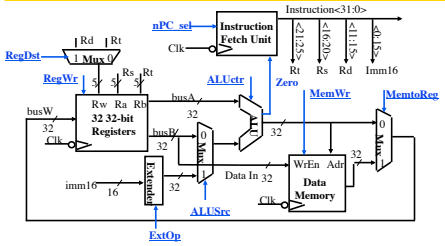
add = rtype $\bullet$ $func_5 \bullet \sim func_4 \bullet \sim func_3 \bullet \sim func_2 \bullet \sim func_1 \bullet \sim func_0$
sub = rtype $\bullet$ $func_5 \bullet \sim func_4 \bullet \sim func_3 \bullet \sim func_2 \bullet func_1 \bullet \sim func_0$

How do we implement this in gates?

Controller Implementation



Peer Instruction



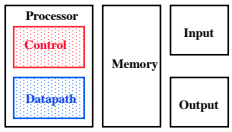
1) MemToReg='x' & ALUctr='sub'. SUB or BEQ?

2) ALUctr='add'. Which 1 signal is different for all 3 of: ADD, LW, & SW? RegDst or ExtOp?

	12
a)	SR
b)	SE
c)	BR
d)	BE

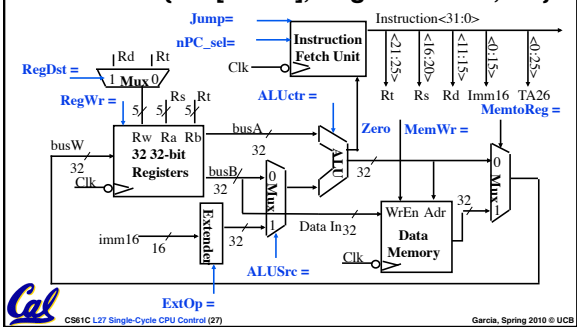
Summary: Single-cycle Processor

- 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
    - Formulate Logic Equations
    - Design Circuits

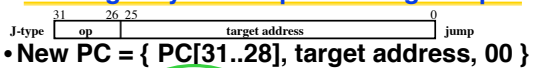


The Single Cycle Datapath during Jump

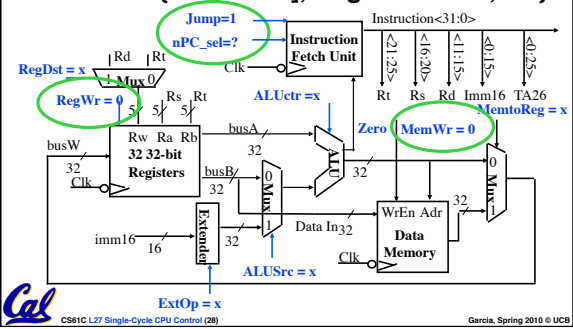
- New PC = { PC[31..28], target address, 00 }



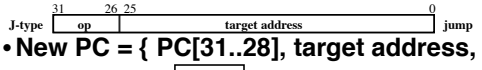
The Single Cycle Datapath during Jump



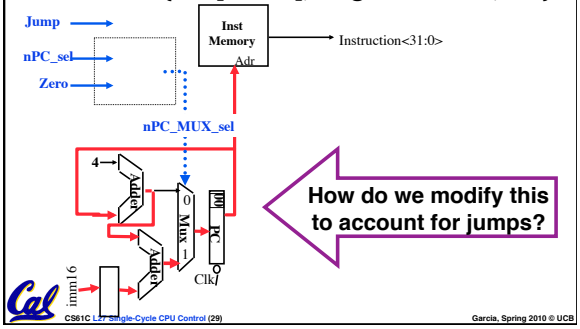
- New PC = { PC[31..28], target address, 00 }



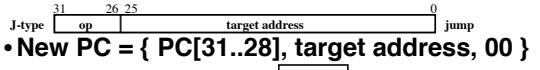
Instruction Fetch Unit at the End of Jump



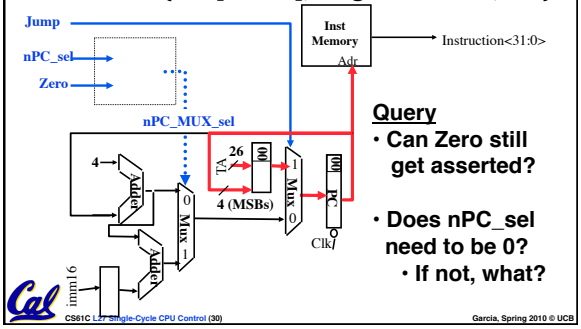
- New PC = { PC[31..28], target address, 00 }



Instruction Fetch Unit at the End of Jump



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



Query

- Can Zero still get asserted?
- Does nPC_sel need to be 0?
  - If not, what?