inst.eecs.berkeley.edu/~cs61c

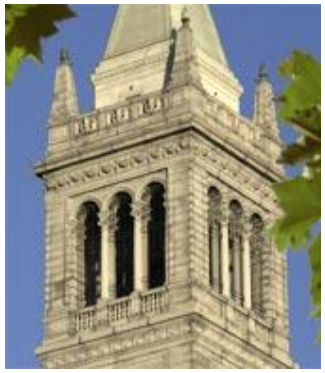
CS61C : Machine Structures

Lecture 26 – Single Cycle CPU Datapath II

Lecturer PSOE Dan Garcia

www.cs.berkeley.edu/~ddgarcia

98,389 UC Identity thefts?! ⇒



A laptop was stolen from Grad division with names, SS #, birth dates of almost 10^6 former grad students at UC Berkeley. The thief may not know what they have! Sensitive data allowed on portables? Good idea…NOT!

newscenter.berkeley.edu/security/grad/

How to Design a Processor: step-by-step

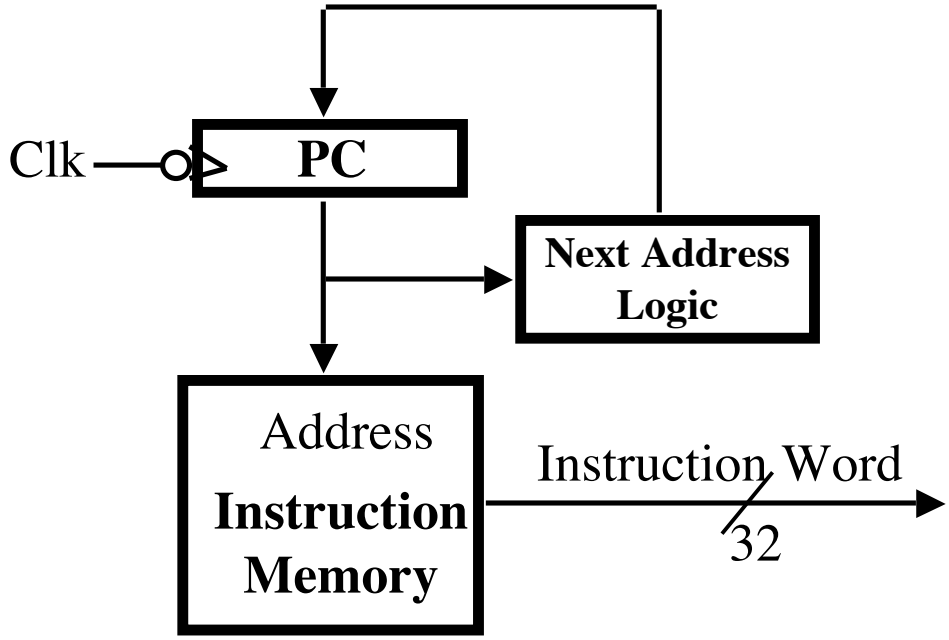
- 1. Analyze instruction set architecture (ISA) => datapath requirements
  - meaning of each instruction is given by the *register transfers*
  - datapath must include storage element for ISA registers
  - datapath must support each register transfer
- 2. Select set of datapath components and establish 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 (hard part!)

Step 3: Assemble DataPath meeting requirements

- Register Transfer Requirements
  ⇒ Datapath Assembly
- Instruction Fetch
- Read Operands and Execute Operation

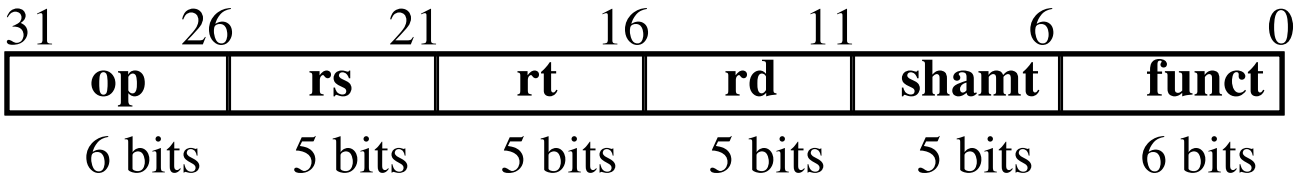
3a: Overview of the Instruction Fetch Unit

- The common RTL operations
  - Fetch the Instruction: mem[PC]
  - Update the program counter:
    - Sequential Code: PC = PC + 4
    - Branch and Jump: PC = "something else"

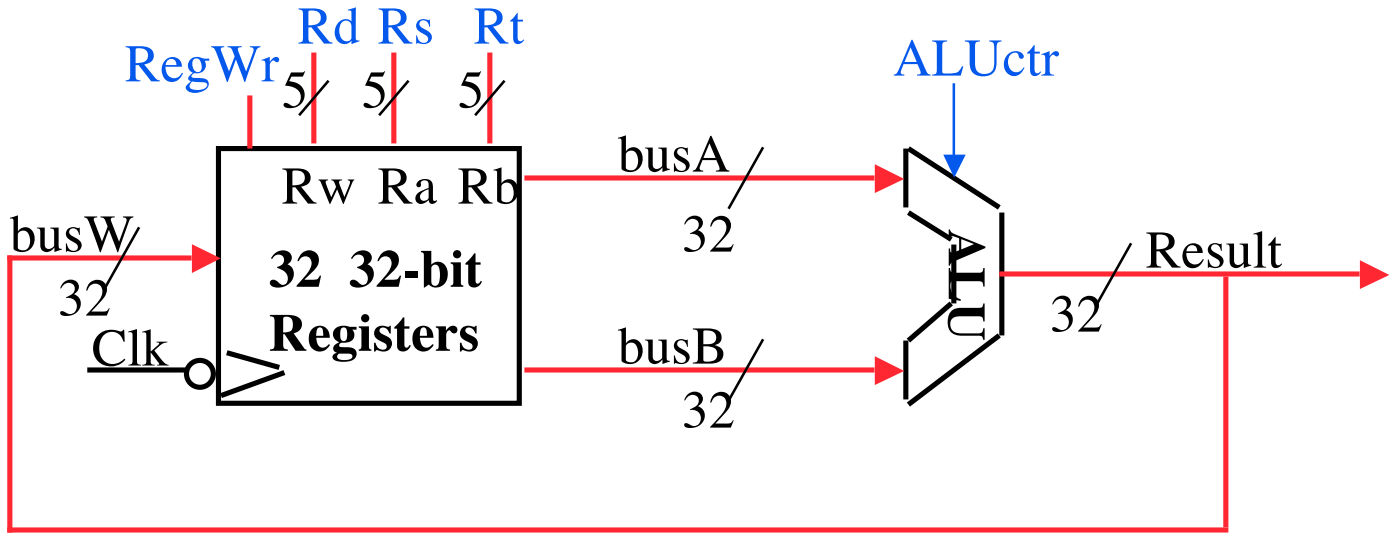


3b: Add & Subtract

- R[rd] = R[rs] op R[rt] Ex.: `addU rd,rs,rt`
  - Ra, Rb, and Rw come from instruction's Rs, Rt, and Rd fields

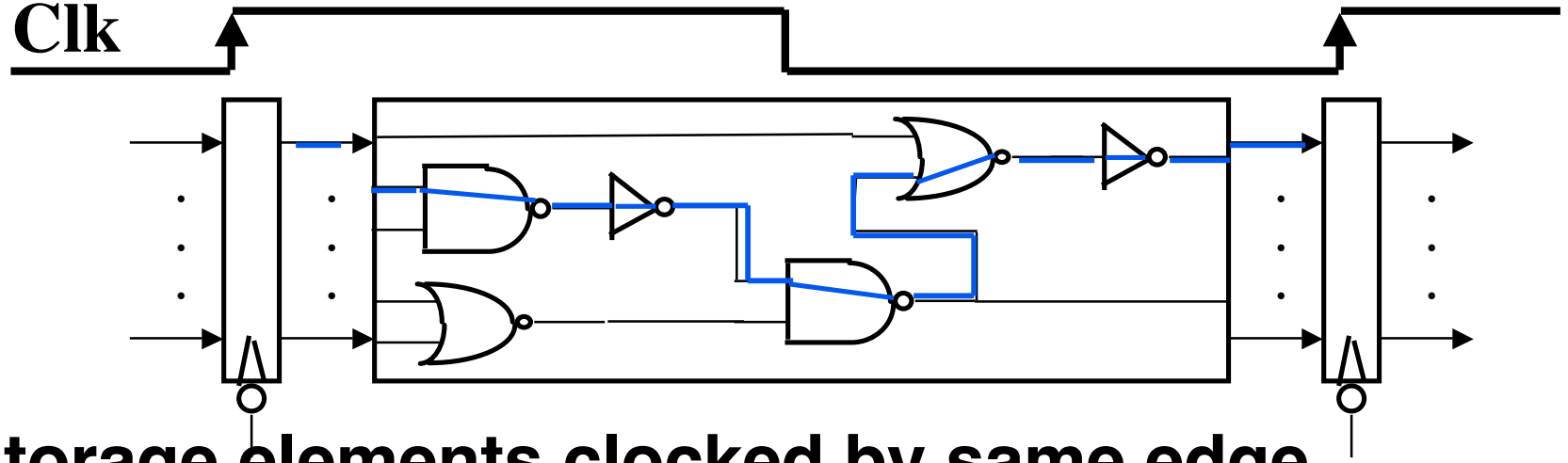


  - ALUctr and RegWr: control logic after decoding the instruction



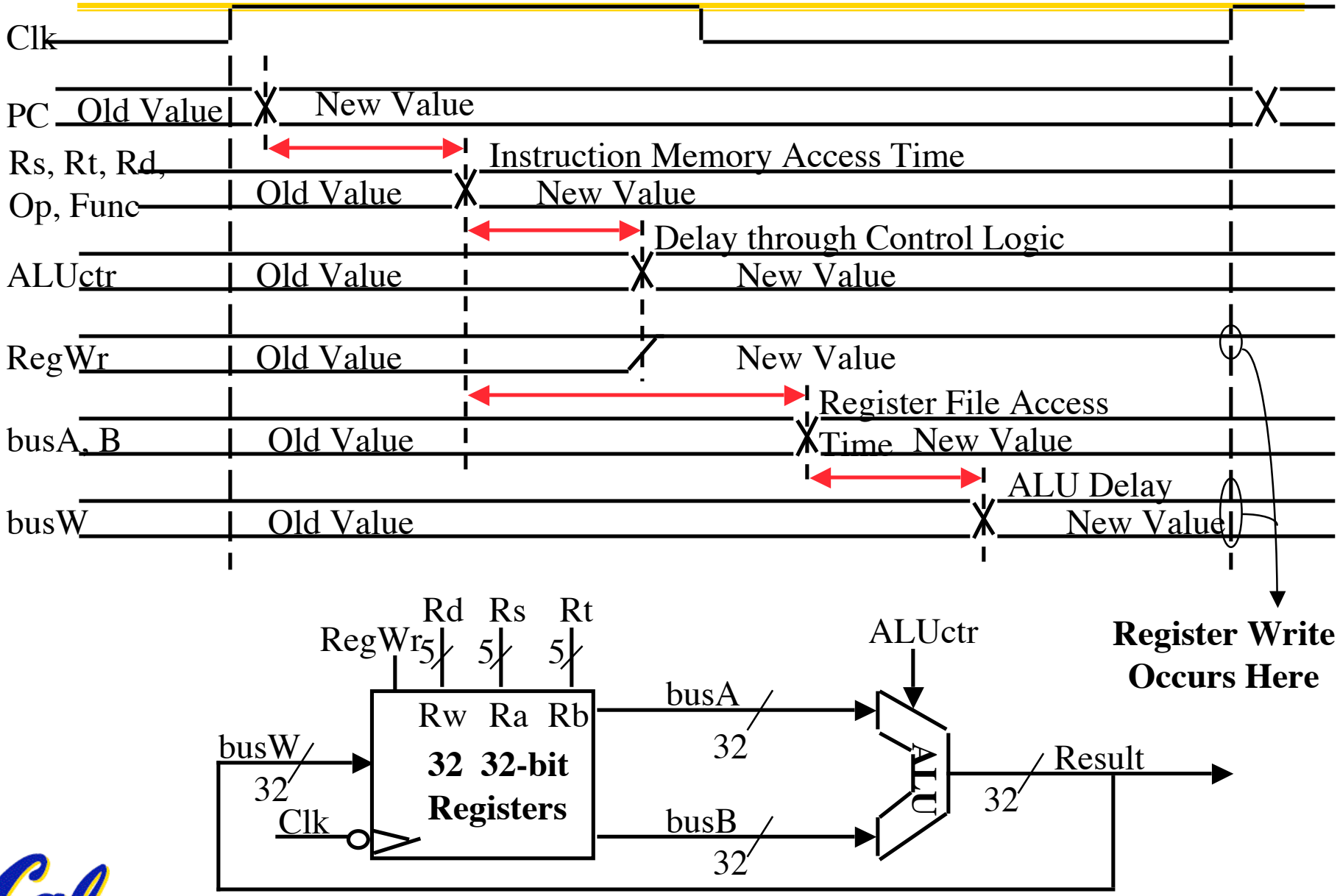
- We've already defined register file, ALU

Clocking Methodology



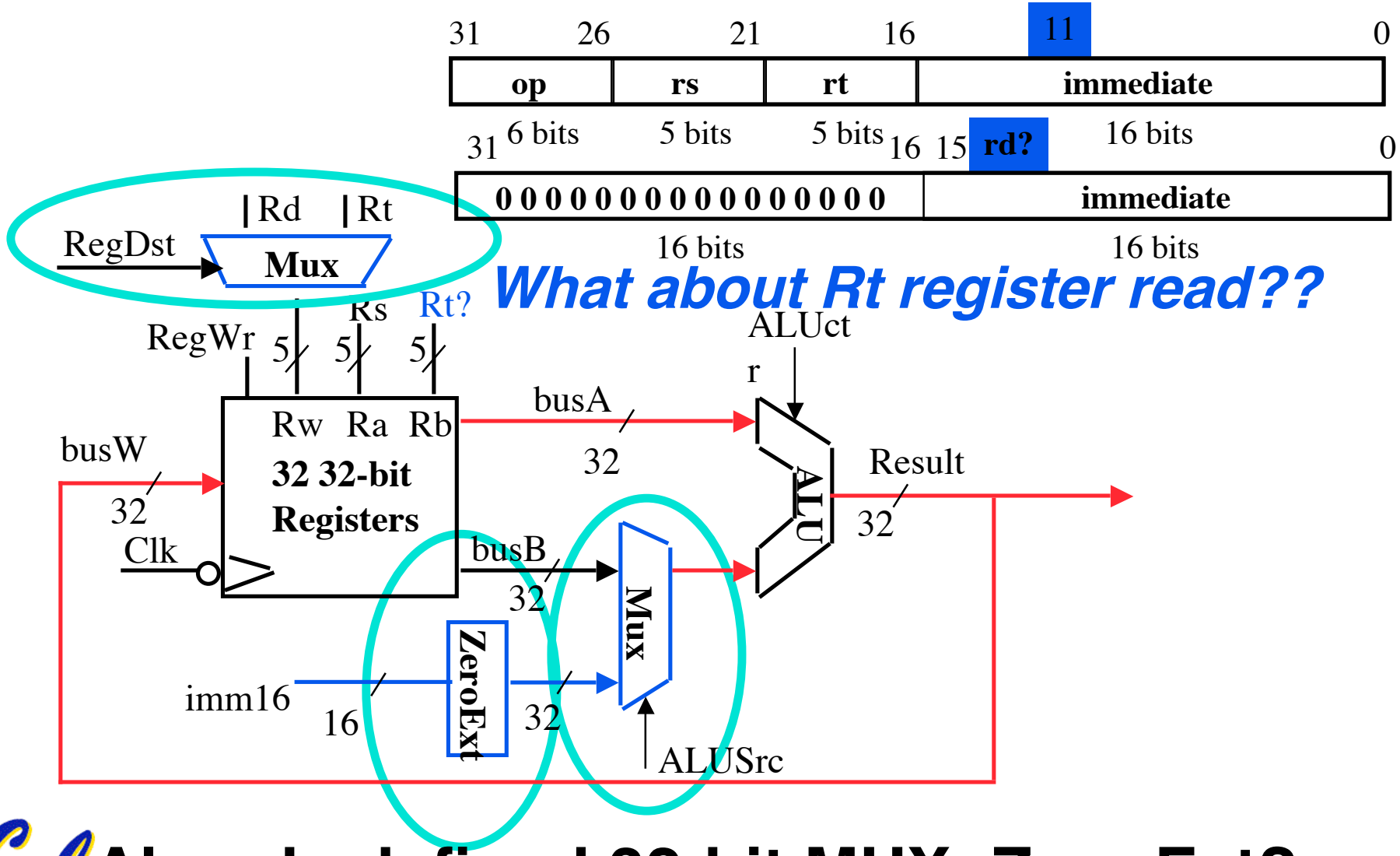
- Storage elements clocked by same edge
- Being physical devices, flip-flops (FF) and combinational logic have some delays
  - Gates: delay from input change to output change
  - Signals at FF D input must be stable before active clock edge to allow signal to travel within the FF, and we have the usual clock-to-Q delay
- "Critical path" (longest path through logic) determines length of clock period

Register-Register Timing: One complete cycle



3c: Logical Operations with Immediate

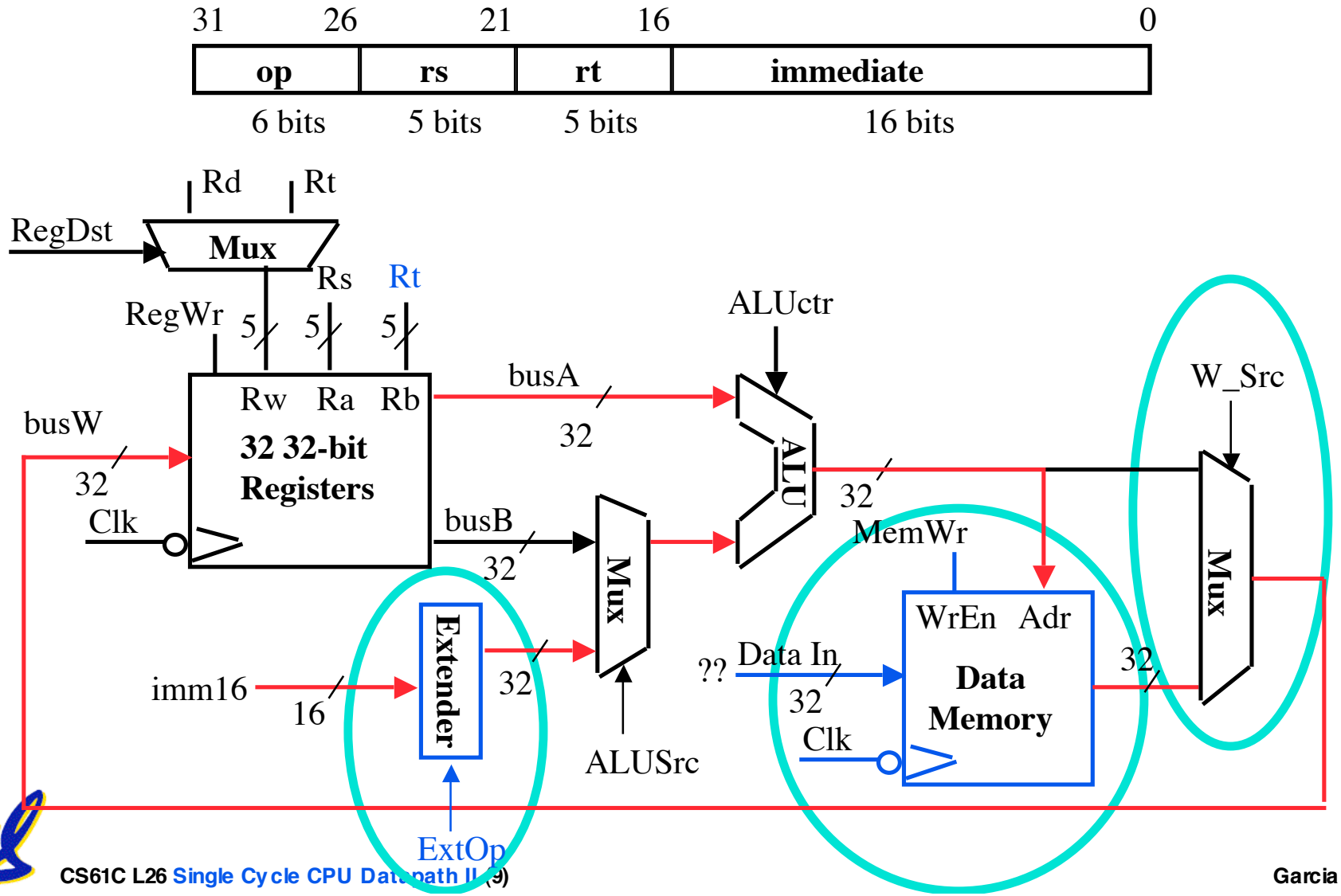
- R[rt] = R[rs] op ZeroExt[imm16]]



- Already defined 32-bit MUX; Zero Ext?

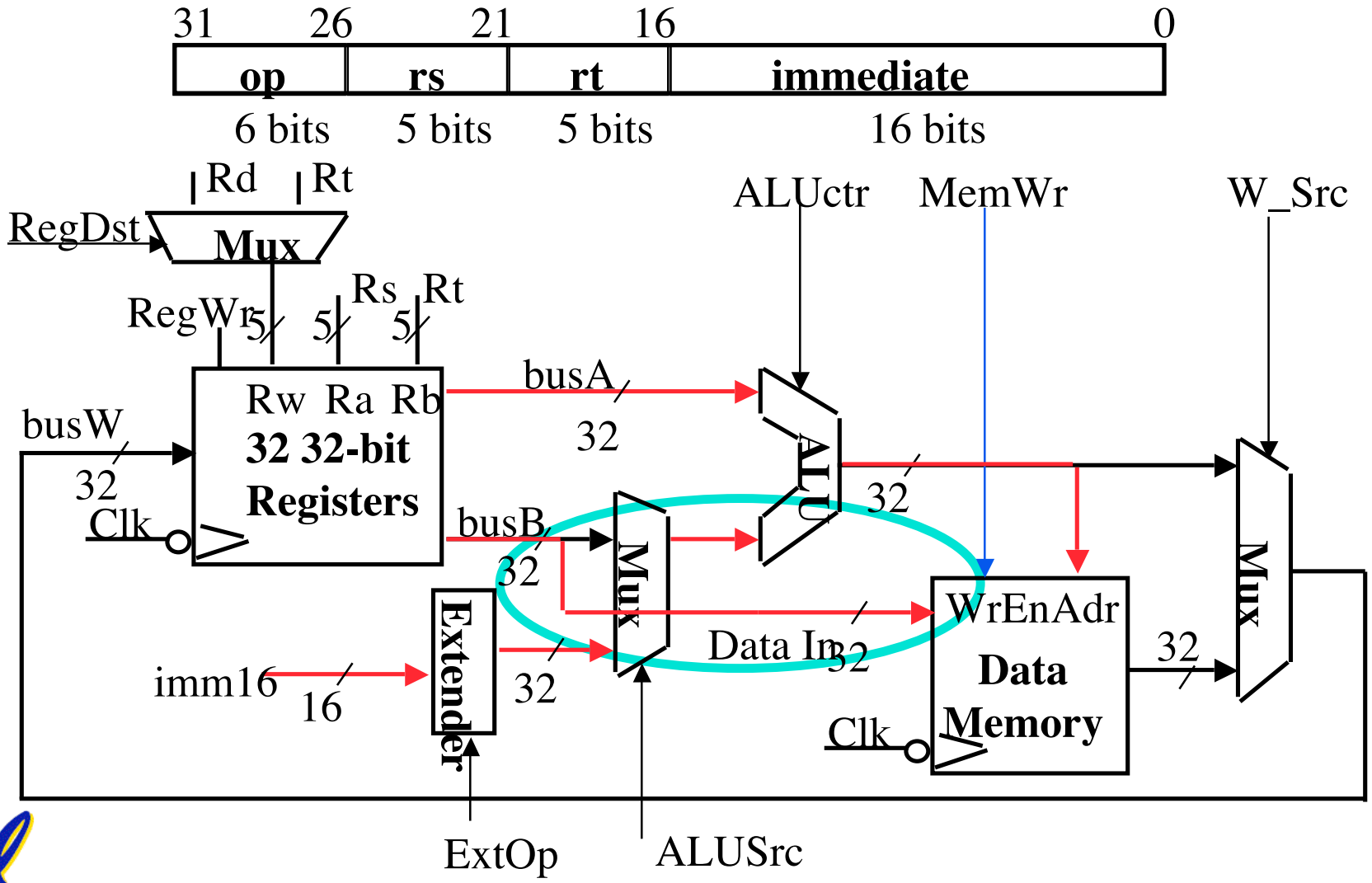
3d: Load Operations

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

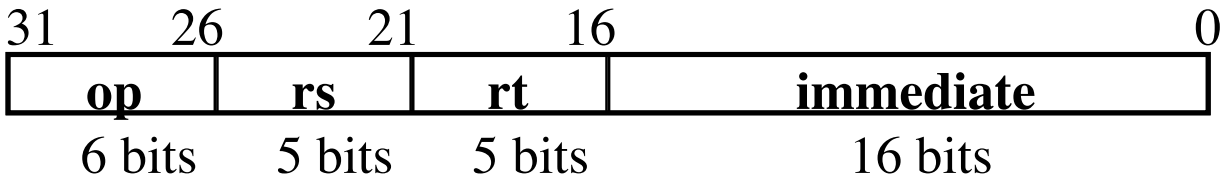


3e: Store Operations

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



3f: The Branch Instruction



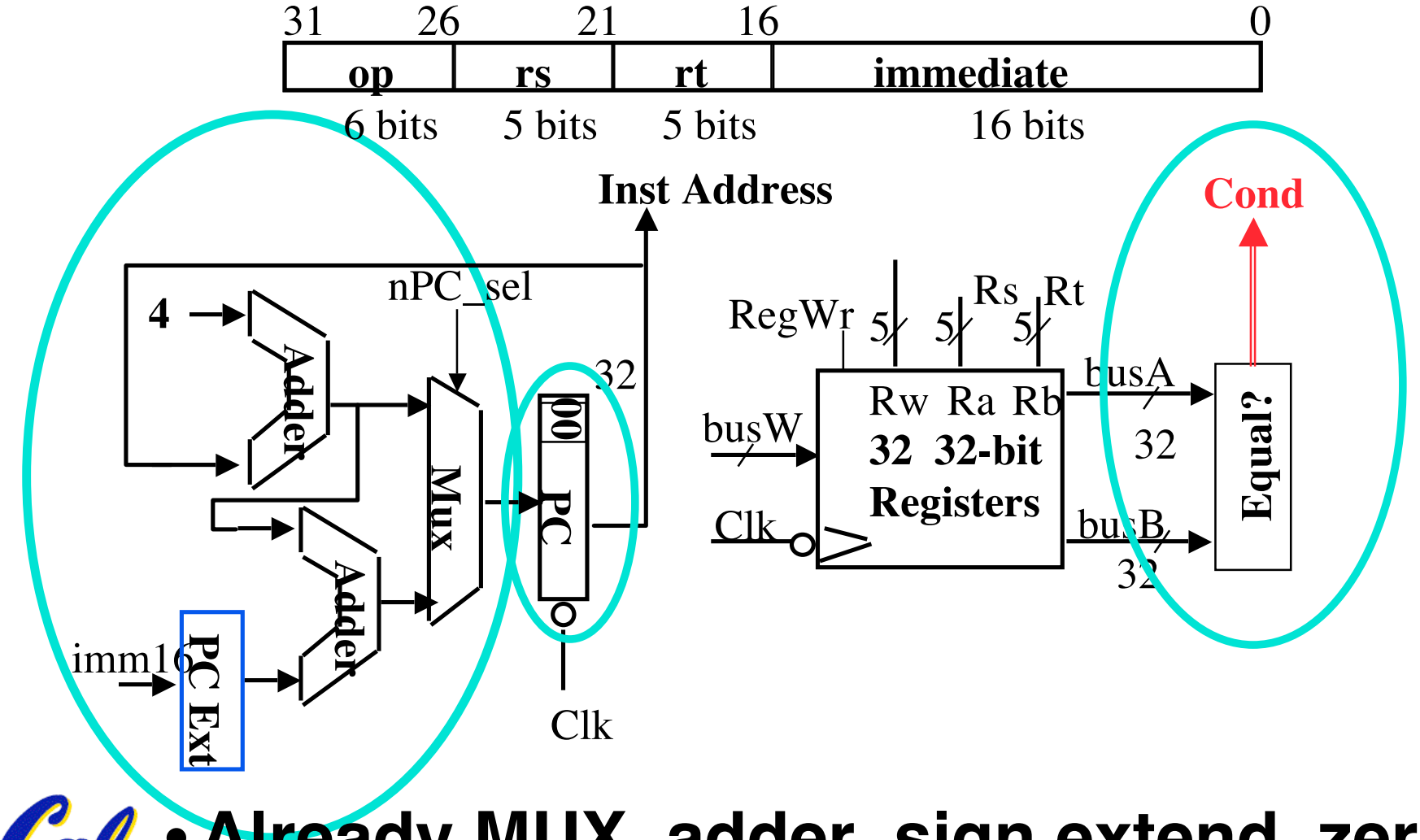
- `beq rs, rt, imm16`
  - mem[PC] Fetch the instruction from memory
  - Equal = R[rs] == R[rt] Calculate branch condition
  - if (Equal) Calculate the next instruction's address
    - PC = PC + 4 + (SignExt(imm16) x 4)

  else
    - PC = PC + 4

Datapath for Branch Operations

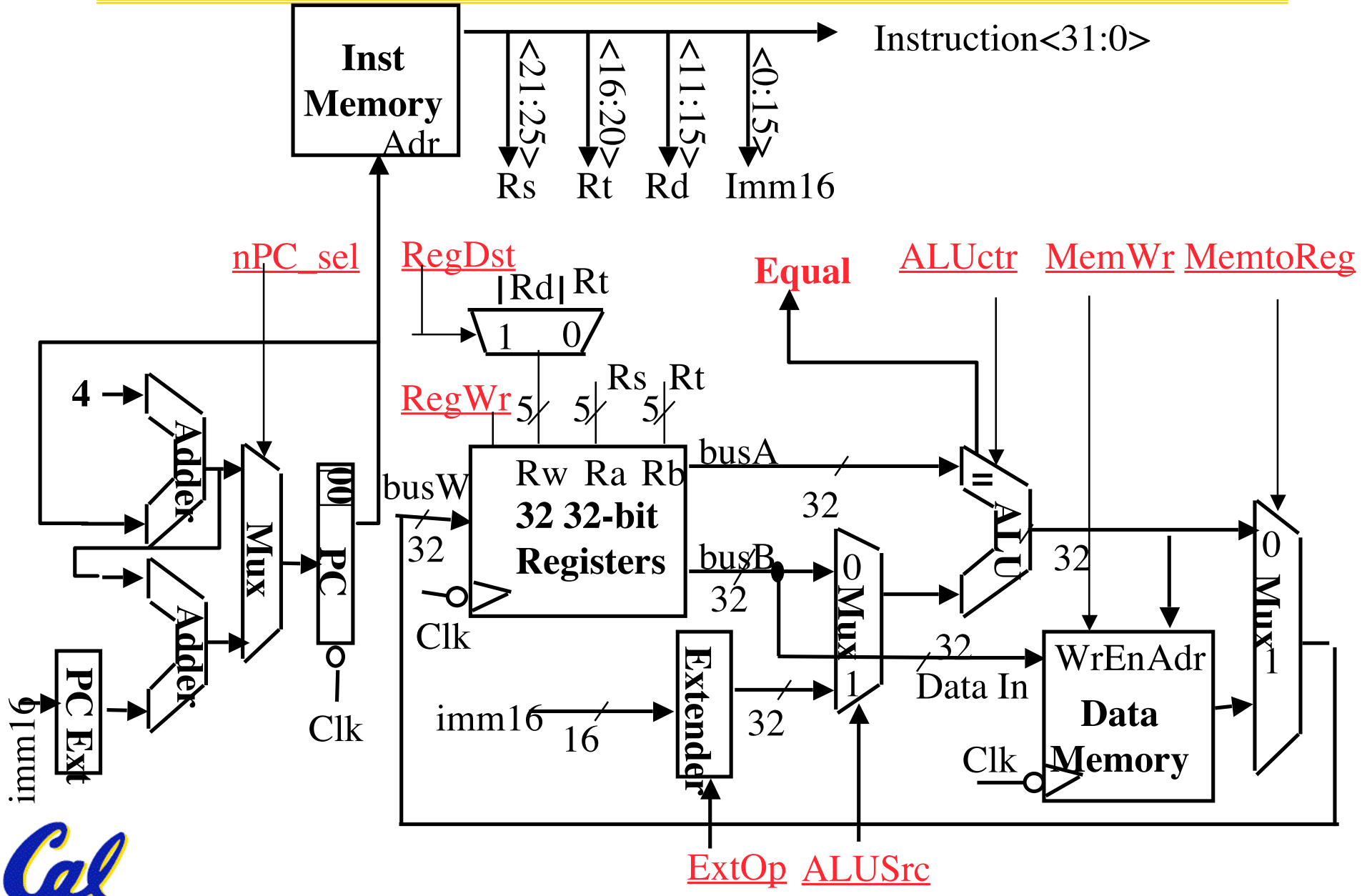
- beq rs, rt, imm16

Datapath generates condition (equal)

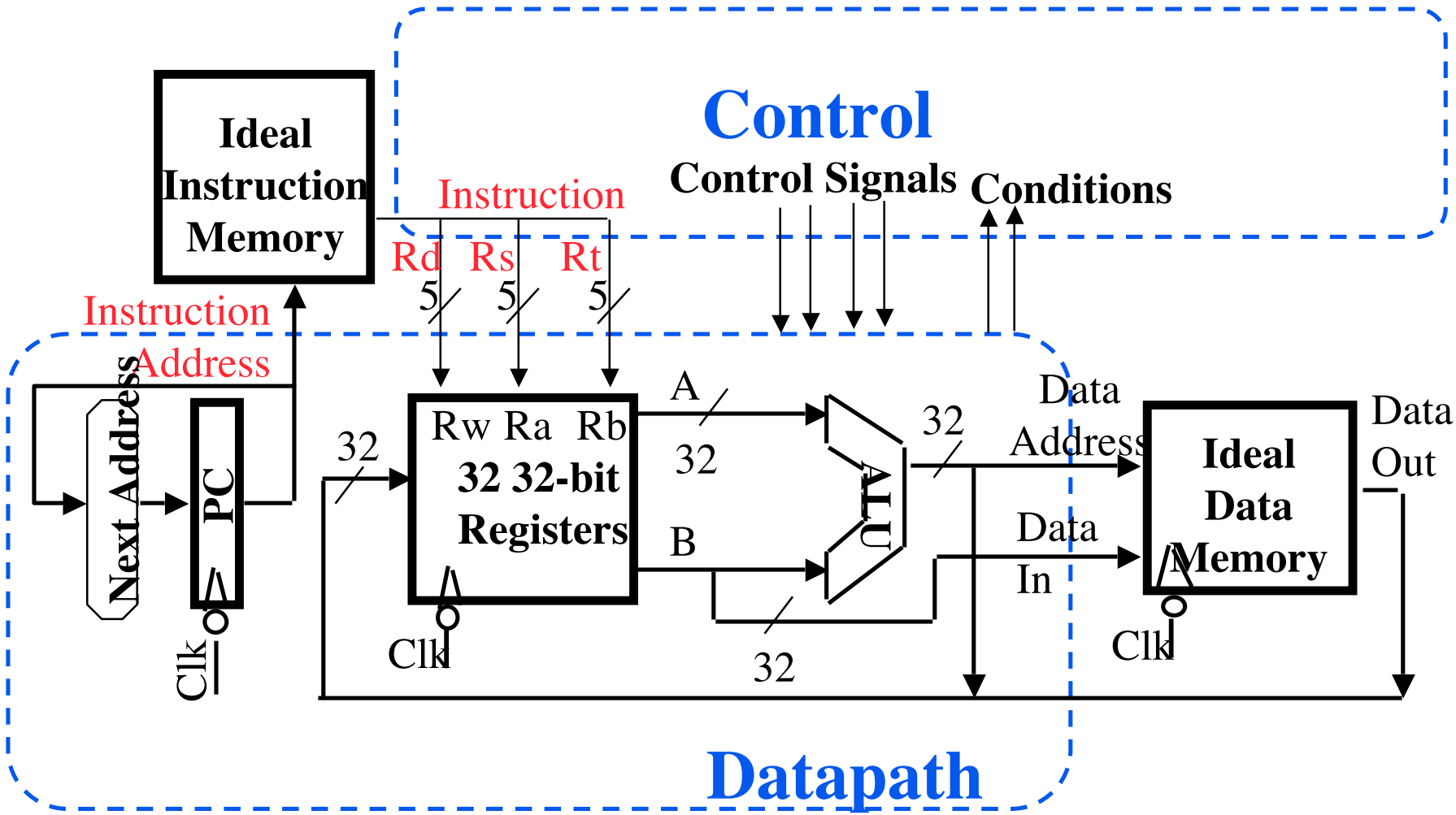


- Already MUX, adder, sign extend, zero

Putting it All Together:A Single Cycle Datapath



An Abstract View of the Implementation



Peer Instruction

A. Our ALU is a synchronous device

B. We could have used tri-state devices instead of a MUX to feed busW, the register write data line

C. The ALU is inactive for memory reads or writes.

	ABC
1:	FFF
2:	FFT
3:	FTF
4:	FTT
5:	TFF
6:	TFT
7:	TTF
8:	TTT

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
- Next time!