UC Berkeley CS61C : Machine Structures

Lecture 26
CPU Design: Designing a Single-cycle CPU, pt 2

2008-04-02

TA Omar Akkawi

inst.eecs.berkeley.edu/~cs61c-to

Creative Sues to Stop Modification of Vista Drivers

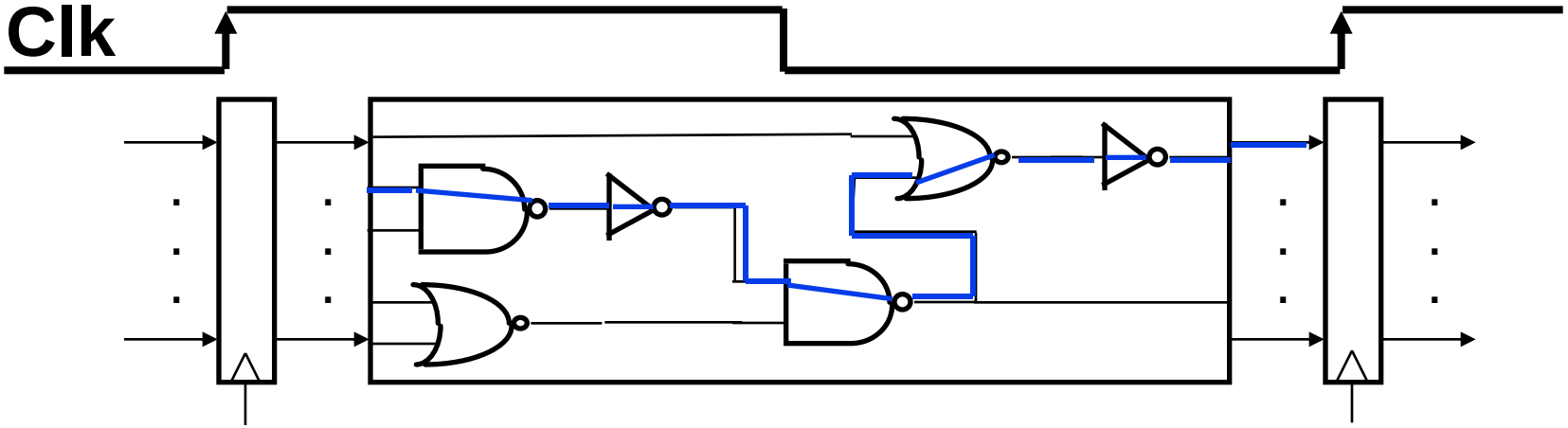
Creative recently sued an individual to prevent them from releasing modified versions of Vista drivers for the X-Fi soundcard.

http://hardware.slashdot.org/hardware/08/03/29/046201.shtml?tid=222

How to Design a Processor: step-by-step

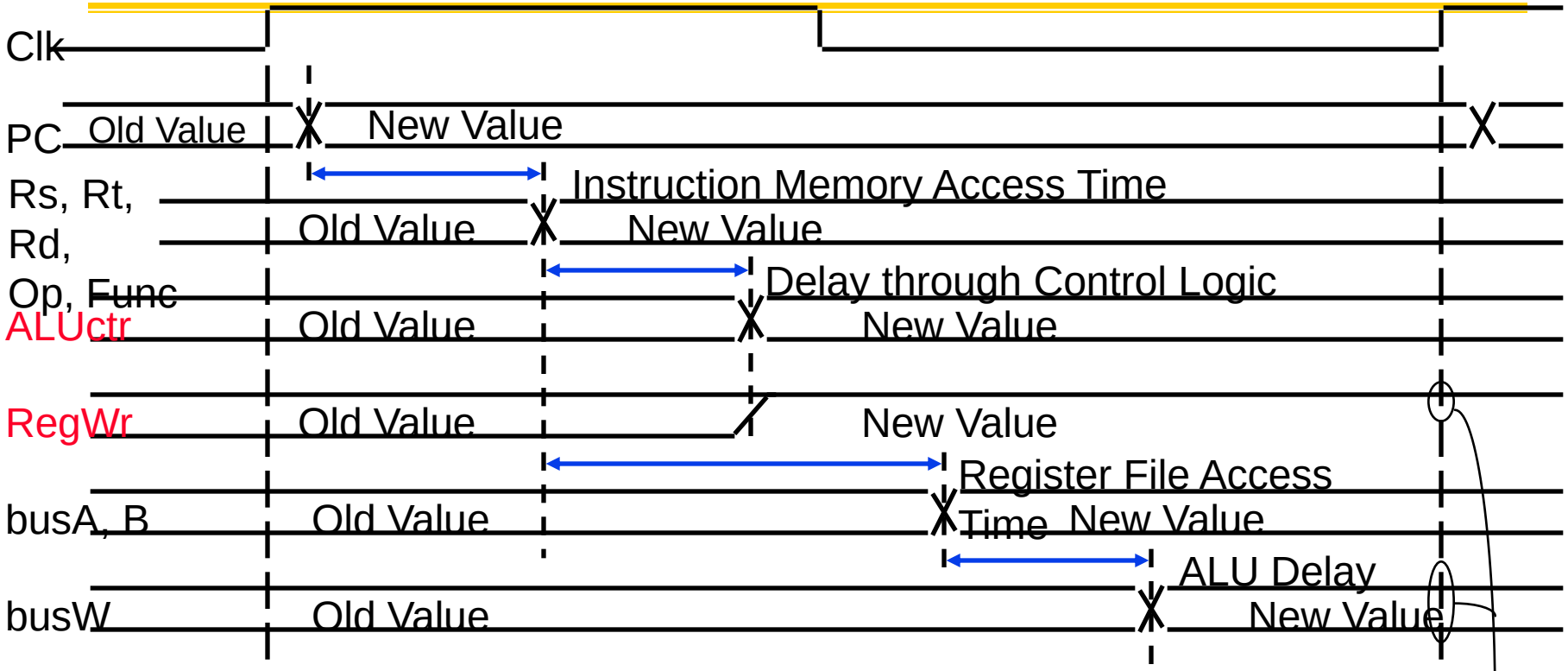
1. Analyze instruction set architecture (ISA) => datapath requirements
   - meaning of each instruction is given by the *register transfers*
   - datapath must include storage element for ISA registers
   - datapath must support each register transfer
2. Select set of datapath components and establish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

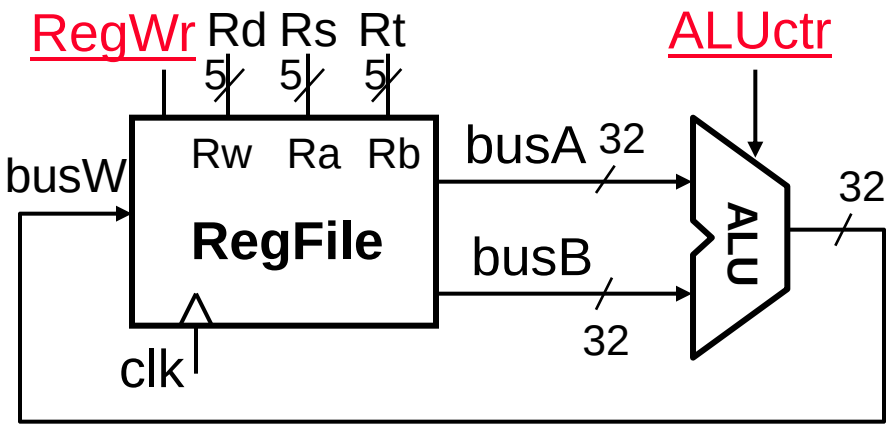
Clocking Methodology



- 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 (set-up time), 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

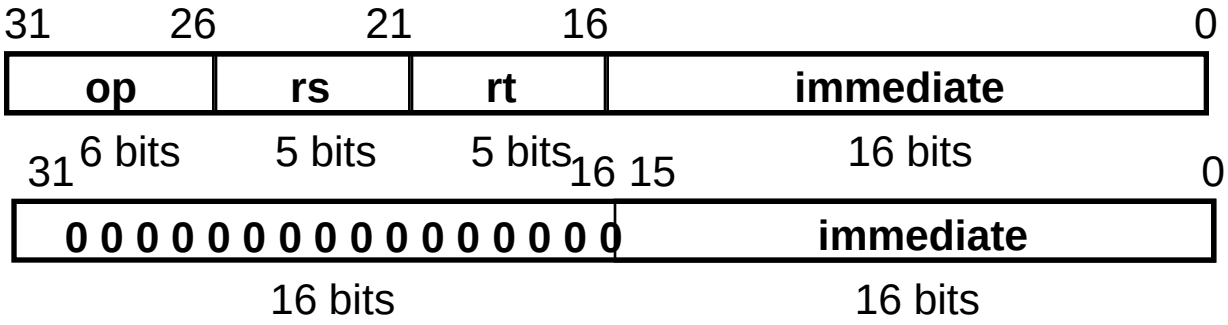




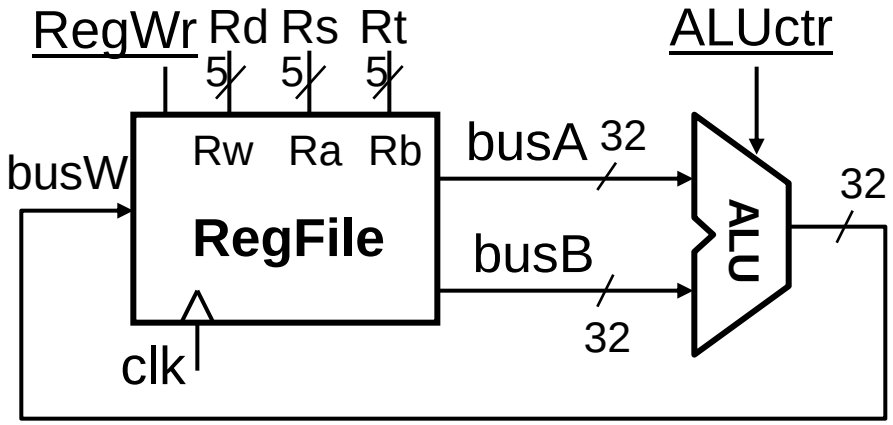
Register Write Occurs Here

3c: Logical Operations with Immediate

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

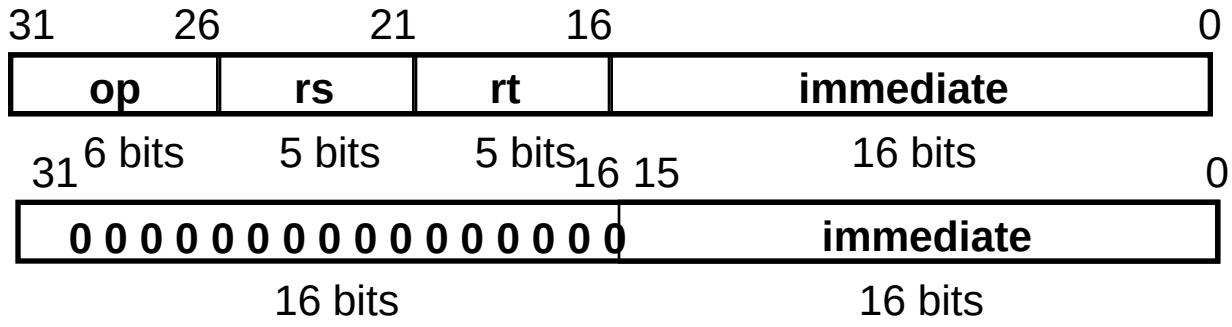


But we're writing to Rt register??

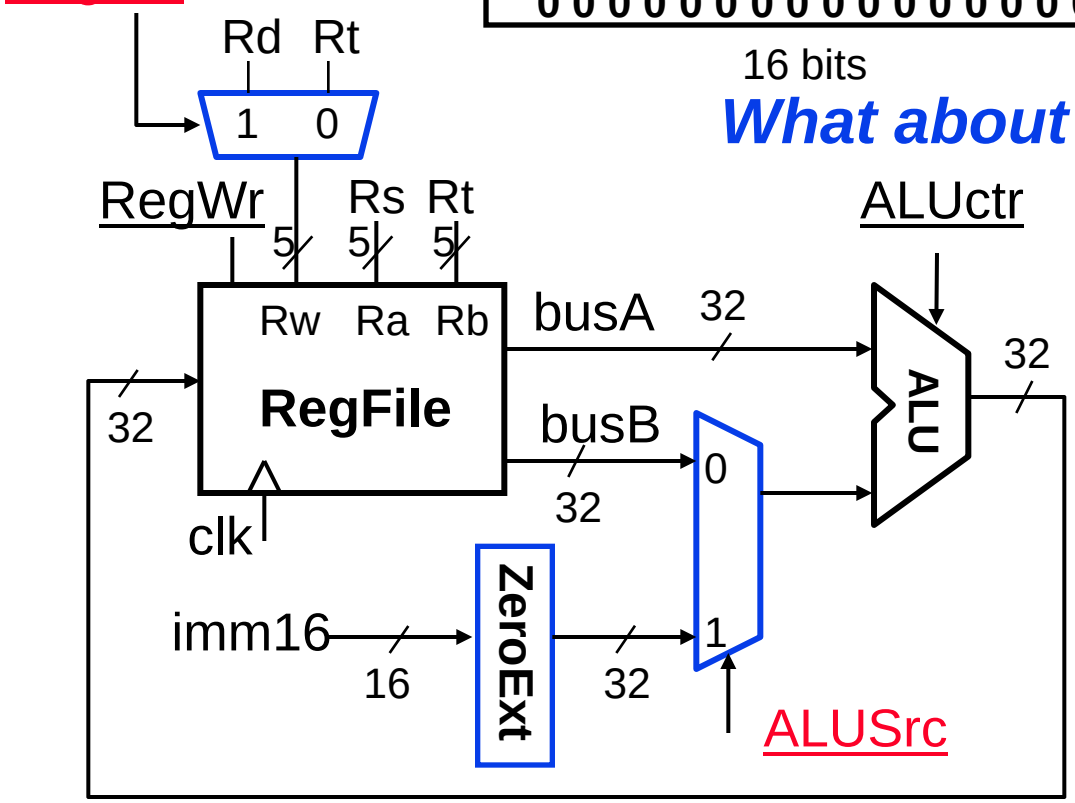


3c: Logical Operations with Immediate

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



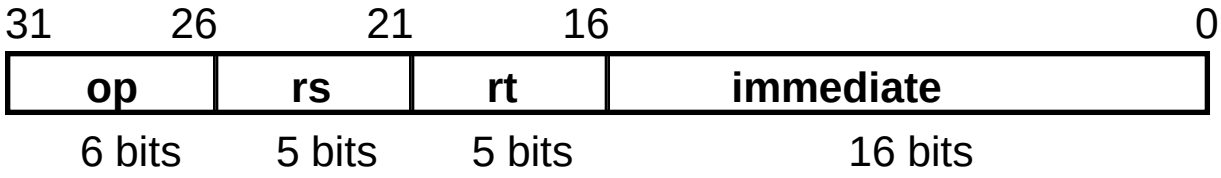
What about Rt register read??

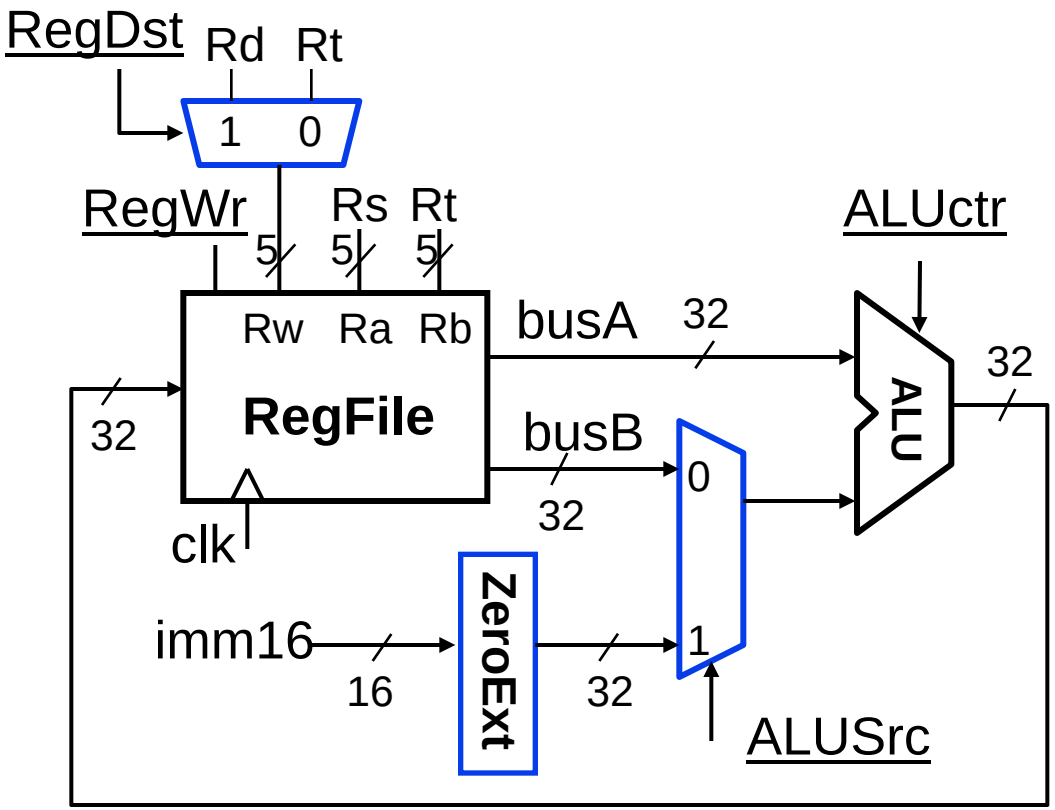


- Already defined 32-bit MUX; Zero Ext?

3d: Load Operations

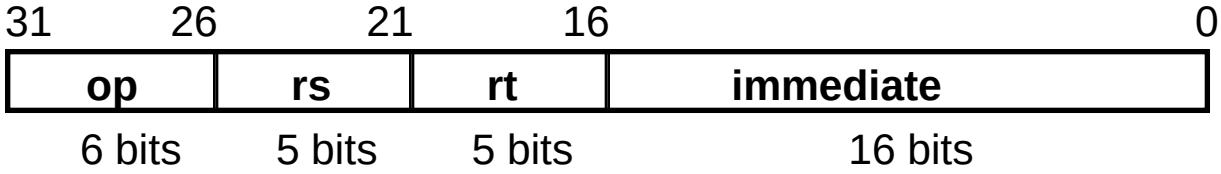
- R[rt] = Mem[R[rs] + SignExt[imm16]]
  Example: `lw rt,rs,imm16`

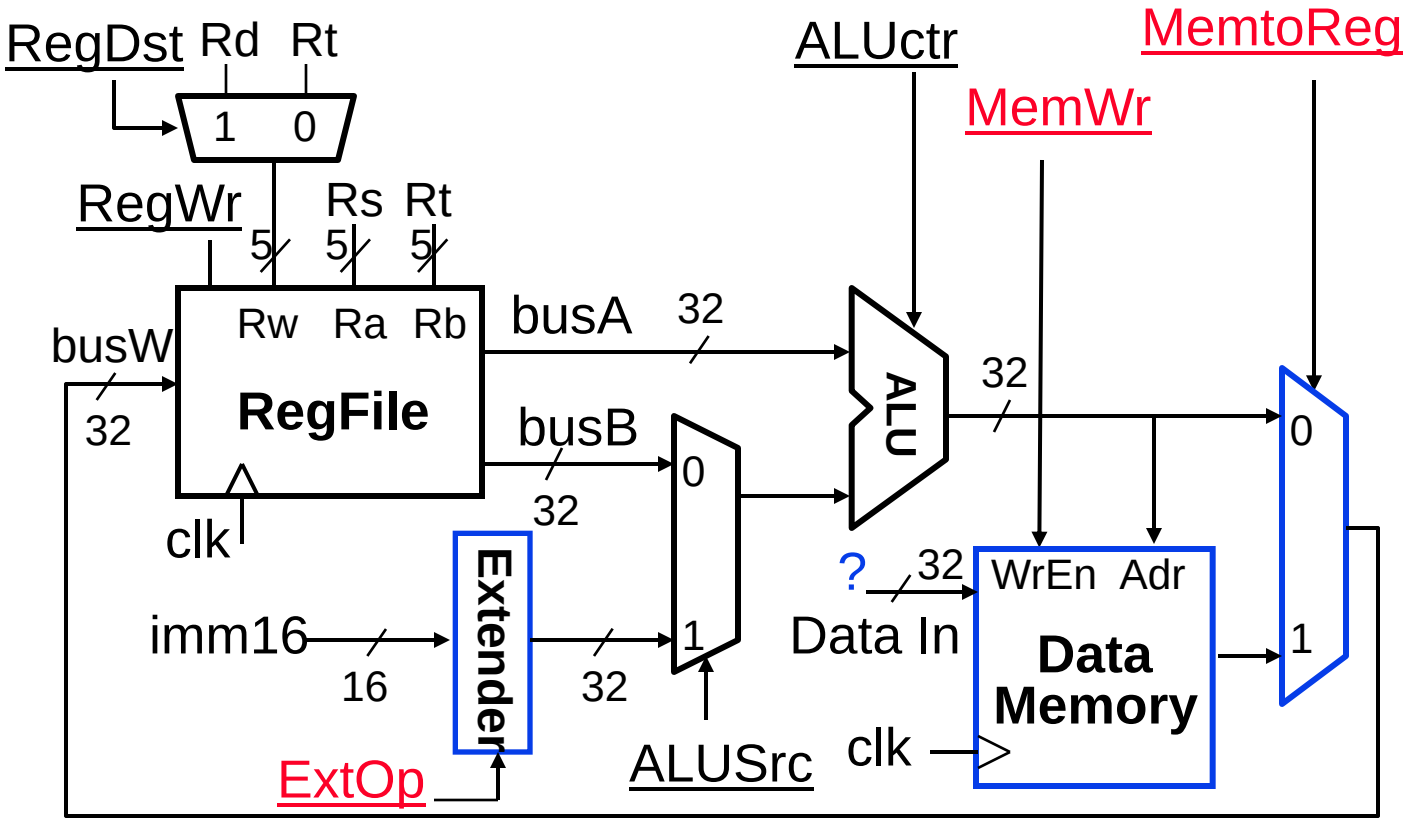




3d: Load Operations

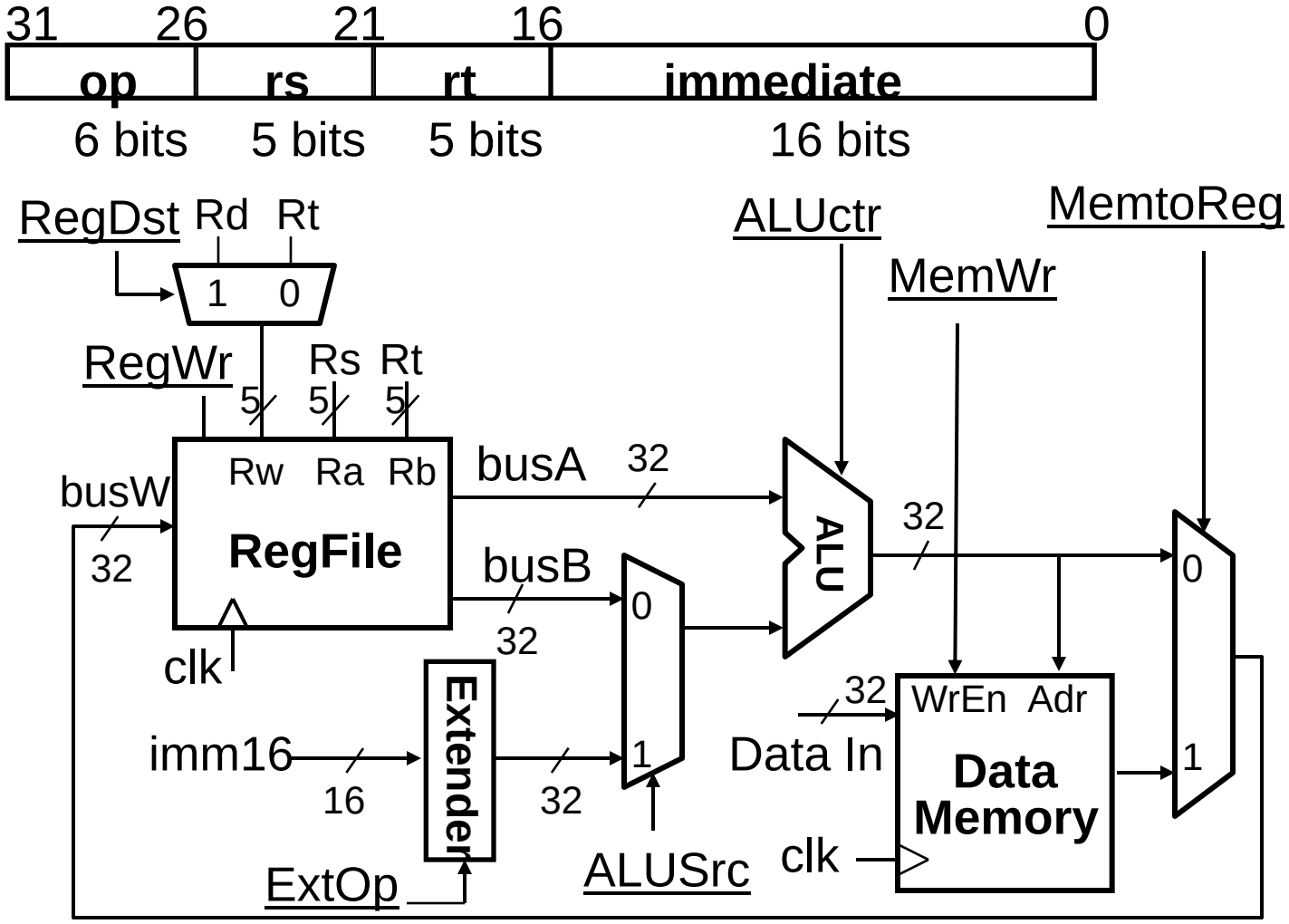
- R[rt] = Mem[R[rs] + SignExt[imm16]]
  Example: `lw rt,rs,imm16`





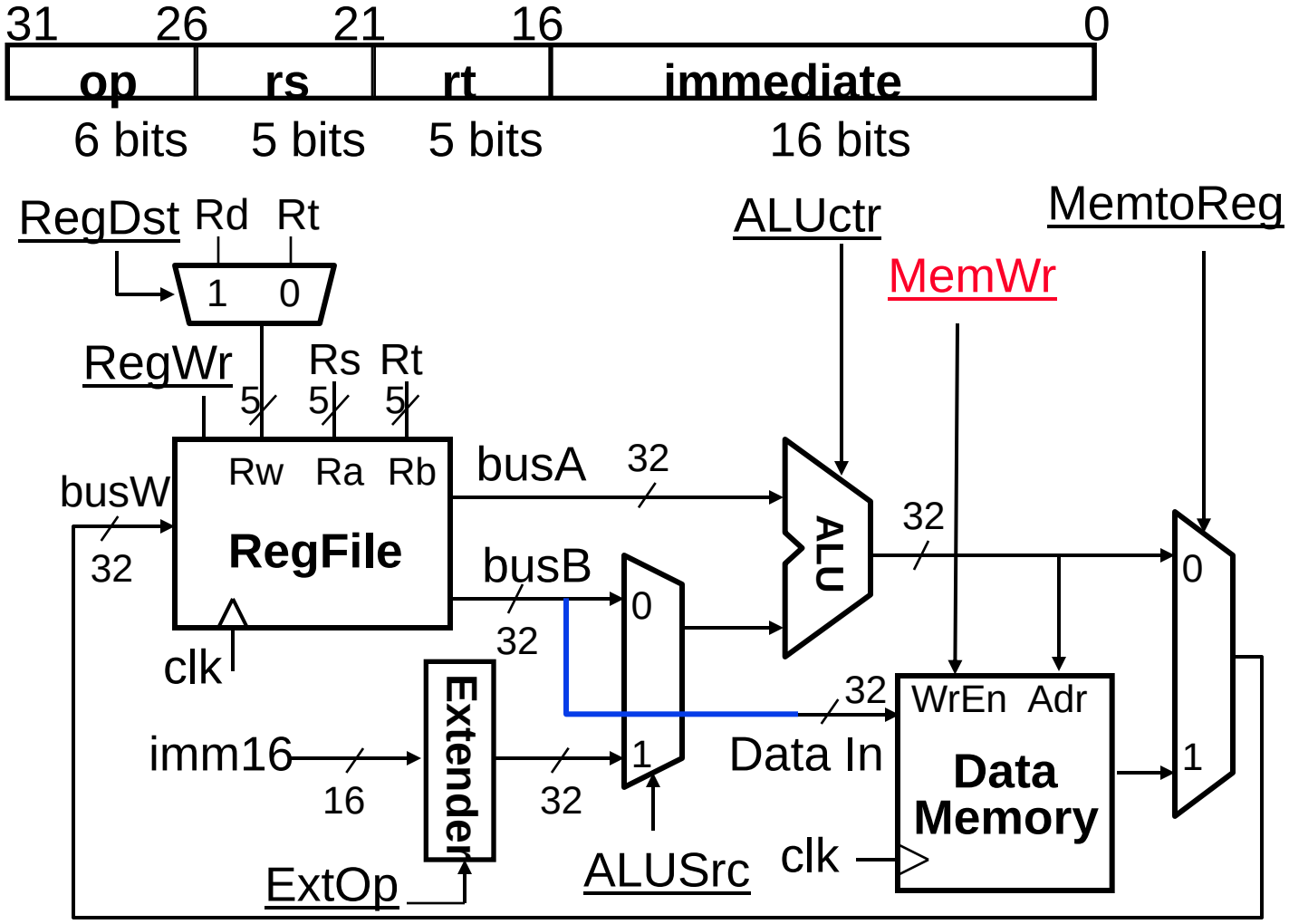
3e: Store Operations

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

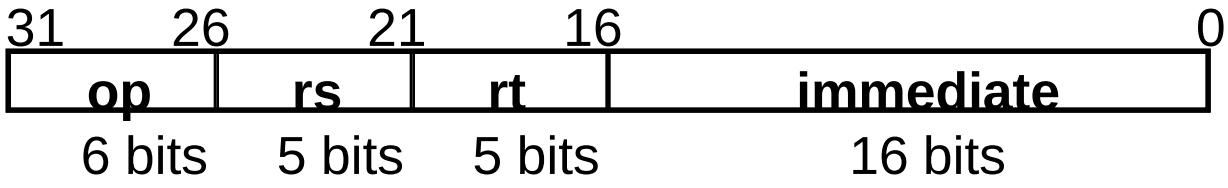


3e: Store Operations

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



3f: The Branch Instruction



`beq rs, rt, imm16`

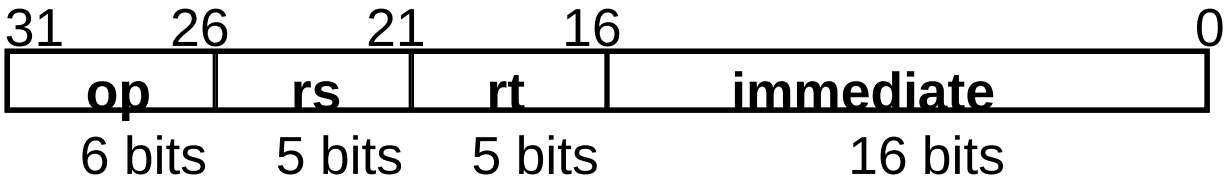
- mem[PC] Fetch the instruction from memory
- Equal = R[rs] == R[rt] Calculate branch condition
- if (Equal) Calculate the next instruction's address
  - PC = PC + 4 + (SignExt(imm16) x 4)

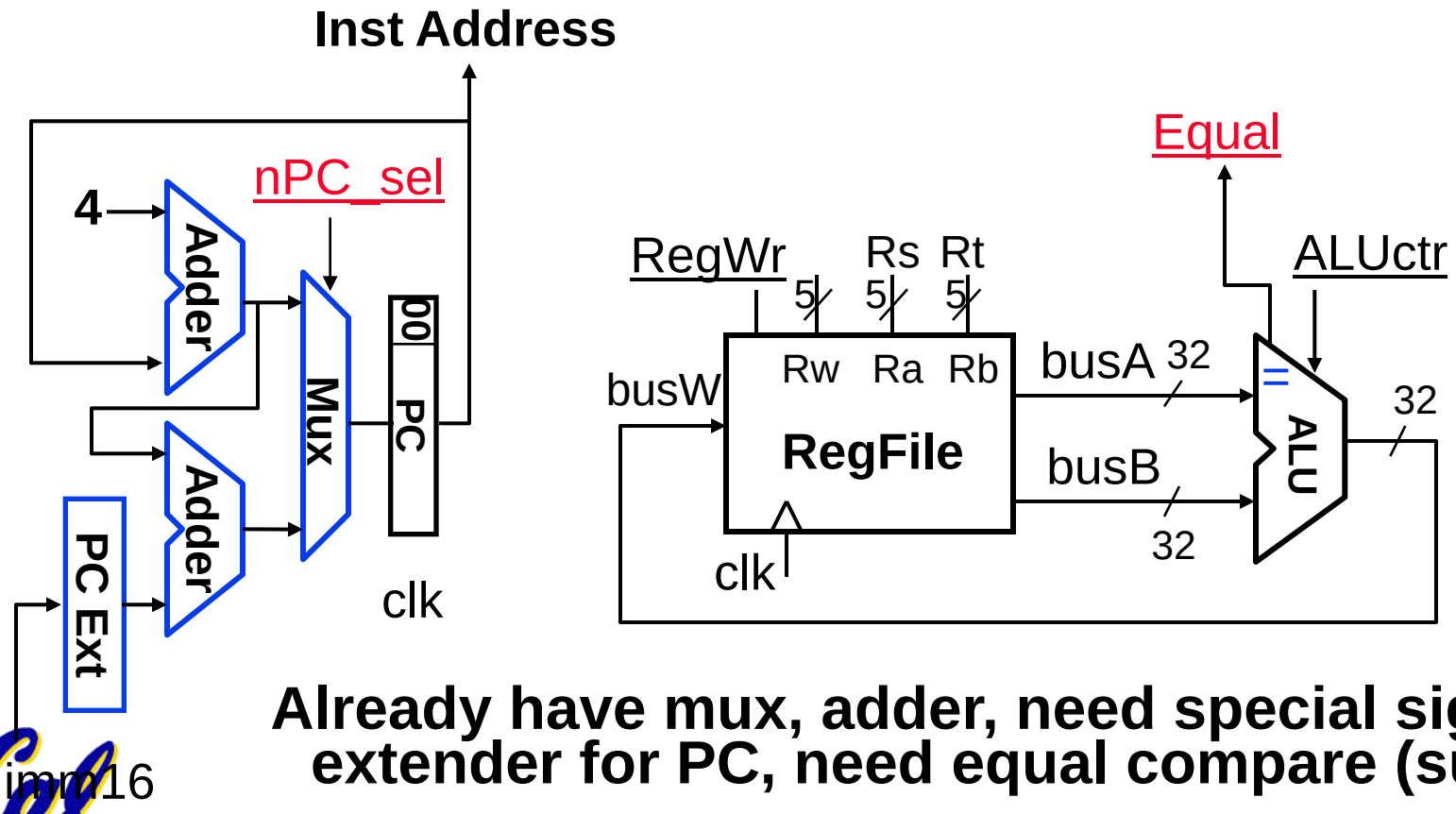
  else
  - PC = PC + 4

Datapath for Branch Operations

- **beq rs, rt, imm16**

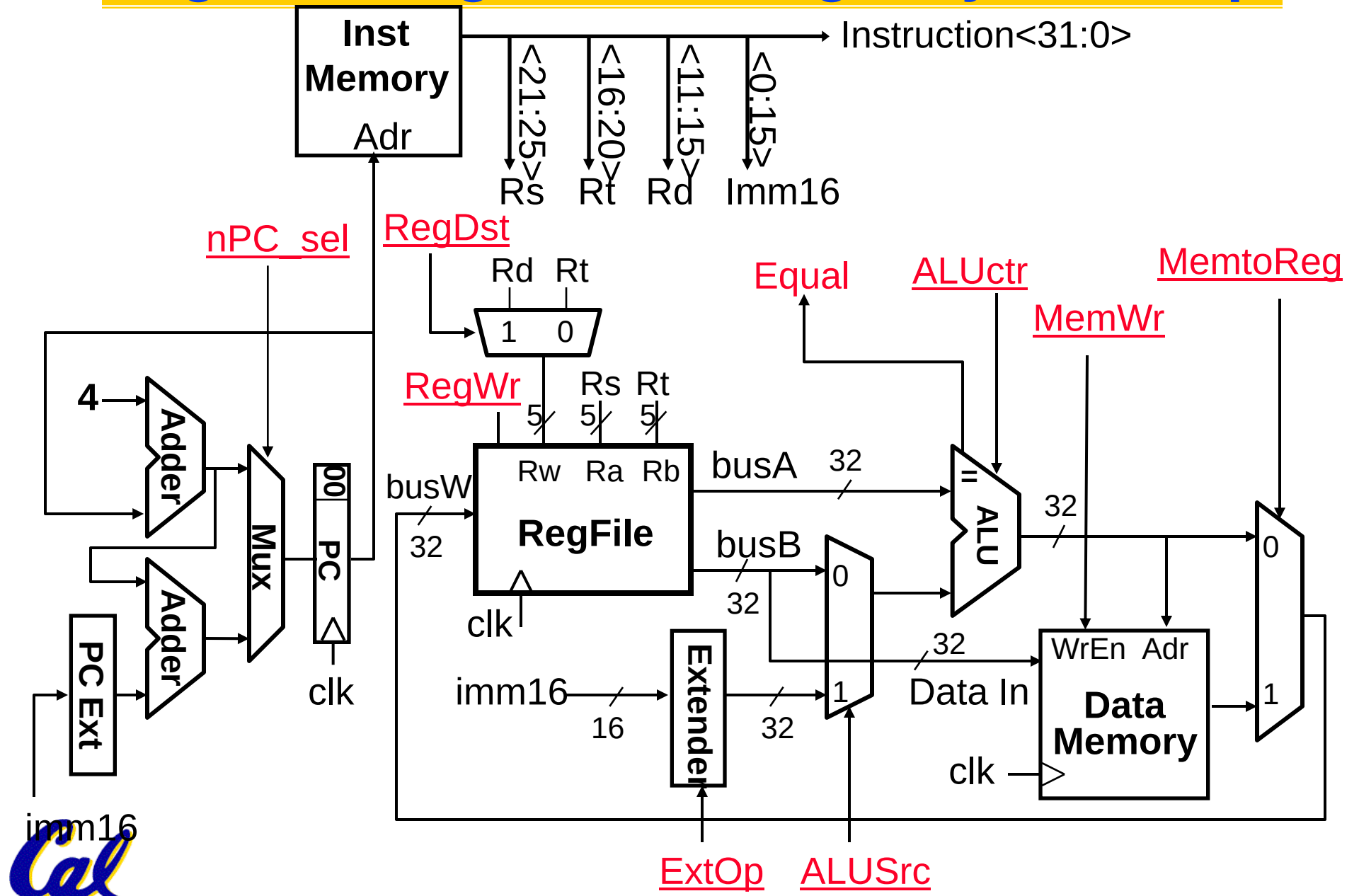
Datapath generates condition (equal)



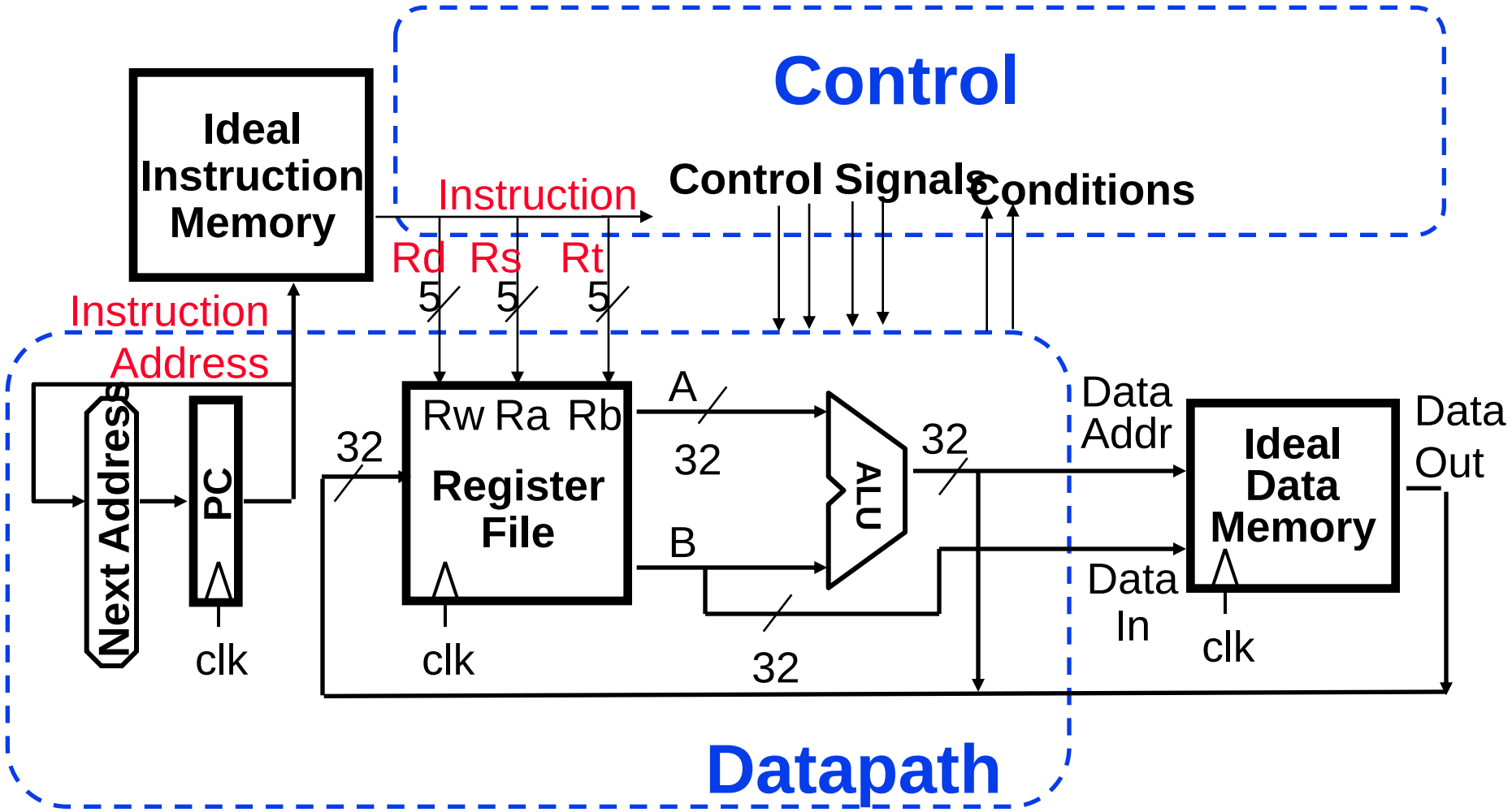


Already have mux, adder, need special sign extender for PC, need equal compare (sub?)

Putting it All Together:A Single Cycle Datapath



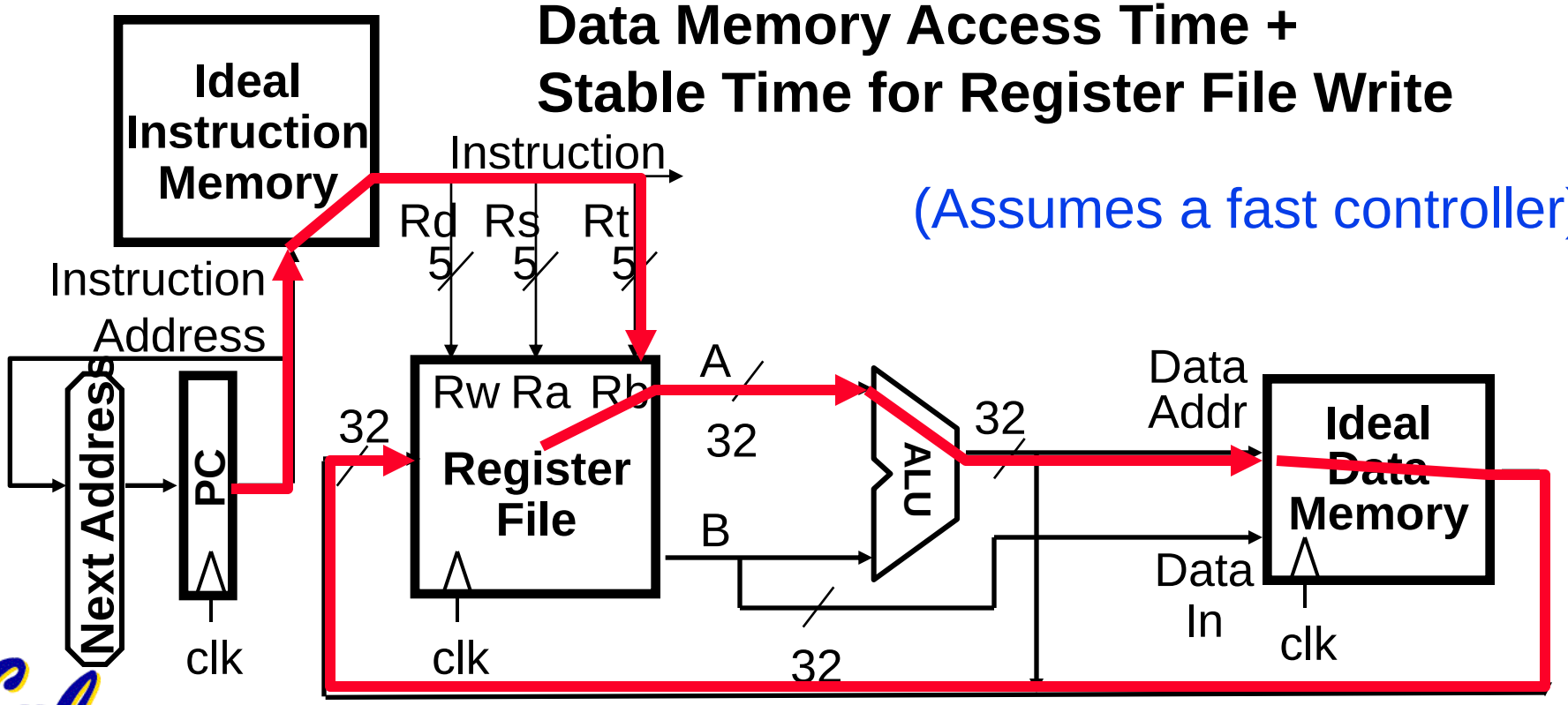
An Abstract View of the Implementation



An Abstract View of the Critical Path

Critical Path (Load Instruction) =
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

(Assumes a fast controller)



Administrivia

- **Authnews is back up!**
  - **The server for off-campus newsgroup access was down for a few days, but it's back up now, so resume using it.**
- **Project 1 grading is being looked into.**
  - **Many students have had questions about grades for project 1, so the TAs in charge of project 1 are investigating it. Stay tuned for further announcements.**
- **Homework 6 due Saturday.**

Peer Instruction

A. Our ALU is a synchronous device

B. We should use the main ALU to compute PC=PC+4

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

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

Summary: A Single Cycle Datapath

- We have everything except control signals