CS 152
Computer Architecture and Engineering

Lecture 2 – Single Cycle Datapaths

2006-8-31

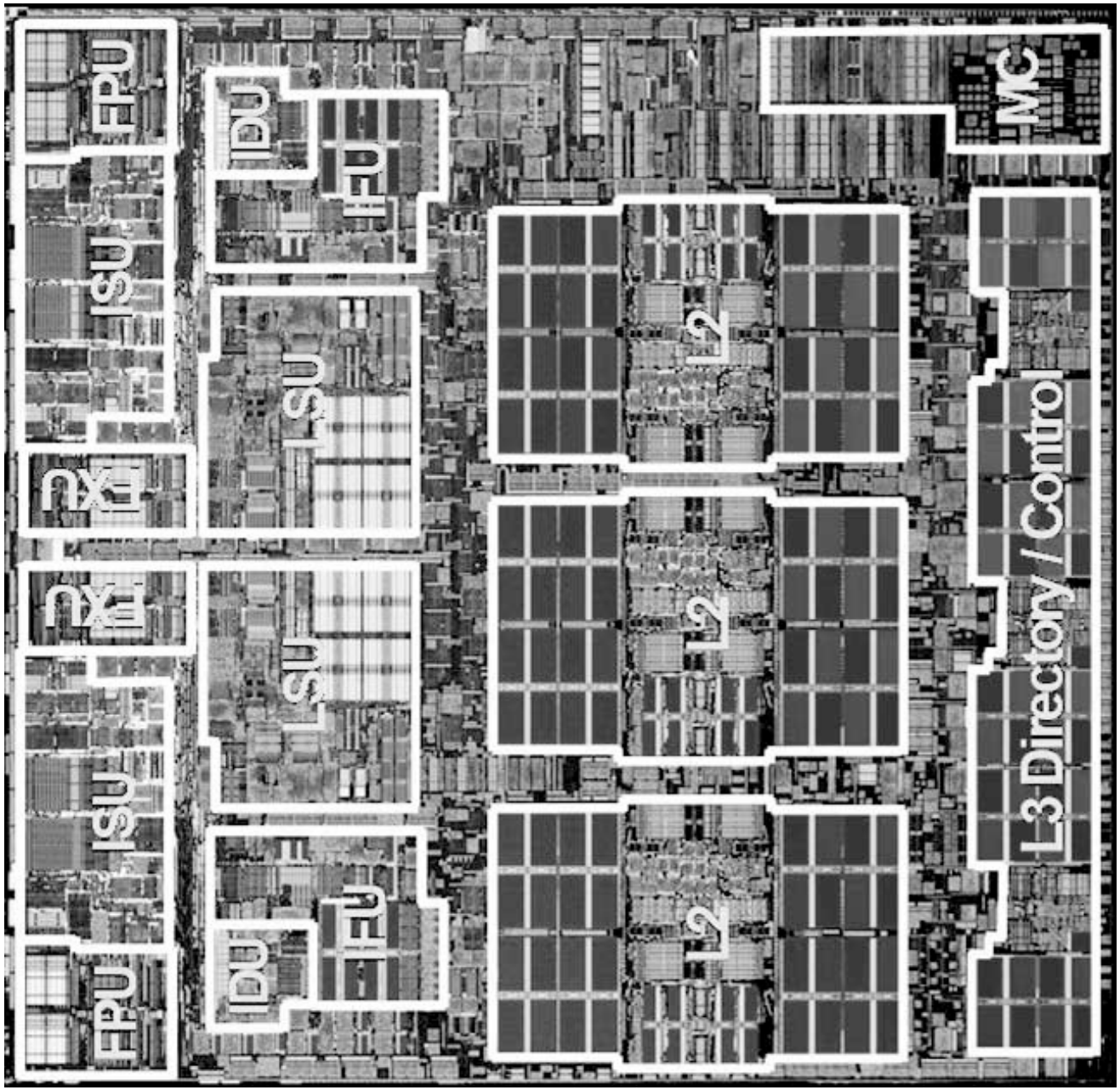
John Lazzaro
(www.cs.berkeley.edu/~lazzaro)

TAs: Udam Saini and Jue Sun

www-inst.eecs.berkeley.edu/~cs152/

UC Regents Fall 2006 © UCB

Last Time: CS 152 Course Introduction



IBM Power 5 “die photo”: a die is an unpackaged part

Single-cycle CPU project
3 weeks

Pipelined CPU
4 weeks

Final Project
5 weeks

Teams of 4-5 students

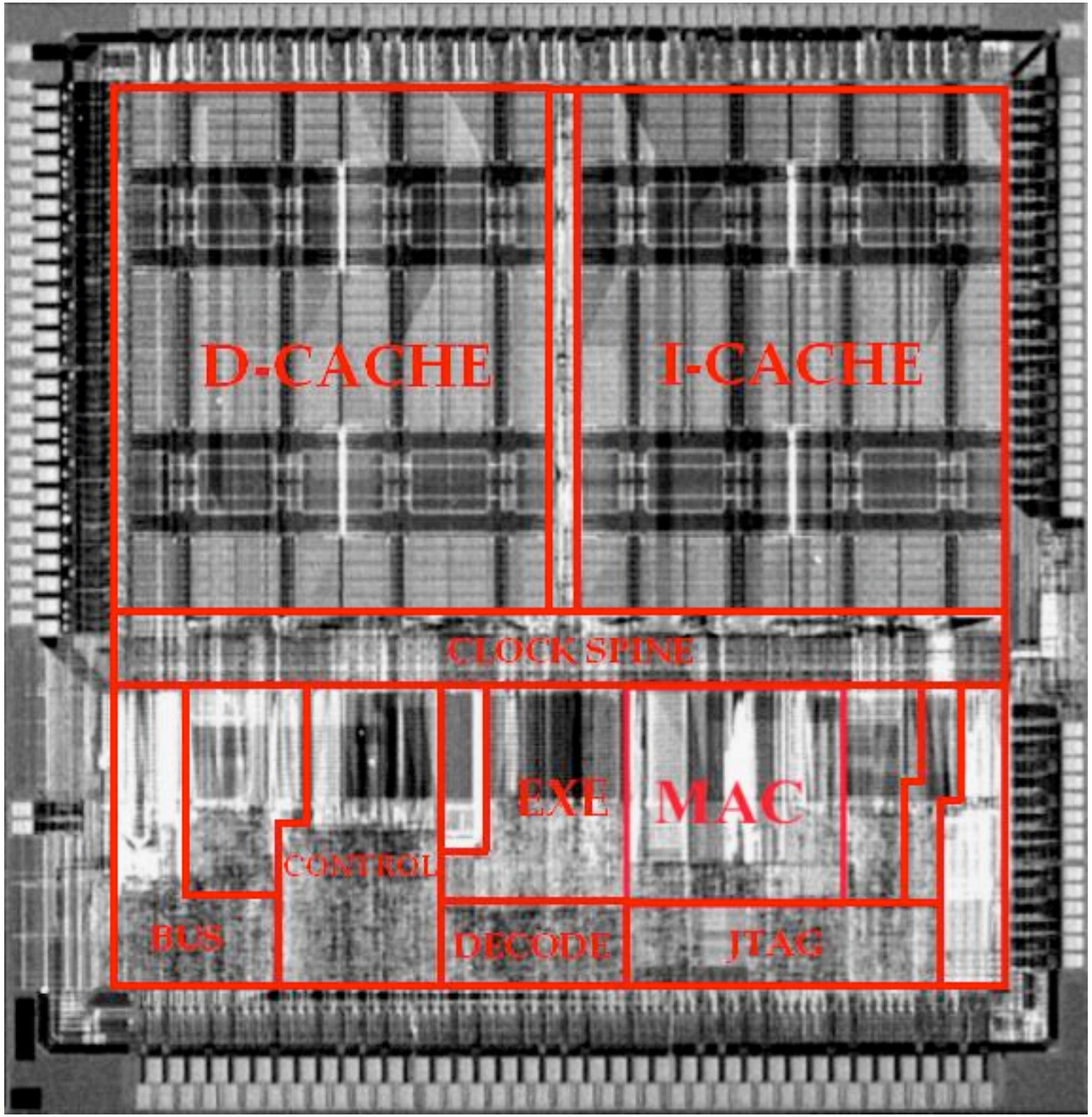
UC Regents Fall 2006 © UCB

Administrivia: Upcoming deadlines ...

- **Friday:** "Teams meet the TAs", 12-2 and 3-5, 125 Cory.
  (1) Decide on group names (2) Collect your NT usernames -- bring your account sheet!
- **Tuesday:** Lab 1 final report due, 11:59 PM, via the submit program.
  I will be around all weekend long -- email (lazzaro@cs) or phone (643-4005) for Cory access.
  Check: Accounts OK? Cardkey woes?
- **Thursday 9/7:** Lab 2 preliminary design document due to TAs via email, 11:59 PM.

UC Regents Fall 2006 © UCB

CS 152: Real hardware, not simulation



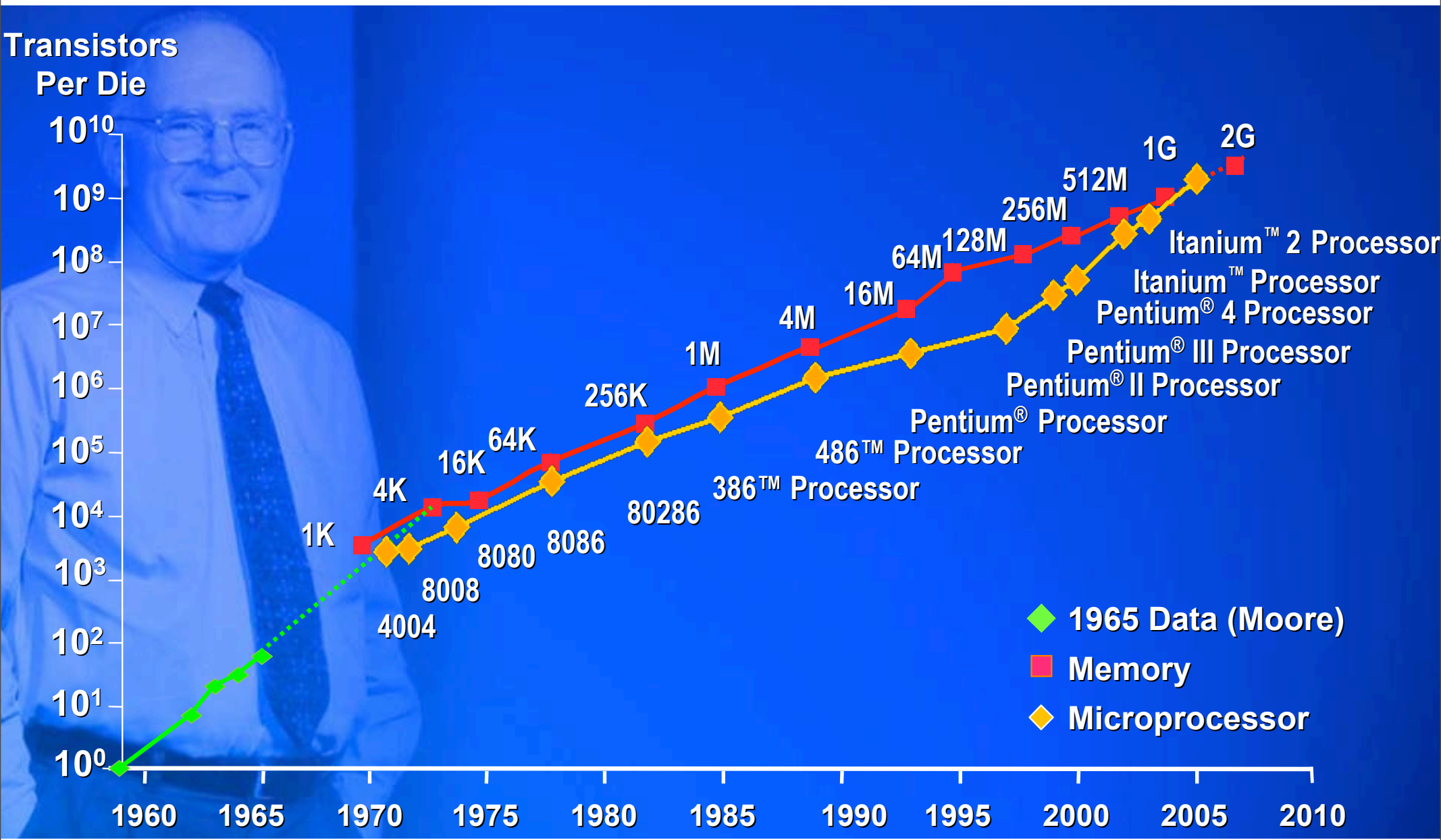
Intel XScale 80200: used in earlier HP PocketPCs

Will we be fabricate CPU dies?

Back when I was taking classes (1984 @ Caltech) our project course did fab chips.

UC Regents Fall 2006 © UCB

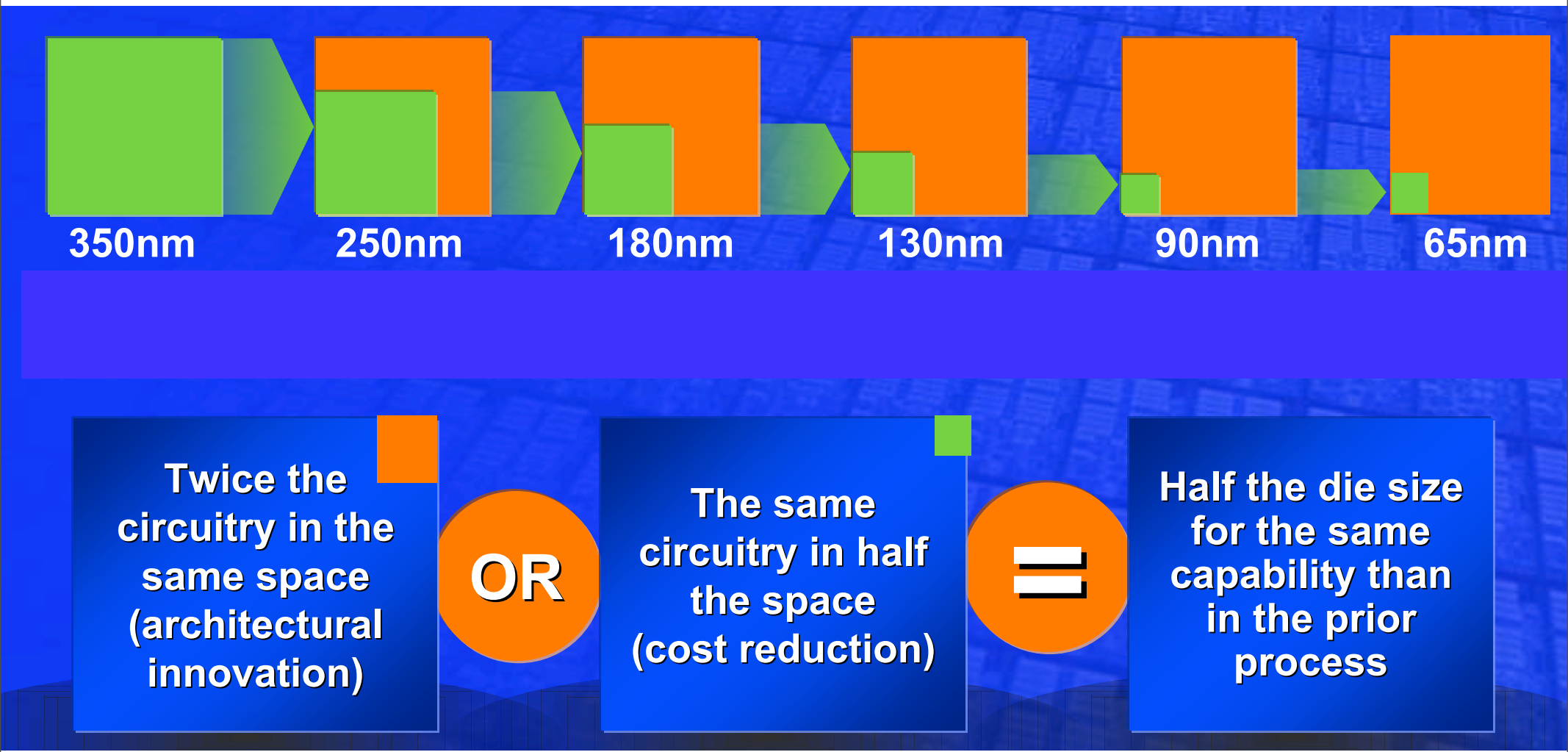
Moore's Law for CPUs and DRAMs



From: "Facing the Hot Chips Challenge Again", Bill Holt, Intel, presented at Hot Chips 17, 2005.

UC Regents Fall 2006 © UCB

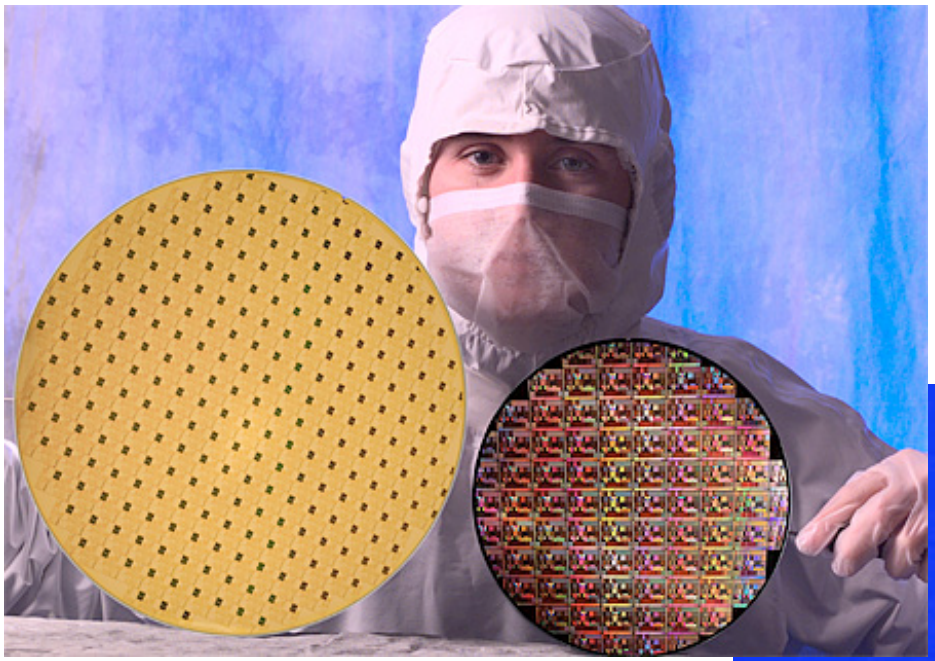
Main driver: device scaling ...

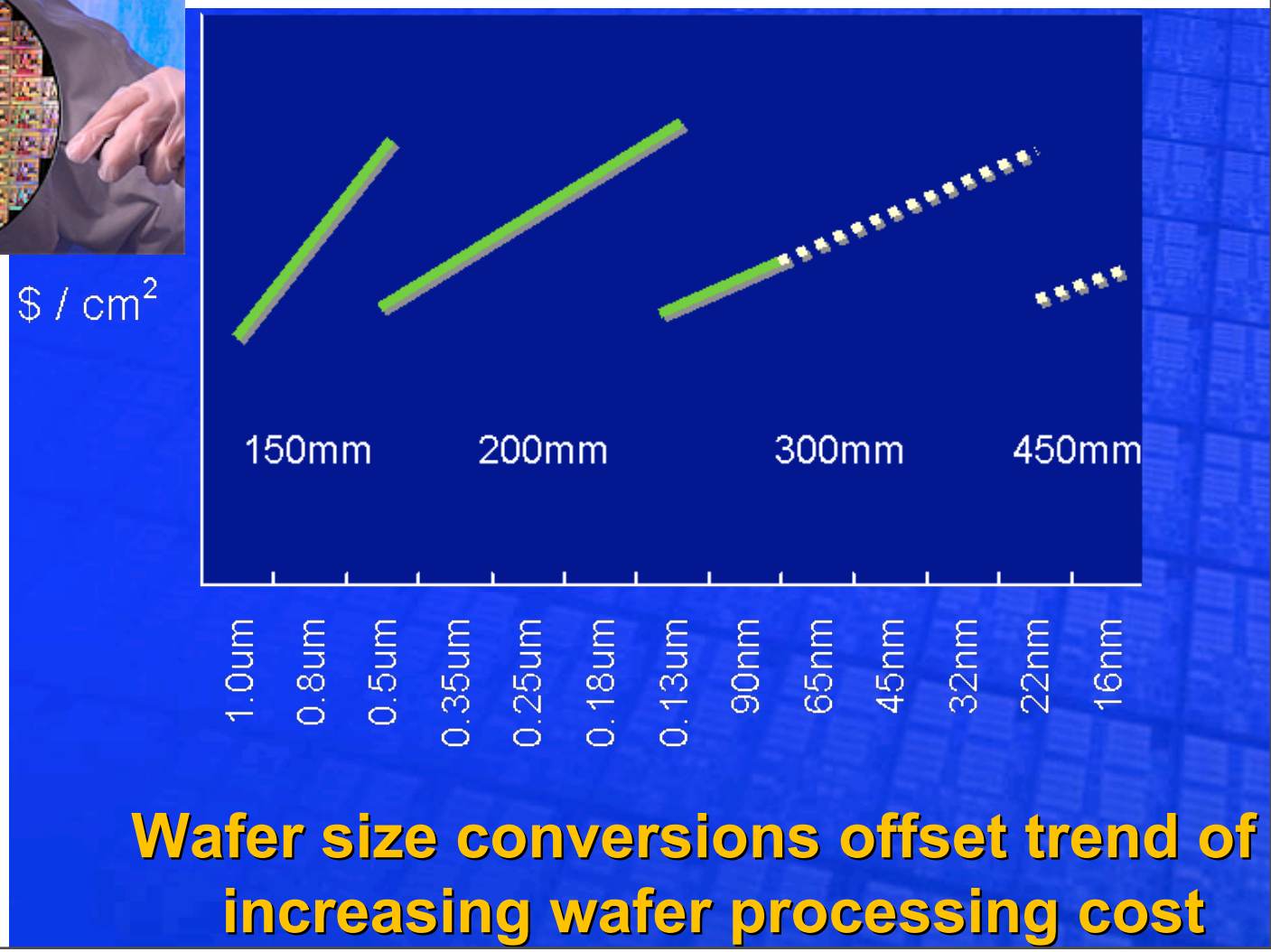


From: “Facing the Hot Chips Challenge Again”, Bill Holt, Intel, presented at Hot Chips 17, 2005.

UC Regents Fall 2006 © UCB

Secondary driver: Wafer size

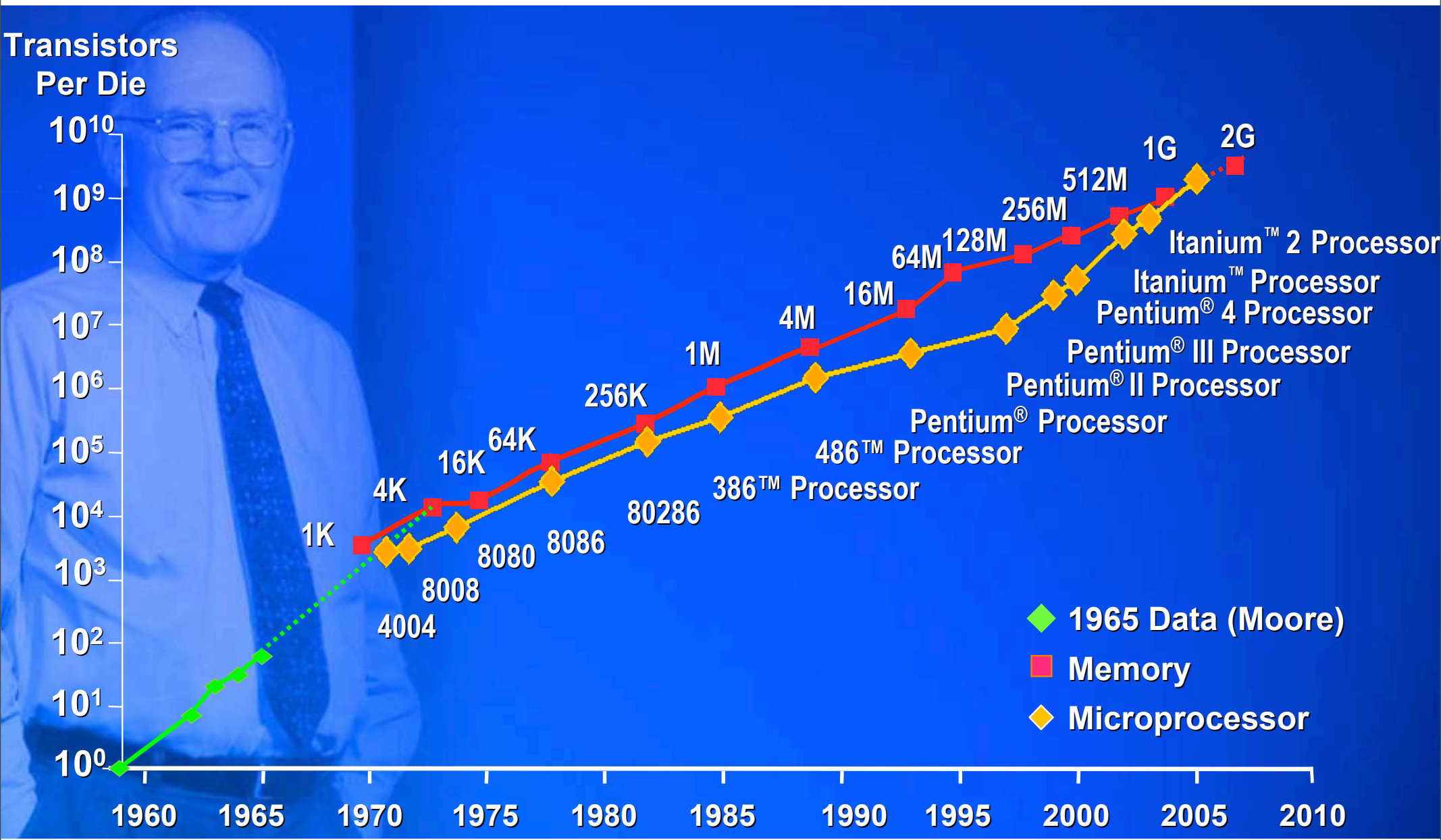




From: "Facing the Hot Chips Challenge Again", Bill Holt, Intel, presented at Hot Chips 17, 2005.

Wafer size conversions offset trend of increasing wafer processing cost

Thus, cost per transistor plummets ...



From: "Facing the Hot Chips Challenge Again", Bill Holt, Intel, presented at Hot Chips 17, 2005.

UC Regents Fall 2006 © UCB

Lab 2: A Single-Cycle MIPS CPU

CS152 Computer Architecture and Engineering

Lab #2: Single Cycle Processor

John Lazzaro

Fall 2006

In Lab 2, your group will build a single-cycle processor, like the one described in Chapter 5 of COD. You will create the design in Verilog. The processor will run in simulation (ModelSim) and on real hardware (Xilinx). You will create a test plan for your processor. You will use the test plan for bug diagnosis, and to verify that your processor executes a subset of the MIPS instruction set.

UC Regents Fall 2006 © UCB

Today: Single Cycle Datapath Design

The book presentation of single cycle processors is sufficient to do Lab 2.

Th 9/1	Single-Cycle Datapaths	PDF	5.1-4, 5.8, Appendix B

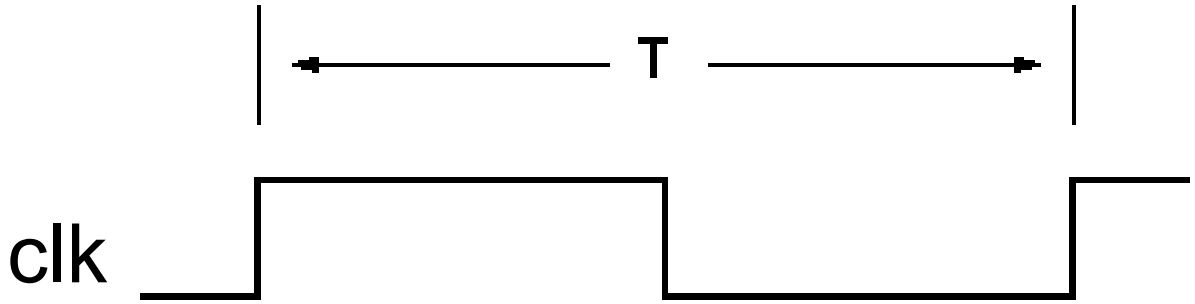
This lecture is not.

This lecture is a gentle introduction, to prepare you to read the book ...

UC Regents Fall 2006 © UCB

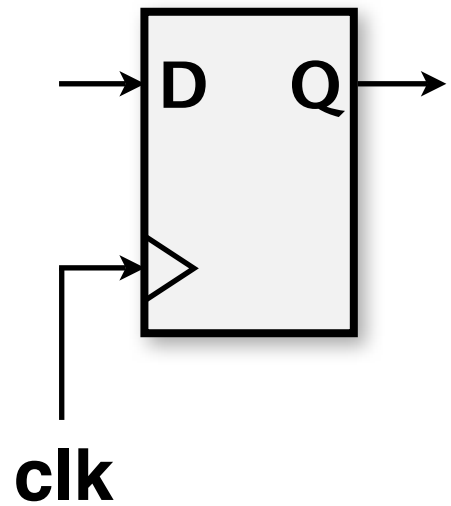
Single cycle data paths: Assumptions

Processor uses synchronous logic design (a "clock").



f	T
1 MHz	1 µs
10 MHz	100 ns
100 MHz	10 ns
1 GHz	1 ns

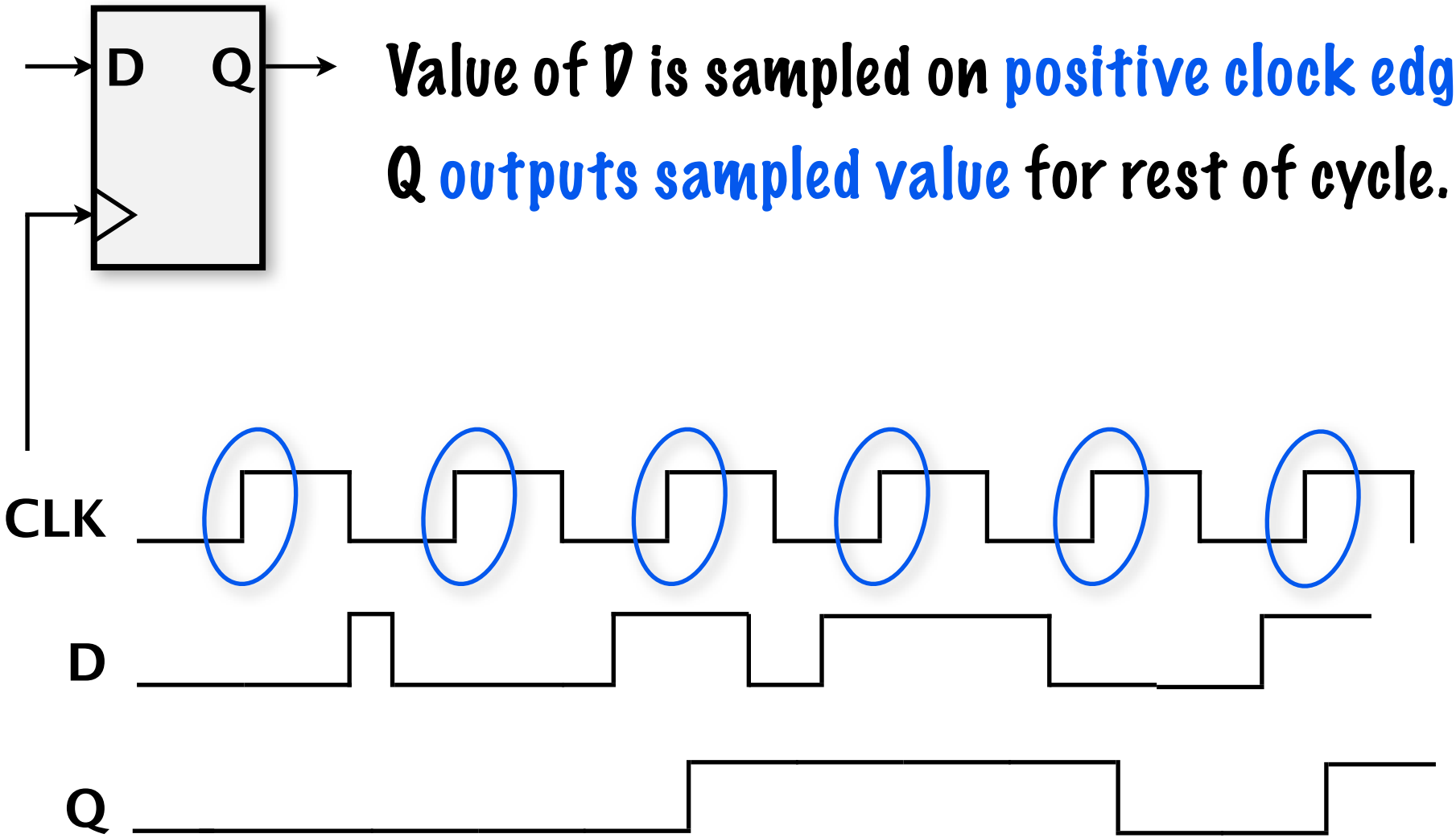
All state elements act like positive edge-triggered flip flops.



Reset ?

UC Regents Fall 2006 © UCB

Review: Edge-Triggered D Flip Flops

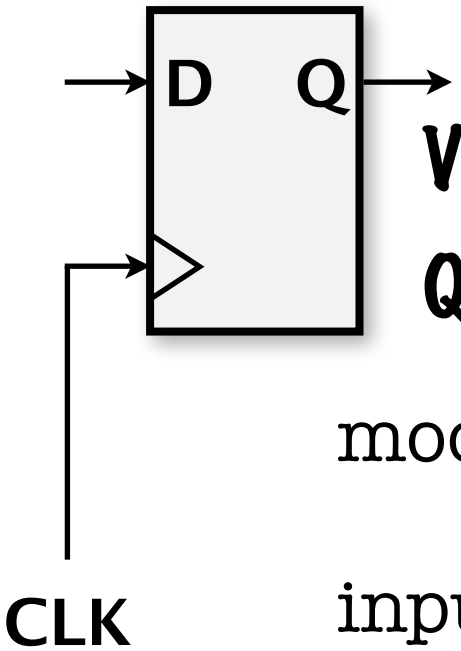


Value of D is sampled on positive clock edge.

Q outputs sampled value for rest of cycle.

UC Regents Fall 2006 © UCB

Review: Edge-Triggering in Verilog



Value of D is sampled on positive clock edge.

Q outputs sampled value for rest of cycle.

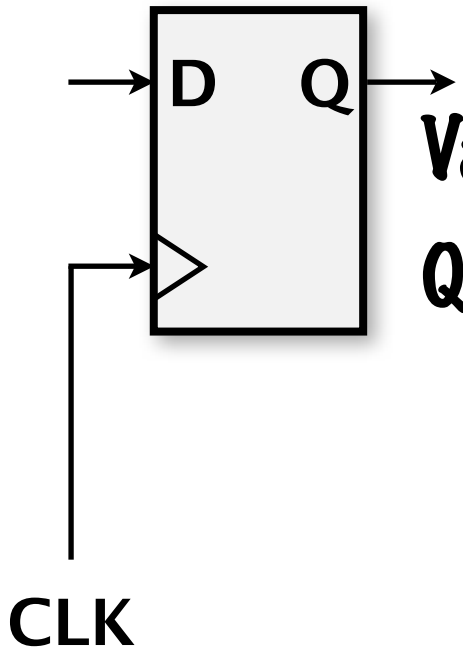
```
module ff(D, Q, CLK);

input D, CLK;
output Q;

always @ (CLK)
   Q  <= D;

endmodule
```

Module code has two bugs.

Where?

UC Regents Fall 2006 © UCB

Review: Edge-Triggered D Flip Flops



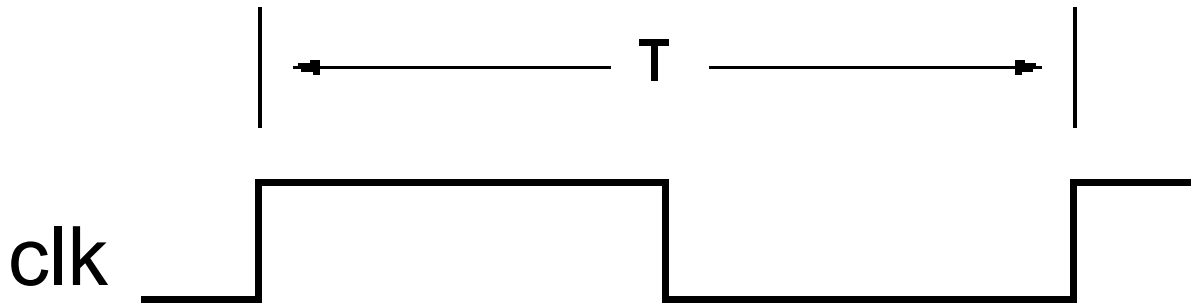
Value of D is sampled on positive clock edge.

Q outputs sampled value for rest of cycle.

```
module ff(D, Q, CLK);

input D, CLK;
output Q;
reg Q;

always @ (posedge CLK)
   Q  <= D;

endmodule
```


UC Regents Fall 2006 © UCB

define: Single-cycle datapath



All instructions execute in a single cycle of the clock (positive edge to positive edge)

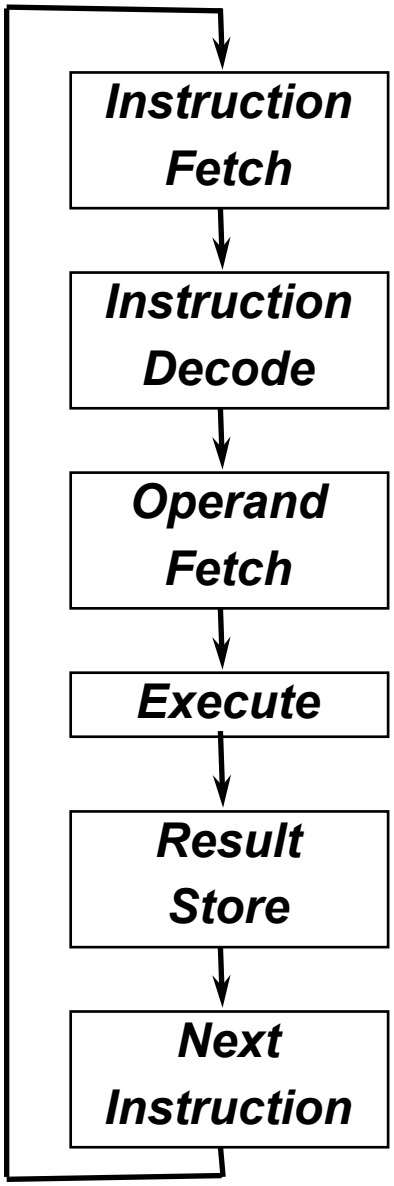
Advantage: a great way to learn CPUs.

Drawbacks: unrealistic hardware assumptions, slow clock period

UC Regents Fall 2006 © UCB

Recall: MIPS R-format instructions

Syntax: ADD $8 $9 $10 **Semantics:** $8 = $9 + $10



Fetch next inst from memory: 012A4020

opcode	rs	rt	rd	shamt	funct

Decode fields to get : ADD $8 $9 $10

"Retrieve" register values: $9 $10

Add $9 **to** $10

Place this sum in $8

Prepare to fetch instruction that follows the ADD in the program.

UC Regents Fall 2006 © UCB

Goal #1: An R-format single-cycle CPU

Syntax: `ADD $8 $9 $10` **Semantics:** `$8 = $9 + $10`

opcode	rs	rt	rd	shamt	funct

Sample program:

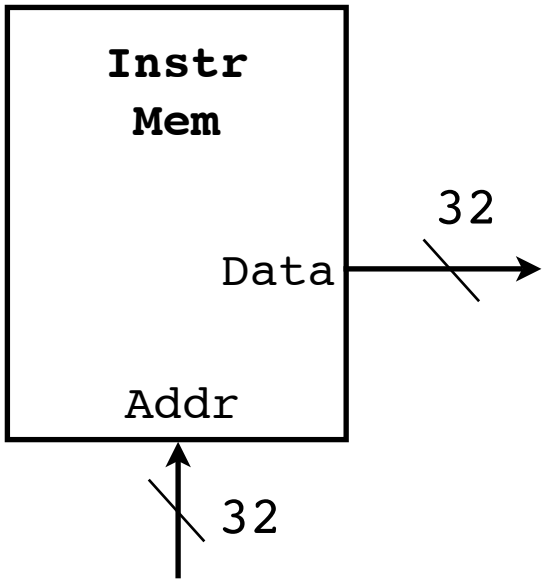
```
ADD $8 $9 $10
SUB $4 $8 $3
AND $9 $8 $4
...
```

How registers get their initial values are not of concern to us right now.

- No branches or jumps: machine only runs straight line code.
- No loads or stores: machine has no use for data memory, only instruction memory.

UC Regents Fall 2006 © UCB

Separate Read-Only Instruction Memory



Reads are combinational: Put a stable address on input, a short time later data appears on output.

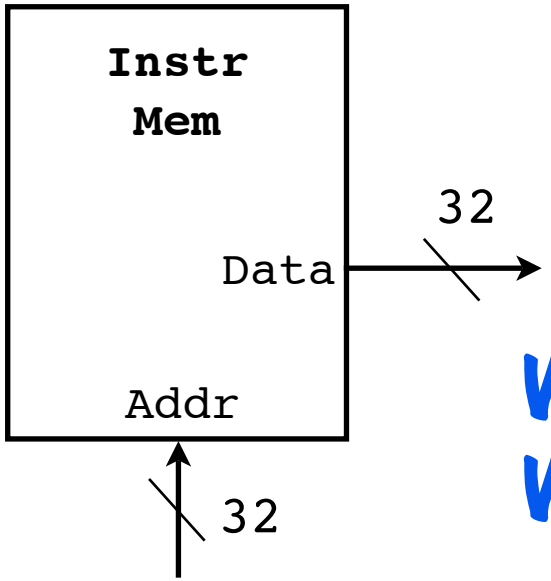
Not concerned about how programs are loaded into this memory.

Related to separate instruction and data caches in "real" designs.

UC Regents Fall 2006 © UCB

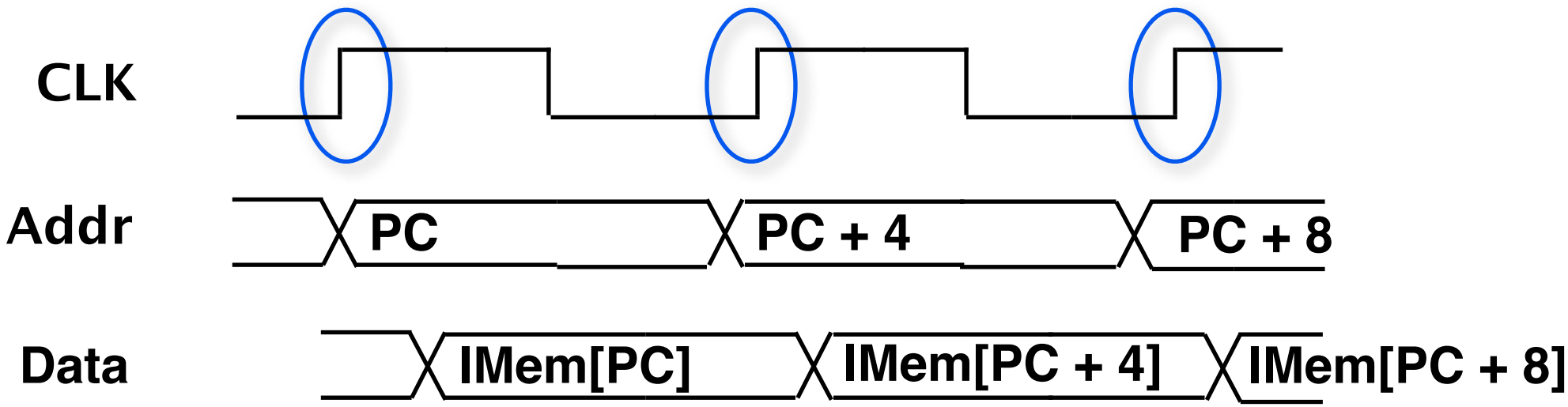
Task #1: Straight-line Instruction Fetch

"Requirements"



Fetching straight-line MIPS instructions requires a machine that generates this timing diagram:

Why do we increment every clock cycle?
Why +4 and not +1?

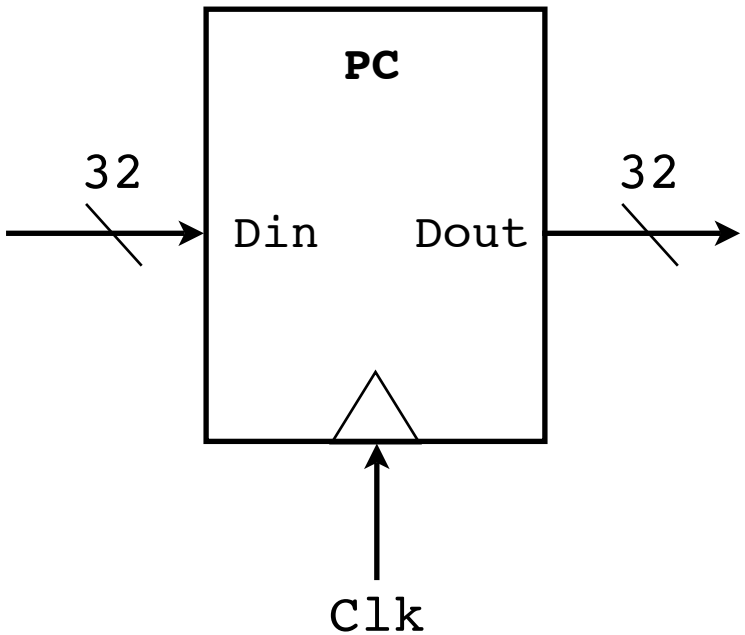


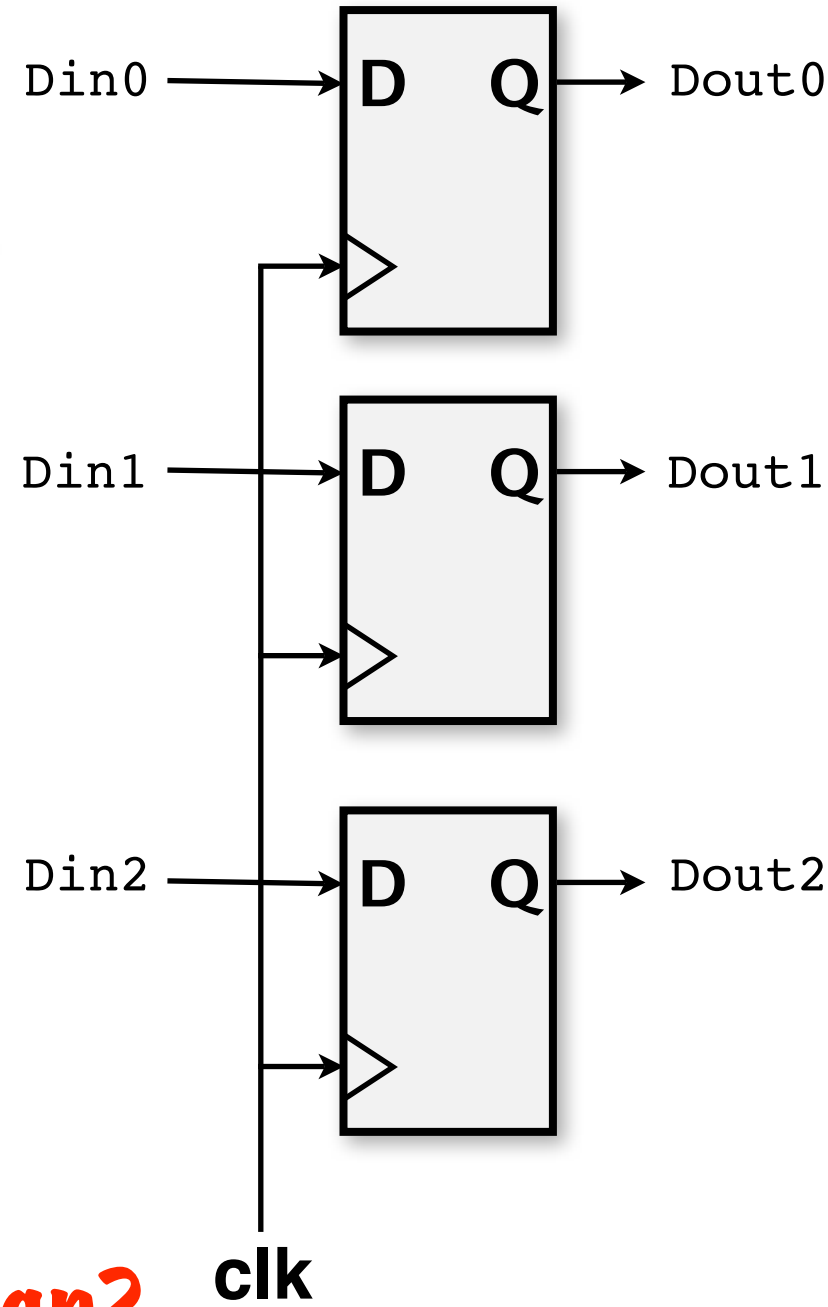
PC == Program Counter, points to next instruction.

UC Regents Fall 2006 © UCB

New Component: Register (for PC)

Built out of an array of flip-flops



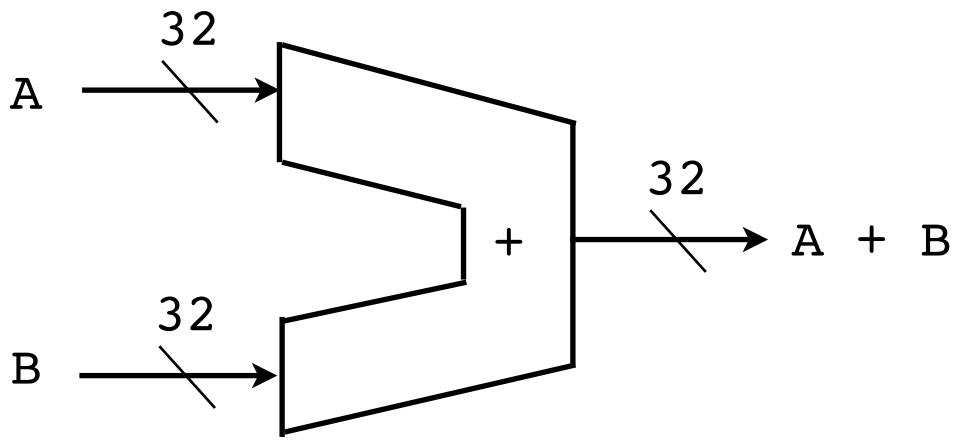


In later examples, we will add an "enable" input: clock edge updates state only if enable is high.

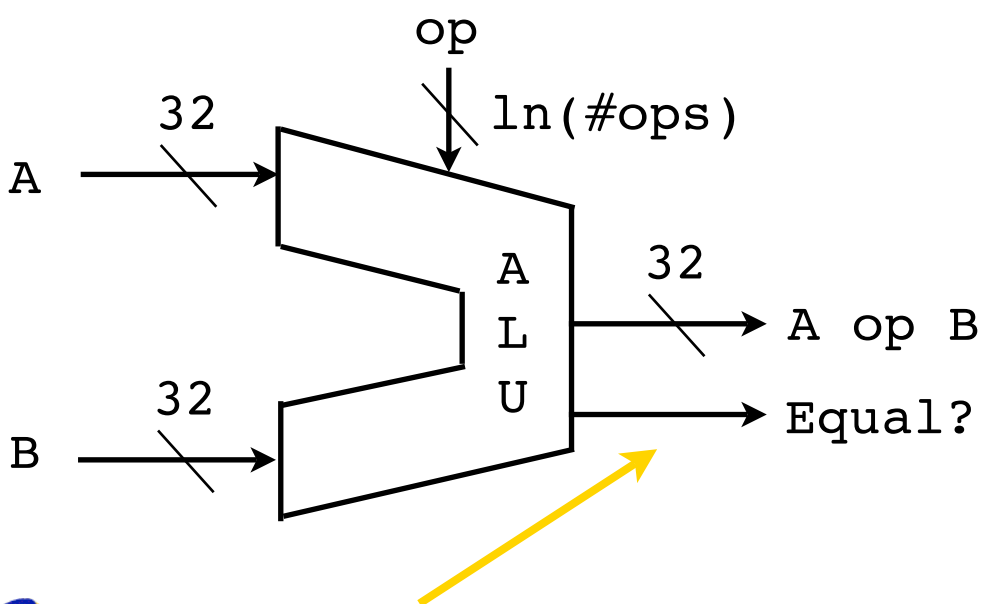
Logic design?

UC Regents Fall 2006 © UCB

New Component: A 32-bit adder (ALU)



Combinational: Put a A and B values on inputs, a short time later A + B appears on output.



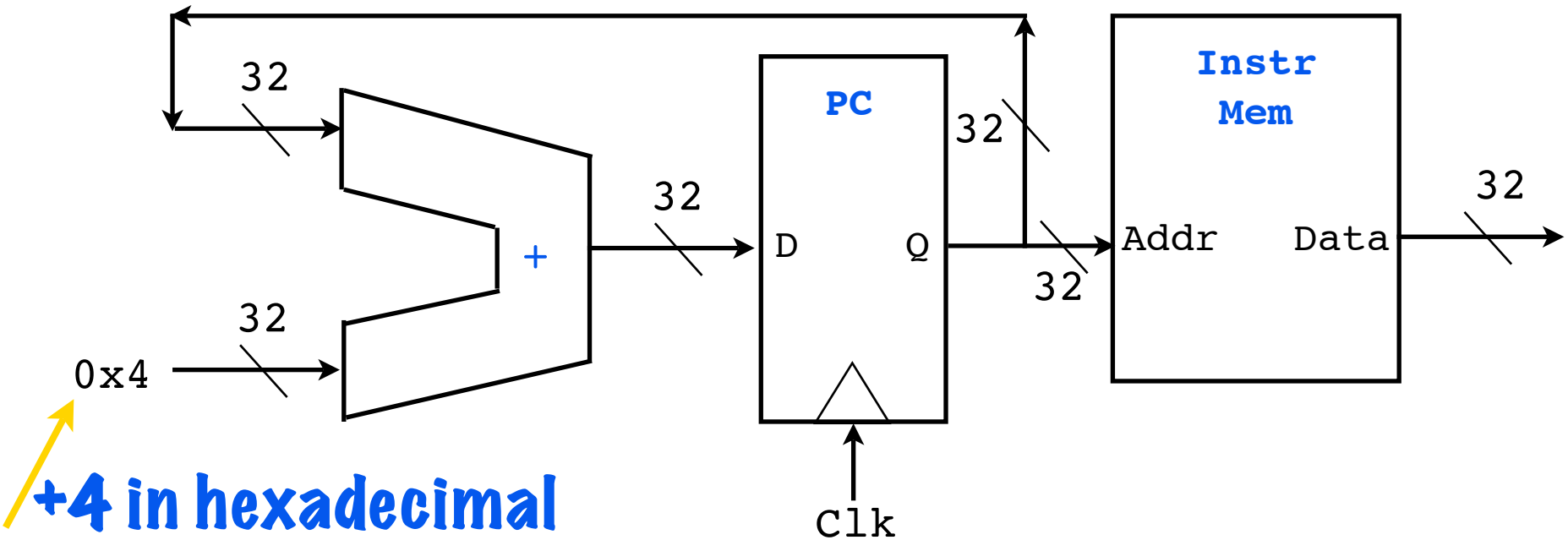
ALU: Combinational part that is able to execute many functions of A and B (add, sub, and, or, ...). The "op" value selects the function.

Sometimes, extra outputs for use by control logic ...

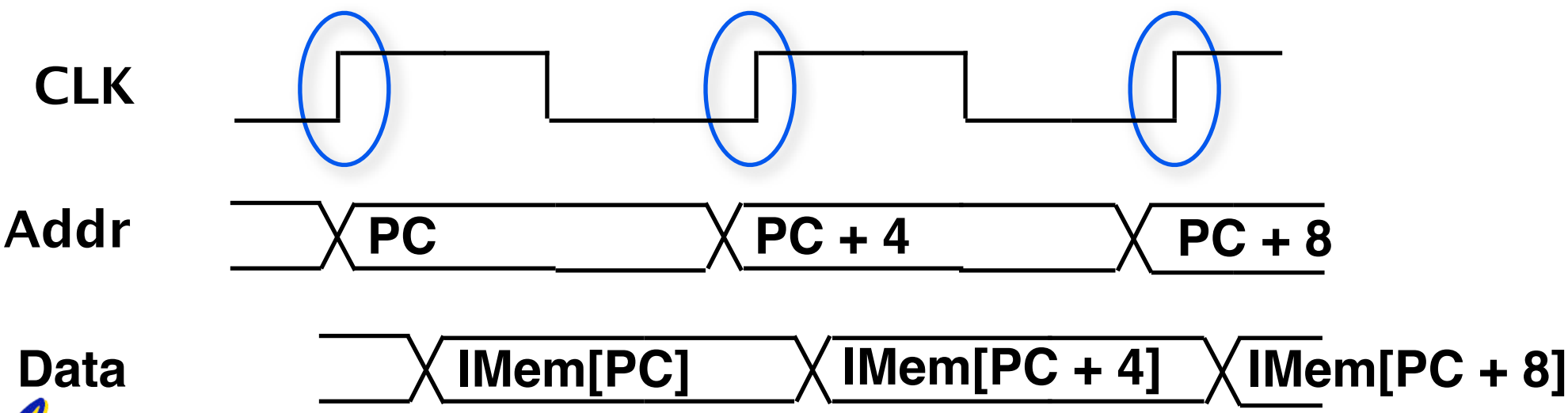
UC Regents Fall 2006 © UCB

Design: Straight-line Instruction Fetch

CS 152: State machine design in the service of an ISA



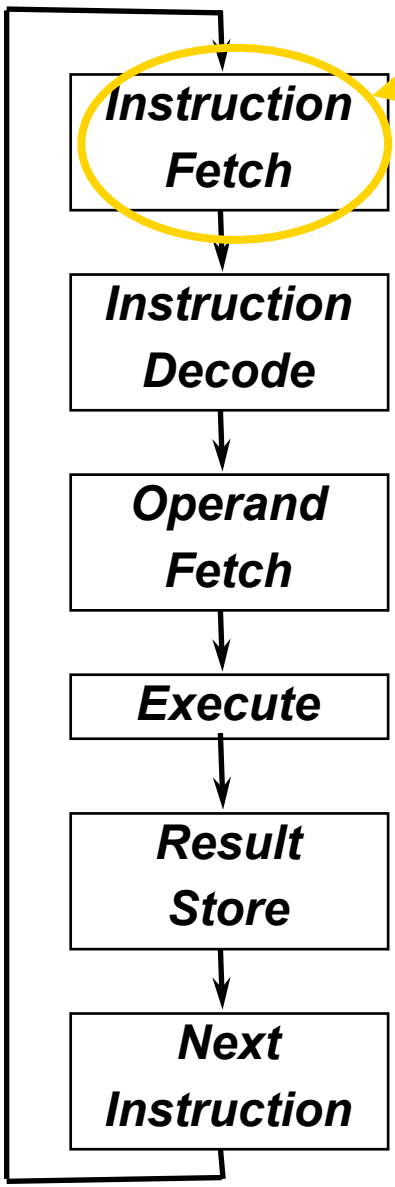
+4 in hexadecimal



UC Regents Fall 2006 © UCB

Goal #1: An R-format single-cycle CPU

Syntax: ADD $8 $9 $10 **Semantics:** $8 = $9 + $10



Done! To continue, we need registers ...

Fetch next inst from memory: 012A4020

opcode	rs	rt	rd	shamt	funct

Decode fields to get : ADD $8 $9 $10

"Retrieve" register values: $9 $10

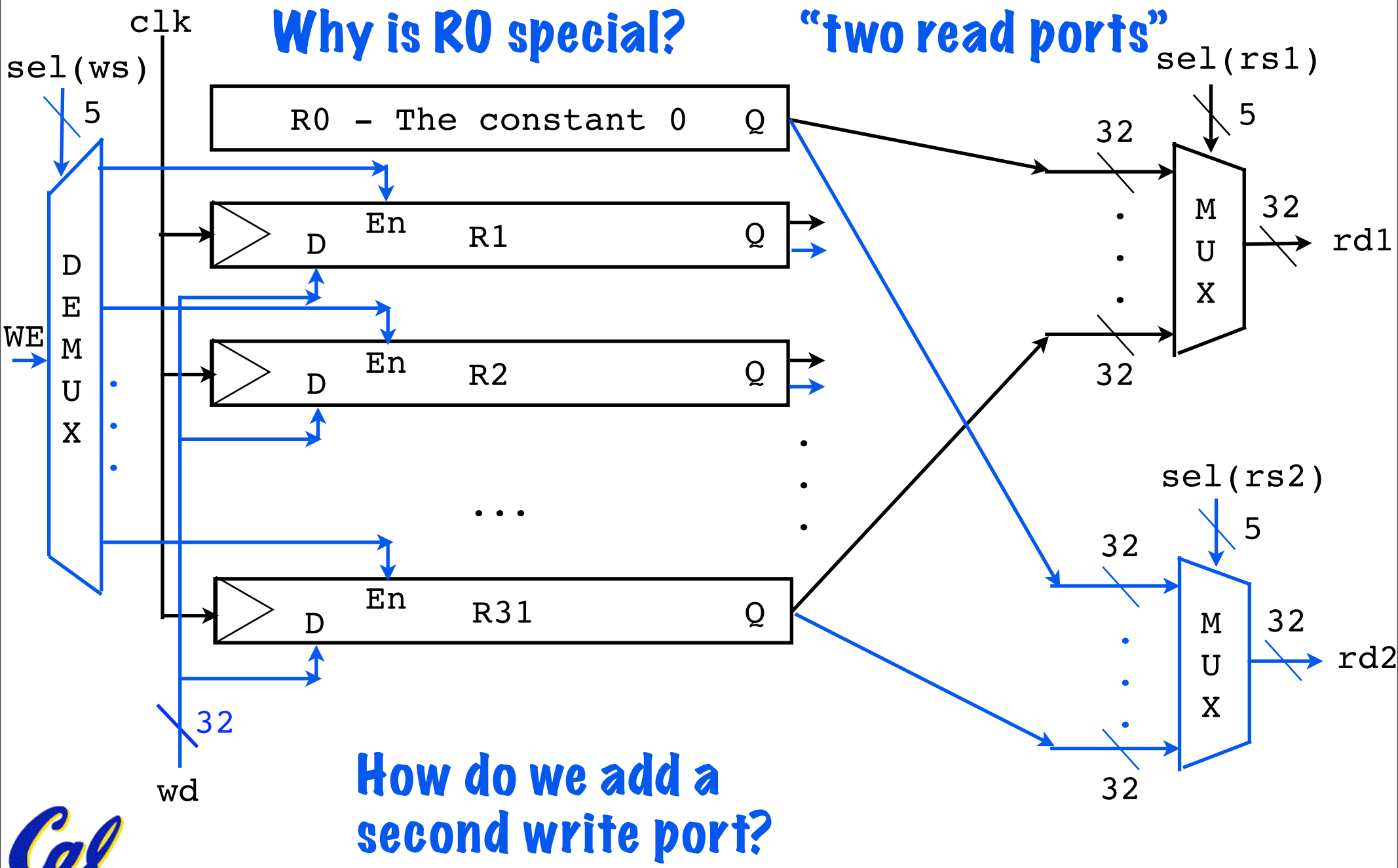
Add $9 **to** $10

Place this sum in $8

Prepare to fetch instruction that follows the ADD in the program.

UC Regents Fall 2006 © UCB

MIPS Register file: From the top down

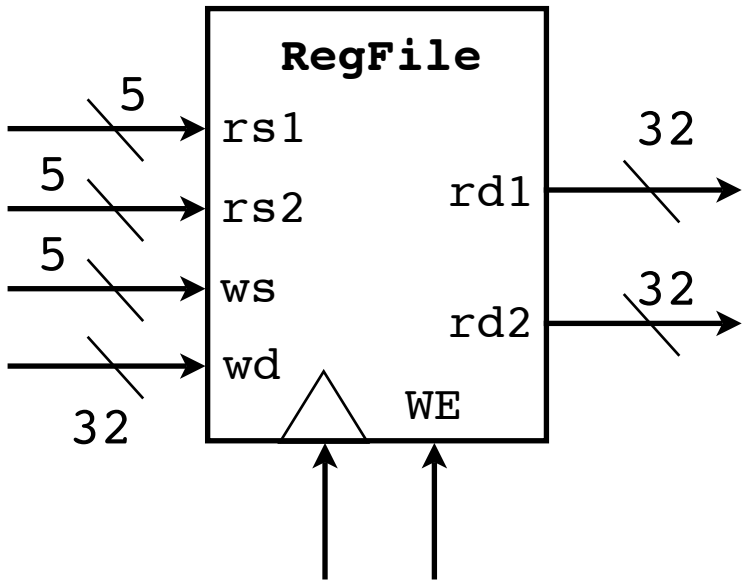


UC Regents Fall 2006 © UCB

Register File Schematic Symbol

Why do we need WE?

If we had a MIPS register file w/o WE, how could we work around it?

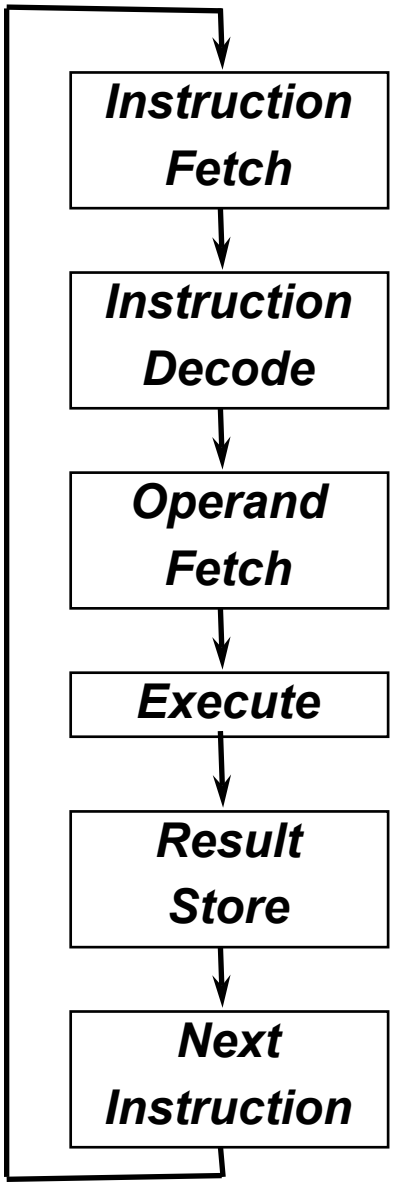


UC Regents Fall 2006 © UCB

Goal #1: An R-format single-cycle CPU

Syntax: ADD $8 $9 $10 **Semantics:** $8 = $9 + $10

What do we do with these?



Fetch next inst from memory: 012A4020

opcode	rs	rt	rd	shamt	funct

Decode fields to get : ADD $8 $9 $10

"Retrieve" register values: $9 $10

Add $9 **to** $10

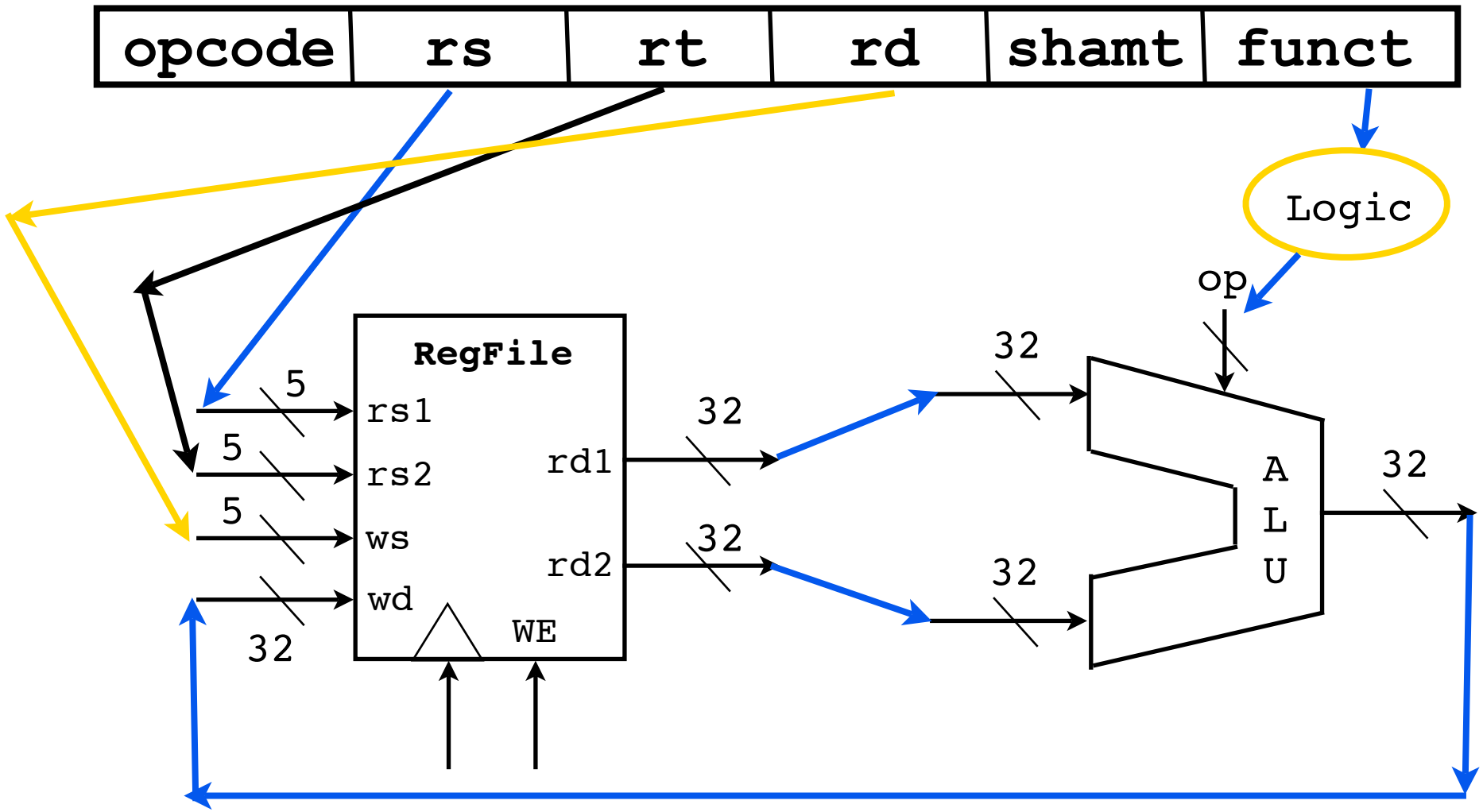
Place this sum in $8

Prepare to fetch instruction that follows the ADD in the program.

UC Regents Fall 2006 © UCB

Computing engine of the R-format CPU

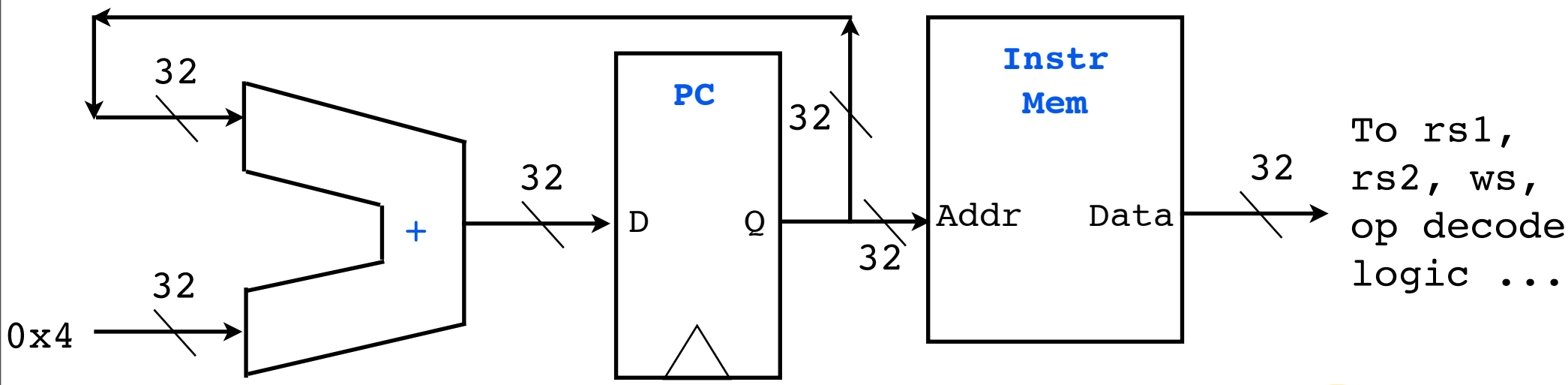
Decode fields to get : ADD $8 $9 $10



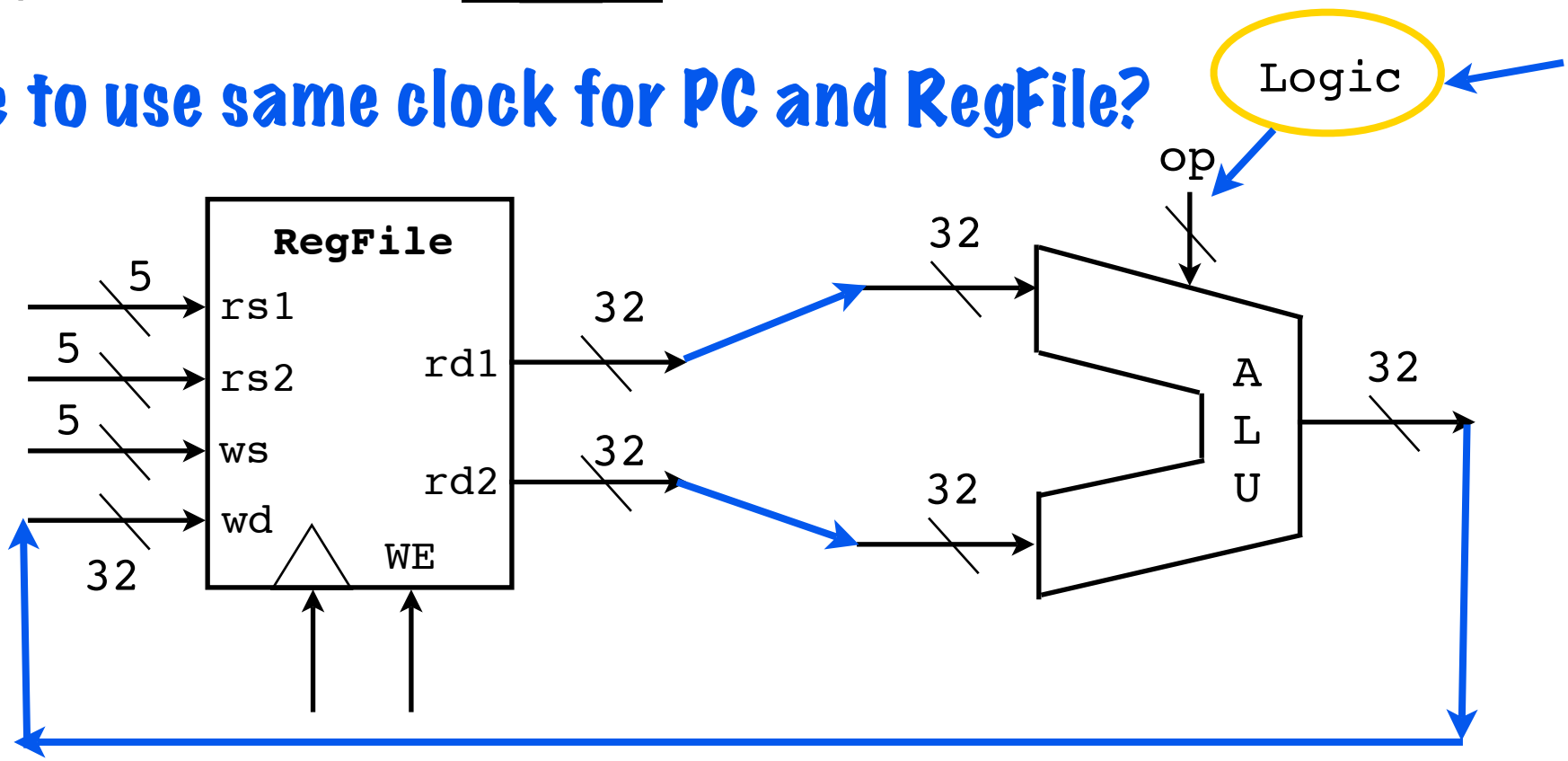
What do we do with WE?

UC Regents Fall 2006 © UCB

Putting it all together ...

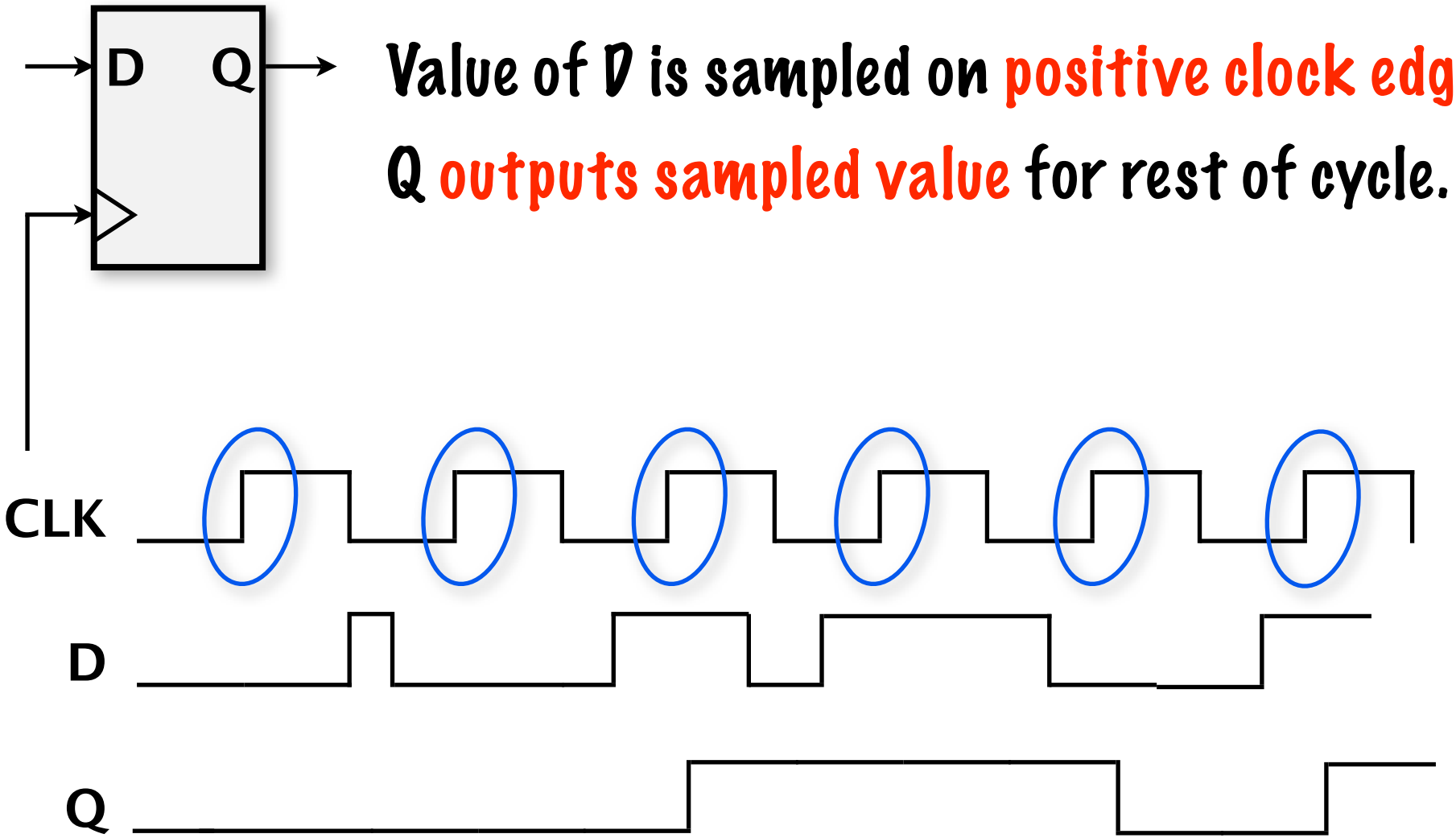


Is it safe to use same clock for PC and RegFile?



UC Regents Fall 2006 © UCB

Recall: Our ideal-world D Flip-Flop



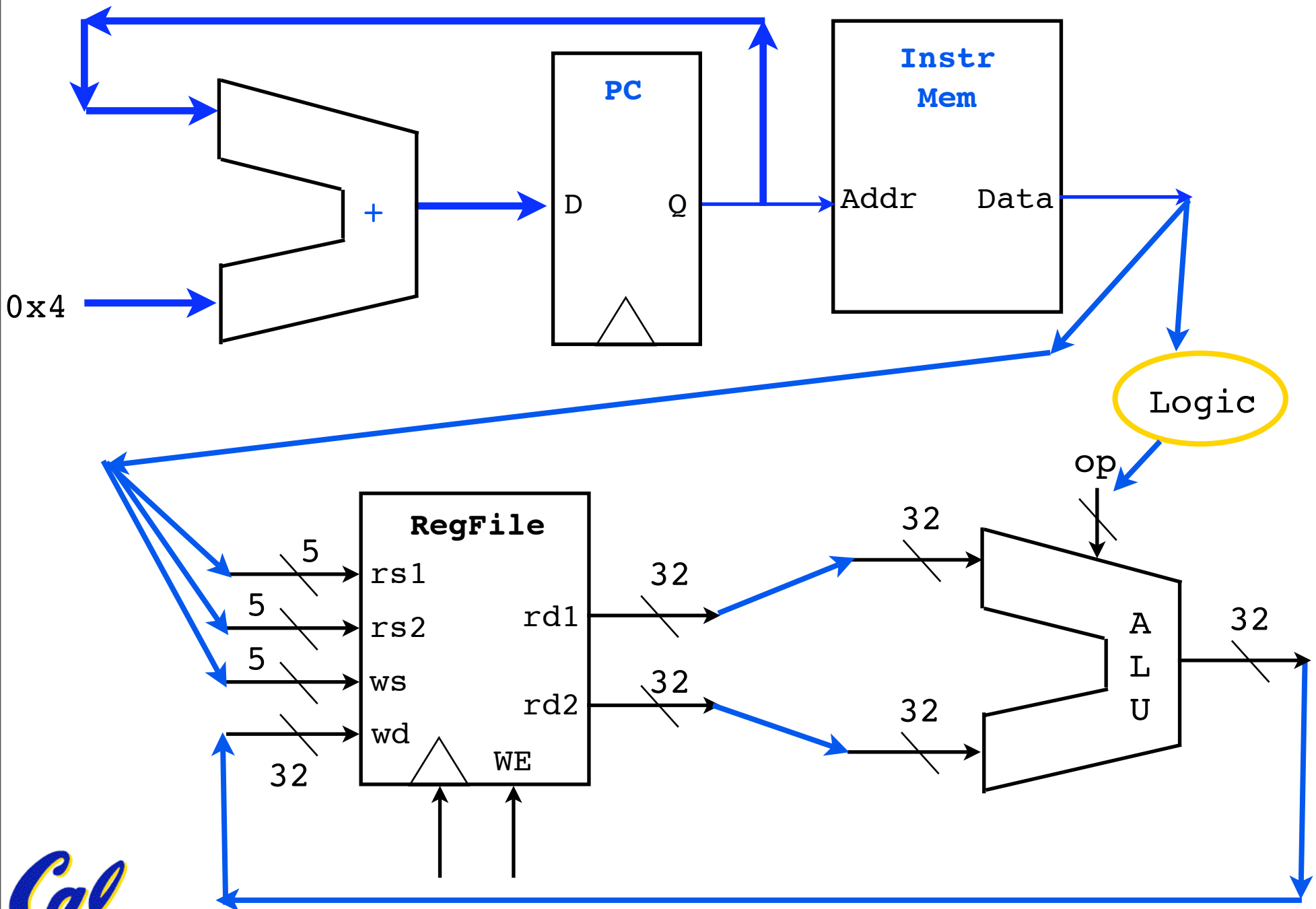
Value of D is sampled on positive clock edge.

Q outputs sampled value for rest of cycle.

Also assume: clocks arrive at all flip flops simultaneously.

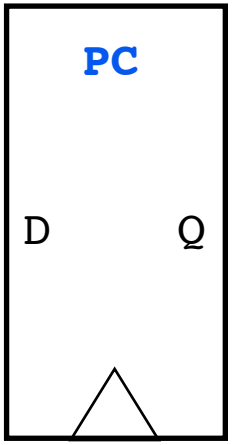
UC Regents Fall 2006 © UCB

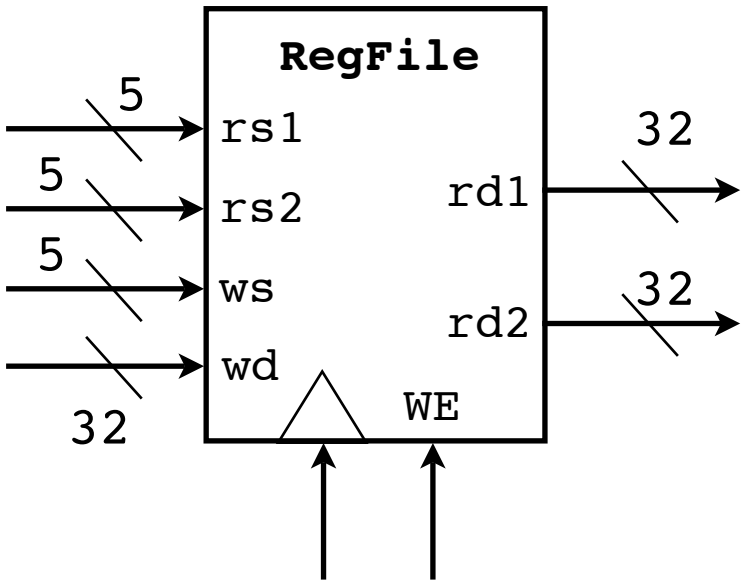
Reminder: How data flows after posedge



UC Regents Fall 2006 © UCB

Next posedge: Update state and repeat





In this ideal world, as long as the clock is slow enough, the machine gets the right answer.

In Timing lecture, we look at the assumptions behind ideality.

UC Regents Fall 2006 © UCB

Next Step ...

- Design stand-alone machines for other major classes of instructions: immediates, branches, load/store.

- Learn how to efficiently "merge" single-function machines to make one general-purpose machine.

UC Regents Fall 2006 © UCB

Goal #2: add I-format ALU instructions

Syntax: `ORI $8 $9 64` **Semantics:** `$8 = $9 | 64`

op	rs	rt	immediate

- In this example, $9 is rs and $8 is rt.
- 16-bit immediate extended to 32 bits.

Zero-extend: `0x8000` ⇨ `0x00008000`

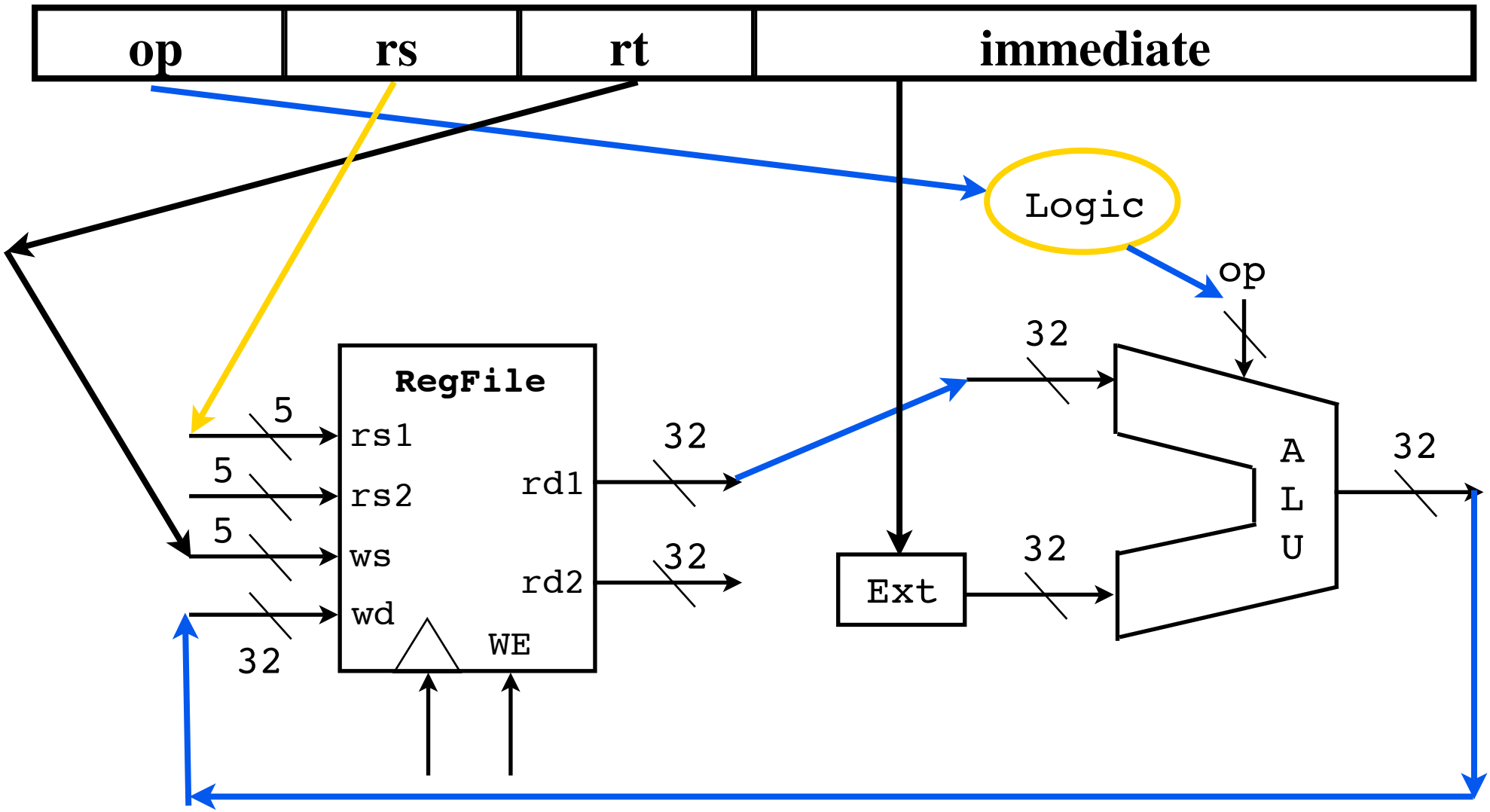
Sign-extend: `0x8000` ⇨ `0xFFFF8000`

Some MIPS instructions zero-extend immediate field, other instructions sign-extend.

UC Regents Fall 2006 © UCB

Computing engine of the I-format CPU

Decode fields to get : ORI $8 $9 64

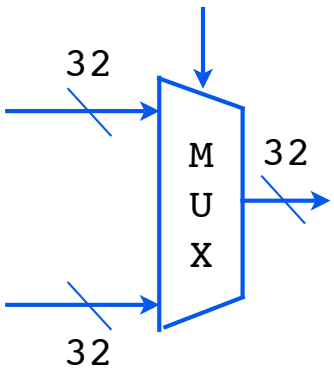


In a Verilog implementation, what should we do with rs2?

UC Regents Fall 2006 © UCB

Merging data paths ...

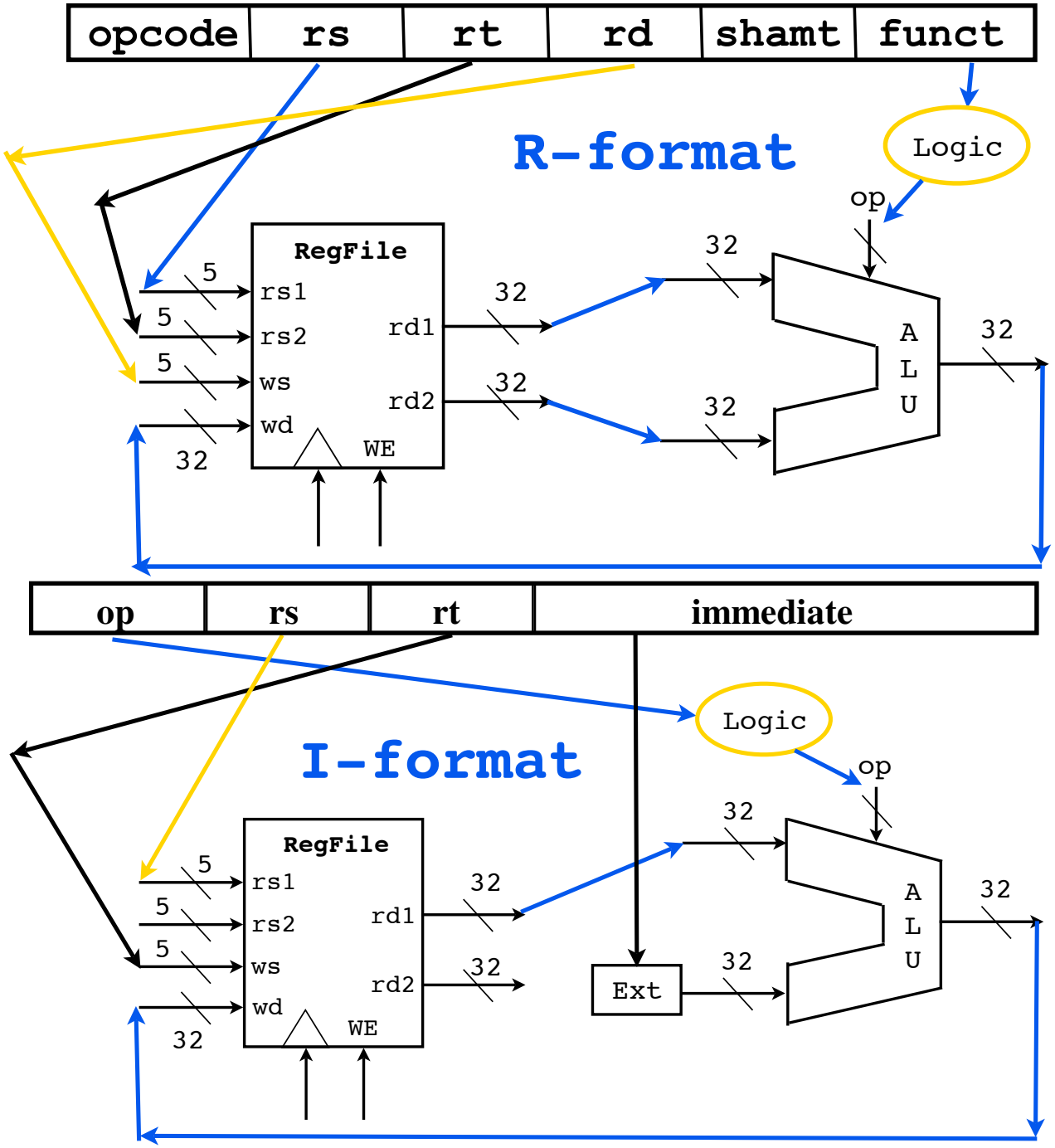
Add muxes



How many ?

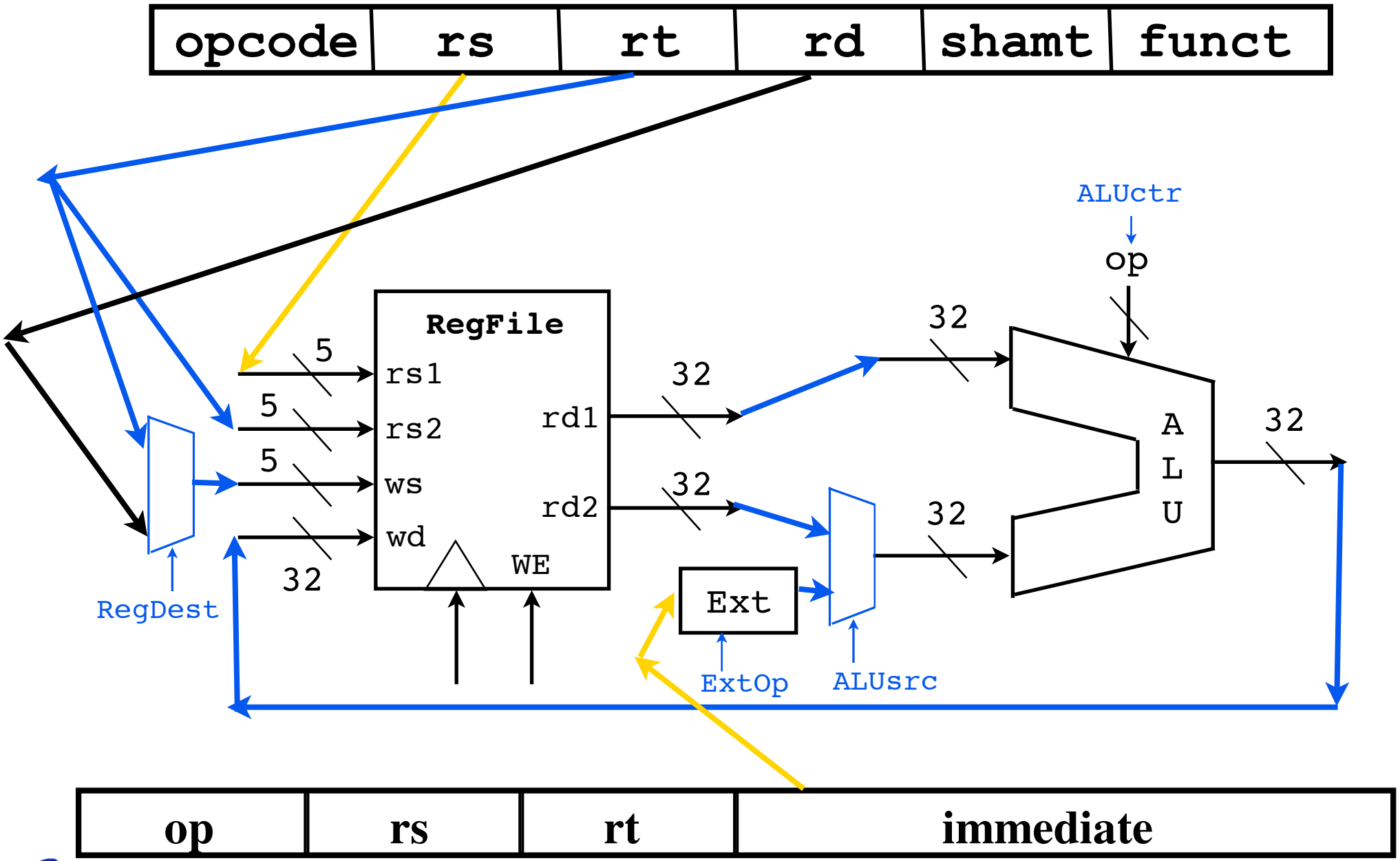
(ignore ALU control)

Where ?



UC Regents Fall 2006 © UCB

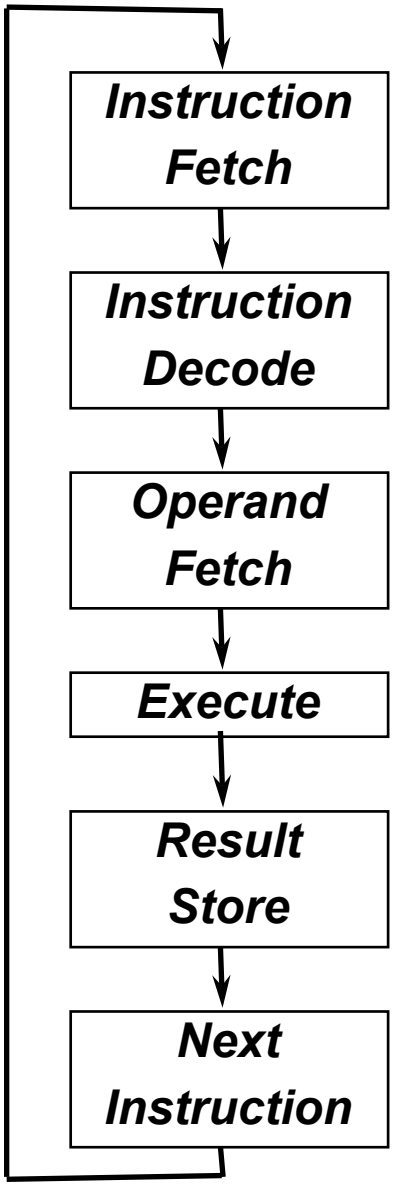
The merged data path ...



If you watched it being designed, it's understandable ...

UC Regents Fall 2006 © UCB

Memory Instructions: `LW $1,32($2)`



For next class ...

Fetch the load inst from memory

opcode	rs	rt	offset

"I-Format"

Decode fields to get : LW $1, 32($2)

"Retrieve" register value: $2

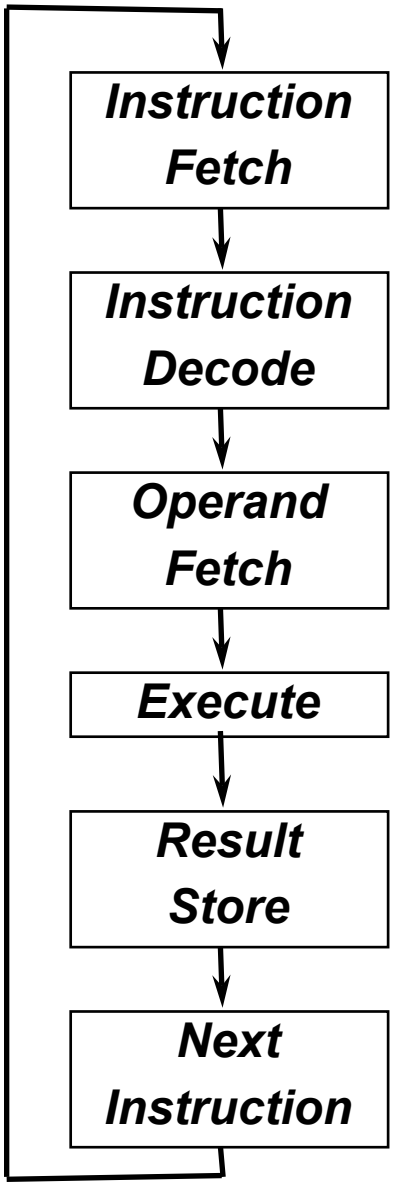
Compute memory address: 32 + $2

Load memory address contents into: $1

Prepare to fetch instr that follows the LW in the program. Depending on load semantics, new $1 is visible to that instr, or not until the following instr ("delayed loads").

UC Regents Fall 2006 © UCB

Branch Instructions: BEQ $1,$2,25



Also for next class ...

Fetch branch inst from memory

opcode	rs	rt	offset

"I-Format"

Decode fields to get: BEQ $1,$2, 25

"Retrieve" register values: $1, $2

Compute if we take branch: $1 == $2 ?

ALWAYS prepare to fetch instr that follows the BEQ in the program ("delayed branch"). IF we take branch, the instr we fetch AFTER that instruction is PC + 4 + 100.

PC == "Program Counter"

UC Regents Fall 2006 © UCB

Office Hours, Mid-terms ...

- **Udam:** TBA
  Jue: TBA
  John: TBA

- **Mid-term 1:** Tuesday October 3rd, 6:00 to 9:00 PM, TBA.

- **Mid-term 2:** Tuesday December 5th, 6:00 to 9:00 PM, TBA.

Last call for schedule conflicts ...

UC Regents Fall 2006 © UCB

Administrivia: Upcoming deadlines ...

- **Friday:** "Teams meet the TAs", 12-2 and 3-5, 125 Cory.
  (1) Decide on group names (2) Collect your NT usernames -- bring your account sheet!
- **Tuesday:** Lab 1 final report due, 11:59 PM, via the submit program.
  I will be around all weekend long -- email (lazzaro@cs) or phone (643-4005) for Cory access.
  Check: Accounts OK? Cardkey woes?
- **Thursday 9/7:** Lab 2 preliminary design document due to TAs via email, 11:59 PM.

UC Regents Fall 2006 © UCB