

# Processor

**Prof. Hakim Weatherspoon**

**CS 3410, Spring 2015**

Computer Science

Cornell University

See P&H Chapter: 4.1-4.4, 1.6, Appendix B

# Announcements

Project Partner finding assignment on CMS

No official office hours over break

Lab1 due tomorrow

HW1 Help Sessions Wed, Feb 18 and Sun, Feb 21

# Announcements

Make sure to go to your Lab Section this week

Lab2 due in class this week (it is *not* homework)

Lab1: Completed Lab1 due *tomorrow* Friday, Feb 13<sup>th</sup>, *before* winter break

Note, a Design Document is due when you submit Lab1 final circuit

Work **alone**

## Save your work!

- *Save often*. Verify file is non-zero. Periodically save to Dropbox, email.
- Beware of MacOSX 10.5 (leopard) and 10.6 (snow-leopard)

## Homework1 is out

Due a week before prelim1, Monday, February 23rd

*Work on problems incrementally, as we cover them in lecture (i.e. part 1)*

Office Hours for help

Work **alone**

Work alone, **BUT** use your resources

- Lab Section, Piazza.com, Office Hours
- Class notes, book, Sections, CSUGLab

# Announcements

Check online syllabus/schedule

- <http://www.cs.cornell.edu/Courses/CS3410/2015sp/schedule.html>
- Slides and Reading for lectures
- Office Hours
- *Pictures of all TAs*
- Homework and Programming Assignments
- **Dates to keep in Mind**
  - **Prelims: Tue Mar 3rd and Thur April 30th**
  - ***Lab 1: Due this Friday, Feb 13th before Winter break***
  - Proj2: Due Thur Mar 26th before Spring break
  - Final Project: Due when final would be (not known until Feb 14t

Schedule is subject to change

# Collaboration, Late, Re-grading Policies

## “Black Board” Collaboration Policy

- Can discuss approach together on a “black board”
- Leave and write up solution independently
- Do not copy solutions

## Late Policy

- Each person has a total of **four** “slip days”
- Max of **two** slip days for any individual assignment
- Slip days deducted first for *any* late assignment, cannot selectively apply slip days
- For projects, slip days are deducted from all partners
- **25%** deducted per day late after slip days are exhausted

## Regrade policy

- Submit written request to lead TA,  
and lead TA will pick a different grader
- Submit another written request,  
lead TA will regrade directly
- Submit yet another written request for professor to regrade.

# Goals for today

## MIPS Datapath

- Memory layout
- Control Instructions

## Performance

- How fast can we make it?
- CPI (Cycles Per Instruction)
- MIPS (Instructions Per Cycle)
- Clock Frequency

# MIPS Instruction Types

## Arithmetic/Logical

- R-type: result and two source registers, shift amount
- I-type: 16-bit immediate with sign/zero extension

## Memory Access

- load/store between registers and memory
- word, half-word and byte operations

## Control flow

- conditional branches: pc-relative addresses
- jumps: fixed offsets, register absolute

# Memory Instructions

101011001010000100000000000000100

I-Type

op      rs      rd                  offset

6 bits    5 bits    5 bits                  16 bits

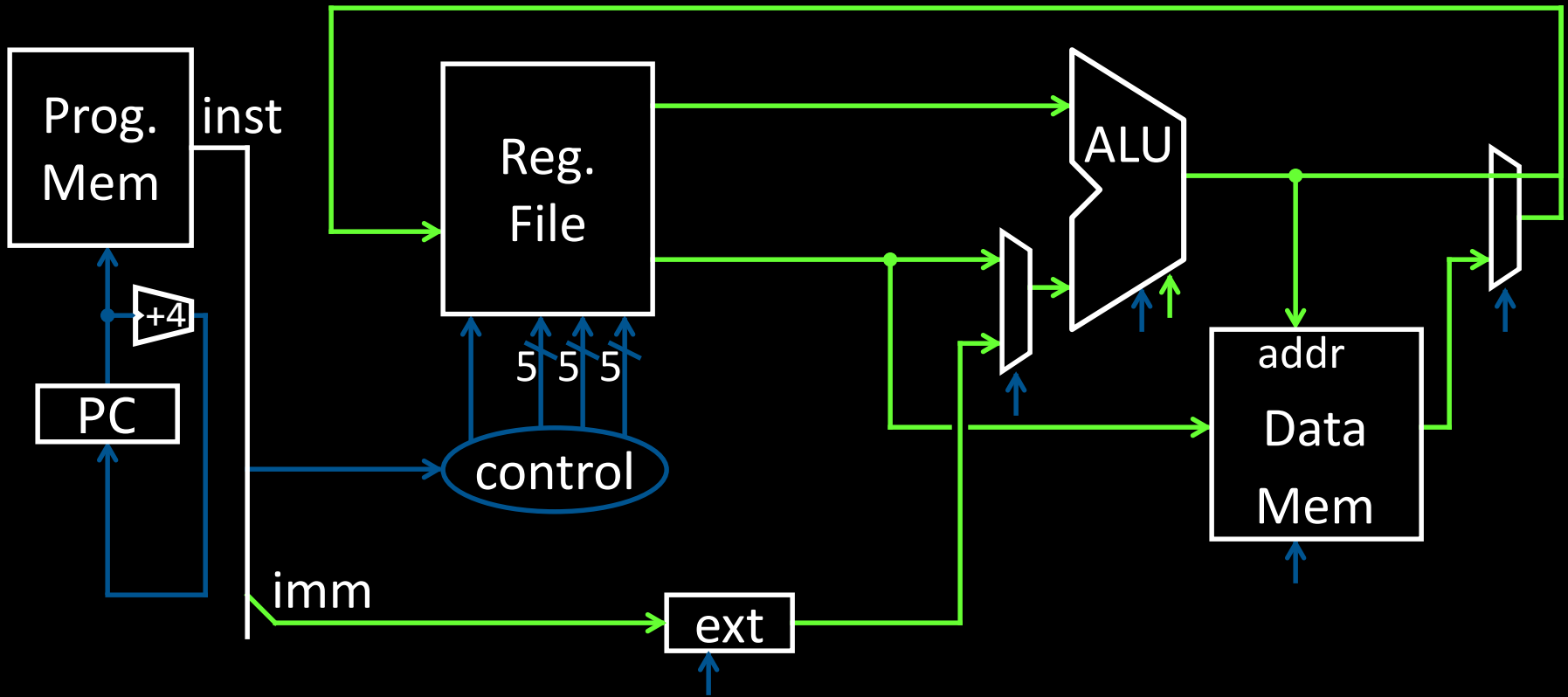
base + offset  
addressing

op	mnemonic	description
0x23	LW rd, offset(rs)	$R[rd] = \text{Mem}[\text{offset} + R[rs]]$
0x2b	SW rd, offset(rs)	$\text{Mem}[\text{offset} + R[rs]] = R[rd]$

signed  
offsets



# Memory Operations



# Memory Instructions

101011001010000100000000000000100

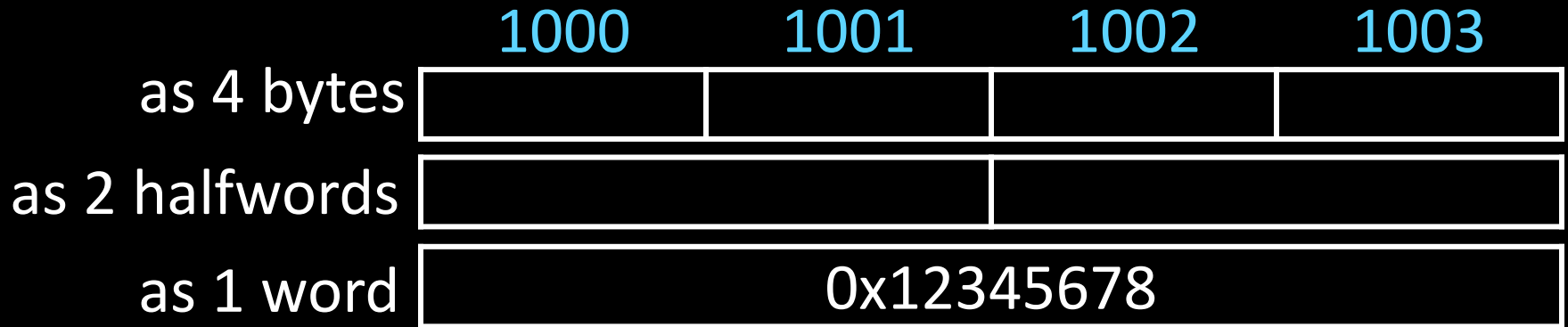
op      rs      rd                  offset  
6 bits   5 bits   5 bits                  16 bits

op	mnemonic	description
0x20	LB rd, offset(rs)	$R[rd] = \text{sign\_ext}(\text{Mem}[\text{offset}+R[rs]])$
0x24	LBU rd, offset(rs)	$R[rd] = \text{zero\_ext}(\text{Mem}[\text{offset}+R[rs]])$
0x21	LH rd, offset(rs)	$R[rd] = \text{sign\_ext}(\text{Mem}[\text{offset}+R[rs]])$
0x25	LHU rd, offset(rs)	$R[rd] = \text{zero\_ext}(\text{Mem}[\text{offset}+R[rs]])$
0x23	LW rd, offset(rs)	$R[rd] = \text{Mem}[\text{offset}+R[rs]]$
0x28	SB rd, offset(rs)	$\text{Mem}[\text{offset}+R[rs]] = R[rd]$
0x29	SH rd, offset(rs)	$\text{Mem}[\text{offset}+R[rs]] = R[rd]$
0x2b	SW rd, offset(rs)	$\text{Mem}[\text{offset}+R[rs]] = R[rd]$

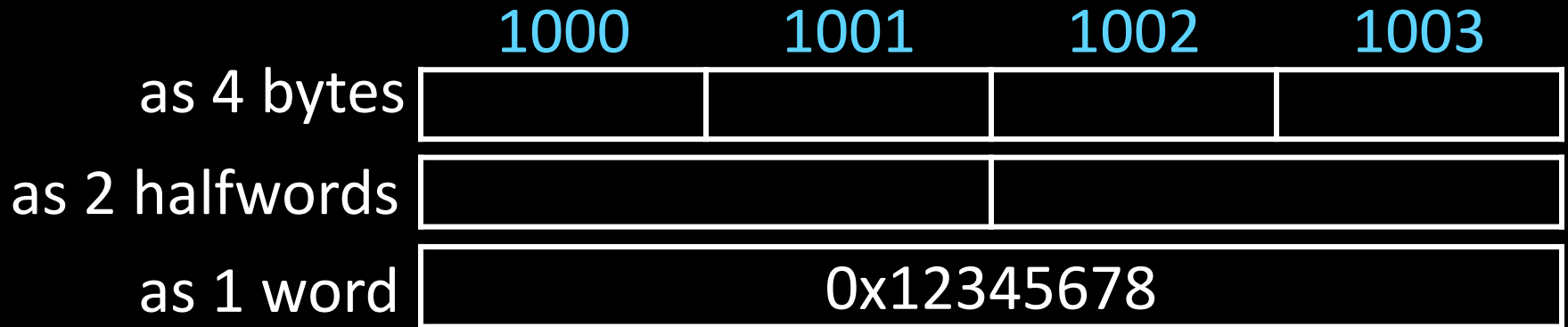
# Endianness

Endianness: Ordering of bytes within a memory word

**Little Endian** = least significant part first (MIPS, x86)



**Big Endian** = most significant part first (MIPS, networks)



# Memory Layout

Examples (big/little endian):

# r5 contains 5 (0x00000005)

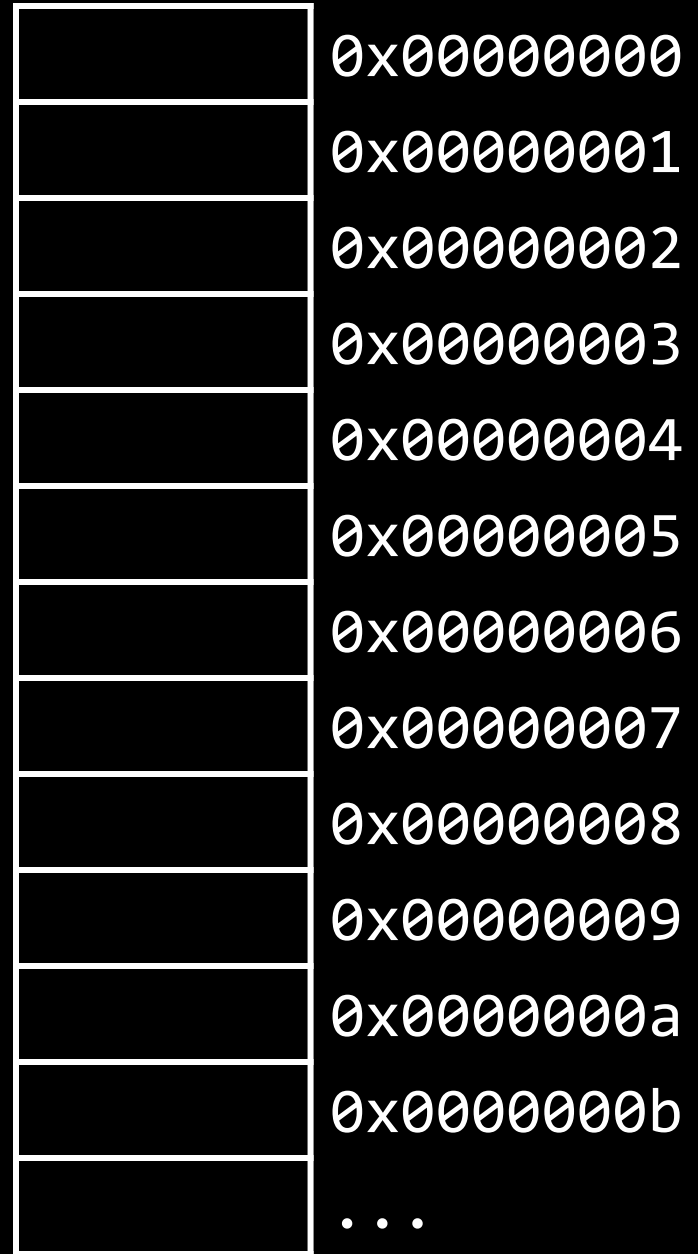
SB r5, 2(r0)

LB r6, 2(r0)

SW r5, 8(r0)

LB r7, 8(r0)

LB r8, 11(r0)



# MIPS Instruction Types

## Arithmetic/Logical

- R-type: result and two source registers, shift amount
- I-type: 16-bit immediate with sign/zero extension

## Memory Access

- load/store between registers and memory
- word, half-word and byte operations

## Control flow

- conditional branches: pc-relative addresses
- jumps: fixed offsets, register absolute

# Control Flow: Absolute Jump

00001001000000000000000000000000000001

op

6 bits

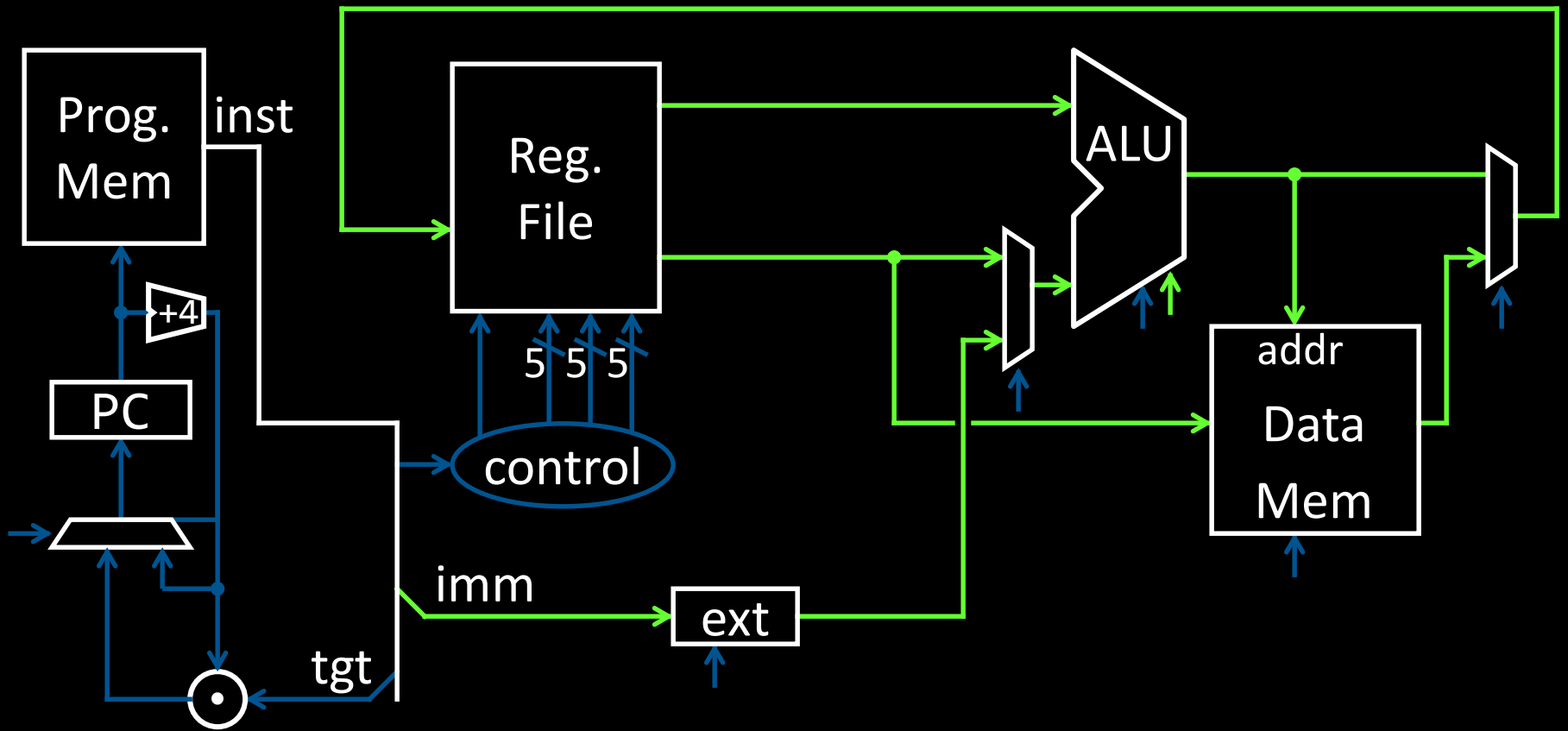
immediate

26 bits

J-Type

op	Mnemonic	Description
0x2	J target	$PC = (PC+4)_{31..28} \cdot \text{target} \cdot 00$

# Absolute Jump



# Control Flow: Jump Register

00000000011000000000000000000000001000

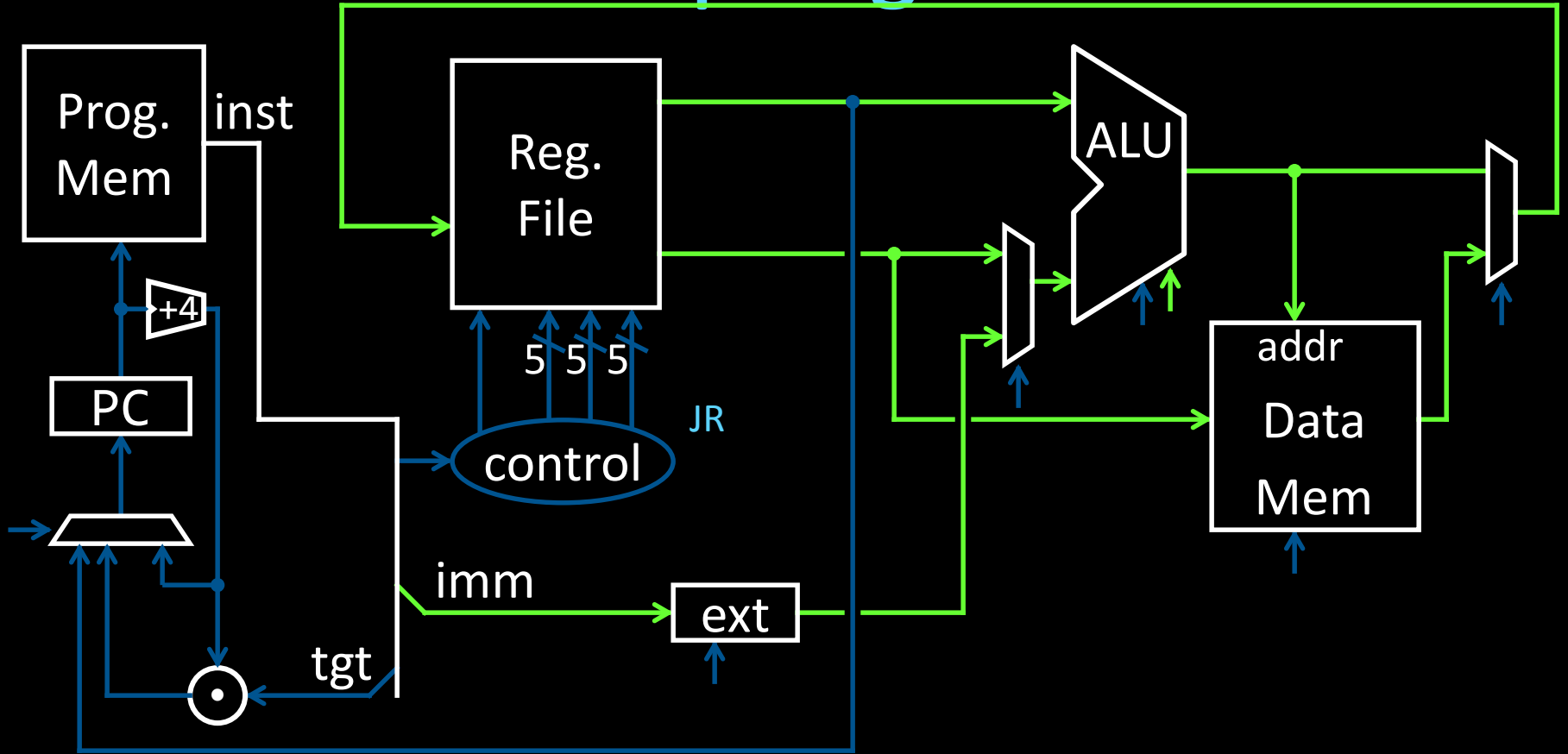
op          rs          -          -          -          func  
6 bits      5 bits      5 bits      5 bits      5 bits      6 bits

R-Type

op	func	mnemonic	description
0x0	0x08	JR rs	PC = R[rs]



# Jump Register



# Examples

E.g. Use Jump or Jump Register instruction to jump to 0xabcd1234

But, what about a jump based on a condition?

# assume  $0 \leq r3 \leq 1$

if ( $r3 == 0$ ) jump to 0xdecafe00

else jump to 0xabcd1234

# Control Flow: Branches

0001000010100001000000000000000011

op

6 bits

rs

5 bits

rd

5 bits

offset

16 bits

I-Type

signed  
offsets

op

mnemonic

description

0x4

BEQ rs, rd, offset

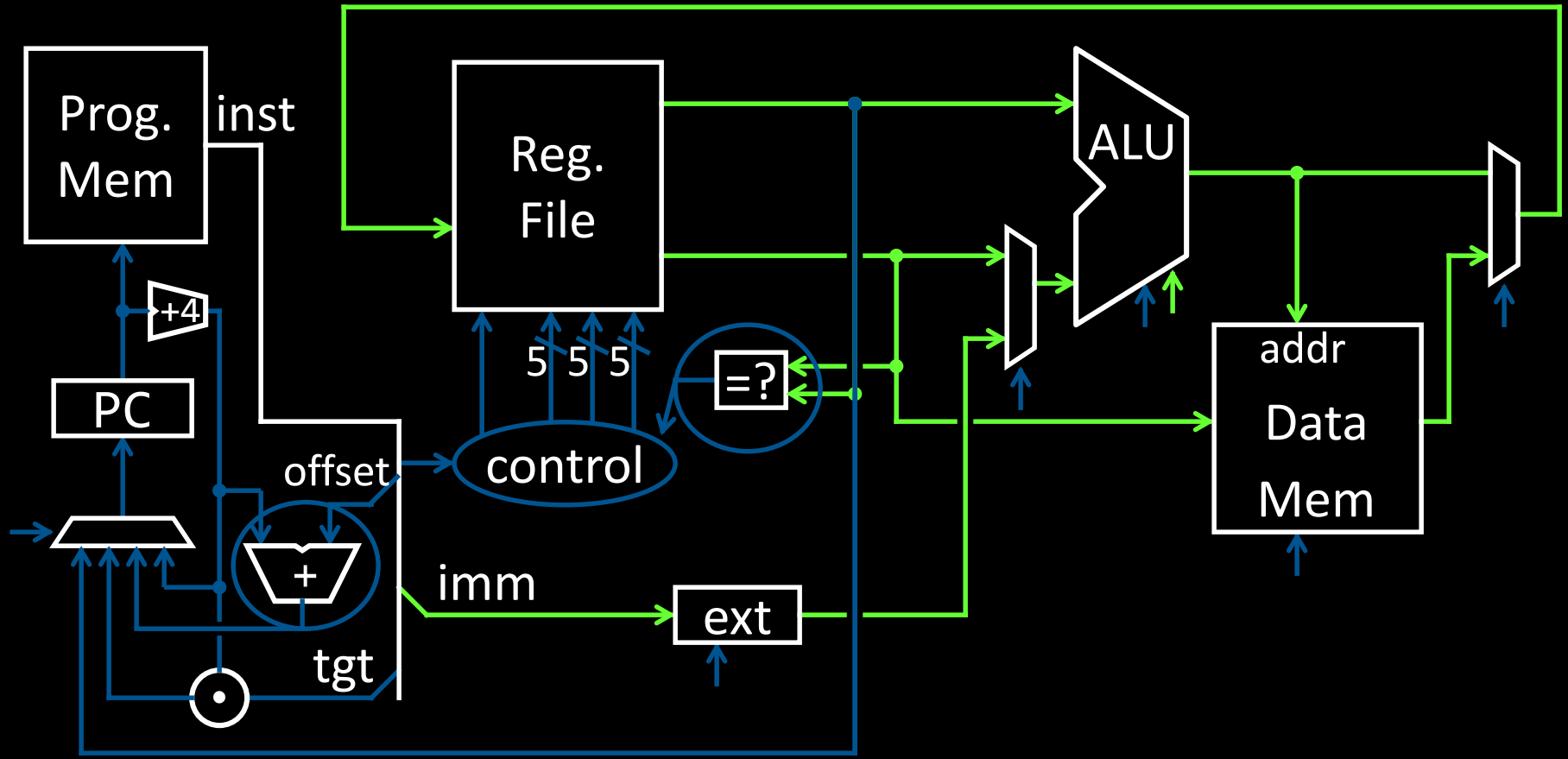
if  $R[rs] == R[rd]$  then  $PC = PC + 4 + (offset \ll 2)$

0x5

BNE rs, rd, offset

if  $R[rs] != R[rd]$  then  $PC = PC + 4 + (offset \ll 2)$

# Control Flow: Branches



# Control Flow: More Branches

## Conditional Jumps (cont.)

000001001010000100000000000000010

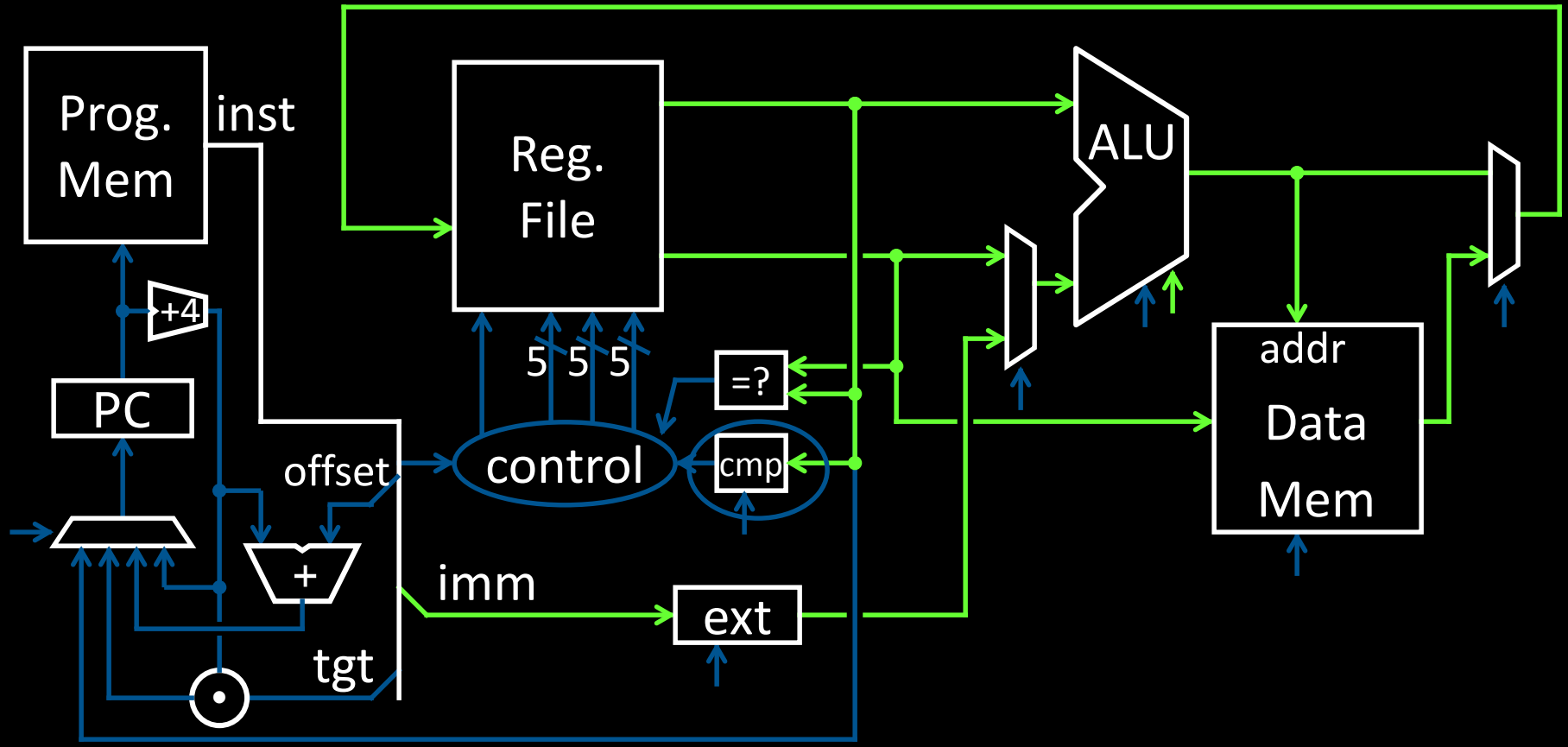
op      rs      subop      offset  
6 bits   5 bits   5 bits      16 bits

almost I-Type

signed  
offsets

op	subop	mnemonic	description
0x1	0x0	BLTZ rs, offset	if $R[rs] < 0$ then $PC = PC+4 + (\text{offset} \ll 2)$
0x1	0x1	BGEZ rs, offset	if $R[rs] \geq 0$ then $PC = PC+4 + (\text{offset} \ll 2)$
0x6	0x0	BLEZ rs, offset	if $R[rs] \leq 0$ then $PC = PC+4 + (\text{offset} \ll 2)$
0x7	0x0	BGTZ rs, offset	if $R[rs] > 0$ then $PC = PC+4 + (\text{offset} \ll 2)$

# Control Flow: More Branches



# Control Flow: Jump and Link

Why? Function/procedure calls

00001101000000000000000000000001

op

immediate

6 bits

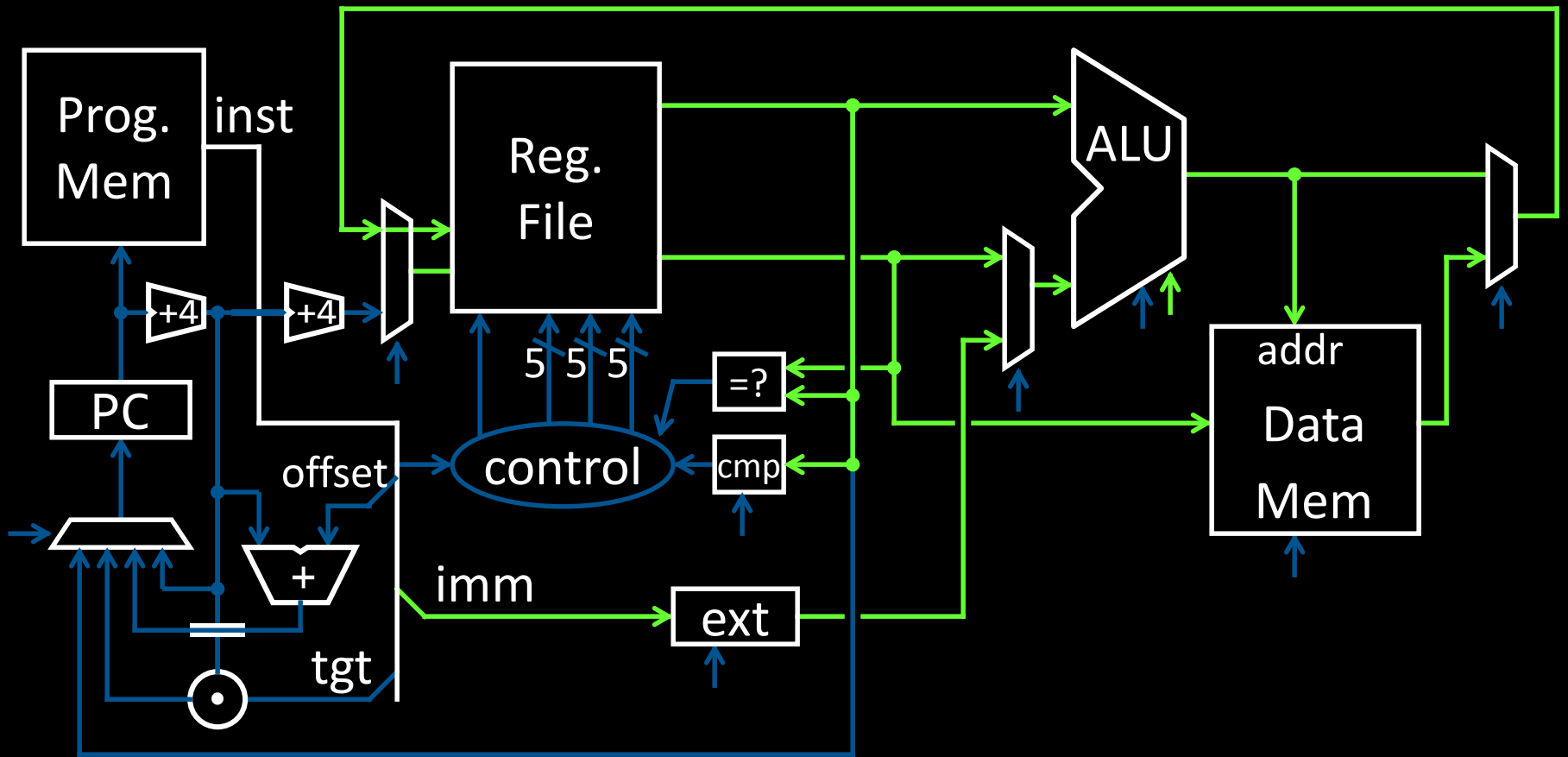
26 bits

J-Type

Discuss later

op	mnemonic	description
0x3	JAL target	$r31 = PC+8$ (+8 due to branch delay slot) $PC = (PC+4)_{31..28} \cdot \text{target} \cdot 00$

# Jump and Link





# Goals for today

## MIPS Datapath

- Memory layout
- Control Instructions

## Performance

- How to get it?
- CPI (Cycles Per Instruction)
- MIPS (Instructions Per Cycle)
- Clock Frequency

## Pipelining

- Latency vs throughput

# Questions

How do we measure performance?

What is the performance of a single cycle CPU?

How do I get performance?

See: P&H 1.6

# Performance

How do I get it?

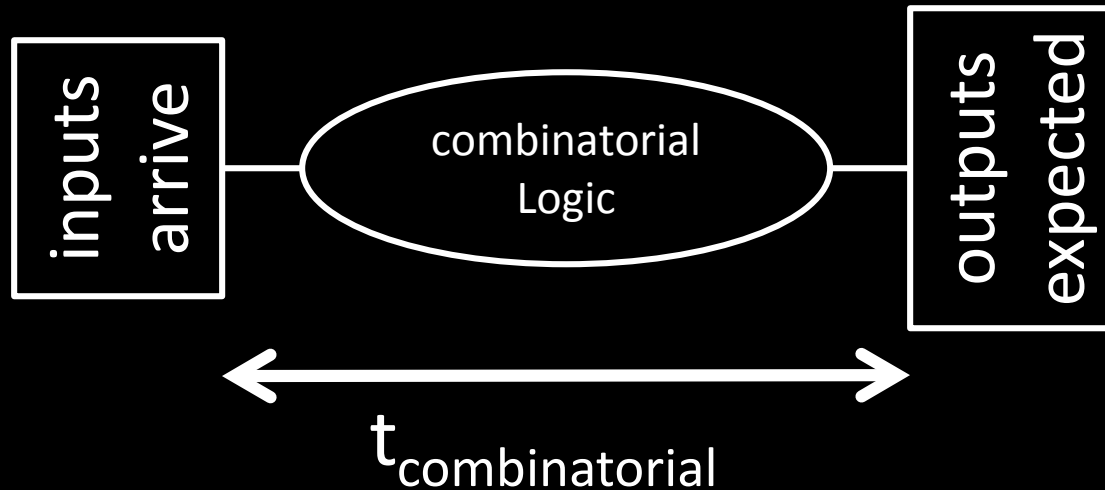
Parallelism

Pipelining

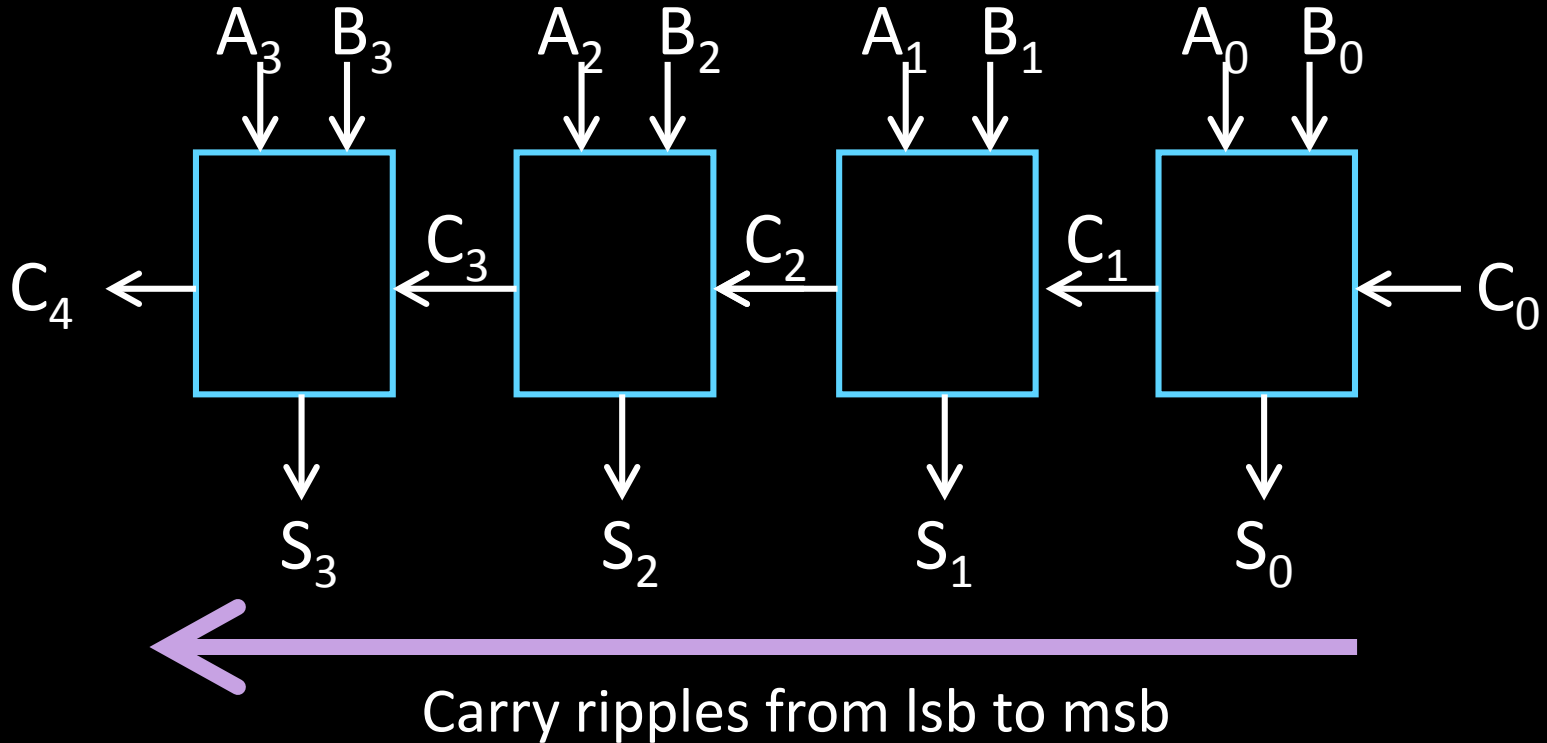
Both!

# Performance: Aside

Speed of a circuit is affected by the number of gates in series (on the *critical path* or the *deepest level of logic*)



# 4-bit Ripple Carry Adder



- First full adder, 2 gate delay
- Second full adder, 2 gate delay
- ...

# Adding

Main ALU, slows us down

Does it need to be this slow?

Observations

- Have to wait for  $C_{in}$
- Can we compute in parallel in some way?
- CLA carry look-ahead adder

# Carry Look Ahead Logic

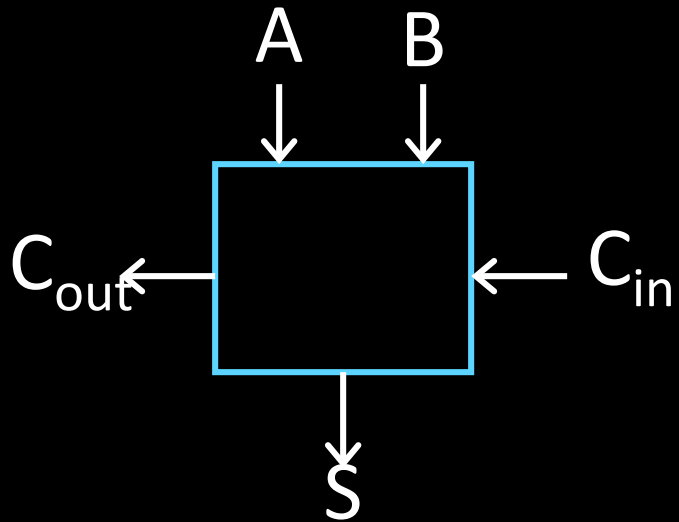
Can we reason  $C_{out}$  independent of  $C_{in}$ ?

- *Just based on (A,B) only*

When is  $C_{out} == 1$ , irrespective of  $C_{in}$ ?

If  $C_{in} == 1$ , when is  $C_{out}$  also  $== 1$

# 1-bit Adder with Carry



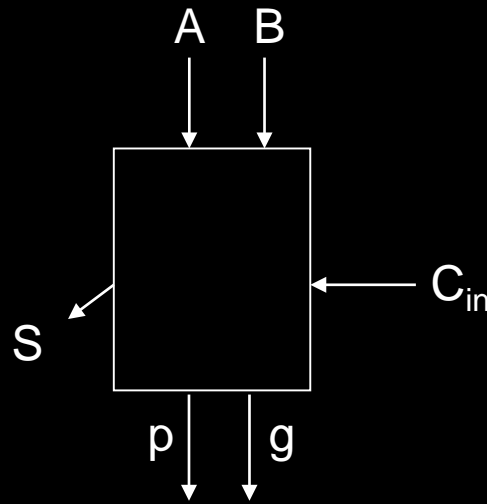
## Full Adder

- Adds three 1-bit numbers
- Computes 1-bit result and 1-bit carry
- Can be cascaded

A	B	C <sub>in</sub>	C <sub>out</sub>	S
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1



# 1-bit CLA adder



Create two terms: **propagator**, **generator**

$g = 1$ , *generates*  $C_{out}$ :  $g = AB$

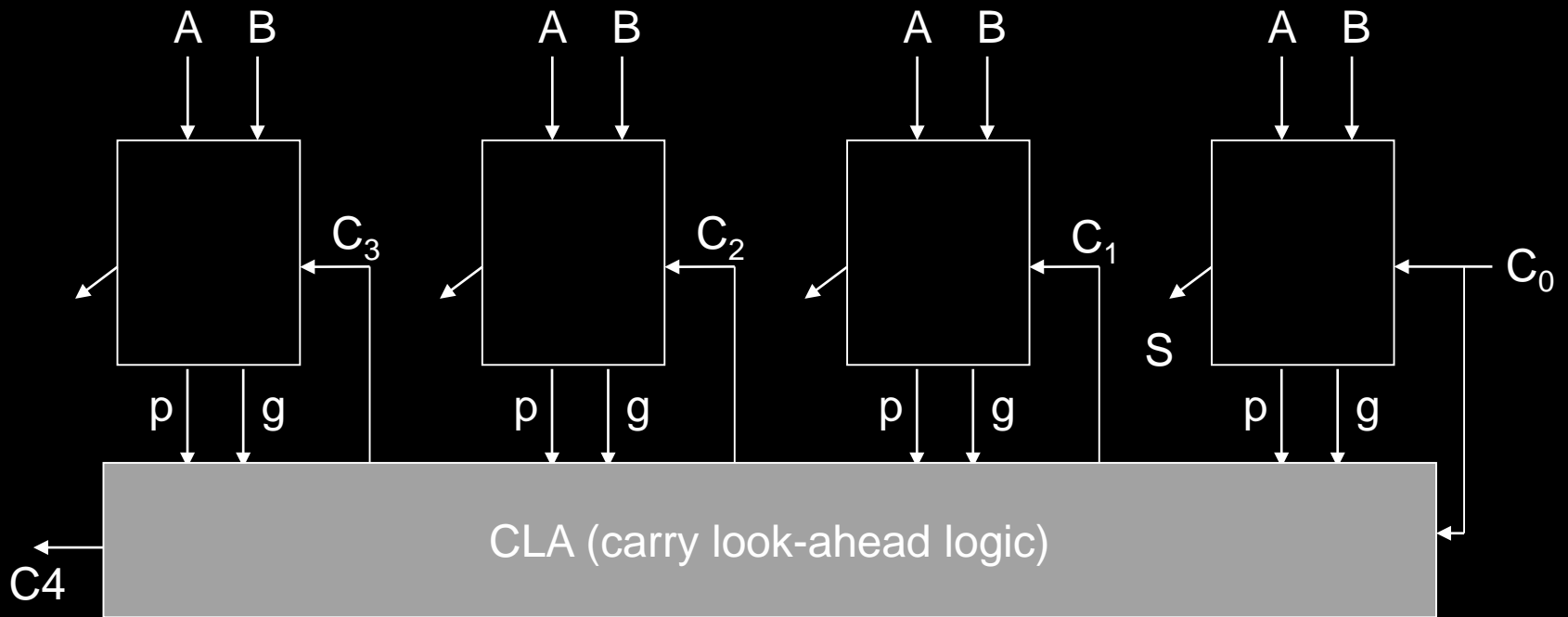
- Irrespective of  $C_{in}$

$p = 1$ , *propagates*  $C_{in}$  to  $C_{out}$ :  $p = A + B$

$p$  and  $g$  generated in 1 gate delay

$S$  is 2 gate delay **after** we get  $C_{in}$

# 4-bit CLA



# Performance

How do I get it?

Parallelism

Pipelining

Both!

# Goals for today

## MIPS Datapath

- Memory layout
- Control Instructions

## Performance

- How to get it? Parallelism and Pipeline!
- CPI (Cycles Per Instruction)
- MIPS (Instructions Per Cycle)
- Clock Frequency

*Next Time*

## Pipelining

- Latency vs throughput