CPSC 313: Computer Hardware and Operating Systems

Unit 1: The y86 (as a sequential processor)

2024 Winter Term 1

Administrivia:

- You are responsible for checking the schedule in plenty of time to :
  - Complete pre-class work before class
  - Complete lab assignments
- **Lab 1 is due a week from Sunday** (but start it soon!)
- **Quiz 0 is due Sep 18**
  - Practice questions, advice, and quiz information are on PrairieLearn (Quiz 0 Practice and Quiz 0 Information).
- **C Refresher Tutorials:**
  - Happened on Thu Sep 5, but...
  - A video version is on Canvas
    (as "A previous term's C Refresher" in the first table entry)

Suggestions from the Field

- We release slides at the time we cover them; we link them directly to the content related to them (e.g., a video or a lecture).
  - If you wish to save them to a centralized place, you may definitely do so you could even start a Piazza post and track them for everyone ☺
- When we ask questions in class: *try to answer them yourself*. You learn nothing from someone else answering ☹
- Calculations: we believe that we (or previous education) have taught you how to do the calculations that we ask for. Tell us if that’s wrong!
- More generally: Help us help you -- ask questions! We really do welcome you to raise your hand and ask questions in class.

Logistics

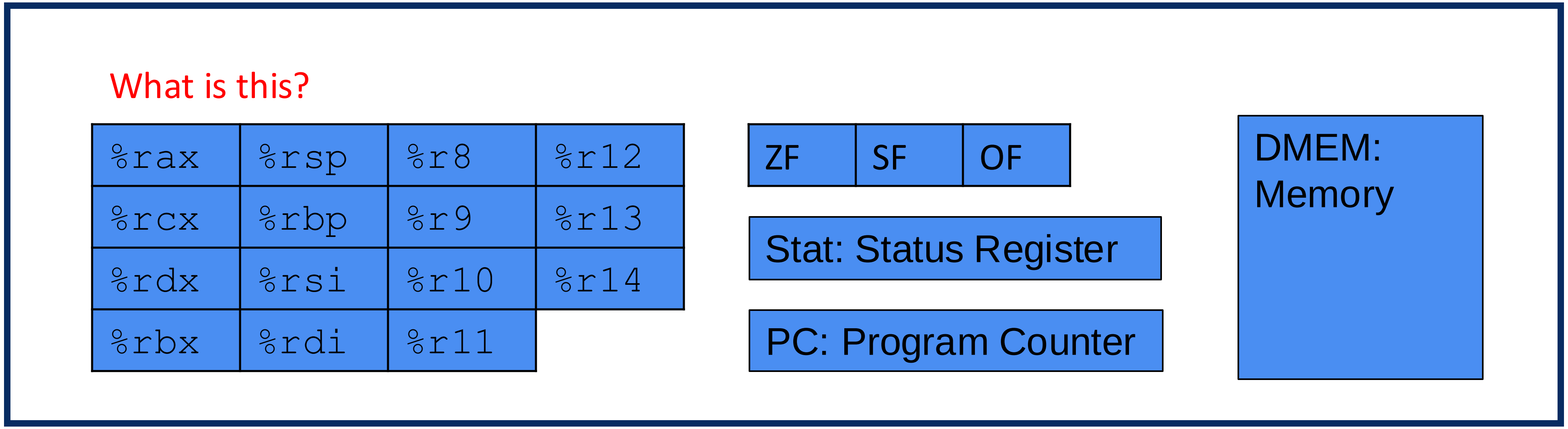
- Running out of time on in-class activity
  - Submit at least some work! It’s graded on participation (> 0% becomes 100%).
  - Complete the remainder of the activity before the next lecture – if possible.

Today

- Topics: These should be things you learned in 213!
  - How is data represented?
  - Little endian representation
  - 2’s complement
  - Hexadecimal
- Learning outcomes
  - Remember how to do arithmetic in hex and what it means
  - Map data representation to the y86 architecture
  - Remember how to read/write C code

Since this is review, we'll go quickly...
And mostly do our first graded in-class exercise!

The y86 in a single slide



What is this?

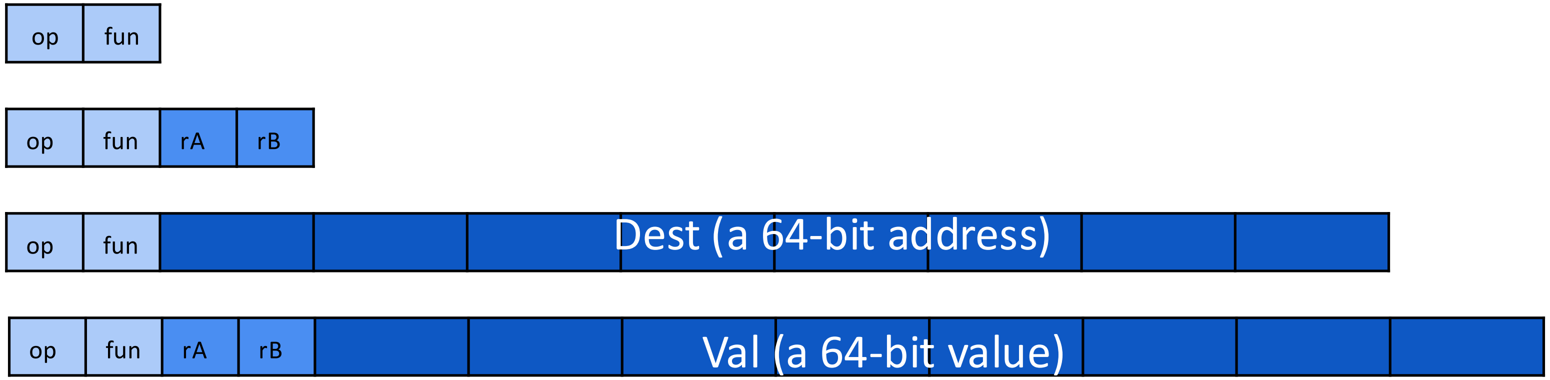
%rax	%rsp	%r8	%r12
%rcx	%rbp	%r9	%r13
%rdx	%rsi	%r10	%r14
%rbx	%rdi	%r11	

ZF	SF	OF

Stat: Status Register

PC: Program Counter

DMEM:
Memory



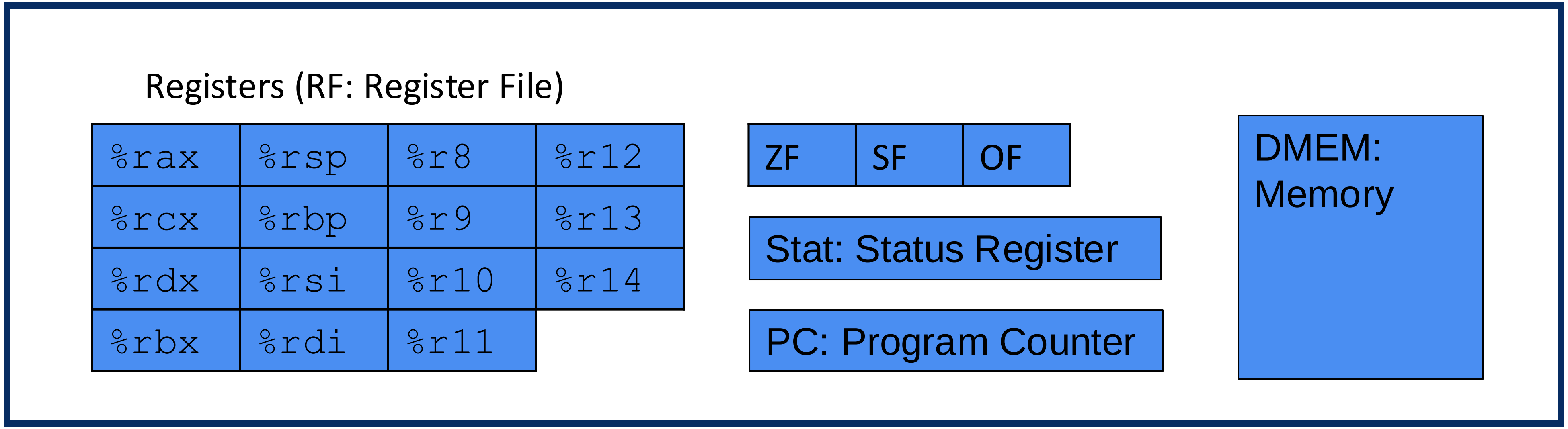
1-byte instructions: op | fun

2-byte instructions: op | fun | rA | rB

9-byte instructions: op | fun | Dest (a 64-bit address)

10-byte instructions: op | fun | rA | rB | Val (a 64-bit value)

The y86 in a single slide



Registers (RF: Register File)

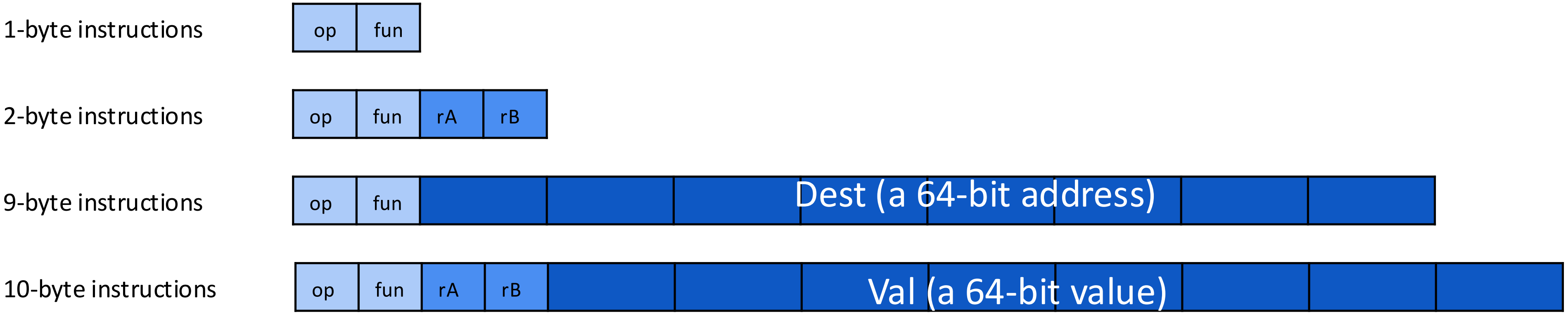
%rax	%rsp	%r8	%r12
%rcx	%rbp	%r9	%r13
%rdx	%rsi	%r10	%r14
%rbx	%rdi	%r11	

ZF	SF	OF

Stat: Status Register

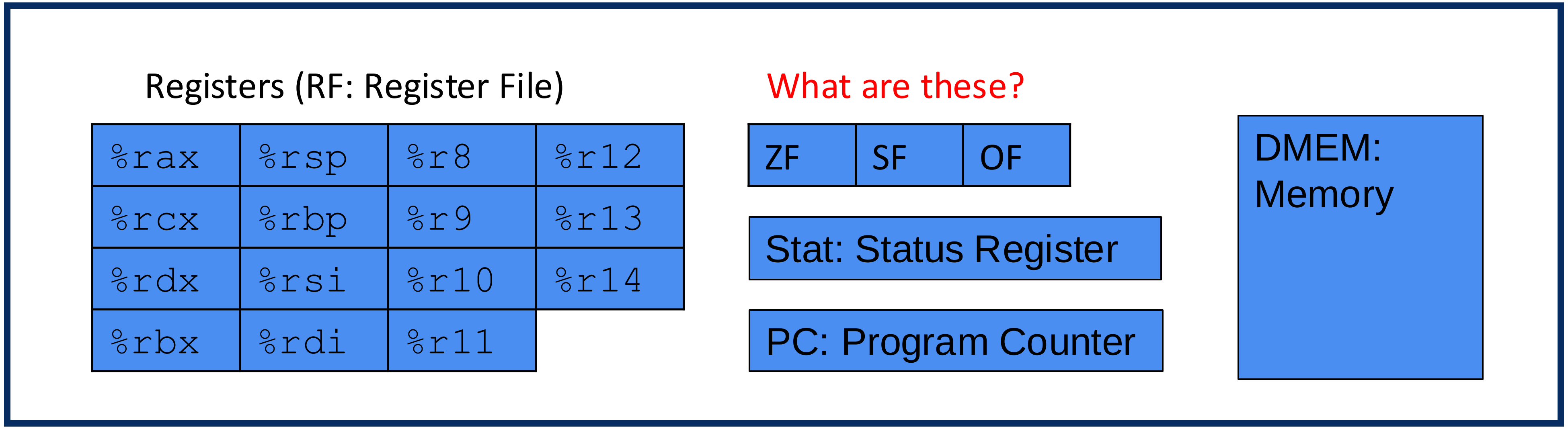
PC: Program Counter

DMEM: Memory



1-byte instructions	op	fun			
2-byte instructions	op	fun	rA	rB	
9-byte instructions	op	fun			Dest (a 64-bit address)
10-byte instructions	op	fun	rA	rB	Val (a 64-bit value)

The y86 in a single slide



Registers (RF: Register File)

%rax	%rsp	%r8	%r12
%rcx	%rbp	%r9	%r13
%rdx	%rsi	%r10	%r14
%rbx	%rdi	%r11	

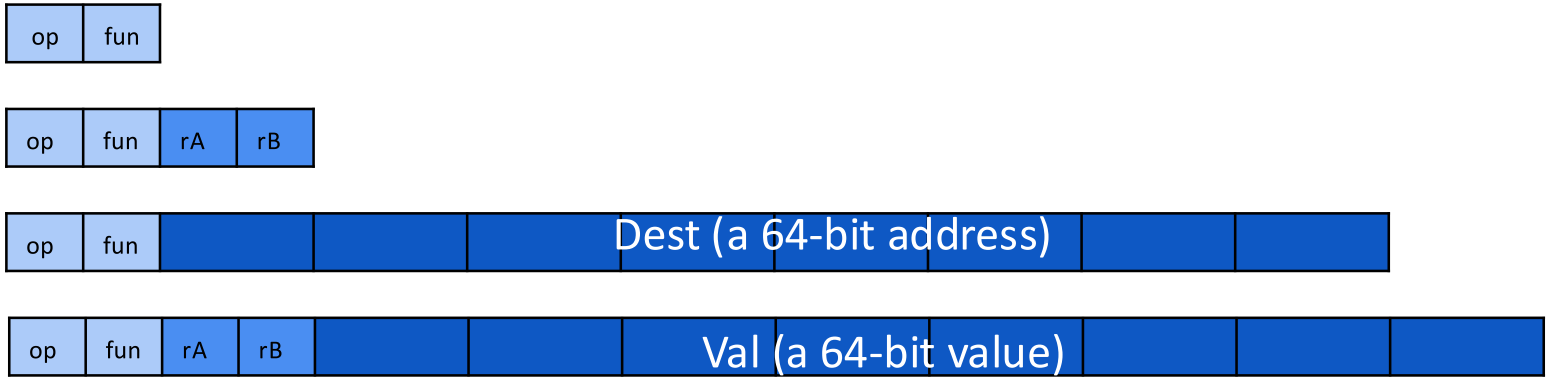
What are these?

ZF	SF	OF

Stat: Status Register

PC: Program Counter

DMEM: Memory



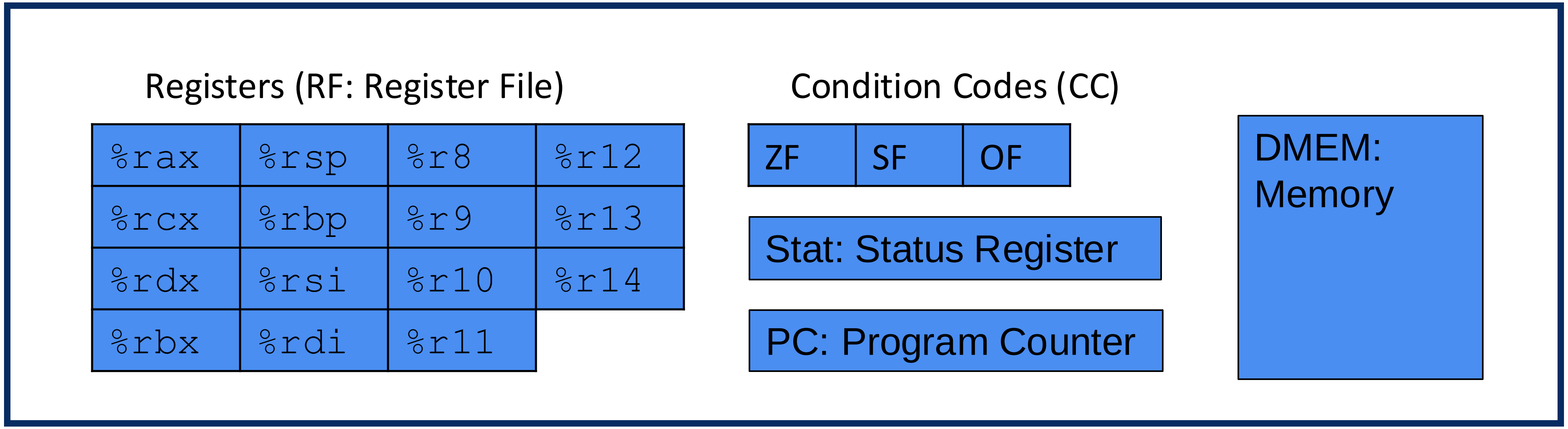
1-byte instructions: op | fun

2-byte instructions: op | fun | rA | rB

9-byte instructions: op | fun | Dest (a 64-bit address)

10-byte instructions: op | fun | rA | rB | Val (a 64-bit value)

The y86 in a single slide



Registers (RF: Register File)

%rax	%rsp	%r8	%r12
%rcx	%rbp	%r9	%r13
%rdx	%rsi	%r10	%r14
%rbx	%rdi	%r11	

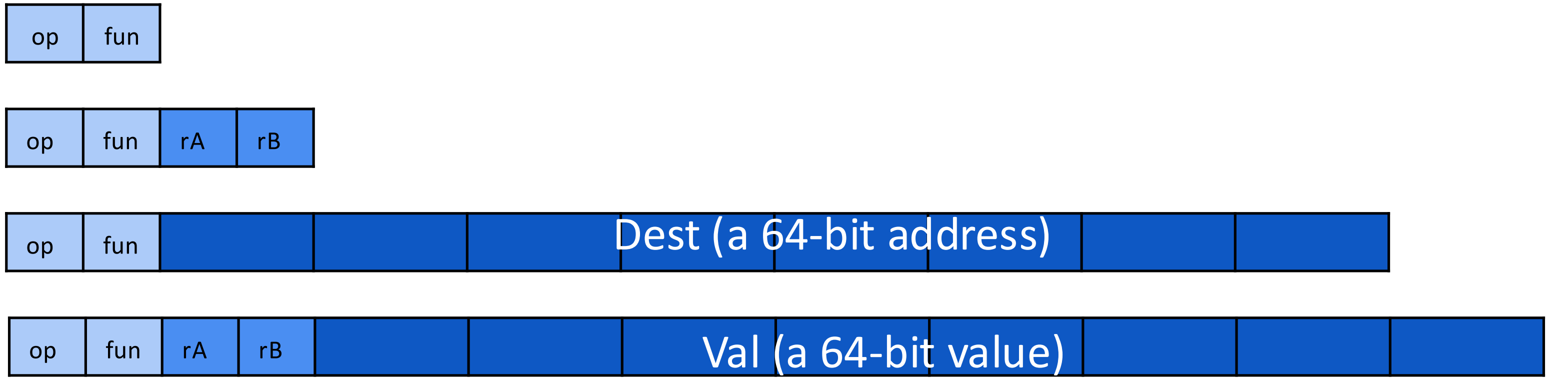
Condition Codes (CC)

ZF	SF	OF

Stat: Status Register

PC: Program Counter

DMEM: Memory



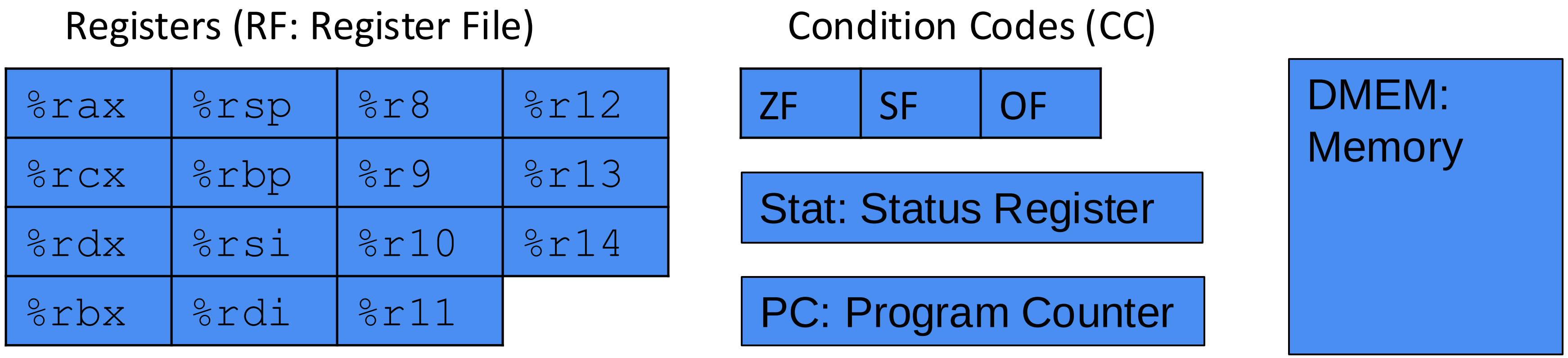
1-byte instructions: op | fun

2-byte instructions: op | fun | rA | rB

9-byte instructions: op | fun | Dest (a 64-bit address)

10-byte instructions: op | fun | rA | rB | Val (a 64-bit value)

The y86 in a single slide



Registers (RF: Register File)

%rax	%rsp	%r8	%r12
%rcx	%rbp	%r9	%r13
%rdx	%rsi	%r10	%r14
%rbx	%rdi	%r11	

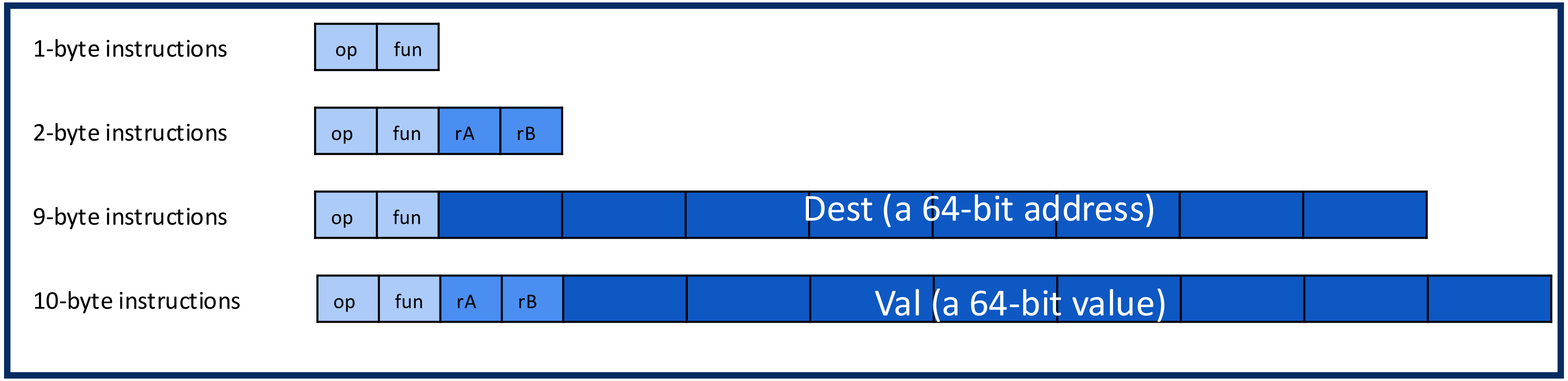
Condition Codes (CC)

ZF	SF	OF

Stat: Status Register

PC: Program Counter

DMEM: Memory

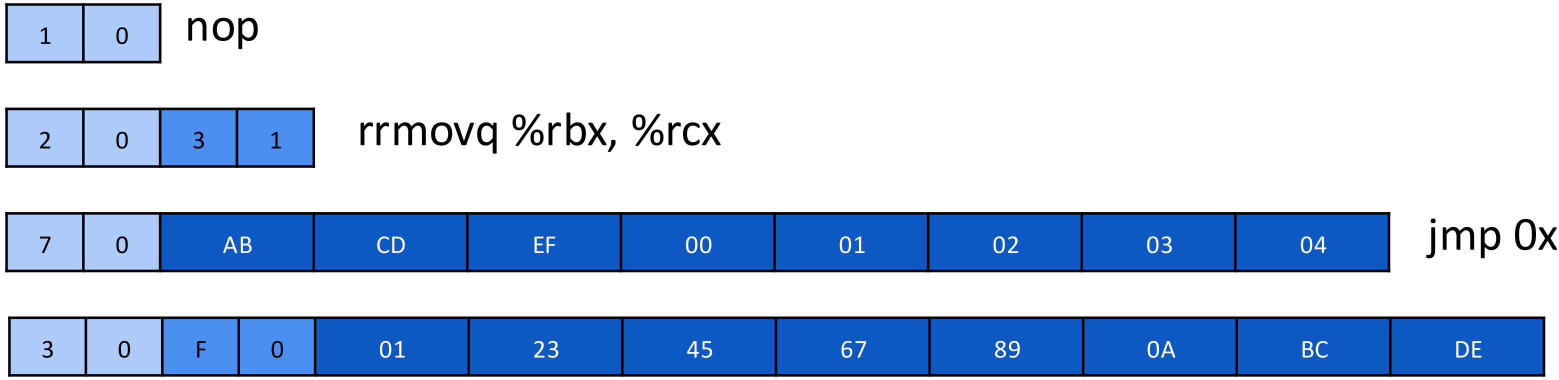


1-byte instructions	op	fun			
2-byte instructions	op	fun	rA	rB	
9-byte instructions	op	fun			Dest (a 64-bit address)
10-byte instructions	op	fun	rA	rB	Val (a 64-bit value)

Representing Data In Memory

- rswap.ys : Swap two values in registers (probably just this one)
- mswap.ys : Swap two values in memory

Little Endian Representation



0x1000 | 1 | 0 | nop

0x1001 | 2 | 0 | 3 | 1 | rrmovq %rbx, %rcx

0x1003 | 7 | 0 | AB | CD | EF | 00 | 01 | 02 | 03 | 04 | jmp 0x0403020100EFCDAB

0x100C | 3 | 0 | F | 0 | 01 | 23 | 45 | 67 | 89 | 0A | BC | DE

irmovq 0xDEBC0A8967452301, %rax

0x1000	10	20	31	70	AB	CD	EF	00
0x1008	01	02	03	04	30	F0	01	23
	45	67	89	0A	BC	DE	XX	XX

A program in memory is just a sequence of bytes!

Expressing Negative Numbers

- 2’s complement
    1. Write the positive number in binary
    2. Flip all the bits
    3. Add 1

In-Class Exercise

- Here is a fun way to gain practice reading stuff in memory and refreshing your C programming skill.
- It’s a scavenger hunt! You will find it as the first in-class exercise on PrairieLearn.
- Recommendations:
  - Everyone open the code on their own screens
  - Everyone also watch one screen where you run experiements and you can all see the results, e.g., “./treasure 15”
  - Help each other understand what different treasures are doing.
  - HAVE FUN!

Coming Up

- Check the Canvas Syllabus and PrairieLearn to know what’s coming!
- In the short-term:
  - Almost always pre- and in-class exercises coming up!
    - Check the pre-class exercise video/slides for textbook readings
  - Start Lab 1
  - Do Quiz 0 (but first work through Quiz 0 Practice/Information and Lab 1)