Department of Computer Science
UNIVERSITY OF COLORADO **BOULDER**

Machine-Level Programming II: Control

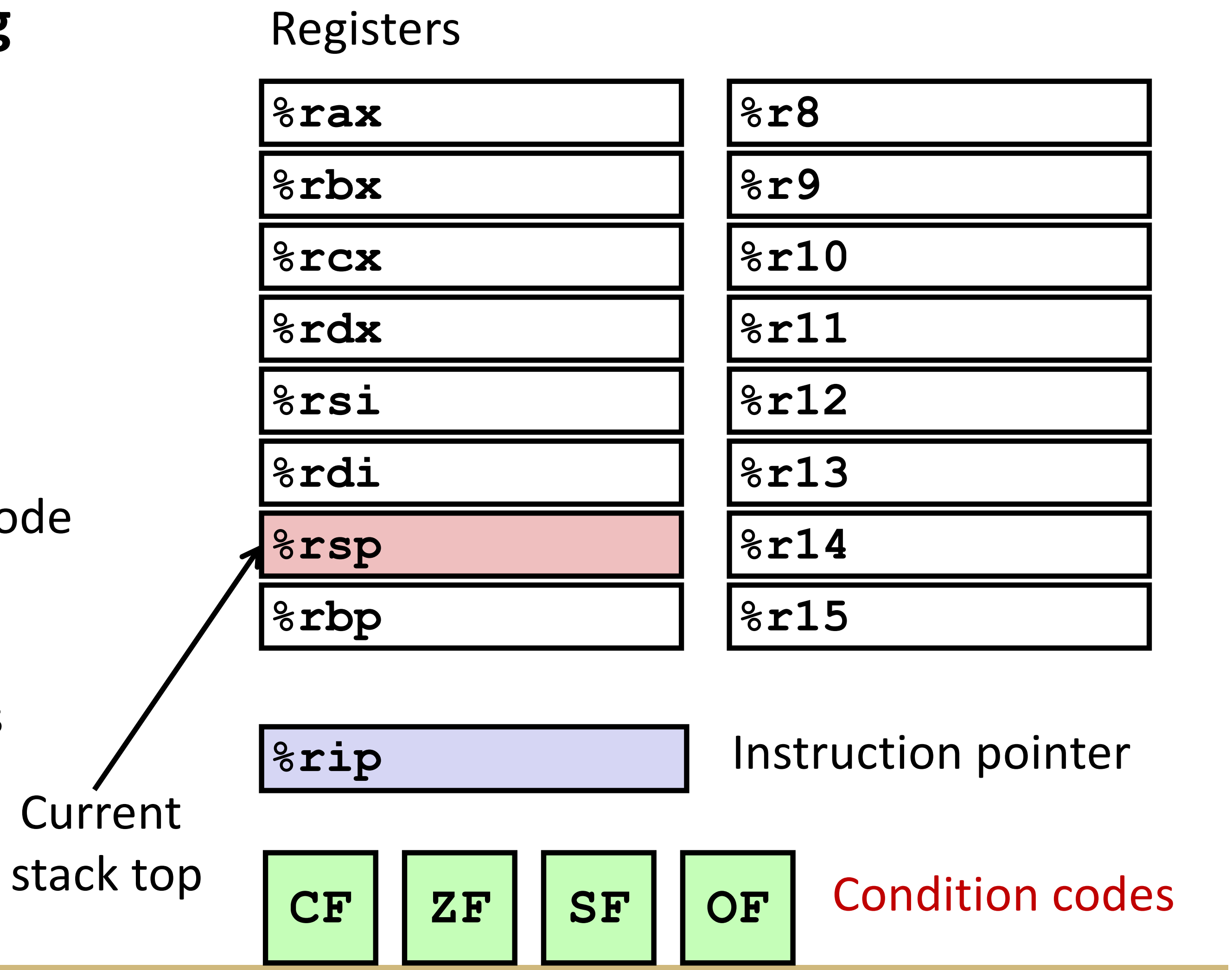
These slides adapted from materials provided by the textbook

Machine-Level Programming II: Control

- **Control: Condition codes**
- Conditional branches
- Loops
- Switch Statements

Processor State (x86-64, Partial)

- **Information about currently executing program**
  - Temporary data (**`%rax`**, …)
  - Location of runtime stack (**`%rsp`**)
  - Location of current code control point (**`%rip`**, …)
  - Status of recent tests (CF, ZF, SF, OF)



Registers

`%rax`	`%r8`
`%rbx`	`%r9`
`%rcx`	`%r10`
`%rdx`	`%r11`
`%rsi`	`%r12`
`%rdi`	`%r13`
`%rsp`	`%r14`
`%rbp`	`%r15`

Current stack top

`%rip` Instruction pointer

CF ZF SF OF Condition codes

Condition Codes (Implicit Setting)

- **Single bit registers**

CF	Carry Flag (for unsigned)	SF	Sign Flag (for signed)
ZF	Zero Flag	OF	Overflow Flag (for signed)

- **Implicitly set (think of it as side effect) by arithmetic operations**

  Example: `addq` Src,Dest ↔ `t = a+b`

  CF set if carry out from most significant bit (unsigned overflow)

  ZF set if `t == 0`

  SF set if `t < 0` (as signed)

  OF set if two's-complement (signed) overflow
  `(a>0 && b>0 && t<0) || (a<0 && b<0 && t>=0)`

- **Not set by `leaq` instruction**

Condition Codes (Explicit Setting: Compare)

- **Explicit Setting by Compare Instruction**
  - **`cmpq`** Src2, Src1
  - **`cmpq b,a`** like computing **`a-b`** without setting destination

  - CF set if carry out from most significant bit (used for unsigned comparisons)
  - ZF set if **`a == b`**
  - SF set if **`(a-b) < 0`** (as signed)
  - OF set if two's-complement (signed) overflow

  `(a>0 && b<0 && (a-b)<0) || (a<0 && b>0 && (a-b)>0)`

Condition Codes (Explicit Setting: Test)

- **Explicit Setting by Test instruction**
  - **`testq`** Src2, Src1
    - **`testq b,a`** like computing **`a&b`** without setting destination

  - Sets condition codes based on value of Src1 & Src2
  - Useful to have one of the operands be a mask

  - ZF set when **`a&b == 0`**
  - SF set when **`a&b < 0`**

Reading Condition Codes

- **SetX Instructions**
  - Set low-order byte of destination to 0 or 1 based on combinations of condition codes
  - Does not alter remaining 7 bytes

SetX	Condition	Description
`sete`	**`ZF`**	Equal / Zero
`setne`	**`~ZF`**	Not Equal / Not Zero
`sets`	**`SF`**	Negative
`setns`	**`~SF`**	Nonnegative
`setg`	**`~(SF^OF)&~ZF`**	Greater (Signed)
`setge`	**`~(SF^OF)`**	Greater or Equal (Signed)
`setl`	**`(SF^OF)`**	Less (Signed)
`setle`	**`(SF^OF)|ZF`**	Less or Equal (Signed)
`seta`	**`~CF&~ZF`**	Above (unsigned)
`setb`	**`CF`**	Below (unsigned)

x86-64 Integer Registers

%rax	%al	%r8	%r8b
%rbx	%bl	%r9	%r9b
%rcx	%cl	%r10	%r10b
%rdx	%dl	%r11	%r11b
%rsi	%sil	%r12	%r12b
%rdi	%dil	%r13	%r13b
%rsp	%spl	%r14	%r14b
%rbp	%bpl	%r15	%r15b

- Can reference low-order byte

Reading Condition Codes (Cont.)

- **SetX Instructions:**
  - Set single byte based on combination of condition codes
- **One of addressable byte registers**
  - Does not alter remaining bytes
  - Typically use **`movzbl`** to finish job
    - 32-bit instructions also set upper 32 bits to 0

```
int gt (long x, long y)
{
  return x > y;
}
```

Register	Use(s)
`%rdi`	Argument **`x`**
`%rsi`	Argument **`y`**
`%rax`	Return value

```
    cmpq   %rsi, %rdi   # Compare x:y
    setg   %al          # Set when >
    movzbl %al, %eax    # Zero rest of %rax
    ret
```

Machine-Level Programming II: Control

- Control: Condition codes
- **Conditional branches**
- Loops
- Switch Statements