CSCI 370: Computer Architecture

x86-64 Assembly Reference

General-Purpose x86-64 Registers

64-bit	32-bit	16-bit	8-bit LSB	8-bit MSB	Saved	Use
%rdi	%edi	%di	%dil		caller	Arg 1
%rsi	%esi	%si	%sil		caller	Arg 2
%rdx	%edx	%dx	%dl	%dh	caller	Arg 3
%rcx	%ecx	%cx	%cl	%ch	caller	Arg 4
%r8	%r8d	%r8w	%r8b		caller	Arg 5
%r9	%r9d	%r9w	%r9b		caller	Arg 6
%rax	%eax	%ax	%al	%ah	caller	Return
%r10	%r10d	%r10w	%r10b		caller	
%r11	%r11d	%r11w	%r11b		caller	
%rbx	%ebx	%bx	%bl	%bh	callee	
%r12	%r12d	%r12w	%r12b		callee	
%r13	%r13d	%r13w	%r13b		callee	
%r14	%r14d	%r14w	%r14b		callee	
%r15	%r15d	%r15w	%r15b		callee	
%rbp	%ebp	%bp	%bpl		callee	Base Ptr
%rsp	%esp	%sp	%spl		callee	Stack Ptr
%rip	%eip	%ip				Instruction Ptr

mov and lea + Instruction Parameters

Imm: Immediate
- C numeric literal
- prefixed with a $
- i.e. $-14 $0x423

Reg: Register
- General register
- prefixed with a %
- i.e. $esi %rax

Mem: Memory Access
- Equivalent to ptr deref
- encapsulated in parens
- see addressing modes

```
mov   src, dest

mov Imm, Mem   *p=4;
mov Imm, Reg    x=3;
mov Mem, Reg    x=*p;
mov Reg, Mem   *p=x;
mov Reg, Reg    x=y;
```

Used for Assignment

```
lea   src, dest

lea Mem, Reg   p1=p2;
```

Used for Address Computation

mov**z** - Zero Extend

mov**s** - Sign Extend

Addressing Modes

Normal	(R_b)	Mem[R_b]
Displacement	D(R_b)	Mem[R_b + D]
Complete	D(R_b,R_i,S)	Mem[R_b + R_i×S + D]
Variations	D(R_b,R_i)	Mem[R_b + R_i + D]
	(R_b,R_i)	Mem[R_b + R_i]
	(R_b,R_i,S)	Mem[R_b + R_i×S + D]

Arithmetic Operations

Name	Operands	Behavior	Description
add	src,dst	dst+=src	addition
sub	src,dst	dst-=src	subtraction
imul	src,dst	dst*=src	multiplication
sal	src,dst	dst<<=src	left shift
sar	src,dst	dst>>=src	arithmetic right shift
shr	src,dst	dst>>=src	logical right shift
xor	src,dst	dst^=src	exclusive or
and	src,dst	dst&=src	bitwise and
or	src,dst	dst|=src	bitwise or
inc	dst	++dst	increment
dec	dst	--dst	decrement
neg	dst	dst=-dst	negation
not	dst	dst=~dst	bitwise not

Condition Codes

CF	Carry	Carry out from MSB (unsigned overflow)
ZF	Zero	result = 0
SF	Sign	result < 0 (as signed)
OF	Overflow	two's complement (signed) overflow

Suffix on j,cmov,set		*Description* cmp	test
e		==	==0
ne		!=	!=0
s		negative	
ns		non-negative	
UInt	TInt		
a	g	>	
ae	ge	>=	
b	l	<	
be	le	<=	

test src2,src1
and

CF ZF SF OF

cmp src2,src1
sub

CF ZF SF OF

Control Flow

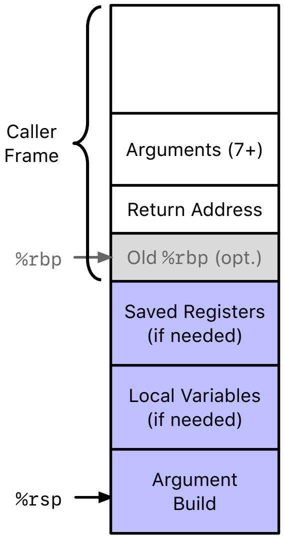
push %r
```
sub  $8, %rsp
mov  %r,(%rsp)
```

pop %r
```
mov  (%rsp),%r
add  $8, %rsp
```

call Fn
- *Push return address*
- *Jump to Fn*

ret
- *Pop address from stack*
- *Jump to address*

Stack Frame


Caller
Frame
Arguments (7+)
Return Address
%rbp
Old %rbp (opt.)
Saved Registers
(if needed)
Local Variables
(if needed)
%rsp
Argument
Build