# Arithmetic Operations

| | | | |
|---|---|---|---|
| `addq` | *Src,Dest* | Dest = Dest + Src | |
| `subq` | *Src,Dest* | Dest = Dest - Src | |
| `imulq` | *Src,Dest* | Dest = Dest * Src | |
| `sarq` | *Src,Dest* | Dest = Dest >> Src | ***Arithmetic*** |
| `shrq` | *Src,Dest* | Dest = Dest >> Src | ***Logical*** |
| `salq` | *Src,Dest* | Dest = Dest << Src | ***Also called shlq*** |
| `xorq` | *Src,Dest* | Dest = Dest ^ Src | |
| `andq` | *Src,Dest* | Dest = Dest & Src | |
| `orq` | *Src,Dest* | Dest = Dest \| Src | |
| `incq` | *Dest* | *Dest = Dest + 1* | |
| `decq` | *Dest* | *Dest = Dest - 1* | |
| `negq` | *Dest* | *Dest = -Dest* | |
| `notq` | *Dest* | *Dest = ~Dest* | |
| `leaq` | *Src,Dest* | Dest = Src (as expr) | ***No memory access!*** |



# Arithmetic Example

```
long arith
(long x, long y, long z)
{
  long t1 = x+y;
  long t2 = z+t1;
  long t3 = x+4;
  long t4 = y * 48;
  long t5 = t3 + t4;
  long rval = t2 * t5;
  return rval;
}
```

`(x,y,z) -> (%rdi,%rsi,%rdx)`

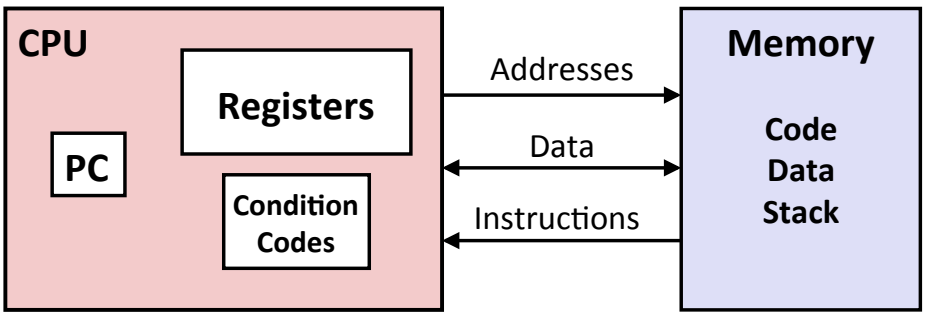
```
arith:
  leaq    (%rdi,%rsi), %rax
  addq    %rdx, %rax
  leaq    (%rsi,%rsi,2), %rdx
  salq    $4, %rdx
  leaq    4(%rdi,%rdx), %rcx
  imulq   %rcx, %rax
  ret
```

# Procedure Call Registers

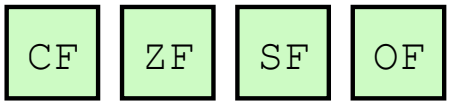
| | | | | | |
|---|---|---|---|---|---|
| Return | %rax | %eax | %r8 | %r8d | Arg 5 |
| | %rbx | %ebx | %r9 | %r9d | Arg 6 |
| Arg 4 | %rcx | %ecx | %r10 | %r10d | |
| Arg 3 | %rdx | %edx | %r11 | %r11d | |
| Arg 2 | %rsi | %esi | %r12 | %r12d | |
| Arg 1 | %rdi | %edi | %r13 | %r13d | |
| Stack ptr | %rsp | %esp | %r14 | %r14d | |
| | %rbp | %ebp | %r15 | %r15d | |



# Assembly View of the Machine



# Condition Codes




Condition codes

CF: Carry flag (for unsigned)

ZF: Zero flag

SF: Sign flag (for signed)

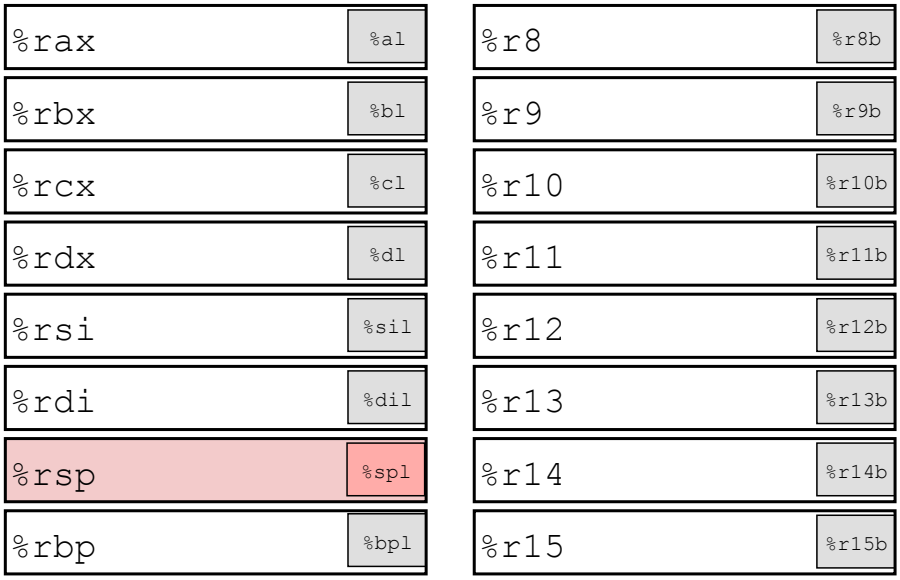
OF: Overflow flag (for signed)



# Reading Condition Codes

| 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) |

# Recap: Single-Byte Virtual Registers






# Example: Greater Than

```
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
```