RUTGERS

THE STATE UNIVERSITY OF NEW JERSEY

Computer Architecture (CS-211) Recitation-7

Song Wen

Topics

- Condition codes
- Loops
- Functions

* Some materials are collected and compiled from previous year's CS 211 lectures and TAs

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
- OF set if two's-complement (signed) overflow

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

Implicit set by arithmetic operations

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

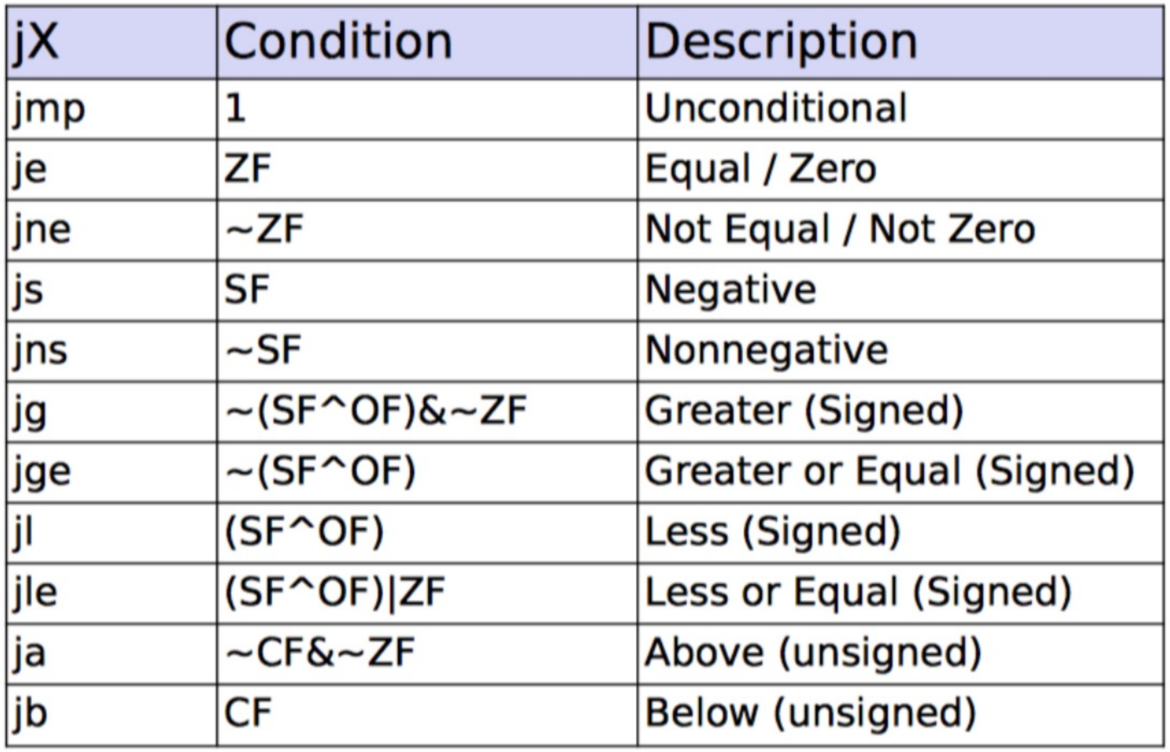
Explicit Setting by test Instruction

- testq src2, src1
- testq b, a like computing a&b without setting destination

- Sets condition codes based on value of src1&src2

- ZF set if a&b==0
- SF set if a&b<0

jX instructions: jump to different part of code depending on condition codes



jX	Condition	Description
jmp	1	Unconditional
je	ZF	Equal / Zero
jne	~ZF	Not Equal / Not Zero
js	SF	Negative
jns	~SF	Nonnegative
jg	~(SF^OF)&~ZF	Greater (Signed)
jge	~(SF^OF)	Greater or Equal (Signed)
jl	(SF^OF)	Less (Signed)
jle	(SF^OF)|ZF	Less or Equal (Signed)
ja	~CF&~ZF	Above (unsigned)
jb	CF	Below (unsigned)

Conditional Branch Example

```
long absdiff
  (long x, long y)
{
  long result;
  if (x > y)
    result = x-y;
  else
    result = y-x;
  return result;
}
```

```
absdiff:
   cmpq    %rsi, %rdi  # x:y
   jle     .L4
   movq    %rdi, %rax
   subq    %rsi, %rax
   ret
.L4:       # x <= y
   movq    %rsi, %rax
   subq    %rdi, %rax
   ret
```

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

Do-While Loop Example

Goto Version

```
long pcount_goto
  (unsigned long x) {
  long result = 0;
 loop:
  result += x & 0x1;
  x >>= 1;
  if(x) goto loop;
  return result;
}
```

Register	Use(s)
%rdi	Argument **x**
%rax	result

```
    movl    $0, %eax     #  result = 0
 .L2:                    # loop:
    movq    %rdi, %rdx
    andl    $1, %edx     #  t = x & 0x1
    addq    %rdx, %rax   #  result += t
    shrq    %rdi         #  x >>= 1
    jne     .L2          #  if (x) goto loop
    ret
```

General "While" Translation

While version

```
while (Test)
  Body
```

```
  goto test;
loop:
  Body
test:
  if (Test)
    goto loop;
done:
```

Function: Procedure Control Flow

- Use stack to support function call and return

- Procedure call: call label
  - Push return address on stack
  - Jump to label

- Return address:
  - Address of the next instruction right after call

- Procedure return: ret
  - Pop address from stack
  - Jump to address

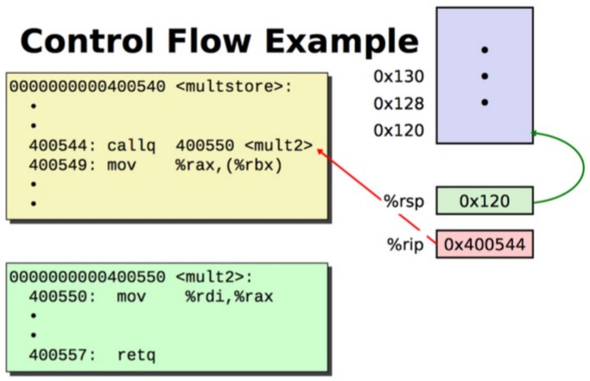
Control Flow Example

```
void multstore(long x,
   long y, long *dest)
{
    long t = mult2(x, y);
    *dest = t;
}
```

```
0000000000400540 <multstore>:
  push    %rbx              # Save %rbx
  mov     %rdx,%rbx         # Save dest
  callq   400550 <mult2># mult2(x,y)
  mov     %rax,(%rbx)       # Save at dest
  pop     %rbx              # Restore %rbx
  retq                      # Return
```

```
long mult2
  (long a, long b)
{
  long s = a * b;
  return s;
}
```

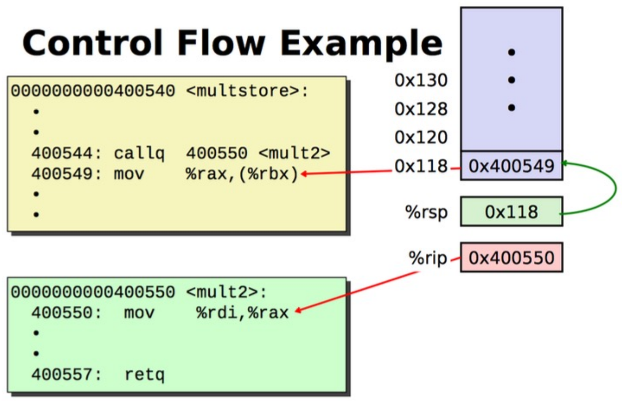
```
0000000000400550 <mult2>:
  mov     %rdi,%rax  # a
  imul    %rsi,%rax  # a * b
  retq               # Return
```



Control Flow Example

```
0000000000400540 <multstore>:
  •
  •
  400544: callq  400550 <mult2>
  400549: mov    %rax,(%rbx)
  •
  •
```

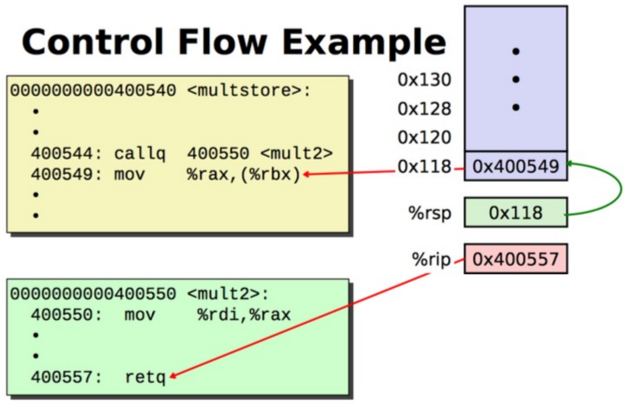
```
0000000000400550 <mult2>:
  400550:  mov    %rdi,%rax
  •
  •
  400557:  retq
```



Control Flow Example

```
0000000000400540 <multstore>:
  •
  •
  400544: callq  400550 <mult2>
  400549: mov    %rax,(%rbx)
  •
  •
```

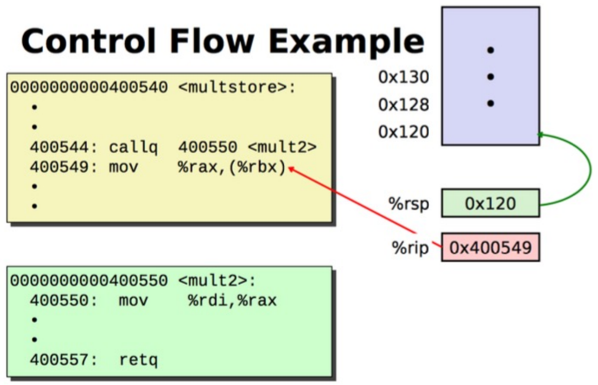
```
0000000000400550 <mult2>:
  400550:  mov    %rdi,%rax
  •
  •
  400557:  retq
```



Control Flow Example

```
0000000000400540 <multstore>:
  •
  •
  400544: callq  400550 <mult2>
  400549: mov    %rax,(%rbx)
  •
  •
```

```
0000000000400550 <mult2>:
  400550:  mov    %rdi,%rax
  •
  •
  400557:  retq
```



Control Flow Example

```
0000000000400540 <multstore>:
  •
  •
  400544: callq  400550 <mult2>
  400549: mov    %rax,(%rbx)
  •
  •
```

```
0000000000400550 <mult2>:
  400550:  mov    %rdi,%rax
  •
  •
  400557:  retq
```

Q&A

Thanks!