

Reading Condition Codes

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

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
```

Goto

```
#include <stdio.h>

int main() {

    int a = 10;

    LABEL:do {

        if (a == 15) {
            /* skip the iteration */
            a = a + 1;
            goto LABEL;
        }

        printf("value of a: %d\n", a);
        a++;

    } while (a < 20);

    return 0;
}
```

Jumping

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)

Example: absdiff

```
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

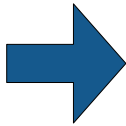
absdiff with Goto

```
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
```

```
long absdiff_j
(long x, long y)
{
    long result;
    int ntest = x <= y;
    if (ntest) goto Else;
    result = x-y;
    goto Done;
Else:
    result = y-x;
Done:
    return result;
}
```

Conditional to Goto

```
if (test-expr)
  then-cmd
else
  else-cmd
...
```

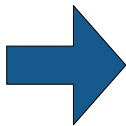


```
t = test-expr
if (!t) goto false;
then-cmd
goto done;

false:
  else-cmd

done:
  ...
```

```
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
```

C: Input with scanf

```
int things_read;

int i;      // declared but uninitialized
char c;

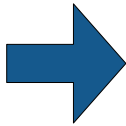
// read an int, store at address &i
things_read = scanf("%d", &i);

// read an int and a char, store at addresses &i and &c
things_read = scanf("%d %c", &i, &c);
```

```
int i;      // declared but uninitialized
...
scanf("%d", i); // DANGER!!!
```

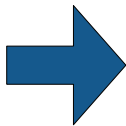
Conditional to Goto

```
if (test-expr)
  then-cmd
else
  else-cmd
...
```



```
t = test-expr
if (!t) goto false;
then-cmd
goto done;
false:
  else-cmd
done:
  ...
```

```
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
```