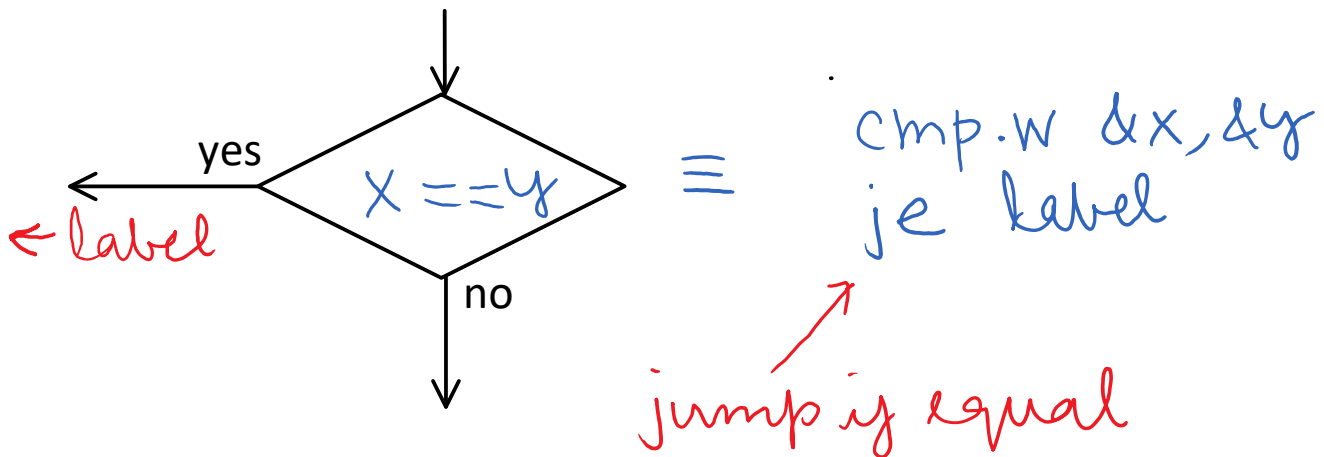
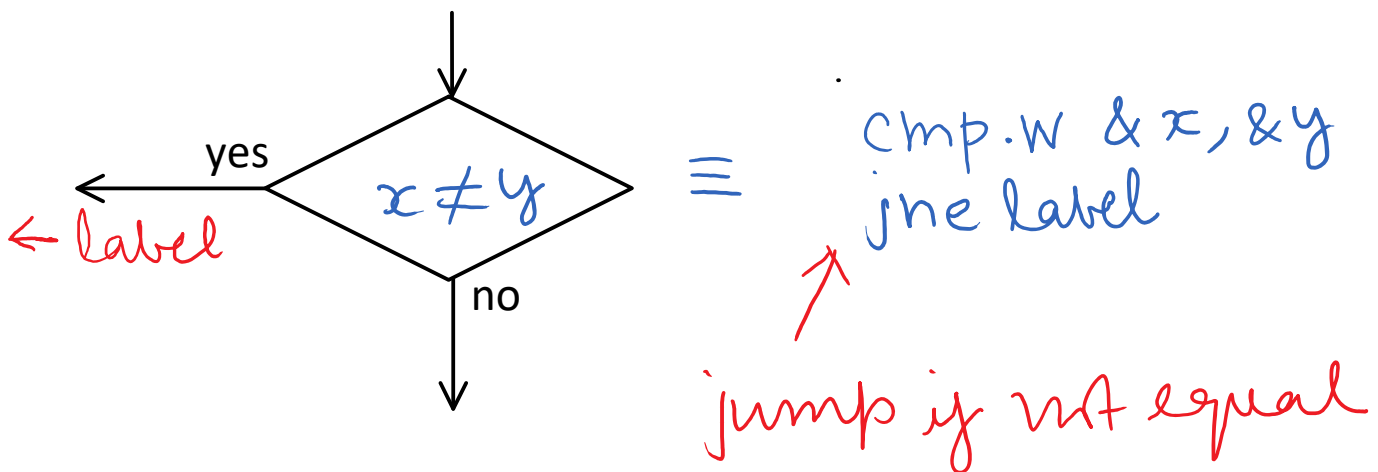


JE : Jump if equal



JNE : Jump if not equal



Conditions you can handle
by now

$x == y$	je
$x \neq y$	jne
$x < y$	jl
$x \geq y$	jge
$x \leq y$	jge
$x > y$	jle

BIT.b, BIT.w : Test for bit set

used to test if a bit is set in a byte (BIT.b) or a word (BIT.w)

bit.b $\times$ 00001000b, &X

↑↑↑↑

 3 2 1 0

Sets the carry bit in the status register if the 3rd bit of X is set, otherwise it resets the carry bit.

bit.b $\times$ 00001000b, &X

jc label ← jump to label if the 3rd bit of X is set

... ← come here if the 3rd bit of X is not set

jump if carry

mov.b ~~x~~ 00001000b, &b3

↑
3

mov.b ~~x~~ 01011011b, &x

↑
3

bit.b &b3, &x

carry status bit is set
since the 3rd bit of x
is set

mov.b ~~x~~ 11110101b, &x

↑
3

bit.b &b3, &x

carry status bit is reset
since the 3rd bit of x
is not set

Recall that every program starts with:

```
.cdecls C,LIST,"msp430.h"
```

header file

In the header file you will find the following ~~x~~ defines (think of these as symbolic constants)

```
#define BIT0
#define BIT1
#define BIT2
#define BIT3
#define BIT4
#define BIT5
#define BIT6
#define BIT7
#define BIT8
#define BIT9
#define BITA
#define BITB
#define BITC
#define BITD
#define BITE
#define BITF
```

```
(0x0001)
(0x0002)
(0x0004)
(0x0008)
(0x0010)
(0x0020)
(0x0040)
(0x0080)
(0x0100)
(0x0200)
(0x0400)
(0x0800)
(0x1000)
(0x2000)
(0x4000)
(0x8000)
```

0th bit set
1st bit set
2nd bit set
3rd bit set
etc.

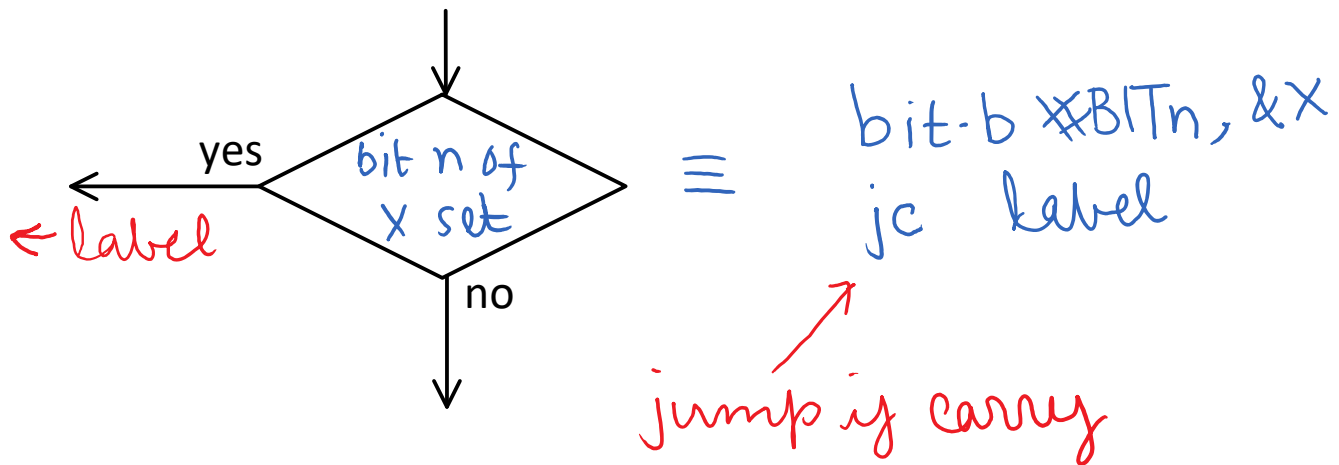
examples: note immediate mode

bit.b ~~x~~BIT3, &x

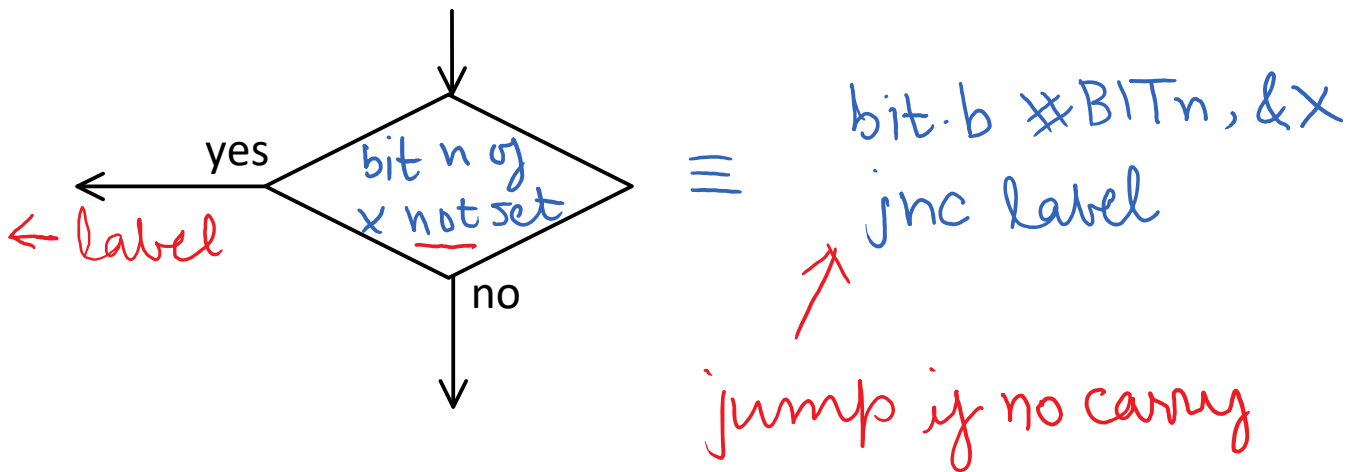
bit.b ~~x~~BIT7, &y

etc

Use JC to test if a bit is set

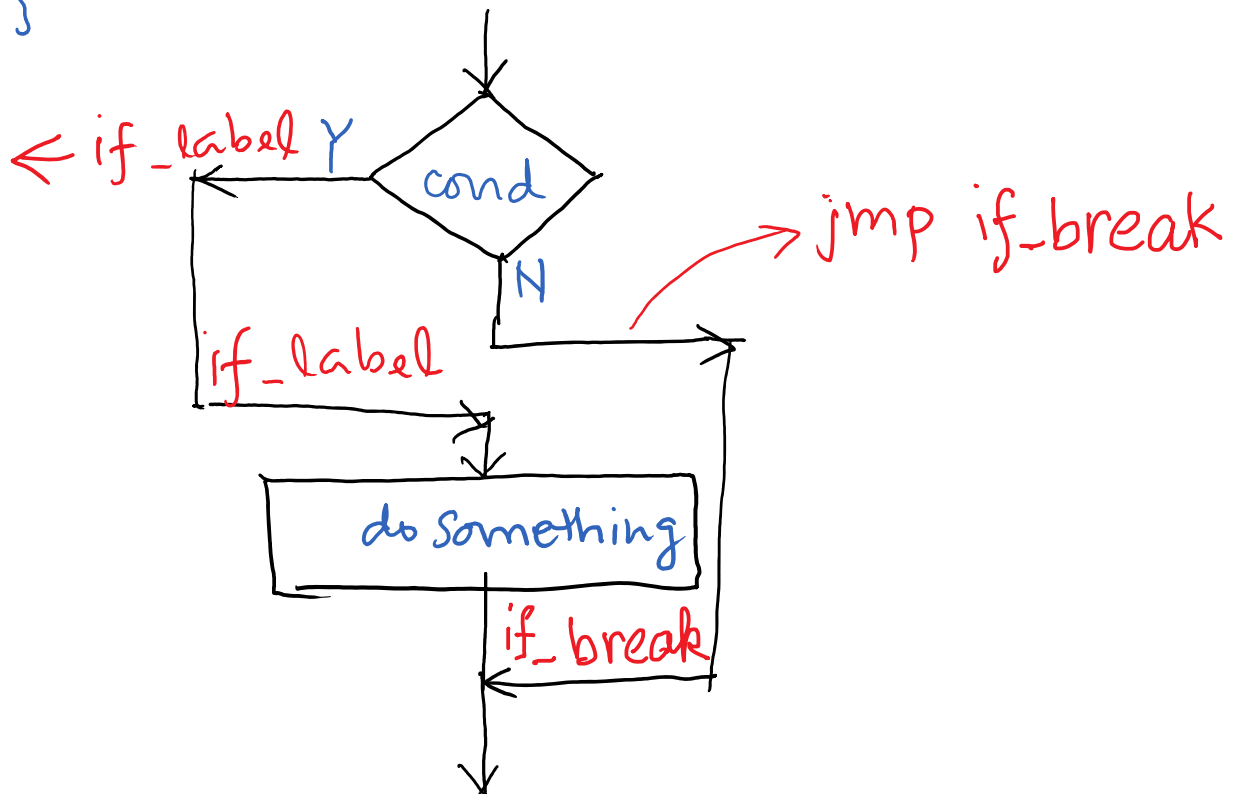


Use JNC to test if a bit is not set



if structure

```
if (cond)
{
    do something
}
```



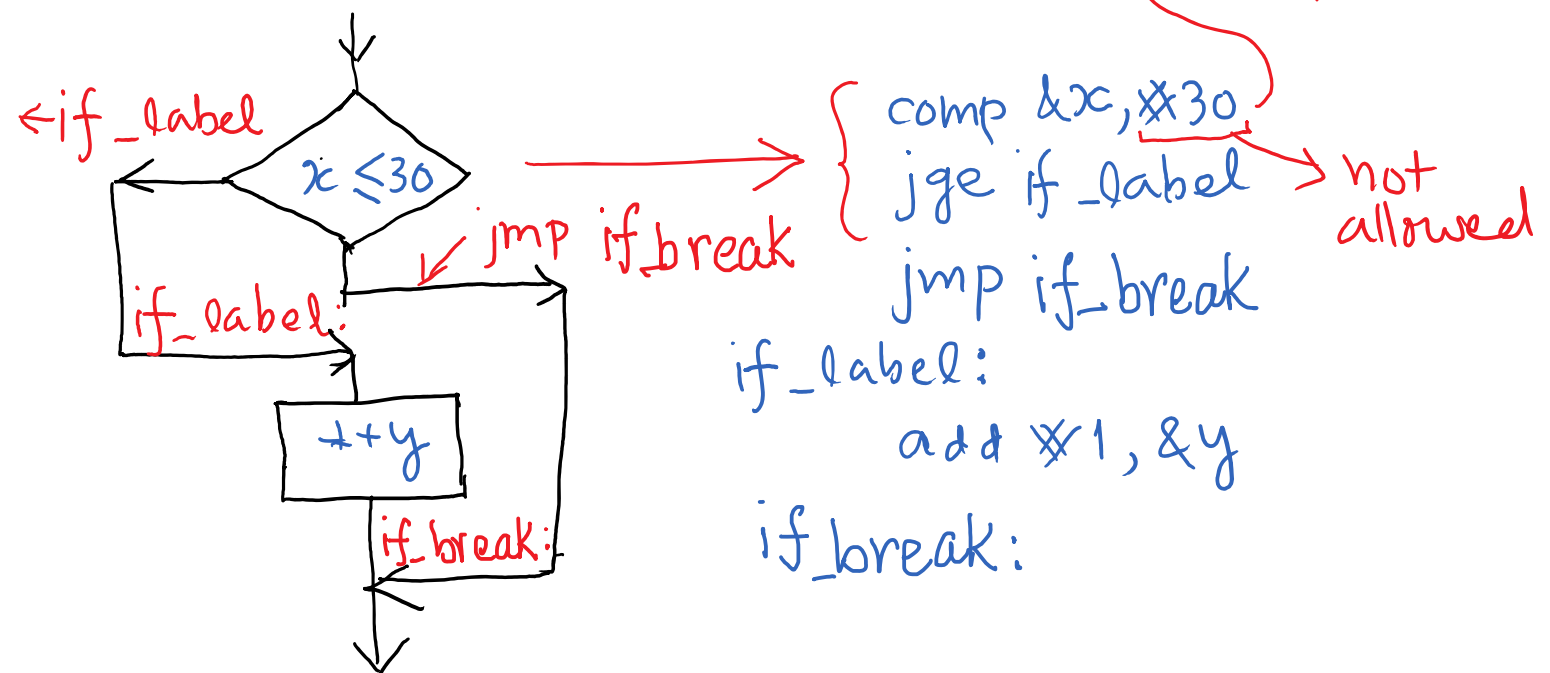
example :

```

if ( x ≤ 30 )
{
    ++y
}

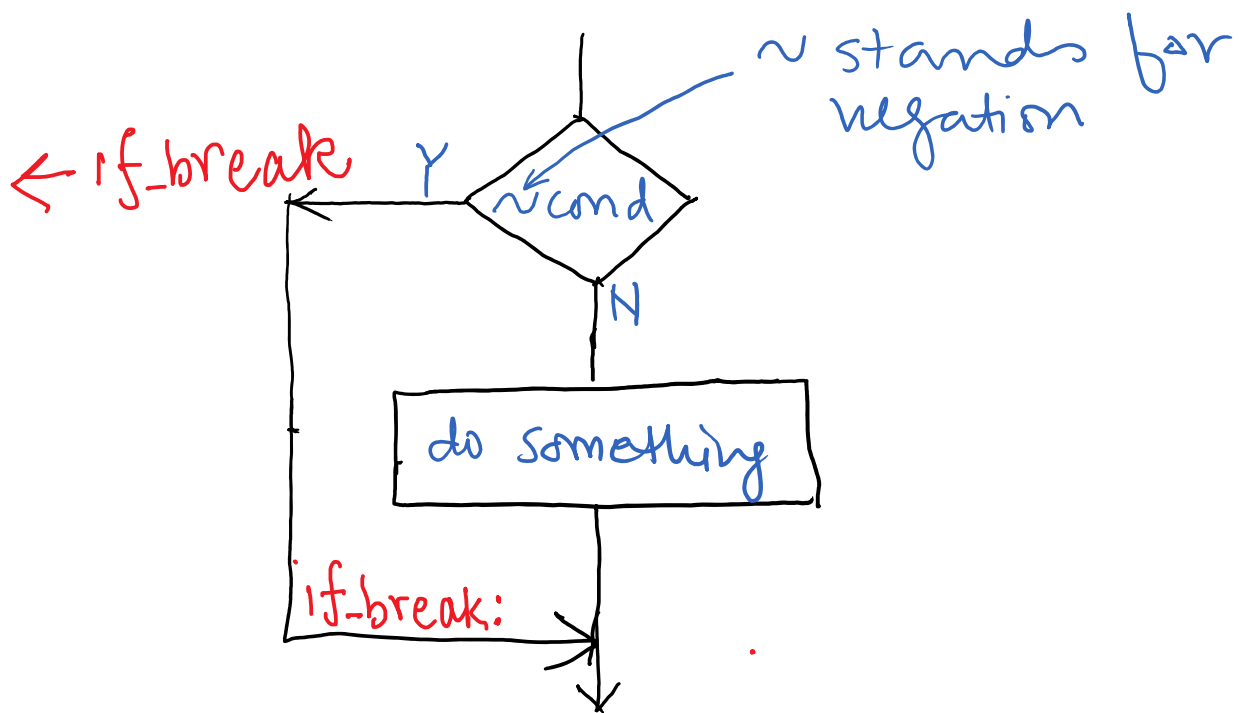
```

$\rightarrow$ mov $\$30$, R5
 $\rightarrow$ cmp &x, R5



if structure (alternate implementation)

```
if (cond)
{
    do something
}
```



Example:

$$\sim(x < 0) = x \geq 0$$

$$\sim(y \geq 0) = y < 0$$

example :

```
if ( x ≤ 30 )
```

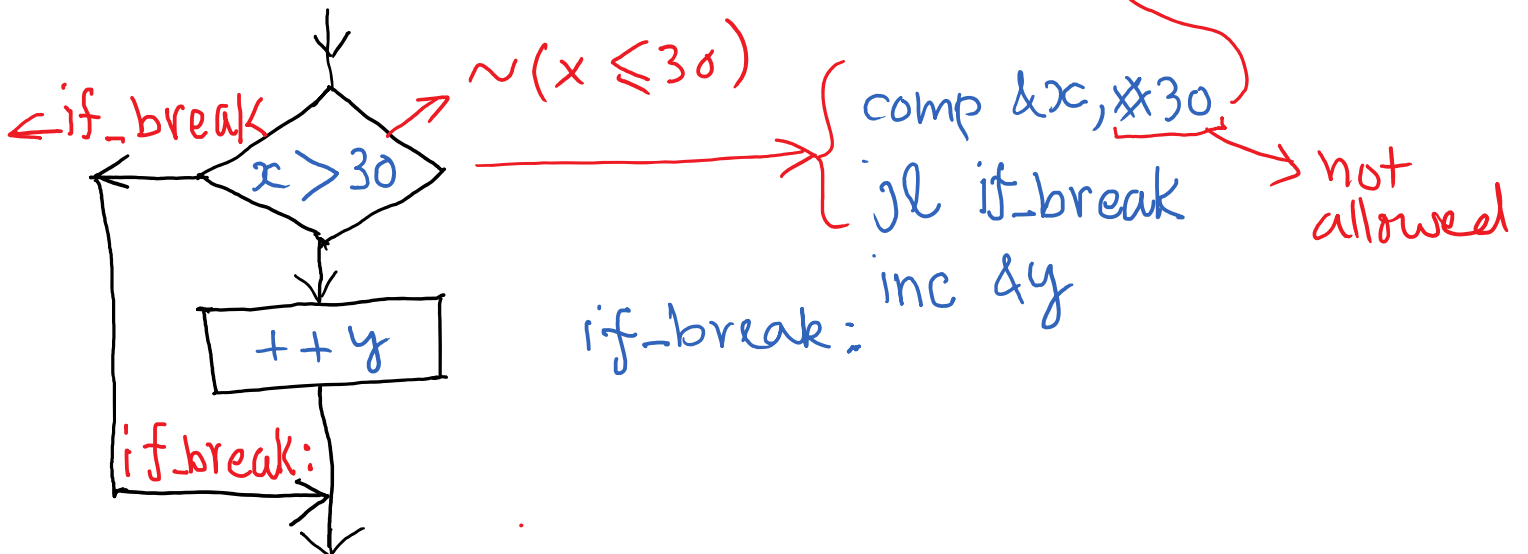
```
{
```

```
  ++y
```

```
}
```

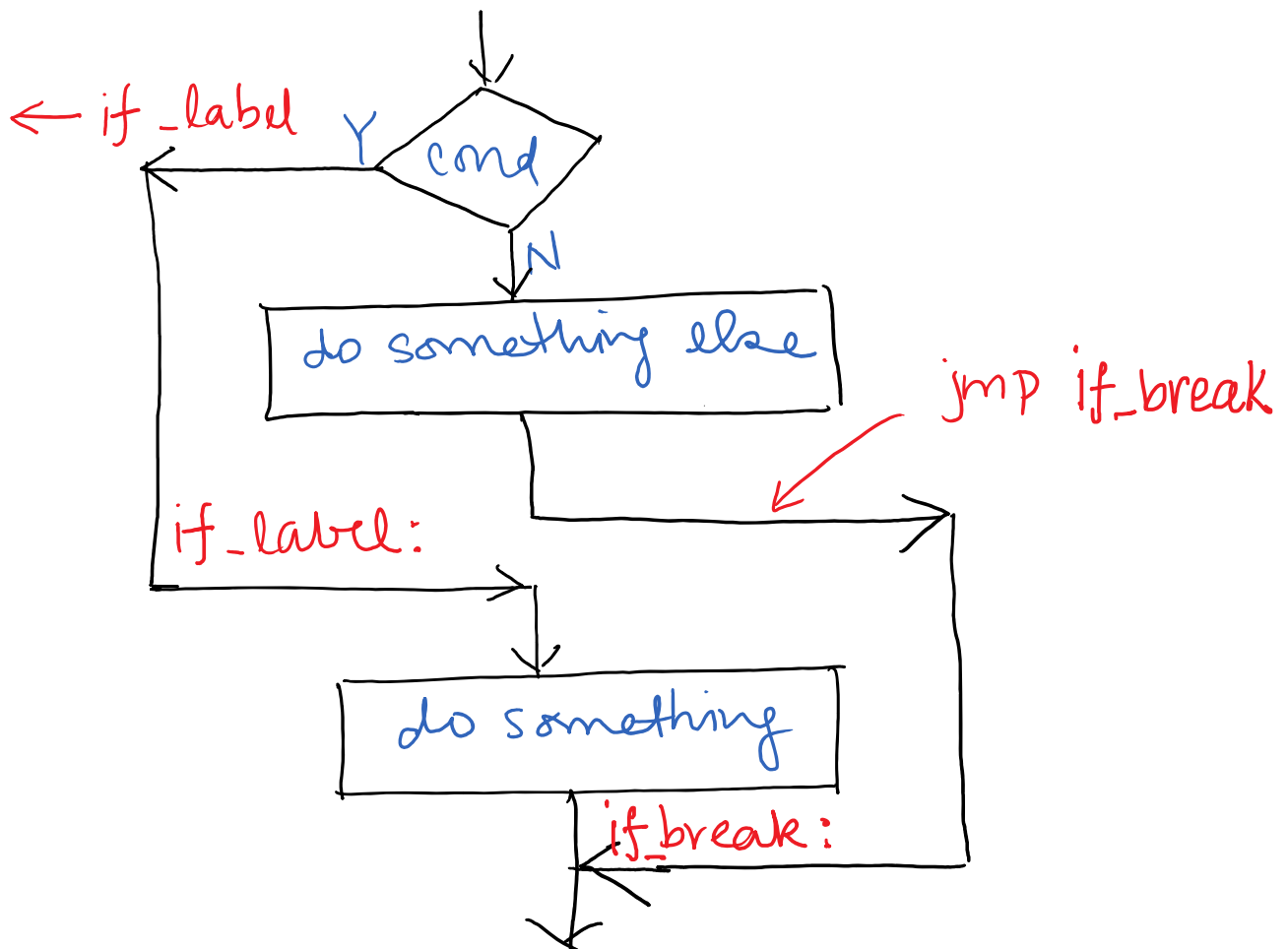
 $\sim(x \leq 30) = x > 30$

```
{ mov $30, R5
  cmp &x, R5
```

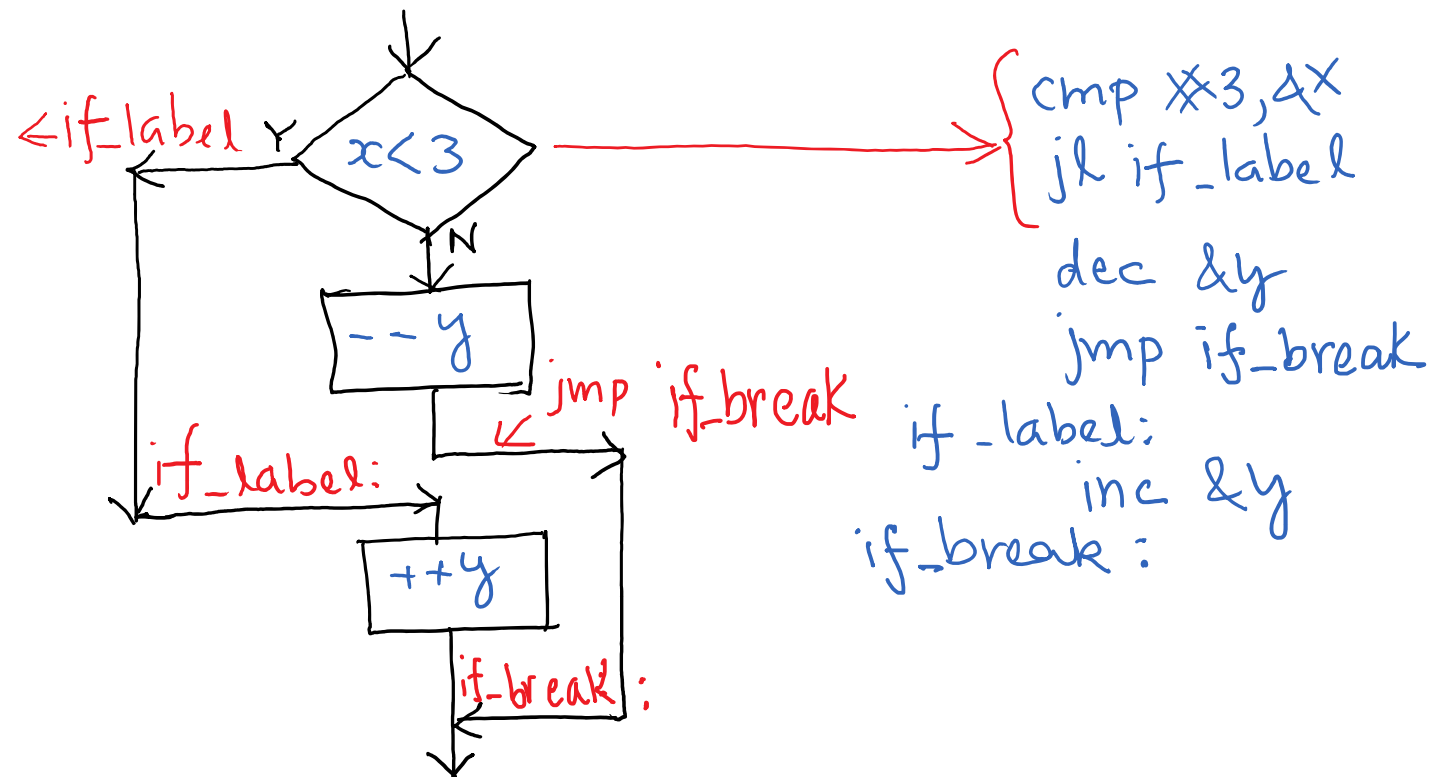


if - else structure

```
if (cond)
{
    do something
}
else
{
    do something else
}
```



```
if (x < 3)
{
    ++y
}
else
{
    --y
}
```



if - else structure
(alternate implementation)

if (cond)

{

do something

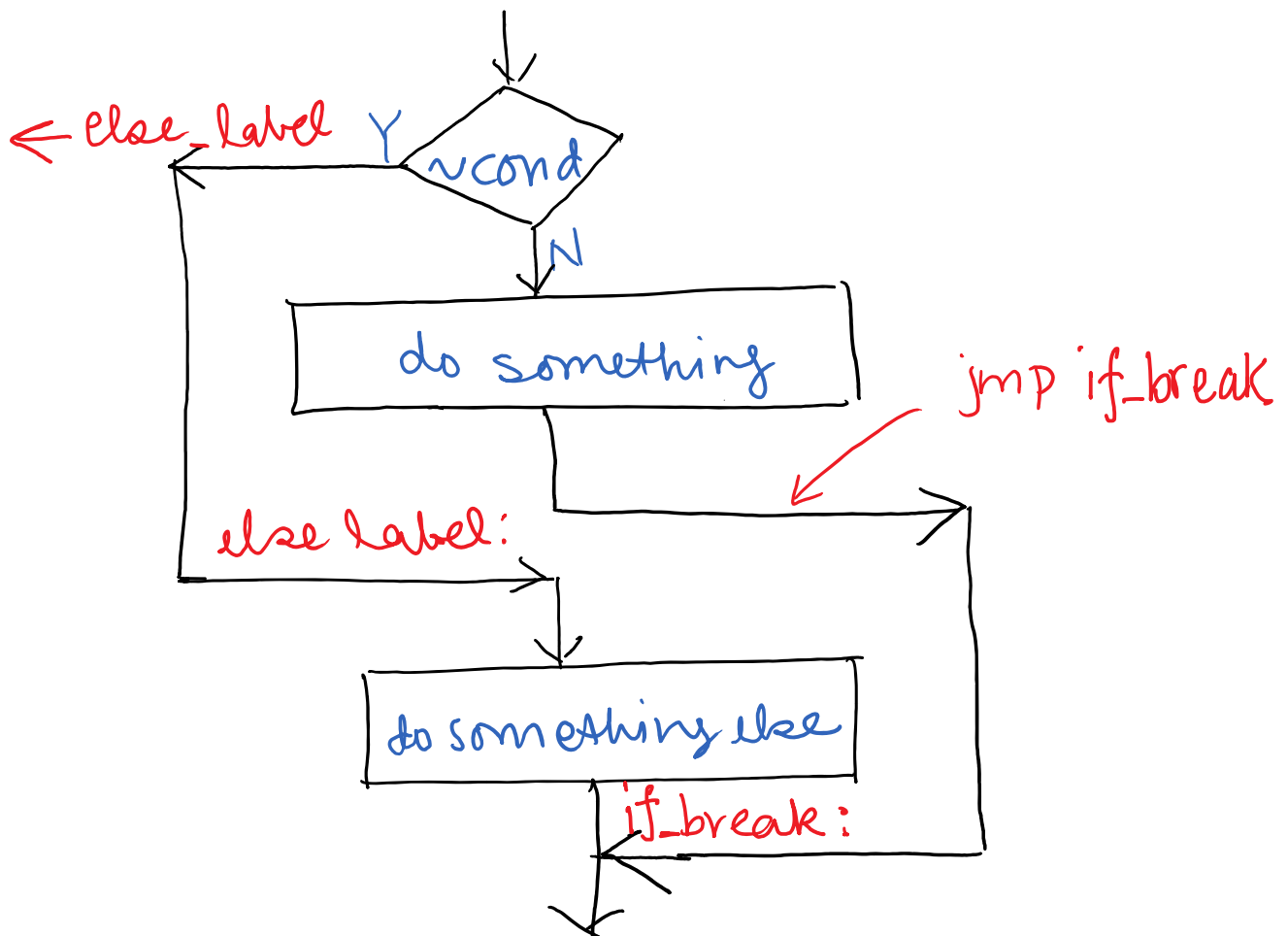
}

else

{

do something else

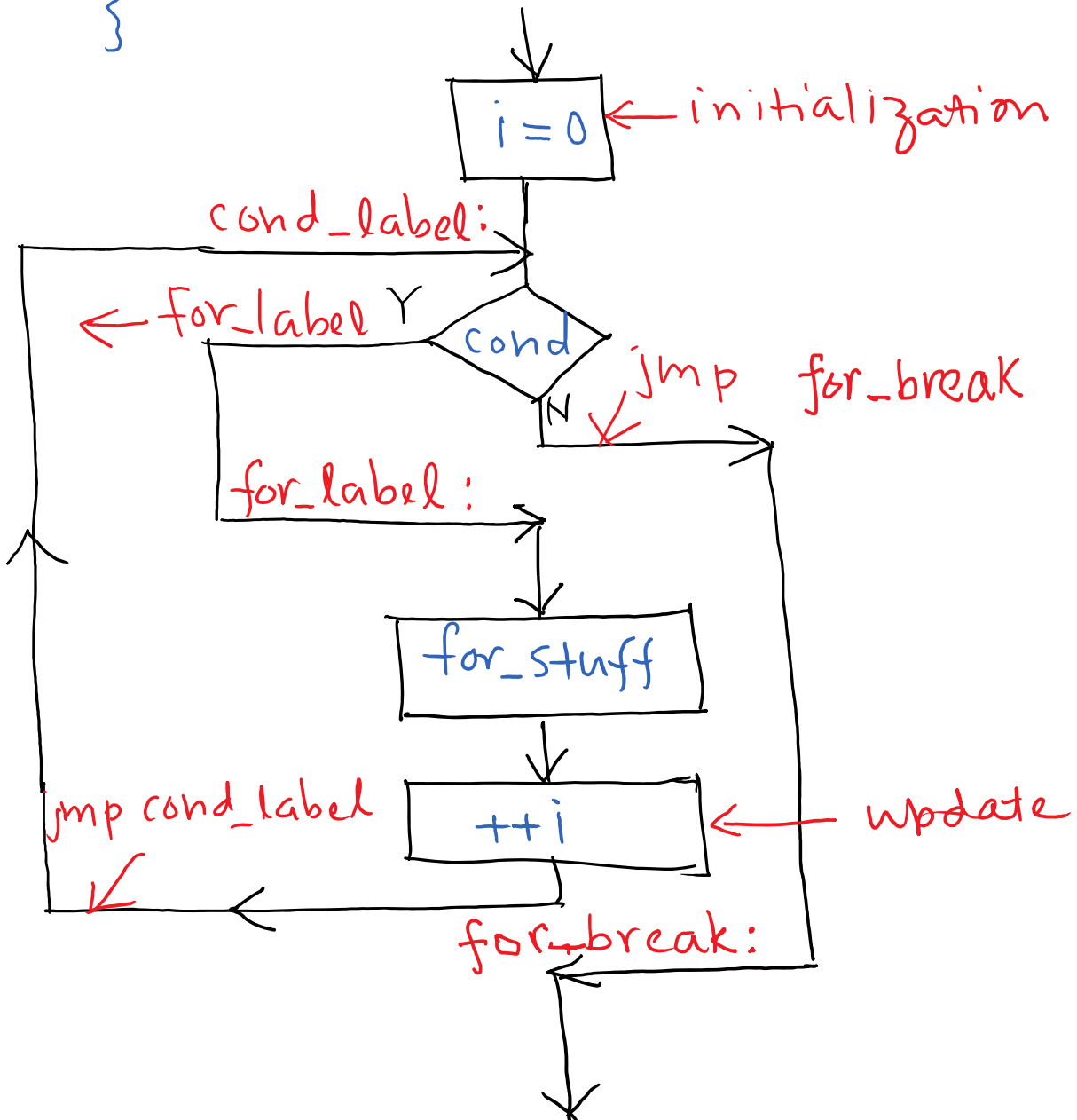
}



initialization LOOPS

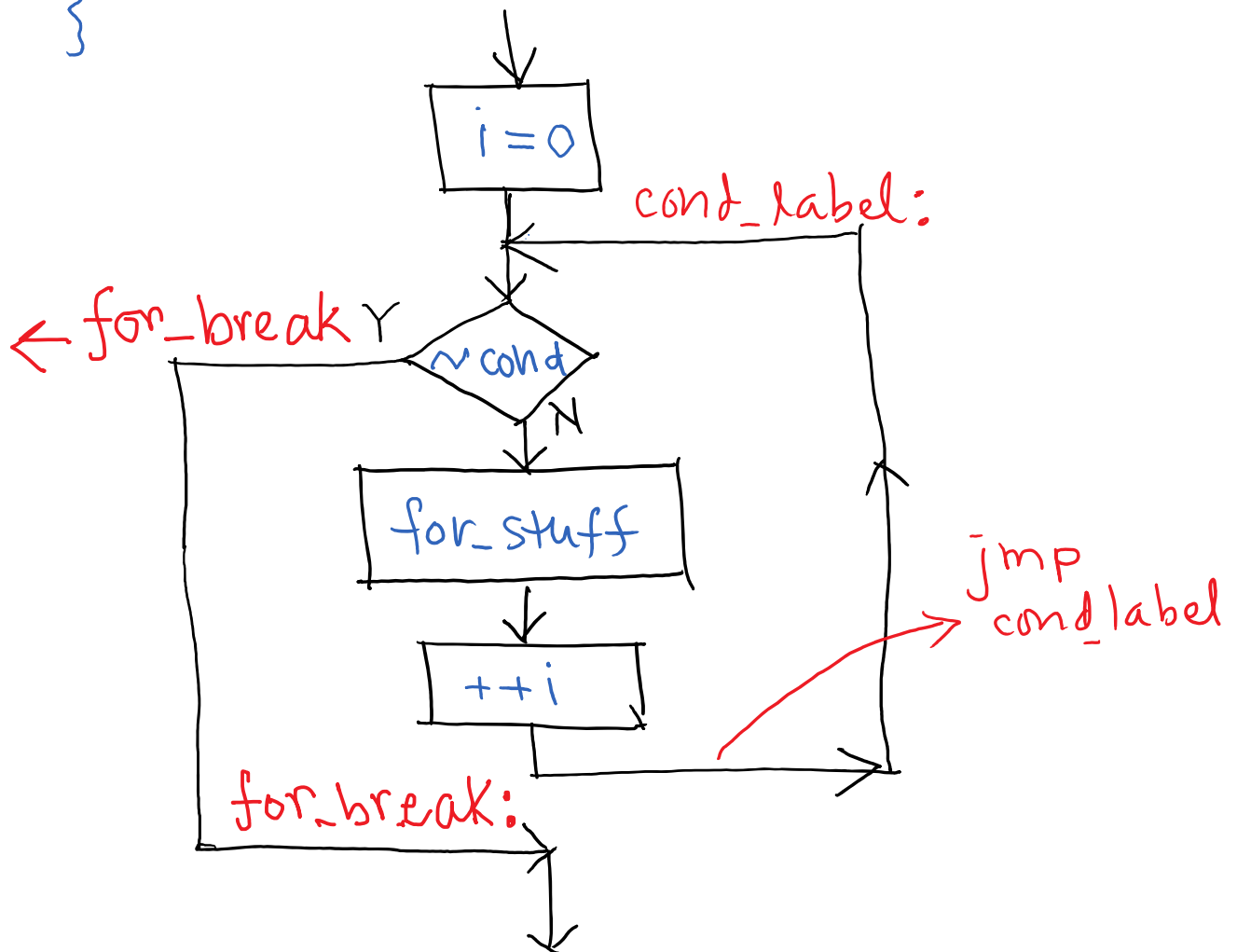
for-loop condition update

```
for ( i = 0 ; cond ; ++i )
{
    for_stuff
}
```



for-loop (alternate implementation)

```
for (i = 0; cond; ++i)
{
    for_stuff
}
```



example

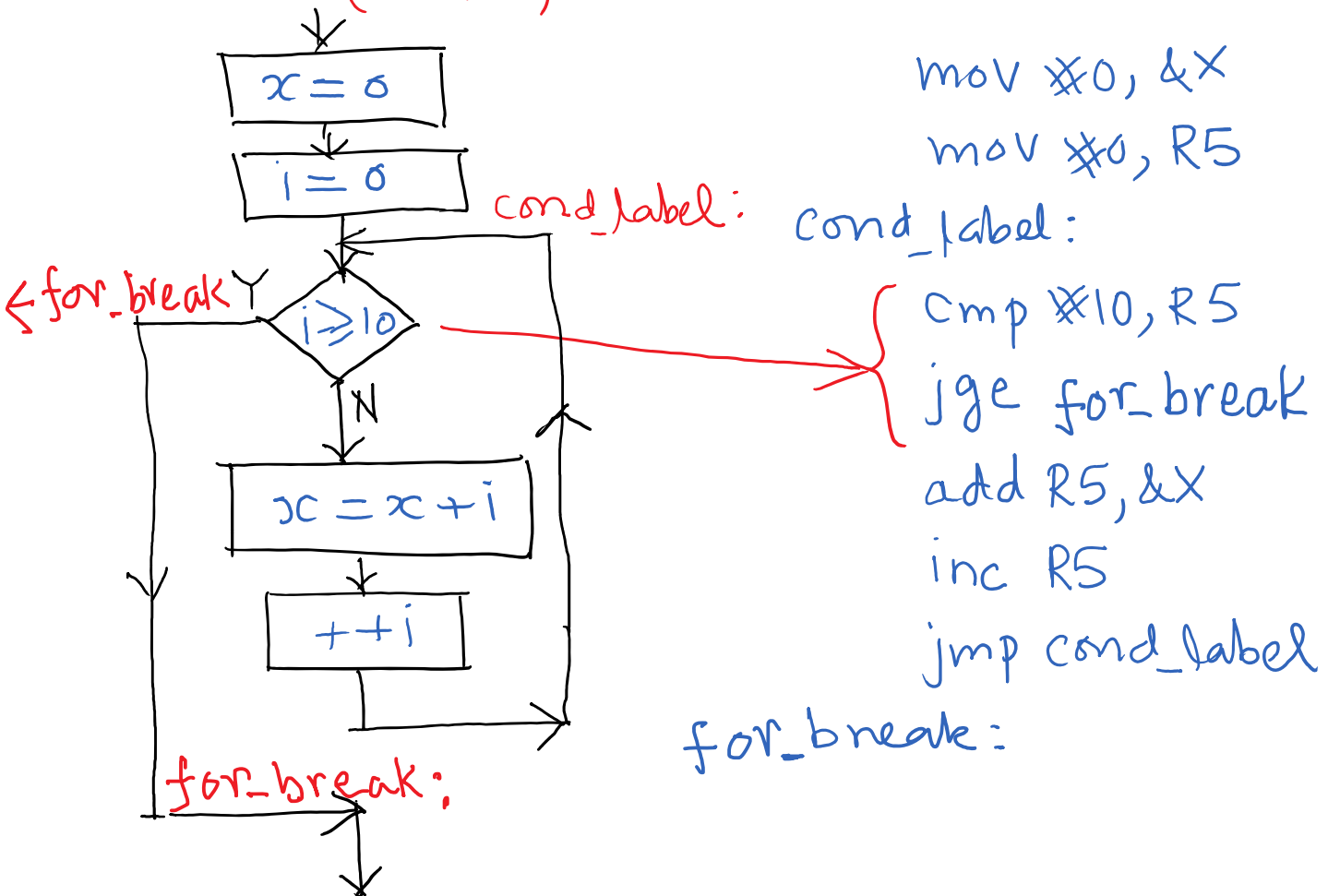
```

x = 0
for (i = 0; i < 10; ++i)
{
    x = x + i
}

```

let us use the alternate implementation

$$\sim(i < 10) = i \geq 10$$



Program "For Loop1"

```
.data
x:    .space    2

.text
      mov      #0, &x
      mov      #0, R5
cond_label:
      cmp      #10, R5
      jge      for_break

      add      R5, &x
      inc      R5

      jmp      cond_label
for_break:

loop:  jmp      loop
```