Sentencias de control y bucles de C a CISCA

1 Traducción de sentencias de control

- 1.1 if
- 1.2 if-else

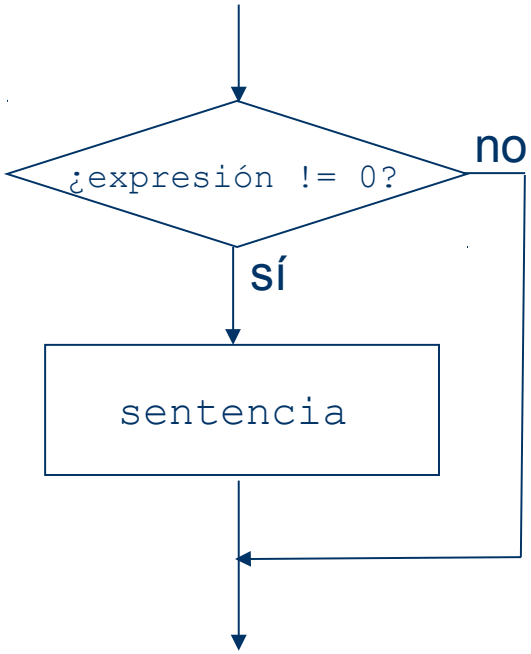
2 Traducción de bucles

- 2.1 while
- 2.2 do-while
- 2.3 for

`if` en alto nivel

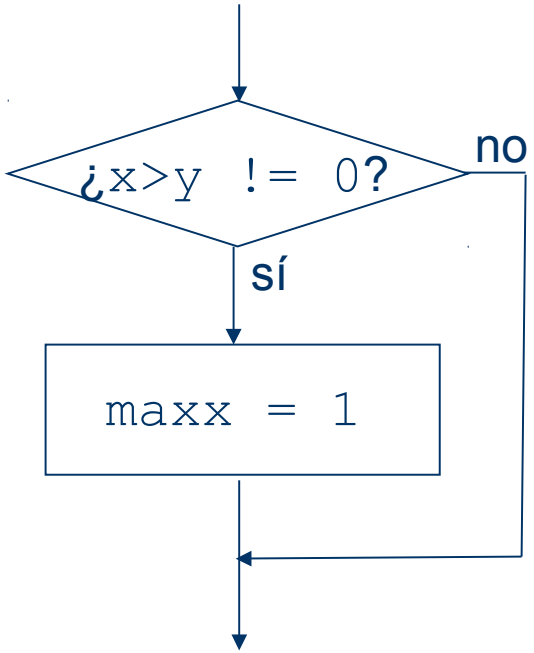
C

```
if (expresión) sentencia;
```



Ejemplo C

```
if (x>y) maxx = 1;
```



Traducción del `if` de C a CISCA

C

```
if (x>y) maxx = 1;
```

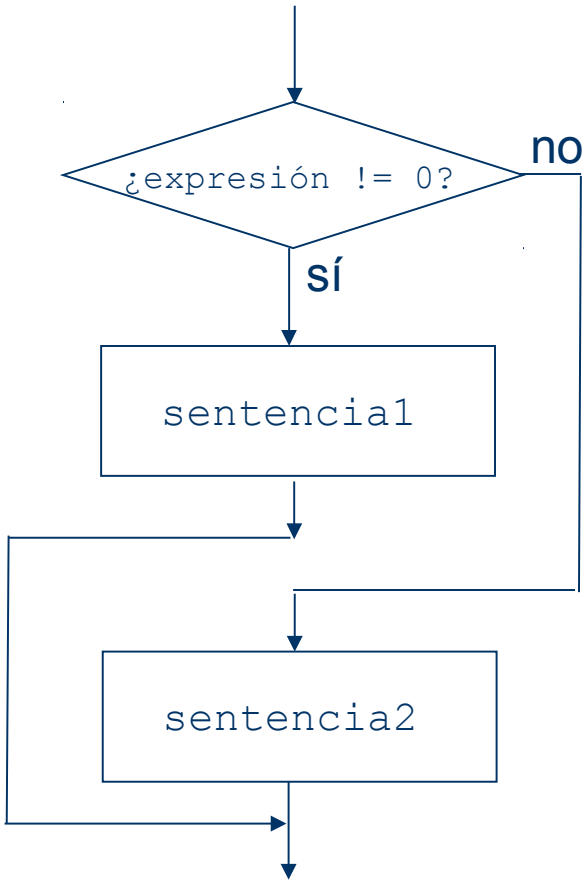
CISCA

```
if:     mov  r1, [x]
        cmp  r1, [y]
        jg   set
        jmp  endif
set:    mov  [maxx], 1
endif:
------------------------------
if:   mov r1, [x]
      cmp r1, [y]
      jle endif ;condición contraria
      mov [maxx], 1
endif:
```

`if-else` en alto nivel

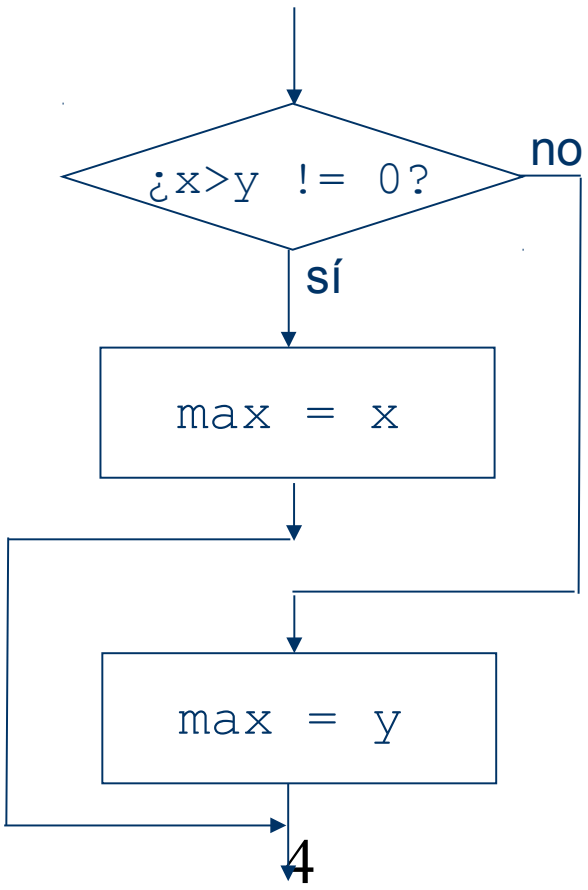
C

```
if (expresión) sentencia1;
    else sentencia2;
```



Ejemplo C

```
if (x>y) max = x;
    else max = y;
```



Traducción del `if-else` de C a CISCA

C

```
if (x>y) max = x;
   else max = y;
```

CISCA

```
if:   mov r1, [x]
      mov r2, [y]
      cmp r1, r2
      jg true
      jmp else
true: mov [max], r1
      jmp  endif
else: mov  [max], r2
endif:
```

Traducción del `if-else` de C a CISCA

C

```
if (x>y) max = x;
   else max = y;
```

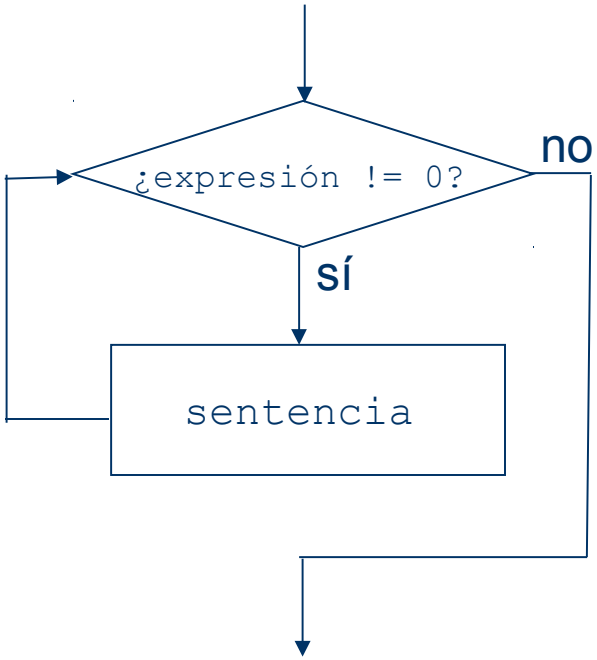
CISCA

```
if:    mov r1, [x]
       mov r2, [y]
       cmp r1, r2
       jle else ;condición contraria
       mov [max], r1
       jmp endif
else:  mov [max], r2
endif:
```

`while` en alto nivel

C

```
while (expresión)
    sentencia;
```



Ejemplo C

```
while (num > 0){
    i = i*num;
    num = num - 1;
}
resul = i;
```

Con más de una sentencia:

```
{
   sent1
   sent2
}
```

Traducción del `while` de C a CISCA

C

```
while(num > 0){
    i = i*num;
    num = num - 1;
}
resul = i;
```

CISCA

```
        mov r1, [i]
        mov r2, [num]
while:  cmp r2, 0
        jg cont
        jmp end_w
cont:   mul r1, r2
        sub r2, 1
        jmp while
end_w:  mov [resul], r1
        mov [i], r1
```

Traducción del `while` de C a CISCA

C

```
while(num > 0){
    i = i*num;
    num = num - 1;
}
resul = i;
```

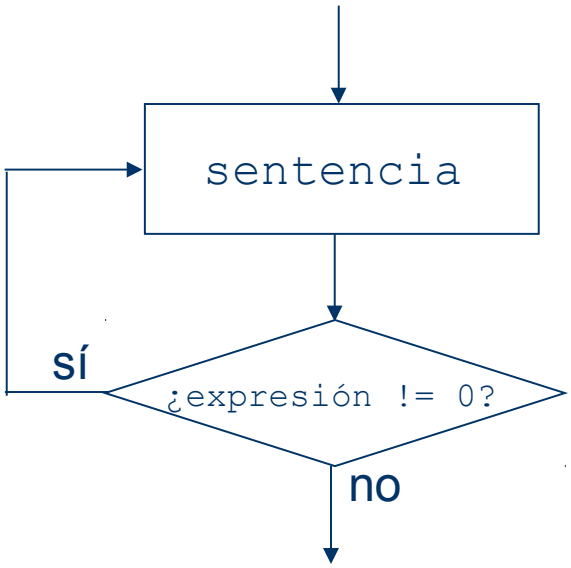
CISCA

```
        mov r1, [i]
        mov r2, [num]
while:  cmp r2, 0
        jle end_w ;condición contrària
        mul r1, r2
        sub r2, 1
        jmp while
end_w:  mov [resul], r1
        mov [i], r1
```

`do-while` en alto nivel

C

```
do
    sentencia;
while (expresión);
```



Ejemplo C

```
do {
    sum = sum + i;
    i = i + 1
} while (i <= valor);
```

Traducción del `do-while` de C a CISCA

C

```
do {
    sum = sum + i;
    i = i + 1
} while (i <= valor);
```

CISCA

```
        mov     r1, [i]

do:     add     [sum], r1
        add     r1, 1
        cmp     r1, [valor]
        jle     do
        mov     [i], r1
```

`for` en alto nivel

C

```
for (expr1; expr2; expr3)
        sentencia;
```

Es equivalente a:

```
expr1;
while (expr2) {
   sentencia;
   expr3;
}
```

Ejemplo C

```
for (i=0; i<=valor; i++)
   sum = sum + i;
```

Traducción del `for` de C a CISCA

C

```
for (i=0; i<=valor; i++)
    sum = sum + i;
```

CISCA

```
      mov r1, 0
for:  cmp r1, [valor]
      jle incr
      jmp endf
incr: add [sum], r1
      add r1, 1
      jmp for
endf: mov [i], r1
```

Traducción del `for` de C a CISCA

C

```
for (i=0; i<=valor; i++)
    sum = sum + i;
```

CISCA

```
        mov r1, 0
for:    cmp r1, [valor]
        jg endf ;condición contraria
        add [sum], r1
        add r1, 1
        jmp for
endf:   mov [i], r1
```