Repasemos

¿Que vimos la clase pasada?

Lógica

Lógica

- Representamos valores de verdad con cero y uno

Lógica

- Representamos valores de verdad con cero y uno
  - Cero es falso

Lógica

- Representamos valores de verdad con cero y uno
  - Cero es falso
  - Uno es verdadero

Lógica

- Representamos valores de verdad con cero y uno
  - Cero es falso
  - Uno es verdadero
- Operaciones lógicas

Lógica

- Representamos valores de verdad con cero y uno
  - Cero es falso
  - Uno es verdadero
- Operaciones lógicas
  - Se aplican sobre las cadenas binarias

Lógica

- Representamos valores de verdad con cero y uno
  - Cero es falso
  - Uno es verdadero
- Operaciones lógicas
  - Se aplican sobre las cadenas binarias
  - Actúan independientemente sobre cada bit de la cadena

Operaciones lógicas

Operaciones lógicas

AND

Operaciones lógicas

AND

OR

Operaciones lógicas

AND

OR

NOT

Operaciones lógicas

AND

OR

NOT

XOR

AND

- “Y” lógico entre los bits de dos cadenas

AND

- “Y” lógico entre los bits de dos cadenas
  - Solo uno si ambos bits son uno

AND

- “Y” lógico entre los bits de dos cadenas
  - Solo uno si ambos bits son uno

AND 11001101
01000011

AND

- “Y” lógico entre los bits de dos cadenas
  - Solo uno si ambos bits son uno

```
   11001101
AND 01000011
-----------
   01000001
```

AND

- “Y” lógico entre los bits de dos cadenas
  - Solo uno si ambos bits son uno

```
  11001101
AND
  01000011
----------
  01000001
```

OR

- “O” lógico entre los bits de dos cadenas

OR

- “O” lógico entre los bits de dos cadenas
  - Solo uno si por lo menos uno de los dos bits es uno

OR

- “O” lógico entre los bits de dos cadenas
  - Solo uno si por lo menos uno de los dos bits es uno

```
   11001101
OR 01000011
-----------
```

OR

- “O” lógico entre los bits de dos cadenas
  - Solo uno si por lo menos uno de los dos bits es uno

```
   11001101
OR 01000011
-----------
   11001111
```

OR

- “O” lógico entre los bits de dos cadenas
  - Solo uno si por lo menos uno de los dos bits es uno

```
  11001101
OR
  01000011
----------
  11001111
```

XOR

- “O” **exclusivo** entre los bits de dos cadenas

XOR

- “O” **exclusivo** entre los bits de dos cadenas
  - Uno si sólo si uno de los dos bits es uno

XOR

- “O” **exclusivo** entre los bits de dos cadenas
  - Uno si sólo si uno de los dos bits es uno

XOR 11001101
01000011

XOR

- “O” **exclusivo** entre los bits de dos cadenas
  - Uno si sólo si uno de los dos bits es uno

XOR

```
  11001101
  01000011
----------
  10001110
```

XOR

- “O” **exclusivo** entre los bits de dos cadenas
  - Uno si sólo si uno de los dos bits es uno

XOR

$$\begin{array}{r} 11001101 \\ \underline{01000011} \\ 10001110 \end{array}$$

NOT

- Negación bit a bit de una sola cadena

NOT

- Negación bit a bit de una sola cadena
  - Uno si es cero

NOT

- Negación bit a bit de una sola cadena
  - Uno si es cero
  - Cero si es uno

NOT

- Negación bit a bit de una sola cadena
  - Uno si es cero
  - Cero si es uno

NOT 11001101

NOT

- Negación bit a bit de una sola cadena
  - Uno si es cero
  - Cero si es uno

NOT 11001101

00110010

Máscaras

Máscaras

- Cadenas binarias elegidas especialmente que se combinan con otras mediante operaciones lógicas

Máscaras

- Cadenas binarias elegidas especialmente que se combinan con otras mediante operaciones lógicas
    - Nos permiten analizar el contenido de la otra cadena

Máscaras

¿Qué pongo en la máscara para dejar pasar un bit si la operación lógica es un AND?

Máscaras

¿Qué pongo en la máscara para dejar pasar un bit si la operación lógica es un AND?

1

Máscaras

¿Qué pongo en la máscara para dejar pasar un bit si la operación lógica es un AND?

1

```
  XXXXXXXX
AND 01000011
```

Máscaras

¿Qué pongo en la máscara para dejar pasar un bit si la operación lógica es un AND?

1

Máscaras

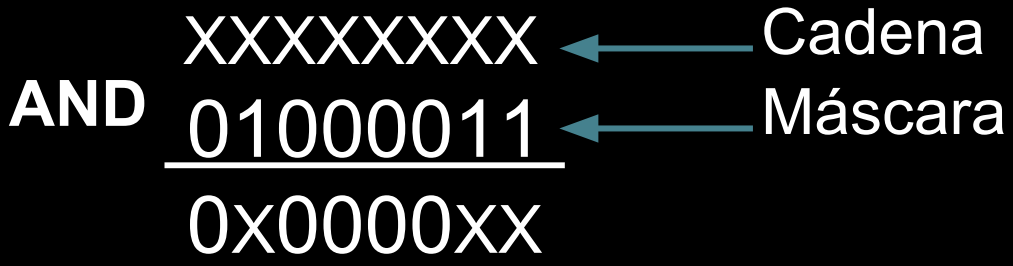
¿Qué pongo en la máscara para dejar pasar un bit si la operación lógica es un AND?

1

Máscaras

¿Qué pongo en la máscara para dejar pasar un bit si la operación lógica es un AND?

1



Máscaras

¿Qué pongo en la máscara para dejar pasar un bit si la operación lógica es un AND?

1

Máscaras

¿Qué pongo en la máscara para dejar pasar un bit si la operación lógica es un OR?

Máscaras

¿Qué pongo en la máscara para dejar pasar un bit si la operación lógica es un OR?

0

Máscaras

¿Qué pongo en la máscara para dejar pasar un bit si la operación lógica es un OR?

0

XXXXXXXX

OR 01000011

Máscaras

¿Qué pongo en la máscara para dejar pasar un bit si la operación lógica es un OR?

0

XXXXXXXX ← Cadena

OR 01000011

Máscaras

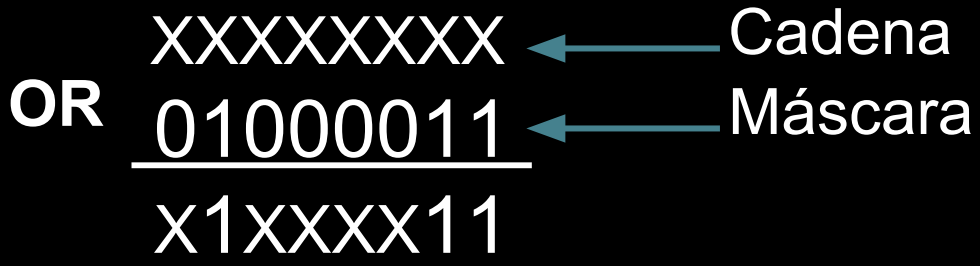
¿Qué pongo en la máscara para dejar pasar un bit si la operación lógica es un OR?

0

Máscaras

¿Qué pongo en la máscara para dejar pasar un bit si la operación lógica es un OR?

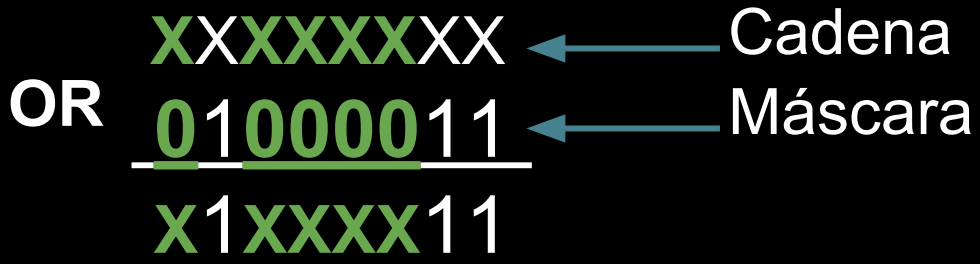
0



Máscaras

¿Qué pongo en la máscara para dejar pasar un bit si la operación lógica es un OR?

0



Arquitecturas Q: Q5

Arquitecturas Q: Q5

Instrucciones:

MUL, MOV, ADD, SUB, DIV, CALL, RET, CMP, JMP, JE, JNE, JLE, JG, JL, JGE, JLEU, JGU, JCS, JNEG, JVS

Arquitecturas Q: Q5

Instrucciones:

MUL, MOV, ADD, SUB, DIV, CALL, RET, CMP, JMP, JE, JNE, JLE, JG, JL, JGE, JLEU, JGU, JCS, JNEG, JVS, AND, OR, NOT.

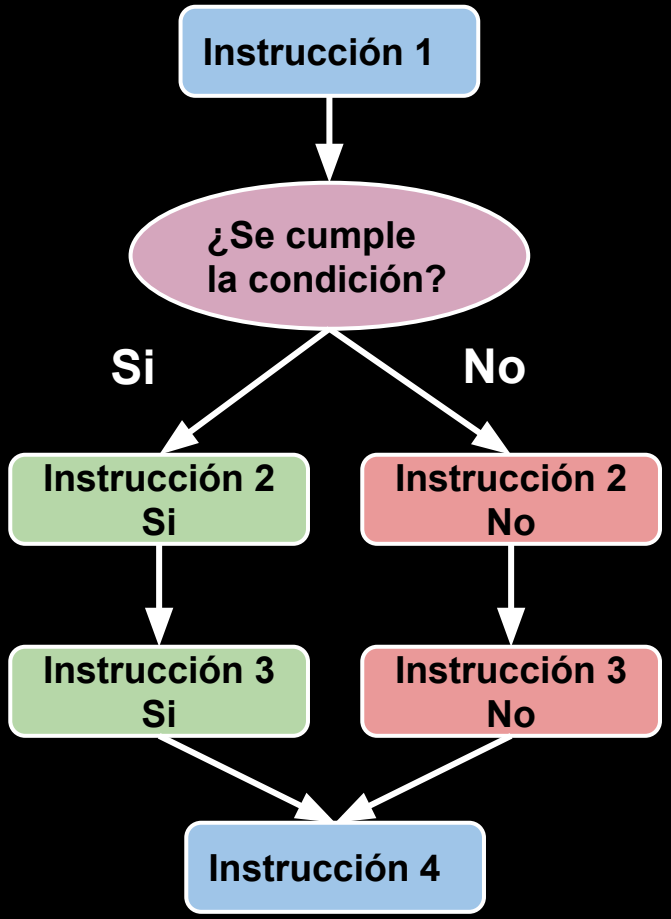
Arquitecturas Q: Q5

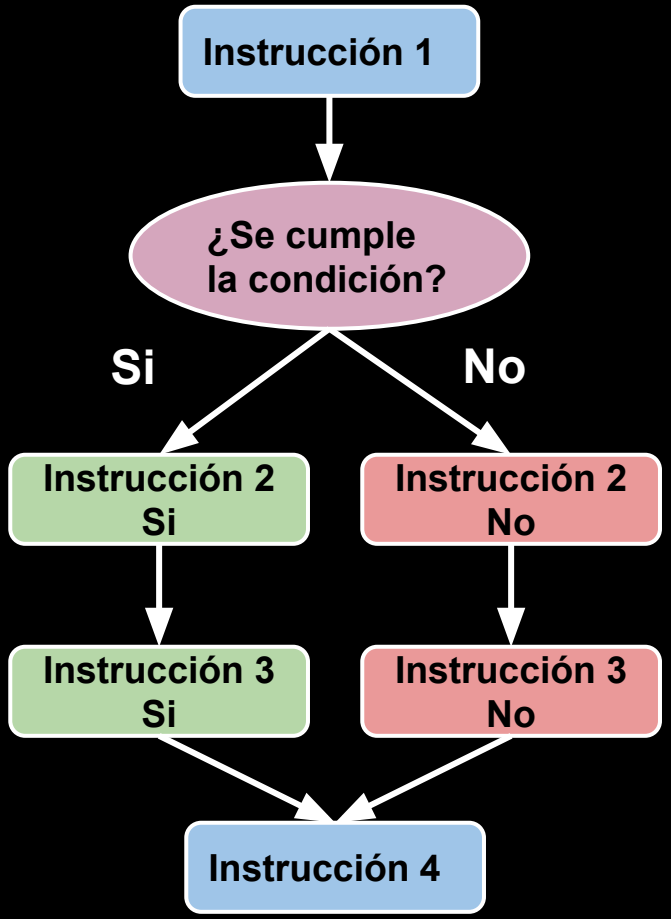
Instrucciones:

MUL, MOV, ADD, SUB, DIV, CALL, RET, CMP, JMP, JE, JNE, JLE, JG, JL, JGE, JLEU, JGU, JCS, JNEG, JVS, AND, OR, NOT.

Operandos (Modos de direccionamiento):

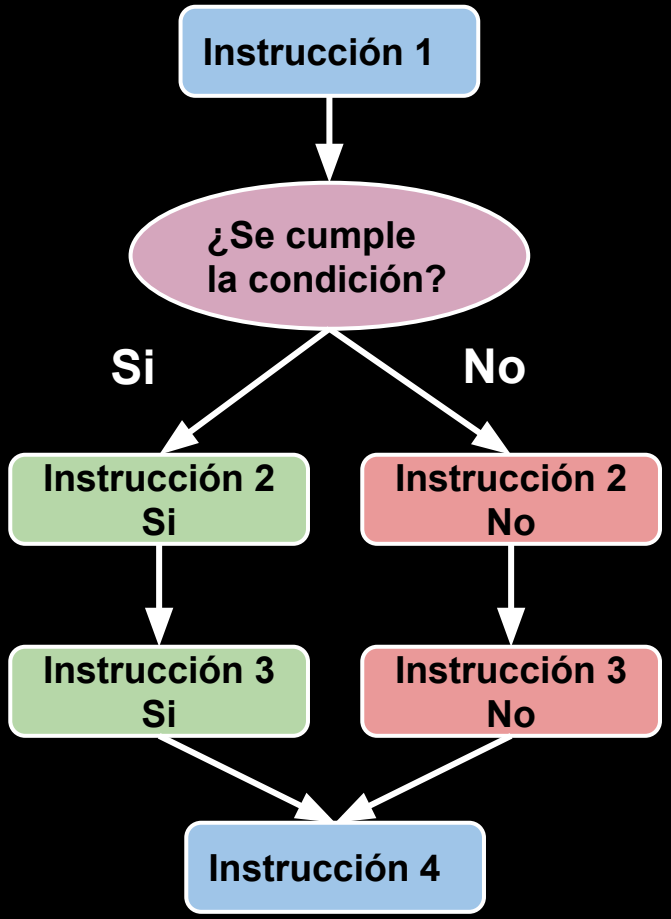
- Registro (modo registro)
- Constante (modo inmediato)
- Dirección memoria (modo directo)

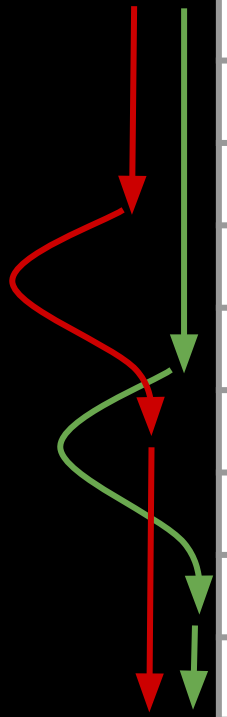

Instrucción 1
¿Se cumple la condición?
Si
No
Instrucción 2 Si
Instrucción 2 No
Instrucción 3 Si
Instrucción 3 No
Instrucción 4



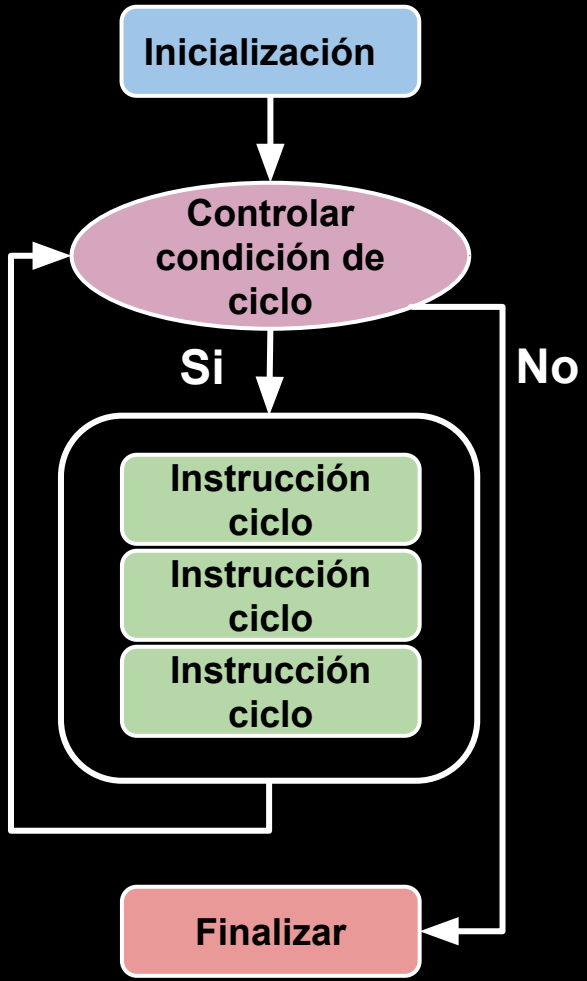
Dirección	Contenido
0x0000	XXXX
0x0001	XXXX
0x0002	XXXX
0x0002	XXXX
0x0003	XXXX
0x0004	XXXX
0x0005	XXXX
0X0006	XXXX
….	….

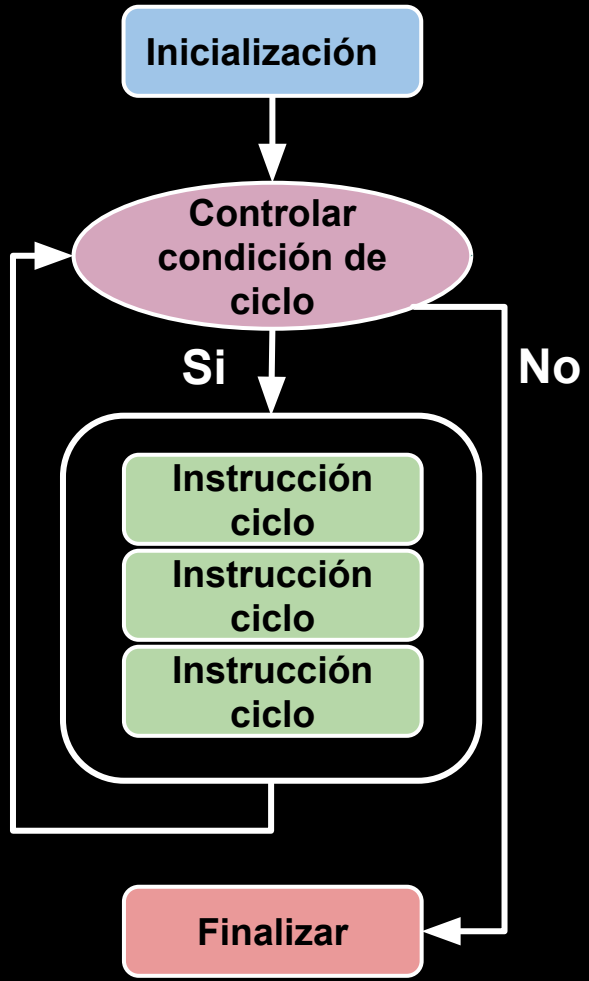
¡Flujos alternativos de ejecución!



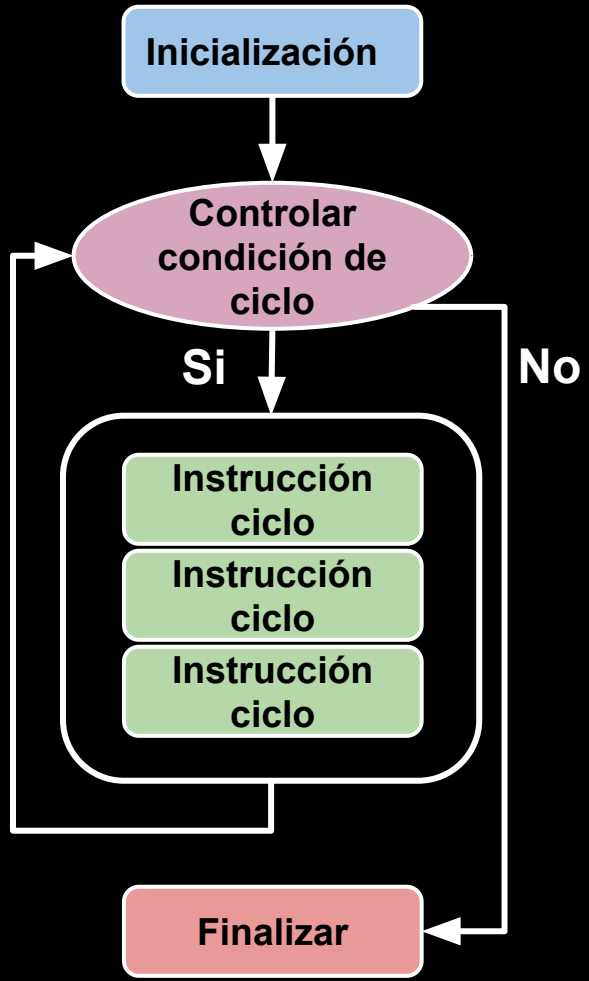


Dirección	Contenido
0x0000	XXXX
0x0001	XXXX
0x0002	XXXX
0x0002	XXXX
0x0003	XXXX
0x0004	XXXX
0x0005	XXXX
0X0006	XXXX
....	....

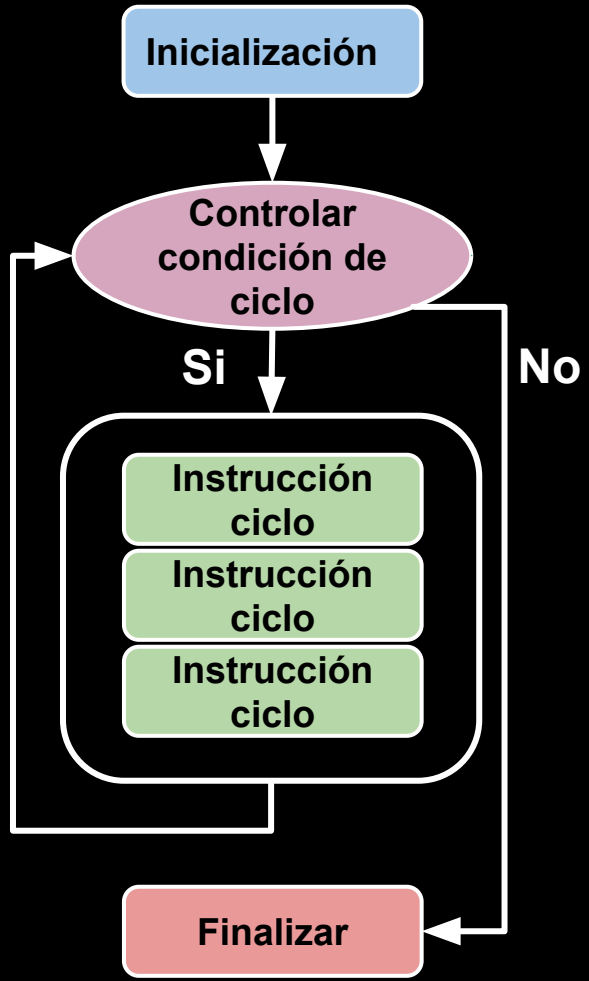

Inicialización
Controlar condición de ciclo
Si
No
Instrucción ciclo
Instrucción ciclo
Instrucción ciclo
Finalizar

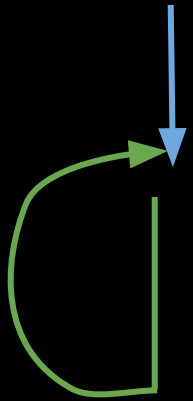


Dirección	Contenido
0x0000	XXXX
0x0001	XXXX
0x0002	XXXX
0x0002	XXXX
0x0003	XXXX
0x0004	XXXX
0x0005	XXXX
0X0006	XXXX
….	….

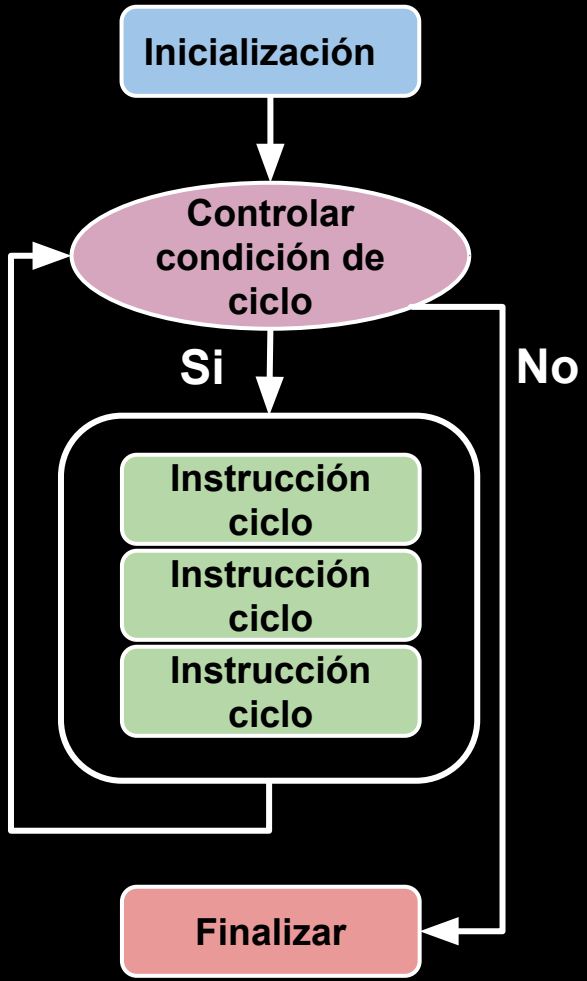


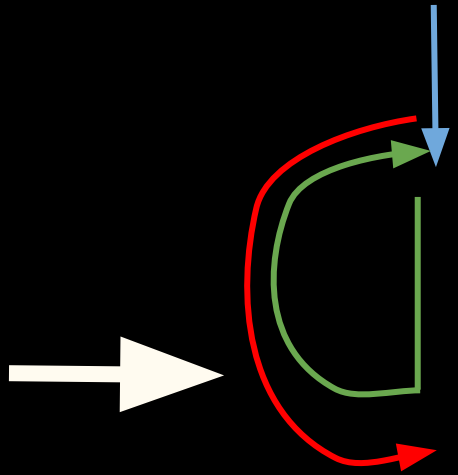
Dirección	Contenido
0x0000	XXXX
0x0001	XXXX
0x0002	XXXX
0x0002	XXXX
0x0003	XXXX
0x0004	XXXX
0x0005	XXXX
0X0006	XXXX
….	….



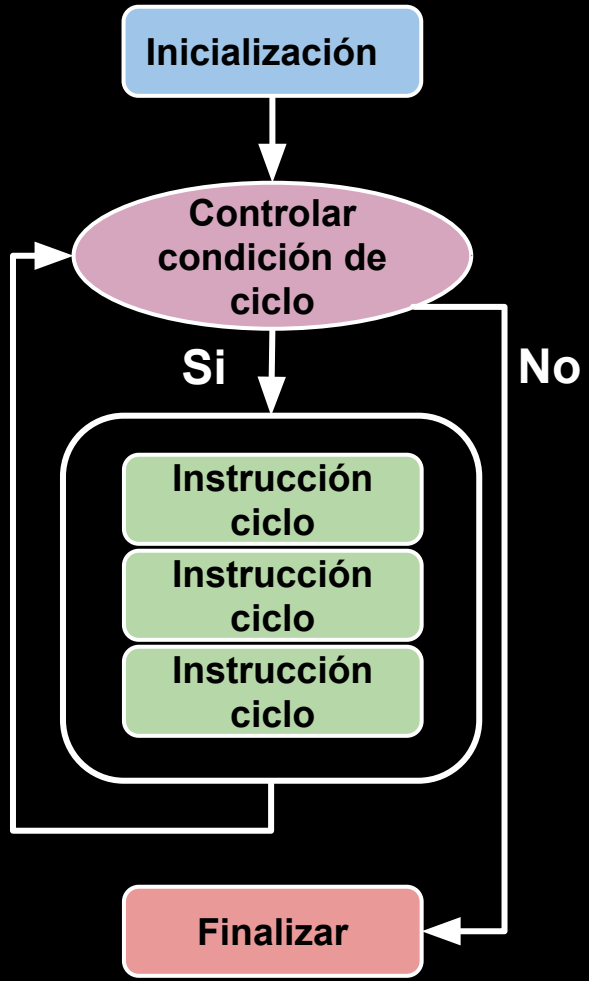


Dirección	Contenido
0x0000	XXXX
0x0001	XXXX
0x0002	XXXX
0x0002	XXXX
0x0003	XXXX
0x0004	XXXX
0x0005	XXXX
0X0006	XXXX
….	….



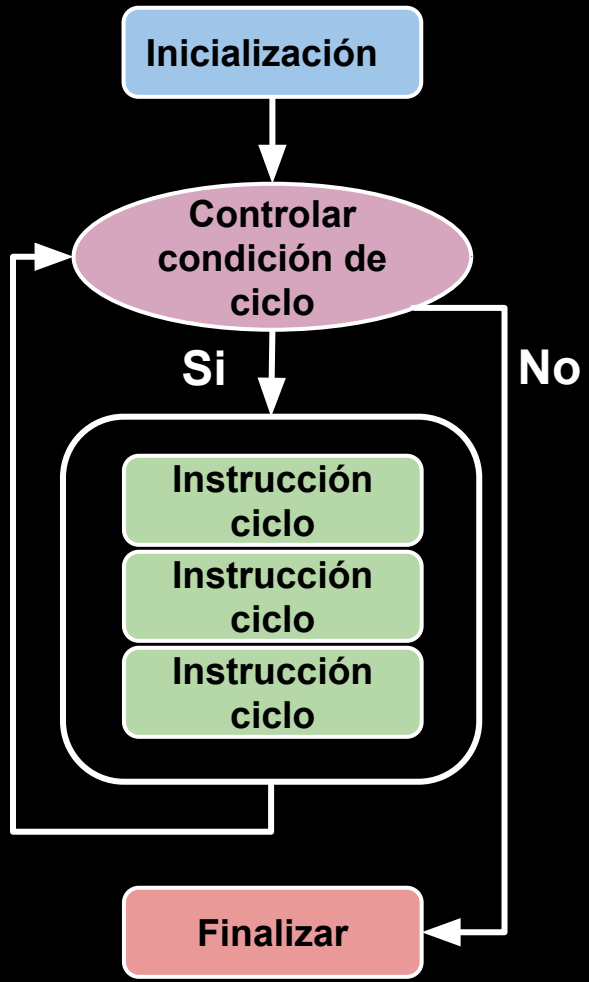


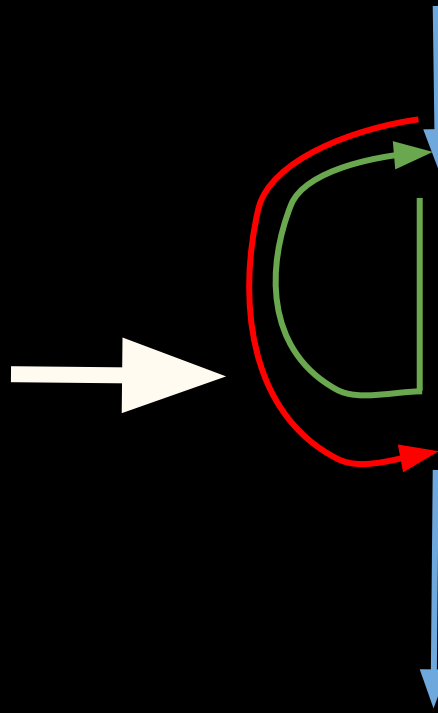
Dirección	Contenido
0x0000	XXXX
0x0001	XXXX
0x0002	XXXX
0x0002	XXXX
0x0003	XXXX
0x0004	XXXX
0x0005	XXXX
0X0006	XXXX
….	….



Dirección	Contenido
0x0000	XXXX
0x0001	XXXX
0x0002	XXXX
0x0002	XXXX
0x0003	XXXX
0x0004	XXXX
0x0005	XXXX
0X0006	XXXX
….	….

Repetición controlada





Dirección	Contenido
0x0000	XXXX
0x0001	XXXX
0x0002	XXXX
0x0002	XXXX
0x0003	XXXX
0x0004	XXXX
0x0005	XXXX
0X0006	XXXX
….	….

Repeticiones controladas

- Rutina que calcule la suma de los primeros n naturales, siendo n un valor mayor o igual cero guardado en R0

Repeticiones controladas

- Rutina que calcule la suma de los primeros n naturales, siendo n un valor mayor o igual cero guardado en R0

MOV R5,

Repeticiones controladas

- Rutina que calcule la suma de los primeros n naturales, siendo n un valor mayor o igual cero guardado en R0

```
MOV R5, 0x0000
```

Repeticiones controladas

- Rutina que calcule la suma de los primeros n naturales, siendo n un valor mayor o igual cero guardado en R0

```
MOV R5, 0x0000
CMP R0, 0x0000
```

Repeticiones controladas

- Rutina que calcule la suma de los primeros n naturales, siendo n un valor mayor o igual cero guardado en R0

```
MOV R5, 0x0000
CMP R0, 0x0000
JLE
```

Repeticiones controladas

- Rutina que calcule la suma de los primeros n naturales, siendo n un valor mayor o igual cero guardado en R0

```
MOV R5, 0x0000
CMP R0, 0x0000
JLE  termine
```

Repeticiones controladas

- Rutina que calcule la suma de los primeros n naturales, siendo n un valor mayor o igual cero guardado en R0

```
MOV R5, 0x0000
CMP R0, 0x0000
JLE   termine
ADD R5, R0
```

Repeticiones controladas

- Rutina que calcule la suma de los primeros n naturales, siendo n un valor mayor o igual cero guardado en R0

```
MOV R5, 0x0000
CMP R0, 0x0000
JLE  termine
ADD R5, R0
SUB R0, 0x0001
```

Repeticiones controladas

- Rutina que calcule la suma de los primeros n naturales, siendo n un valor mayor o igual cero guardado en R0

```
MOV R5, 0x0000
CMP R0, 0x0000
JLE   termine
ADD R5, R0
SUB R0, 0x0001
JMP
```

Repeticiones controladas

- Rutina que calcule la suma de los primeros n naturales, siendo n un valor mayor o igual cero guardado en R0

```
MOV R5, 0x0000
CMP R0, 0x0000
JLE  termine
ADD R5, R0
SUB R0, 0x0001
JMP bucle
```

Repeticiones controladas

- Rutina que calcule la suma de los primeros n naturales, siendo n un valor mayor o igual cero guardado en R0

```
MOV R5, 0x0000
CMP R0, 0x0000
JLE   termine
ADD R5, R0
SUB R0, 0x0001
JMP bucle
RET
```

Repeticiones controladas

- Rutina que calcule la suma de los primeros n naturales, siendo n un valor mayor o igual cero guardado en R0

```
        MOV R5, 0x0000
        CMP R0, 0x0000
        JLE   termine
        ADD R5, R0
        SUB R0, 0x0001
        JMP bucle
termine: RET
```

Repeticiones controladas

- Rutina que calcule la suma de los primeros n naturales, siendo n un valor mayor o igual cero guardado en R0

```
        MOV R5, 0x0000
bucle:  CMP R0, 0x0000
        JLE  termine
        ADD R5, R0
        SUB R0, 0x0001
        JMP bucle
termine: RET
```

Arreglos

Arreglos

- Datos en posiciones de memoria consecutiva

Arreglos

- Datos en posiciones de memoria consecutiva
- Los datos no necesariamente ocupan una celda

Arreglos

- Datos en posiciones de memoria consecutiva
- Los datos no necesariamente ocupan una celda
  - Ej: cada dato es un número de 4 bits (4 datos por celda)

Arreglos

- Datos en posiciones de memoria consecutiva
- Los datos no necesariamente ocupan una celda
  - Ej: cada dato es un número de 4 bits (4 datos por celda)
- La cantidad de datos es el tamaño del arreglo

Arreglos

- Datos en posiciones de memoria consecutiva
- Los datos no necesariamente ocupan una celda
  - Ej: cada dato es un número de 4 bits (4 datos por celda)
- La cantidad de datos es el tamaño del arreglo
- Sabemos cuando el arreglo termina porque

Arreglos

- Datos en posiciones de memoria consecutiva
- Los datos no necesariamente ocupan una celda
  - Ej: cada dato es un número de 4 bits (4 datos por celda)
- La cantidad de datos es el tamaño del arreglo
- Sabemos cuando el arreglo termina porque
  - a) Nos dicen el tamaño

Arreglos

- Datos en posiciones de memoria consecutiva
- Los datos no necesariamente ocupan una celda
  - Ej: cada dato es un número de 4 bits (4 datos por celda)
- La cantidad de datos es el tamaño del arreglo
- Sabemos cuando el arreglo termina porque
  - a) Nos dicen el tamaño
  - b) Nos dicen que al final hay un valor especial, por ej cero

Arreglos

- ¿Cómo podría recorrer el arreglo usando un bucle?

Arreglos

- ¿Cómo podría recorrer el arreglo usando un bucle?
  - Haciendo que el programa modifique su propio código

Arreglos

- ¿Cómo podría recorrer el arreglo usando un bucle?
  - Haciendo que el programa modifique su propio código
- Usamos un nuevo modo de direccionamiento

Arreglos

- ¿Cómo podría recorrer el arreglo usando un bucle?
  - Haciendo que el programa modifique su propio código
- Usamos un nuevo modo de direccionamiento
  - Modo indirecto

Arreglos

- ¿Cómo podría recorrer el arreglo usando un bucle?
  - Haciendo que el programa modifique su propio código
- Usamos un nuevo modo de direccionamiento
  - Modo indirecto
    - Indirecto [[valor]]

Arreglos

- ¿Cómo podría recorrer el arreglo usando un bucle?
  - Haciendo que el programa modifique su propio código
- Usamos un nuevo modo de direccionamiento
  - Modo indirecto
    - Indirecto [[valor]]
    - Registro indirecto [Ri]

Indirecto

Registro	Contenido
R0	0xFD34
R1	0xAA56
R2	0x00A6
R3	0xF6CD
R4	0x00A3
R5	0x00A1
R6	0x00A3
R7	0xABCC

Dirección	Contenido
0x00A0	0x00A6
0x00A1	0xF6CD
0x00A2	0x00A3
0x00A3	0x00A1
0x00A4	0xABC2
0x00A5	0xFD31
0x00A6	0x0000
0X00A7	0x00A0

Indirecto

R0 =
[R5] =
[0x00A0] =
[[0x00A0]] =
[[0x00A2]] =

Registro	Contenido
R0	0xFD34
R1	0xAA56
R2	0x00A6
R3	0xF6CD
R4	0x00A3
R5	0x00A1
R6	0x00A3
R7	0xABCC

Dirección	Contenido
0x00A0	0x00A6
0x00A1	0xF6CD
0x00A2	0x00A3
0x00A3	0x00A1
0x00A4	0xABC2
0x00A5	0xFD31
0x00A6	0x0000
0X00A7	0x00A0

Indirecto

R0 =
[R5] =
[0x00A0] =
[[0x00A0]] =
[[0x00A2]] =

Registro	Contenido
R0	0xFD34
R1	0xAA56
R2	0x00A6
R3	0xF6CD
R4	0x00A3
R5	0x00A1
R6	0x00A3
R7	0xABCC

Dirección	Contenido
0x00A0	0x00A6
0x00A1	0xF6CD
0x00A2	0x00A3
0x00A3	0x00A1
0x00A4	0xABC2
0x00A5	0xFD31
0x00A6	0x0000
0X00A7	0x00A0

Indirecto

R0 = 0xFD34
[R5] =
[0x00A0] =
[[0x00A0]] =
[[0x00A2]] =

Registro	Contenido
R0	0xFD34
R1	0xAA56
R2	0x00A6
R3	0xF6CD
R4	0x00A3
R5	0x00A1
R6	0x00A3
R7	0xABCC

Dirección	Contenido
0x00A0	0x00A6
0x00A1	0xF6CD
0x00A2	0x00A3
0x00A3	0x00A1
0x00A4	0xABC2
0x00A5	0xFD31
0x00A6	0x0000
0X00A7	0x00A0

Indirecto

R0 = 0xFD34
[R5] =
[0x00A0] =
[[0x00A0]] =
[[0x00A2]] =

Registro	Contenido
R0	0xFD34
R1	0xAA56
R2	0x00A6
R3	0xF6CD
R4	0x00A3
R5	0x00A1
R6	0x00A3
R7	0xABCC

Dirección	Contenido
0x00A0	0x00A6
0x00A1	0xF6CD
0x00A2	0x00A3
0x00A3	0x00A1
0x00A4	0xABC2
0x00A5	0xFD31
0x00A6	0x0000
0X00A7	0x00A0

Indirecto

R0 = 0xFD34
[R5] =
[0x00A0] =
[[0x00A0]] =
[[0x00A2]] =

Registro	Contenido
R0	0xFD34
R1	0xAA56
R2	0x00A6
R3	0xF6CD
R4	0x00A3
R5	0x00A1
R6	0x00A3
R7	0xABCC

Dirección	Contenido
0x00A0	0x00A6
0x00A1	0xF6CD
0x00A2	0x00A3
0x00A3	0x00A1
0x00A4	0xABC2
0x00A5	0xFD31
0x00A6	0x0000
0X00A7	0x00A0

Indirecto

R0 = 0xFD34
[R5] = 0xF6CD
[0x00A0] =
[[0x00A0]] =
[[0x00A2]] =

Registro	Contenido
R0	0xFD34
R1	0xAA56
R2	0x00A6
R3	0xF6CD
R4	0x00A3
R5	0x00A1
R6	0x00A3
R7	0xABCC

Dirección	Contenido
0x00A0	0x00A6
0x00A1	0xF6CD
0x00A2	0x00A3
0x00A3	0x00A1
0x00A4	0xABC2
0x00A5	0xFD31
0x00A6	0x0000
0X00A7	0x00A0

Indirecto

R0 = 0xFD34
[R5] = 0xF6CD
[0x00A0] =
[[0x00A0]] =
[[0x00A2]] =

Registro	Contenido
R0	0xFD34
R1	0xAA56
R2	0x00A6
R3	0xF6CD
R4	0x00A3
R5	0x00A1
R6	0x00A3
R7	0xABCC

Dirección	Contenido
0x00A0	0x00A6
0x00A1	0xF6CD
0x00A2	0x00A3
0x00A3	0x00A1
0x00A4	0xABC2
0x00A5	0xFD31
0x00A6	0x0000
0X00A7	0x00A0

Indirecto

R0 = 0xFD34
[R5] = 0xF6CD
[0x00A0] = 0x00A6
[[0x00A0]] =
[[0x00A2]] =

Registro	Contenido
R0	0xFD34
R1	0xAA56
R2	0x00A6
R3	0xF6CD
R4	0x00A3
R5	0x00A1
R6	0x00A3
R7	0xABCC

Dirección	Contenido
0x00A0	0x00A6
0x00A1	0xF6CD
0x00A2	0x00A3
0x00A3	0x00A1
0x00A4	0xABC2
0x00A5	0xFD31
0x00A6	0x0000
0X00A7	0x00A0

Indirecto

R0 = 0xFD34
[R5] = 0xF6CD
[0x00A0] = 0x00A6
[[0x00A0]] =
[[0x00A2]] =

Registro	Contenido
R0	0xFD34
R1	0xAA56
R2	0x00A6
R3	0xF6CD
R4	0x00A3
R5	0x00A1
R6	0x00A3
R7	0xABCC

Dirección	Contenido
0x00A0	0x00A6
0x00A1	0xF6CD
0x00A2	0x00A3
0x00A3	0x00A1
0x00A4	0xABC2
0x00A5	0xFD31
0x00A6	0x0000
0X00A7	0x00A0

Indirecto

R0 = 0xFD34
[R5] = 0xF6CD
[0x00A0] = 0x00A6
[[0x00A0]] =
[[0x00A2]] =

Registro	Contenido
R0	0xFD34
R1	0xAA56
R2	0x00A6
R3	0xF6CD
R4	0x00A3
R5	0x00A1
R6	0x00A3
R7	0xABCC

Dirección	Contenido
0x00A0	0x00A6
0x00A1	0xF6CD
0x00A2	0x00A3
0x00A3	0x00A1
0x00A4	0xABC2
0x00A5	0xFD31
0x00A6	0x0000
0X00A7	0x00A0

Indirecto

R0 = 0xFD34
[R5] = 0xF6CD
[0x00A0] = 0x00A6
[[0x00A0]] = 0x0000
[[0x00A2]] =

Registro	Contenido
R0	0xFD34
R1	0xAA56
R2	0x00A6
R3	0xF6CD
R4	0x00A3
R5	0x00A1
R6	0x00A3
R7	0xABCC

Dirección	Contenido
0x00A0	0x00A6
0x00A1	0xF6CD
0x00A2	0x00A3
0x00A3	0x00A1
0x00A4	0xABC2
0x00A5	0xFD31
0x00A6	0x0000
0X00A7	0x00A0

Indirecto

R0 = 0xFD34
[R5] = 0xF6CD
[0x00A0] = 0x00A6
[[0x00A0]] = 0x0000
[[0x00A2]] =

Registro	Contenido
R0	0xFD34
R1	0xAA56
R2	0x00A6
R3	0xF6CD
R4	0x00A3
R5	0x00A1
R6	0x00A3
R7	0xABCC

Dirección	Contenido
0x00A0	0x00A6
0x00A1	0xF6CD
0x00A2	0x00A3
0x00A3	0x00A1
0x00A4	0xABC2
0x00A5	0xFD31
0x00A6	0x0000
0X00A7	0x00A0

Indirecto

R0 = 0xFD34
[R5] = 0xF6CD
[0x00A0] = 0x00A6
[[0x00A0]] = 0x0000
[[0x00A2]] = 0x00A1

Registro	Contenido
R0	0xFD34
R1	0xAA56
R2	0x00A6
R3	0xF6CD
R4	0x00A3
R5	0x00A1
R6	0x00A3
R7	0xABCC

Dirección	Contenido
0x00A0	0x00A6
0x00A1	0xF6CD
0x00A2	0x00A3
0x00A3	0x00A1
0x00A4	0xABC2
0x00A5	0xFD31
0x00A6	0x0000
0X00A7	0x00A0

Arquitecturas Q: Q6

Instrucciones:

MUL, MOV, ADD, SUB, DIV, CALL, RET, CMP, JMP, JE, JNE, JLE, JG, JL, JGE, JLEU, JGU, JCS, JNEG, JVS, AND, OR, NOT.

Arquitecturas Q: Q6

Instrucciones:

MUL, MOV, ADD, SUB, DIV, CALL, RET, CMP, JMP, JE, JNE, JLE, JG, JL, JGE, JLEU, JGU, JCS, JNEG, JVS, AND, OR, NOT.

Operandos (Modos de direccionamiento):

- Registro (modo registro)
- Constante (modo inmediato)
- Dirección memoria (modo directo)

Arquitecturas Q: Q6

Instrucciones:

MUL, MOV, ADD, SUB, DIV, CALL, RET, CMP, JMP, JE, JNE, JLE, JG, JL, JGE, JLEU, JGU, JCS, JNEG, JVS, AND, OR, NOT.

Operandos (Modos de direccionamiento):

- Registro (modo registro)
- Constante (modo inmediato)
- Dirección memoria (modo directo)
- Registro indirecto
- Indirecto

Ejercicios

1. Ensamblar:

```
ADD [R0], R1
SUB [0x0FFF], 0x0001
MUL [[0x0FFF]], [R0]
```

2. Armar una rutina que recorra un arreglo y deje un 0 si el número es **impar** y un 1 si el número es **par** en esa misma celda. El arreglo comienza en la celda 0x00F0. El último elemento del arreglo es un número negativo.