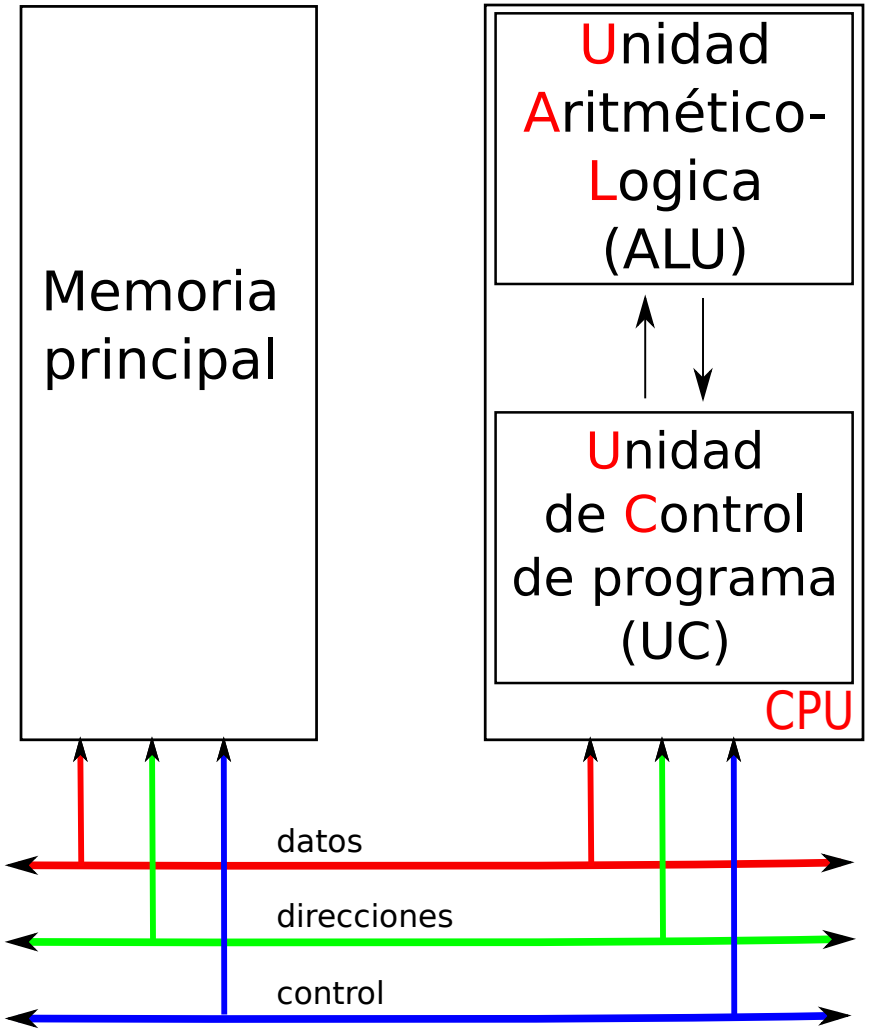
Estructura condicional

Organización de computadoras

Universidad Nacional de Quilmes

http://orga.blog.unq.edu.ar

Repaso


Memoria
principal
Unidad
Aritmético-
Logica
(ALU)
Unidad
de Control
de programa
(UC)
CPU
datos
direcciones
control

Repaso

1. **Memoria Principal**
   1. Organización: celdas y direcciones
   2. Lectura/Escritura
2. **Buses**
   1. ¿Qué son?
   2. Bus de Control / Bus de Datos / Bus de Direcciones
   3. Relación con la Memoria Principal
3. **Q2**
   1. Modo de direccionamiento directo
4. **Ejecución de un programa**
   1. Accesos a memoria en la búsqueda de **instrucción**
   2. Accesos a memoria en la búsqueda de **operandos**

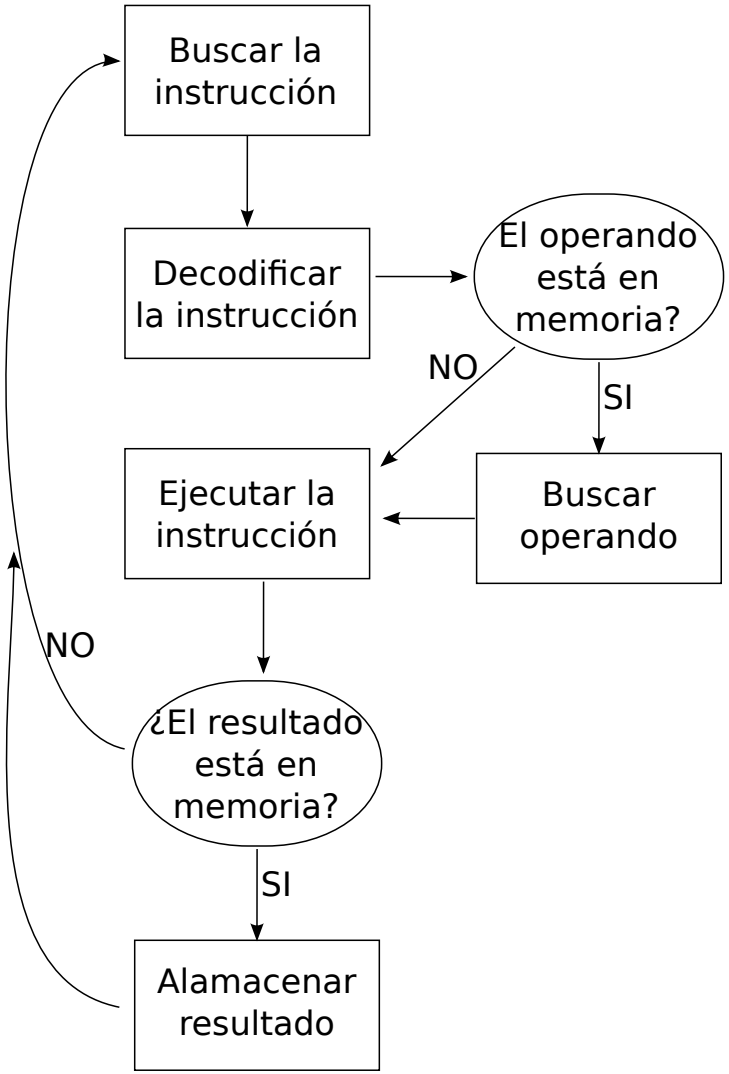
Ejercicio

Ejecutar y contar accesos de la siguiente instrucción

MOV [0A0A],[B0B0]

Ciclo de ejecución revisado

Ciclo de ejecución revisado



¿En que parte de la memoria se busca la instrucción?

Ciclo de ejecución revisado

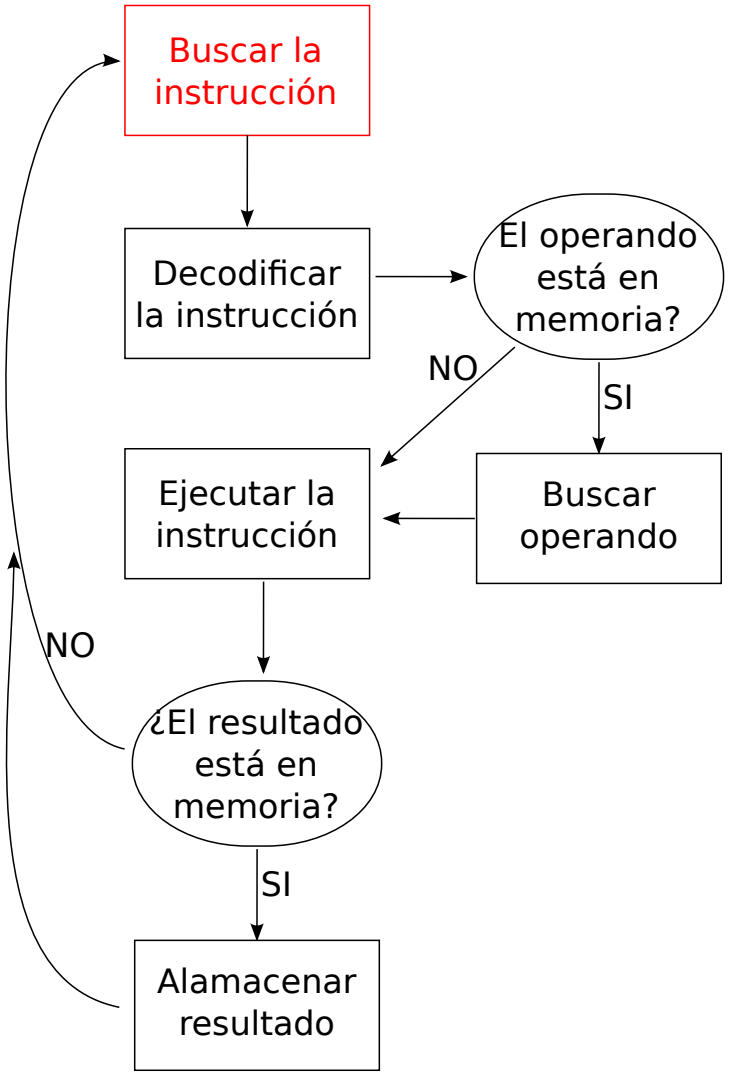
¿Cómo sabe la UC cuál es la celda que debe leer cada vez?

Utiliza un registro especial: Program Counter (PC)

¿Dónde se almacena la instrucción leída?

En otro registro especial: Instruction Register (IR)

Ciclo de ejecución revisado



1. Se hace una lectura de la celda de memoria que indica PC.
2. El contenido de la celda leida se carga en IR
3. Se incrementa PC en 1

Ciclo de ejecución revisado

0	0101
1	1010
2	0000
3	1111
4	1100

- La instrucción actual es 0101
- El valor de PC es 1

Registros reservados

Entonces...

PC (Program Counter) indica la dirección de la **siguiente instrucción** a ejecutar

IR (Instruction Register) Almacena el código máquina de la **instrucción actual**

Program Counter: Ejemplo

0000	1200
0001	000F
0002	1111
0003	29C8
0004	A0A0
0005	0000
0006	0000
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	0000

1. PC=0

Program Counter: Ejemplo

0000	1200
0001	000F
0002	1111
0003	29C8
0004	A0A0
0005	0000
0006	0000
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	0000

1. PC=0
2. Búsqueda de instrucción:
   1. Lectura de la celda 0000
   2. IR=1200
   3. PC=0001

Program Counter: Ejemplo

0000	1200
0001	000F
0002	1111
0003	29C8
0004	A0A0
0005	0000
0006	0000
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	0000

1. PC=0
2. Búsqueda de instrucción (celda 0000)
3. Decodificación de la instrucción: MOV [??], 0x??

Program Counter: Ejemplo

0000	1200
0001	000F
0002	1111
0003	29C8
0004	A0A0
0005	0000
0006	0000
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	0000

1. PC=0
2. Búsqueda de instrucción (celda 0000)
3. Decodificación de la instrucción: MOV [??], 0x??
4. Búsqueda de instrucción:
   1. Lectura de la celda 0001
   2. IR=1200000F
   3. PC=0002

Program Counter: Ejemplo

0000	1200
0001	000F
0002	1111
0003	29C8
0004	A0A0
0005	0000
0006	0000
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	0000

1. PC=0
2. Búsqueda de instrucción (celda 0000)
3. Decodificación de la instrucción: MOV [??], 0x??
4. Búsqueda de instrucción (celda 0001)
5. Búsqueda de instrucción:
   1. Lectura de la celda 0002
   2. IR=1200FFFF0000
   3. PC=0003

Program Counter: Ejemplo

0000	1200
0001	000F
0002	1111
0003	29C8
0004	A0A0
0005	0000
0006	0000
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	0000

1. PC=0
2. Búsqueda de instrucción (celda 0000)
3. Decodificación de la instrucción: MOV [??], 0x??
4. Búsqueda de instrucción (celda 0001)
5. Búsqueda de instrucción (celda 0002)
6. Decodificación de la instrucción: MOV [FFFF], 0x0000

Program Counter: Ejemplo

0000	1200
0001	000F
0002	1111
0003	29C8
0004	A0A0
0005	0000
0006	0000
0007	0000
0008	0000
0009	0000
000A	0000
000B	0000
000C	0000
000D	0000
000E	0000
000F	1111

1. PC=0
2. Búsqueda de instrucción (celda 0000)
3. Decodificación de la instrucción: MOV [??], 0x??
4. Búsqueda de instrucción (celda 0001)
5. Búsqueda de instrucción (celda 0002)
6. Decodificación de la instrucción: MOV [FFFF], 0x0000
7. Ejecución de la instrucción

La decodificación de la instrucción **NO** es el desensamblado

¡Desafío!

Ejecución condicional

Hacer un programa que, si el valor en R0 es igual al valor en R1, ponga en R2 un 1 ó 0 en caso contrario

??

Flags: motivación

¿Cómo determinar si dos números son iguales?

¿Cómo expresar otras condiciones?

¿Cómo ejecutar una instrucción dependiendo de una condición?

Flags: motivación

¿Cómo determinar si dos números son iguales?

Mediante la resta:

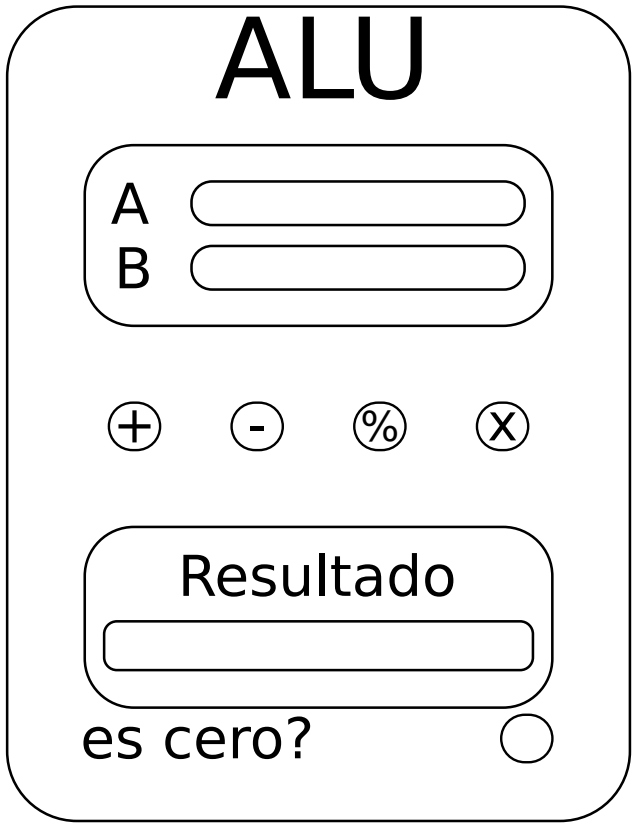
$$\begin{array}{r} A \\ - \quad B \\ \hline C \end{array}$$

si $A = B$ entonces $C = 0$

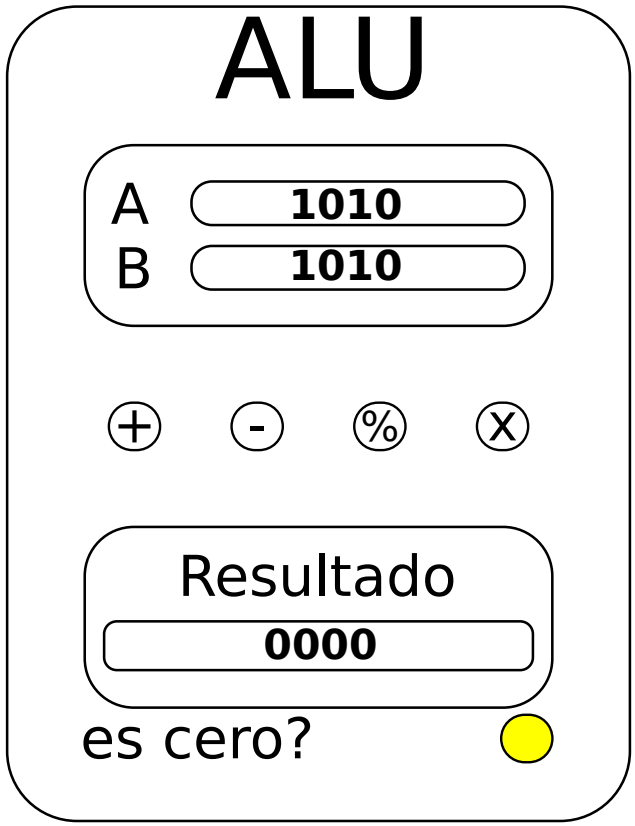
Se necesita una operación de resta (SUB) ✓

Se necesita observar el resultado de la ALU ✗

Flags: motivación



Flags: motivación



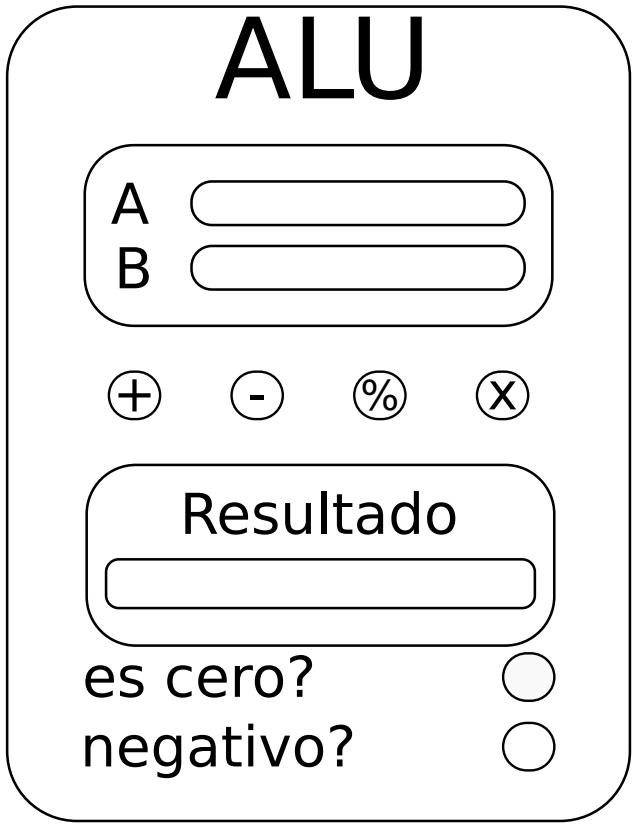
Flags: motivación

¿Cómo determinar si un número es negativo?

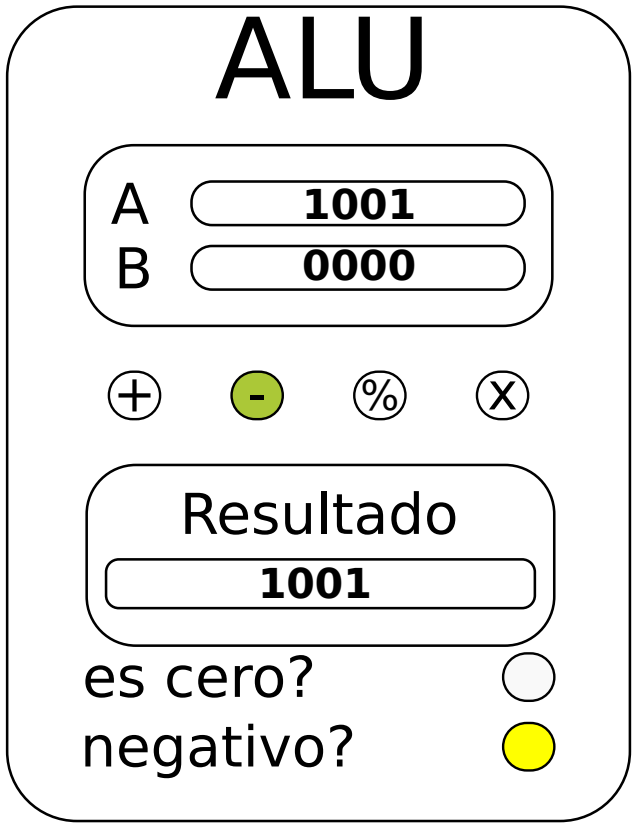
Mirando su primer bit (En ca2)

¿Cómo lo hago con lo que ya tengo?

Flags: motivación



Flags: motivación



Ejecución condicional

Ejercicio...

Hacer un programa que ponga en R2 el máximo valor entre R0 y R1

??

Flags: motivación

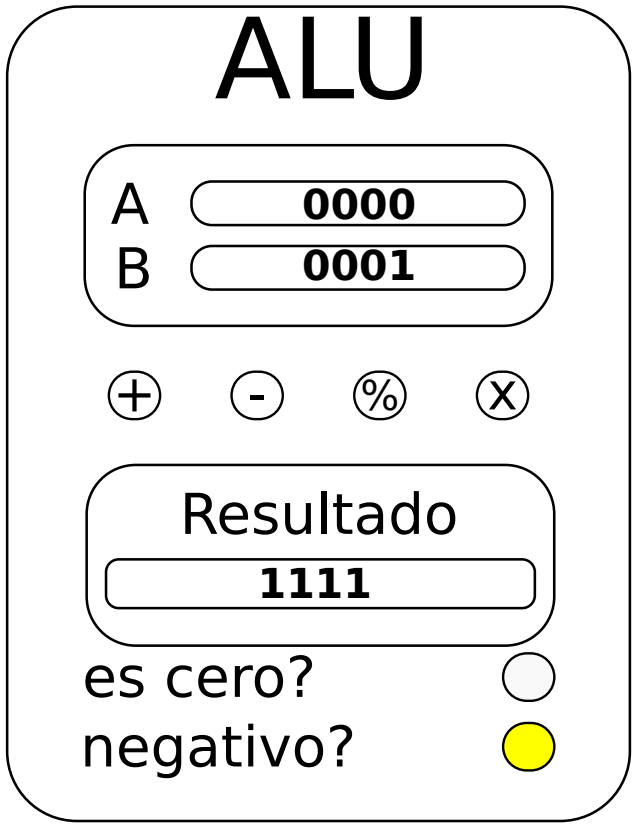
¿Cómo determinar si un número es mayor a otro?

Mediante la resta:

$$\begin{array}{rr} & A \\ - & B \\ \hline & C \end{array}$$ si A>B entonces C>0

¿Cómo lo hago con lo que ya tengo?

Flags: motivación



Flags

Flags

- Son bits que se usan para caracterizar el resultado de la ALU
- La ALU los calcula cuando lleva a cabo una operación aritmética
- Cada uno indica una condición distinta
- La arquitectura provee instrucciones para conocer su valor y actuar en consecuencia.

Flags: Z, N, C y V

Son 4

- Z (Zero)
- N (Negative)
- C (Carry)
- V (Overflow)

Flag Z (Zero)

Z=1 cuando todos los bits del resultado son 0

En una resta:

$$\begin{array}{r} 111 \\ -\ 111 \\ \hline 000 \end{array} \quad Z{=}1$$

En una suma:

$$\begin{array}{r} 111 \\ +\ 001 \\ \hline 000 \end{array} \quad Z{=}1$$

Flag Z (Zero)

¿Cuanto vale Z?

$$\begin{array}{rr} & 101 \\ - & 001 \\ \hline & 100 \end{array} \qquad Z{=}0$$

Flag N (Negative)

N=1 cuando el primer bit del resultado es 1

En una resta:

$$\begin{array}{rr} & 100 \\ - & 001 \\ \hline & 011 \end{array}$$

N=0

En una suma:

$$\begin{array}{rr} & 101 \\ + & 001 \\ \hline & 110 \end{array}$$

N=1

Flag N (Negative)

¿Cuanto vale N?

$$\begin{array}{rr} & 011 \\ + & 011 \\ \hline & 110 \end{array} \qquad \text{N=1}$$

Flag C (Carry)

C=1 cuando luego al terminar la suma hay un acarreo (o al terminar la resta hay un préstamo)

En una resta:

$$\begin{array}{rr} & 100 \\ - & 001 \\ \hline & 011 \end{array}$$

C=0

En una suma:

$$\begin{array}{rr} & 111 \\ + & 001 \\ \hline & 000 \end{array}$$

C=1

Flag C (Carry)

¿Cuanto vale C?

$$\begin{array}{rr} & 011 \\ + & 011 \\ \hline & 110 \end{array} \quad C{=}0$$

$$\begin{array}{rr} & 011 \\ - & 101 \\ \hline & 110 \end{array} \quad C{=}1$$

Flag C (Carry)

¿Que significa C en *BSS()*?

$$\begin{array}{r} 011_{\text{2}}3 \\ +\ 011_{\text{2}}3 \\ \hline 110_{\text{2}}6 \end{array} \quad \text{C=0} \ \checkmark$$

$$\begin{array}{r} 101_{\text{2}}5 \\ +\ 101_{\text{2}}5 \\ \hline 010_{\text{2}}\text{¿2?} \end{array} \quad \text{C=1} \ \times \qquad \begin{array}{r} 011_{\text{2}}3 \\ -\ 101_{\text{2}}5 \\ \hline 110_{\text{2}}\text{¿6?} \end{array} \quad \text{C=1} \ \times$$

Flag C (Carry)

¿Que significa C en *BSS()*?

El resultado no se puede representar

$$\begin{array}{rr} & 101_{BSS(3)} \\ + & 101_{BSS(3)} \\ \hline & \text{¿10?} \end{array} \qquad \begin{array}{rr} & 011_{BSS(3)} \\ - & 101_{BSS(3)} \\ \hline & \text{¿-2?} \end{array}$$

Fuera de rango de $BSS(3)$: $[0,7]$

Flag C (Carry)

¿Que significa C en *CA2()*?

$$\begin{array}{r} 011_{\cdots 3} \\ + \quad 011_{\cdots 3} \\ \hline 110_{\cdots -2} \end{array} \quad C{=}0 \times \qquad \begin{array}{r} 011_{\cdots 3} \\ + \quad 101_{\cdots -3} \\ \hline 000_{\cdots 0} \end{array} \quad C{=}1 \surd$$

$$\begin{array}{r} 011_{\cdots 3} \\ - \quad 101_{\cdots -3} \\ \hline 110_{\cdots -2} \end{array} \quad C{=}1 \times \qquad \begin{array}{r} 111_{\cdots -1} \\ - \quad 010_{\cdots 2} \\ \hline 101_{\cdots -3} \end{array} \quad C{=}0 \surd$$

Flag C (Carry)

¿Que significa C en *CA2()*?

¡Nada!

Flag V (Overflow)

V=1 cuando, en *CA2*() el resultado no se puede representar

(1) positivo + positivo = negativo ✗

(2) negativo + negativo = positivo ✗

(3) negativo - positivo = positivo ✗

(4) positivo - negativo = negativo ✗

Flag V en *CA2()*

$$\begin{array}{rr} \text{Caso 1} & \begin{array}{r} \text{positivo} \\ + \quad \text{positivo} \\ \hline \text{negativo} \end{array} \end{array}$$

$$\begin{array}{r} 010\cdots2 \\ + \quad 010\cdots2 \\ \hline 100\cdots\text{-}4 \end{array} \text{ , V=1 } \times$$

Flag V en *CA2()*

$$\begin{array}{rr} \text{Caso 2} & \begin{array}{r} \text{negativo} \\ + \quad \text{negativo} \\ \hline \text{positivo} \end{array} \end{array}$$

$$\begin{array}{r} 100\cdots4 \\ + \quad 100\cdots4 \\ \hline 000\cdots0 \end{array} \quad , \; V{=}1 \; \times$$

Flag V en *CA2*()

$$\begin{array}{ll} \text{Caso 3} & \begin{array}{l} \text{negativo} \\ \text{- positivo} \\ \hline \text{positivo} \end{array} \end{array}$$

$$\begin{array}{r} 110\ldots2 \\ - \quad 011\ldots3 \\ \hline 011\ldots3 \end{array} \quad \text{, V=1} \times$$

Flag V en *CA2()*

Caso 4	-	positivo
		negativo
		negativo

$$
\begin{array}{rl}
 & 001\cdots 1 \\
- & 100\cdots -4 \\
\hline
 & 101\cdots -3
\end{array}
\quad , \; V{=}1 \; \times
$$

Hasta Acá

Entonces

¿Cómo determinar si dos números son iguales? √

¿Cómo determinar si un número es negativo? √

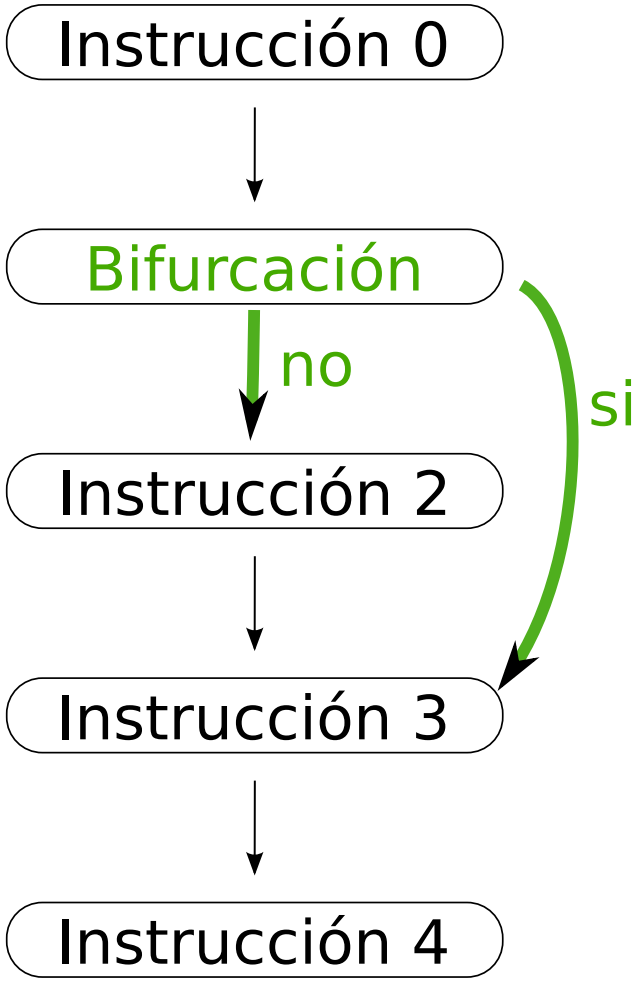
¿Cómo ejecutar una instrucción dependiendo de una condición? ✗

Saltos: motivación

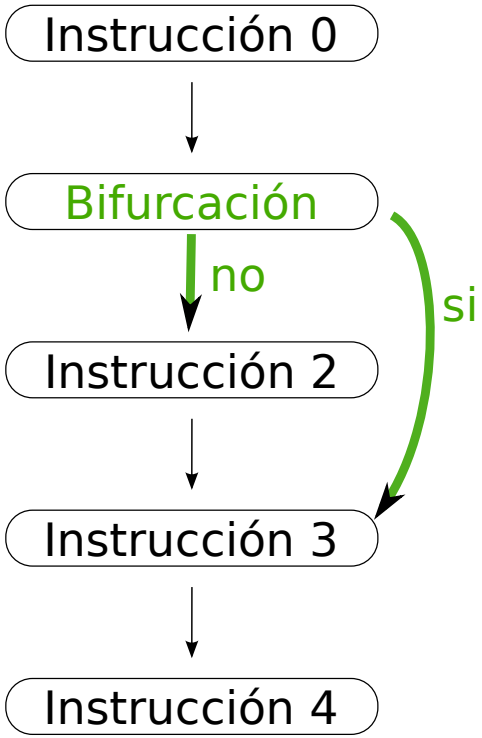
¿Cómo ejecutar una instrucción dependiendo de una condición?

Se necesita una instrucción que observe los resultados de la ALU para desviar el programa

Bifurcación: idea



Bifurcación: idea



¿Cómo hacer para que la UC ejecute la instrucción 3 y no la instrucción 2?

¡Alterando el valor de PC!

Saltos

Saltos: Ejemplo

0000	Instrucción 1
0001	Instrucción 2
0002	Instrucción de salto a 0005
0003	Instrucción 4
0004	Instrucción 5
0005	Instrucción 6
0006	Instrucción 7

PC=0000

Saltos: Ejemplo

0000	Instrucción 1
0001	Instrucción 2
0002	Instrucción de salto a 0005
0003	Instrucción 4
0004	Instrucción 5
0005	Instrucción 6
0006	Instrucción 7

PC=0001

IR = Instrucción 1

Saltos: Ejemplo

0000	Instrucción 1
0001	Instrucción 2
0002	Instrucción de salto a 0005
0003	Instrucción 4
0004	Instrucción 5
0005	Instrucción 6
0006	Instrucción 7

PC=0002

IR = Instrucción 2

Saltos: Ejemplo

0000	Instrucción 1
0001	Instrucción 2
0002	Instrucción de salto a 0005
0003	Instrucción 4
0004	Instrucción 5
0005	Instrucción 6
0006	Instrucción 7

PC=0003

IR = Instrucción de salto

Saltos: Ejemplo

0000	Instrucción 1
0001	Instrucción 2
0002	Instrucción de salto a 0005
0003	Instrucción 4
0004	Instrucción 5
0005	Instrucción 6
0006	Instrucción 7

PC=0005

IR = Instrucción de salto

Ejecución

Saltos: Ejemplo

0000	Instrucción 1
0001	Instrucción 2
0002	Instrucción de salto a 0005
0003	Instrucción 4
0004	Instrucción 5
0005	Instrucción 6
0006	Instrucción 7

PC=0005

IR = Instrucción 6

Ejecución

Saltos absolutos y relativos

Salto Relativo

El nuevo valor de PC se expresa en términos de un desplazamiento con respecto a la siguiente instrucción

0000	Saltar una celda mas adelante
0001	Instrucción 2
0002	Instrucción 3
0003	Instrucción 4

$$PC \leftarrow PC + 1$$

Saltos absolutos y relativos

Salto Absoluto

El nuevo valor de PC se expresa en términos de una dirección de memoria

0000	Saltar a la celda 0003
0001	Instrucción 2
0002	Instrucción 3
0003	Instrucción 4

$$PC \leftarrow 0003$$

Saltos condicionales e incondicionales

Salto condicional
La actualización de PC se lleva a cabo si se cumple determinada condición sobre los flags

Saltos condicionales

Ejemplo

```
0000  SUB R0, R1
0001  saltarSiEsCero 0x0002
0002  SUB R1, 0x0001
0003
0004  ADD R0, 0x000A
```

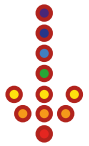
Saltos condicionales

Ejemplo

```
0000  SUB R0, R1
0001  saltarSiEsCero 0x0002← ¿Qué significa?
0002  SUB R1, 0x0001
0003
0004  ADD R0, 0x000A
```

Saltos condicionales

¿Si la condición no se cumple el salto no se ejecuta?



¡NO!

La instrucción de salto si se ejecuta, lo que no ocurre es la alteración de PC

Saltos condicionales e incondicionales

Salto incondicional
La actualización de PC se lleva a cabo siempre que se ejecute el salto

Saltos incondicionales

Ejemplo

```
0000  SUB R0, R1
0001  saltar 0x0002
0002  SUB R1, 0x0001
0003
0004  ADD R0, 0x000A
```

Saltos incondicionales

Ejemplo

0000	SUB R0, R1
0001	saltar 0x0002← ¿Qué significa?
0002	SUB R1, 0x0001
0003	
0004	ADD R0, 0x000A

Saltos condicionales e incondicionales

Saltos condicionales ¿relativos o absolutos?

Saltos incondicionales ¿relativos o absolutos?

Saltos condicionales e incondicionales ¿absolutos o relativos?

Salto condicional absoluto

```
0000  SUB R0, R1
0001  saltarSiEsCero 0x0002
0002  SUB R1, 0x0001
0003
0004  ADD R0, 0x000A
```

Salto condicional relativo

```
0000  SUB R0, R1
0001  saltarSiEsCero 0x0002
0002  SUB R1, 0x0001
0003
0004  ADD R0, 0x000A
```

Etiquetas

Etiqueta
Pseudo-instrucción que permite evitar el cálculo de la dirección (o desplazamiento del salto)

Etiquetas

Ejemplo de uso

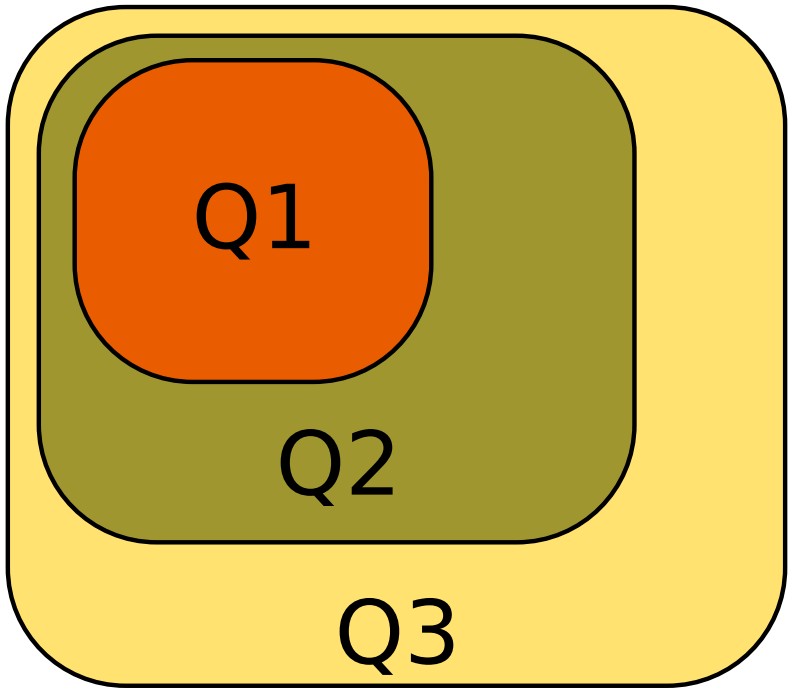
```
            MOV R0, 0x0001
            SUB R1, 0x0001
            saltar-si-es-cero meDioCero
            SUB R1 0x0001
meDioCero   ADD R0, 0x000A
```

Arquitecturas **Q**

... La saga estaba incompleta ...

Arquitectura **Q3**

Arquitectura **Q3**



Arquitectura **Q3**

- Tiene 8 registros de uso general de 16 bits: R0..R7
- Tiene direcciones de 16 bits
- Los operandos pueden estar en registros, ser constantes o estar en direcciones de memoria
- permite 3 modos de direccionamiento:
  - modo registro: el valor buscado está en un registro
  - modo inmediato: el valor buscado está codificado dentro de la instrucción
  - modo directo: el valor buscado está contenido en una celda de memoria

(Idem **Q2**)

Arquitectura **Q3**: formato de instrucciones

- Operaciones de tipo 1 (MUL,MOV,ADD,SUB,DIV)

Cod_Op (4b)	Modo Destino (6b)	Modo Origen (6b)	Operando Destino (16b)	Operando Origen (16b)

- Operaciones de tipo 2 (Salto incondicional y absoluto)

Cod_Op (4b)	Relleno (000000)	Modo Origen (6b)	Operando Origen (16b)

- Operaciones de tipo 3 (Saltos incondicionales y relativos)

Prefijo (1111)	Cod_Op (4)	Desplazamiento(8) (8b)

Arquitectura **Q3**: Operaciones de tipo 1

Tipo 1: Aritméticas

Cod_Op (4b)	Modo Destino (6b)	Modo Origen (6b)	Operando Destino (16b)	Operando Origen (16b)

Operación	**CodOp**
MUL	0000
MOV	0001
ADD	0010
SUB	0011
DIV	0111

(Idem **Q2**)

Arquitectura **Q3**: Operaciones de tipo 2

Tipo 2: Salto incondicional (absoluto)

Cod_Op (4b)	Relleno (000000)	Modo Origen (6b)	Operando Origen (16b)

Operación	**CodOp**	**Efecto**
JMP	1010	PC $\leftarrow$ dirección

Arquitectura **Q3**: Operaciones de tipo 3

Tipo 3: Salto condicional (relativo) - 1 de 2

Prefijo (1111)	Cod_Op (4)	Desplazamiento(8) (8b)

Salto	**Codop**	**Descripción**	**Condición**
JE	0001	Igual / Cero	Z
JNE	1001	No igual	$\overline{Z}$
JLEU	0100	Menor o igual sin signo	$C + Z$
JGU	1100	Mayor sin signo	$\overline{(C + Z)}$
JCS	0101	Menor sin signo	C
JNEG	0110	Negativo	N
JVS	0111	Overflow	V

Arquitectura **Q3**: Operaciones de tipo 3

Tipo 3: Salto condicional (relativo) - 2 de 2

Prefijo (1111)	Cod_Op (4)	Desplazamiento(8) (8b)

Salto	**Codop**	**Descripción**	**Condición**
JLE	0010	Menor o igual con signo	$Z + (N \oplus V)$
JG	1010	Mayor con signo	$\overline{(Z + (N \oplus V))}$
JL	0011	Menor con signo	$N \oplus V$
JGE	1011	Mayor o igual con signo	$\overline{(N \oplus V)}$

Arquitectura **Q3**: Ejercicio

Ensamblar el siguiente programa, ubicándolo a partir de la celda 0FF0

```
        MOV R0, R2
        SUB R0,R1
        JE afuera
        ADD [0001], 0x0005
        ADD R1,R2
afuera: MOV R2, 0x0005
```

Arquitectura **Q3**: Ejercicio

(1)

Hacer un programa que, si el valor en R0 es igual al valor en R1, ponga en R2 un 1 ó 0 en caso contrario

Arquitectura **Q3**: Ejercicio

(2)

Hacer un programa que, si el valor en R7 es negativo, le sume 1, o le reste 1 en caso contrario

Arquitectura **Q3**: Ejercicio

(3)

Hacer un programa que ponga en R2 el máximo valor entre R0 y R1. Considerar que los valores están en *BSS*()

Arquitectura **Q3**: Ejercicio

(4)

Hacer un programa que ponga en R2 el máximo valor entre R0 y R1. Considerar que los valores están en *CA*2()

¿Preguntas?