

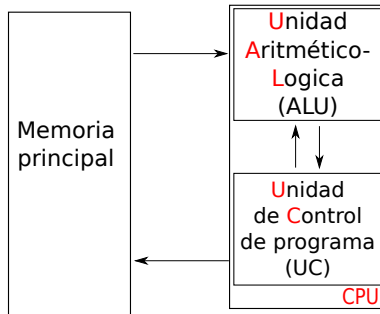
Ejecución de programas simples en una arquitectura simple

Organización de computadoras

Universidad Nacional de Quilmes

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

Arquitectura de Von Neumann



Unidad aritmético lógica

ALU (Unidad aritmético lógica)

Dispositivo que realiza las operaciones aritméticas y lógicas sobre los datos de entrada que se le proveen.

Unidad aritmético lógica

ALU (Unidad aritmético lógica)

Dispositivo que realiza las operaciones aritméticas y lógicas sobre los datos de entrada que se le proveen.

¿Cómo se usa?

Unidad aritmético lógica

ALU (Unidad aritmético lógica)

Dispositivo que realiza las operaciones aritméticas y lógicas sobre los datos de entrada que se le proveen.

¿Cómo se usa?

- 1 La UC (unidad de control) suministra los operandos a la ALU

Unidad aritmético lógica

ALU (Unidad aritmético lógica)

Dispositivo que realiza las operaciones aritméticas y lógicas sobre los datos de entrada que se le proveen.

¿Cómo se usa?

- 1 La UC (unidad de control) suministra los operandos a la ALU
- 2 La UC indica a la ALU la operación a llevar a cabo.

Unidad aritmético lógica

ALU (Unidad aritmético lógica)

Dispositivo que realiza las operaciones aritméticas y lógicas sobre los datos de entrada que se le proveen.

¿Cómo se usa?

- 1 La UC (unidad de control) suministra los operandos a la ALU
- 2 La UC indica a la ALU la operación a llevar a cabo.
- 3 La ALU realiza la operación.

Unidad aritmético lógica

ALU (Unidad aritmético lógica)

Dispositivo que realiza las operaciones aritméticas y lógicas sobre los datos de entrada que se le proveen.

¿Cómo se usa?

- 1 La UC (unidad de control) suministra los operandos a la ALU
- 2 La UC indica a la ALU la operación a llevar a cabo.
- 3 La ALU realiza la operación.
- 4 La UC toma el resultado.

Registros internos

Registro

Espacio de almacenamiento interno a la CPU. Es la tecnología de almacenamiento mas rápida del sistema de cómputos.

Registros internos

Registro

Espacio de almacenamiento interno a la CPU. Es la tecnología de almacenamiento mas rápida del sistema de cómputos.

- Para ser procesado, todo dato debe alojarse en un **registro** interno a la CPU

Registros internos

Registro

Espacio de almacenamiento interno a la CPU. Es la tecnología de almacenamiento mas rápida del sistema de cómputos.

- Para ser procesado, todo dato debe alojarse en un **registro** interno a la CPU
- **Algunos** están disponibles para ser usados por los programas.

Registros internos

Registro

Espacio de almacenamiento interno a la CPU. Es la tecnología de almacenamiento mas rápida del sistema de cómputos.

- Para ser procesado, todo dato debe alojarse en un **registro** interno a la CPU
- **Algunos** están disponibles para ser usados por los programas.
- Otros están **reservados** para uso interno de la Unidad de Control

Código binario

Código binario

Las computadoras tienen la capacidad de traducir una cadena binaria en una acción determinada

Código binario

Las computadoras tienen la capacidad de traducir una cadena binaria en una acción determinada



Los programas deben estar codificados en binario

Código binario

Ejemplo: decodificando una receta

001  agregar huevo

Código binario

Ejemplo: decodificando una receta

001		agregar huevo
000		mezclar

Código binario

Ejemplo: decodificando una receta

001		agregar huevo
000		mezclar
010		agregar harina

Código binario

Ejemplo: decodificando una receta

001		agregar huevo
000		mezclar
010		agregar harina
000		mezclar

Código binario

Ejemplo: decodificando una receta

001		agregar huevo
000		mezclar
010		agregar harina
000		mezclar
110		agregar leche

Código binario

Ejemplo: decodificando una receta

001		agregar huevo
000		mezclar
010		agregar harina
000		mezclar
110		agregar leche
000		mezclar

Código binario

Código Fuente

Código comprensible por un humano

Código Máquina

Código directamente interpretable por la CPU

Código binario

¿Cuándo se **ensambla** (código fuente $\rightarrow$ código máquina)?

¿Cuándo se **desensambla** (código máquina $\rightarrow$ código fuente)?

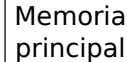
Ciclo de vida de un programa

Ciclo de vida de un programa

El **programador** escribe el programa



El ensamblador lo traduce a **código máquina** y lo carga en memoria



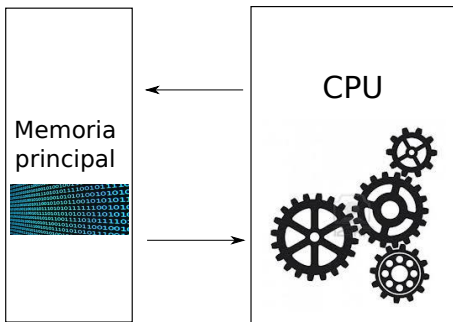
Ciclo de vida de un programa

El **usuario** pide la ejecución del programa



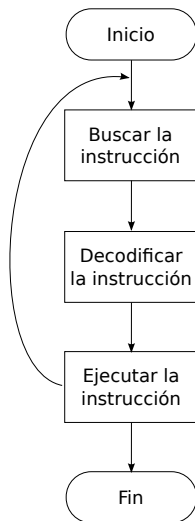
Ciclo de vida de un programa

La CPU ejecuta el programa



Ciclo de ejecución de una instrucción

Ciclo de ejecución de una instrucción



Arquitectura Q1

Arquitectura Q1

- Tiene 8 registros de uso general de 16 bits: R0..R7

Arquitectura Q1

- Tiene 8 registros de uso general de 16 bits: R0..R7
- Tiene instrucciones de 2 operandos:

Arquitectura Q1

- Tiene 8 registros de uso general de 16 bits: R0..R7
- Tiene instrucciones de 2 operandos:

instrucción	sintaxis	efecto
ADD	ADD destino, origen	$\text{destino} \leftarrow \text{destino} + \text{origen}$
SUB	SUB destino, origen	$\text{destino} \leftarrow \text{destino} - \text{origen}$
MUL	MUL destino, origen	$(R7, \text{destino}) \leftarrow \text{destino} * \text{origen}$
DIV	DIV destino, origen	$\text{destino} \leftarrow \text{destino} \% \text{origen}$
MOV	MOV destino, origen	$\text{destino} \leftarrow \text{origen}$

Arquitectura Q1

- Tiene 8 registros de uso general de 16 bits: R0..R7
- Tiene instrucciones de 2 operandos:

instrucción	sintaxis	efecto
ADD	ADD destino, origen	$\text{destino} \leftarrow \text{destino} + \text{origen}$
SUB	SUB destino, origen	$\text{destino} \leftarrow \text{destino} - \text{origen}$
MUL	MUL destino, origen	$(R7, \text{destino}) \leftarrow \text{destino} * \text{origen}$
DIV	DIV destino, origen	$\text{destino} \leftarrow \text{destino} \% \text{origen}$
MOV	MOV destino, origen	$\text{destino} \leftarrow \text{origen}$

- Los operandos pueden ser **variables** (alguno de los registros) o **constantes**.

Arquitectura **Q1**: modos de direccionamiento

Modo de direccionamiento

Mecanismo por el cual la unidad de control obtiene el operando requerido

Arquitectura Q1: modos de direccionamiento

Modo de direccionamiento

Mecanismo por el cual la unidad de control obtiene el operando requerido

Q1 permite 2 modos de direccionamiento:

- modo **registro**: el valor buscado está en un registro
- modo **inmediato**: el valor buscado está codificado dentro de la instrucción (constante)

Arquitectura Q1: modos de direccionamiento

Modo de direccionamiento Inmediato

MOV R0, 0x3456

ADD R6, 0xFEFE

Arquitectura Q1: modos de direccionamiento

Modo de direccionamiento Registro

MOV R0, 0x3456

ADD R6, R1

Arquitectura Q1: formato de instrucciones

Formato de instrucción

Define la organización de los bits dentro de una instrucción, en términos de las partes que la componen. Debe incluir el **código de la operación** y los **operandos**

Arquitectura Q1: formato de instrucciones

Formato de instrucción

Define la organización de los bits dentro de una instrucción, en términos de las partes que la componen. Debe incluir el **código de la operación** y los **operandos**

Cuando se tiene una cadena así:

```
0000100000000000000001111000011110000111100001111
```

Arquitectura Q1: formato de instrucciones

Formato de instrucción

Define la organización de los bits dentro de una instrucción, en términos de las partes que la componen. Debe incluir el **código de la operación** y los **operandos**

Cuando se tiene una cadena así:

```
0000100000000000000001111000011110000111100001111
```



Se separan las partes así

```
0000100000000000000001111000011110000111100001111
```

Arquitectura **Q1**: formato de instrucciones

0000100000000000000001111100001111100001111100001111

Arquitectura Q1: formato de instrucciones

0000100000000000000001111000011110000111100001111



0000100000000000000001111000011110000111100001111

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

Arquitectura **Q1**: Códigos de Operación

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

Arquitectura **Q1**: Códigos de los modos de direccionamiento

Modo	Codificación
Inmediato	000000
Registro	100rrr

donde *rrr* es una codificación (en 3 bits) del número de registro.

Arquitectura Q1: formato de instrucciones

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

Los campos de los operandos Destino y Origen...

- contienen valores constantes (si el modo respectivo es *inmediato*)
- o no existen (si el modo respectivo es *registro*).

Arquitectura **Q1**: Ejemplos

Ejercicio: Ensamblar MOV R1,0x0003

Arquitectura **Q1**: Ejemplos

Ejercicio: Ensamblar MOV R1,0x0003

Efecto	
--------	--

Arquitectura **Q1**: Ejemplos

Ejercicio: Ensamblar MOV R1,0x0003

Efecto	R1 ← 0x0003
--------	-------------

Arquitectura Q1: Ejemplos

Ejercicio: Ensamblar MOV R1,0x0003

Efecto	R1 ← 0x0003
Código de operación	

Arquitectura Q1: Ejemplos

Ejercicio: Ensamblar MOV R1,0x0003

Efecto	R1 $\leftarrow$ 0x0003
Código de operación	0001

Arquitectura Q1: Ejemplos

Ejercicio: Ensamblar MOV R1,0x0003

Efecto	$R1 \leftarrow 0x0003$
Código de operación	0001
Modo Destino	

Arquitectura Q1: Ejemplos

Ejercicio: Ensamblar MOV R1,0x0003

Efecto	$R1 \leftarrow 0x0003$
Código de operación	0001
Modo Destino	R1 está en modo registro: 100rrr
Modo Origen	

Arquitectura Q1: Ejemplos

Ejercicio: Ensamblar MOV R1,0x0003

Efecto	$R1 \leftarrow 0x0003$
Código de operación	0001
Modo Destino	R1 está en modo registro: 100rrr
Modo Origen	0x0003 está en modo inmediato: 000000

Arquitectura Q1: Ejemplos

Ejercicio: Ensamblar MOV R1,0x0003

Efecto	$R1 \leftarrow 0x0003$
Código de operación	0001
Modo Destino	R1 está en modo registro: 100rrr
Modo Origen	0x0003 está en modo inmediato: 000000



00011000010000000000000000000011

Arquitectura **Q1**: Ejemplos

Ejercicio: ensamblar MOV R1,R6

Arquitectura **Q1**: Ejemplos

Ejercicio: ensamblar MOV R1,R6

Efecto

Arquitectura Q1: Ejemplos

Ejercicio: ensamblar MOV R1,R6

Efecto	R1 $\leftarrow$ R6
--------	--------------------

Arquitectura Q1: Ejemplos

Ejercicio: ensamblar MOV R1,R6

Efecto	R1 $\leftarrow$ R6
Código de operación	

Arquitectura Q1: Ejemplos

Ejercicio: ensamblar MOV R1,R6

Efecto	$R1 \leftarrow R6$
Código de operación	0001

Arquitectura Q1: Ejemplos

Ejercicio: ensamblar MOV R1,R6

Efecto	$R1 \leftarrow R6$
Código de operación	0001
Modo Destino	

Arquitectura **Q1**: Ejemplos

Ejercicio: ensamblar MOV R1,R6

Efecto	$R1 \leftarrow R6$
Código de operación	0001
Modo Destino	R1 está en modo registro: 100001
Modo Origen	

Arquitectura Q1: Ejemplos

Ejercicio: ensamblar MOV R1,R6

Efecto	$R1 \leftarrow R6$
Código de operación	0001
Modo Destino	R1 está en modo registro: 100001
Modo Origen	R6 está en modo registro: 100110

Arquitectura Q1: Ejemplos

Ejercicio: ensamblar MOV R1,R6

Efecto	$R1 \leftarrow R6$
Código de operación	0001
Modo Destino	R1 está en modo registro: 100001
Modo Origen	R6 está en modo registro: 100110



0001100001100110

Arquitectura **Q1**: Ejercicios

Ejercicios

Arquitectura **Q1**: Ejercicios

Ejercicios

- 1 Hacer un programa que multiplique por 12 el valor de R0.
- 2 Hacer un programa que suma R0 con R1 y guarde el resultado en R2

Arquitectura **Q1**: Ejercicios

Ejercicio: ensamblar el siguiente programa

```
SUB R0, R1  
ADD R2, R0  
DIV R2, 7
```

Arquitectura Q1: Ejercicios

Ejercicio: Desensamblar

<i>cadena (hexa)</i>	Codop	Modo Destino	Modo Origen	Origen	Destino
1821					

Arquitectura Q1: Ejercicios

Ejercicio: Desensamblar

cadena (hexa)	Codop	Modo Destino	Modo Origen	Origen	Destino
1821	MOV	Registro	Registro	R1	R0
0860					

Arquitectura Q1: Ejercicios

Ejercicio: Desensamblar

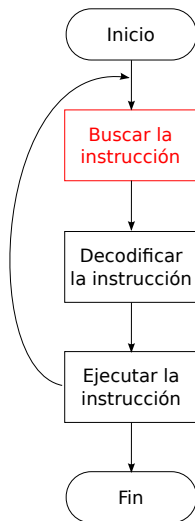
<i>cadena (hexa)</i>	Codop	Modo Destino	Modo Origen	Origen	Destino
1821	MOV	Registro	Registro	R1	R0
0860	MUL	Registro	Registro	R1	R0
28400005					

Arquitectura Q1: Ejercicios

Ejercicio: Desensamblar

<i>cadena (hexa)</i>	Codop	Modo Destino	Modo Origen	Origen	Destino
1821	MOV	Registro	Registro	R1	R0
0860	MUL	Registro	Registro	R1	R0
28400005	ADD	Registro	Inmediato	0005	R1

Ciclo de ejecución de una instrucción



Ciclo de ejecución de una instrucción

Búsqueda de la instrucción

- 1 La UC pide a la memoria un conjunto de bits
- 2 La memoria le envía el sector pedido

Ciclo de ejecución de una instrucción

Búsqueda de la instrucción

- 1 La UC pide a la memoria un conjunto de bits
- 2 La memoria le envía el sector pedido

La UC **sabe** que el sector pedido contiene una instrucción

Ciclo de ejecución de una instrucción

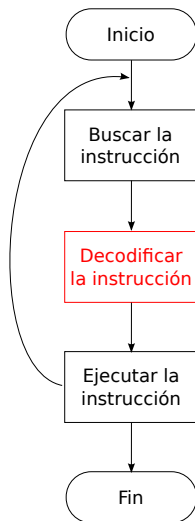
Búsqueda de la instrucción

- 1 La UC pide a la memoria un conjunto de bits
- 2 La memoria le envía el sector pedido

La UC **sabe** que el sector pedido contiene una instrucción

La memoria **no lo sabe**

Ciclo de ejecución de una instrucción

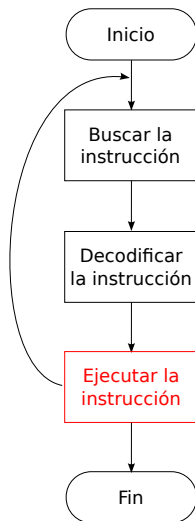


Ciclo de ejecución de una instrucción

Decodificación de la instrucción

- 1 La UC extrae la operación de determinados bits de la cadena leída

Ciclo de ejecución de una instrucción



Ciclo de ejecución de una instrucción

Ejecución de la instrucción

- 1 La UC traduce la operación en señales eléctricas a:
 - la ALU

Ciclo de ejecución de una instrucción

Ejecución de la instrucción

- 1 La UC traduce la operación en señales eléctricas a:
 - la ALU (dándole parámetros y obteniendo resultados)

Ciclo de ejecución de una instrucción

Ejecución de la instrucción

- 1 La UC traduce la operación en señales eléctricas a:
 - la ALU (dándole parámetros y obteniendo resultados)
 - la memoria

Ciclo de ejecución de una instrucción

Ejecución de la instrucción

- 1 La UC traduce la operación en señales eléctricas a:
 - la ALU (dándole parámetros y obteniendo resultados)
 - la memoria (pidiendole la lectura o escritura de bits)

Ciclo de ejecución de una instrucción

Ejecución de la instrucción

- 1 La UC traduce la operación en señales eléctricas a:
 - la ALU (dándole parámetros y obteniendo resultados)
 - la memoria (pidiendole la lectura o escritura de bits)
 - los registros

Ciclo de ejecución de una instrucción

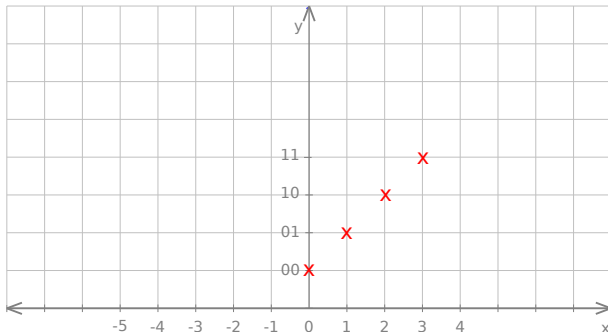
Ejecución de la instrucción

- ❶ La UC traduce la operación en señales eléctricas a:
- la ALU (dándole parámetros y obteniendo resultados)
 - la memoria (pidiendole la lectura o escritura de bits)
 - los registros (poniendoles valor o leyendo el contenido)

Sistemas de numeración enteros

Binario Sin Signo

¿Que números podemos representar en $BSS(2)$?



¿Cómo representar números negativos?

¿Cómo representar números negativos?

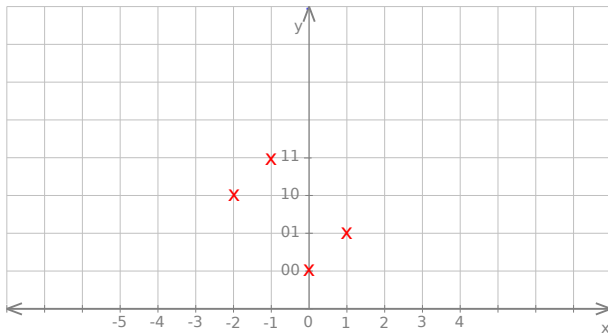
¿Cómo representar números negativos?

¿Cómo representar números negativos?

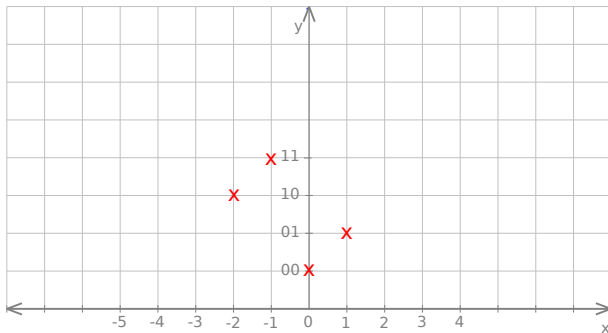


Asociando las cadenas a **otros números**

¿Cómo representar números negativos?



¿Cómo representar números negativos?



Complemento a 2

Complemento a 2

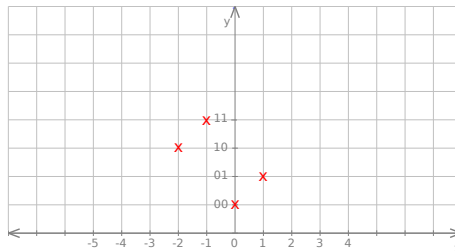
Mecanismo para interpretar una cadena $C=b_{n-1}...b_0$

- Las cadenas se dividen en 2:
 - Las mas **bajas** para los positivos (y el cero)
 - Las mas **altas** para los negativos

Complemento a 2

cadenas bajas 00
(positivos) 01

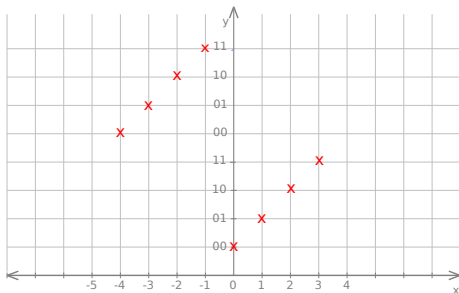
cadenas altas 10
(negativos) 11



Complemento a 2

cadena**s** baja**s** 000
 (positivos) 001
 010
 011

cadena**s** alta**s** 100
 (negativos) 101
 110
 111



Complemento a 2

cadenas bajas	0000	cadenas altas	1010
(positivos)	0001	(negativos)	1011
	0010		1010
	0011		1011
	0100		1100
	0101		1101
	0110		1110
	0111		1111

Interpretación en CA2

Interpretación en CA2

Mecanismo para interpretar una cadena $C=b_{n-1}...b_0$

a) Si comienza con 0 ($b_{n-1}=0$) entonces interpretar como *Binario Sin Signo*

Interpretación en CA2

Mecanismo para interpretar una cadena $C=b_{n-1}...b_0$

- a) Si comienza con 0 ($b_{n-1}=0$) entonces interpretar como *Binario Sin Signo*
- b) Si comienza con 1 ($b_{n-1}=1$) entonces:

Interpretación en CA2

Mecanismo para interpretar una cadena $C=b_{n-1}...b_0$

- a) Si comienza con 0 ($b_{n-1}=0$) entonces interpretar como *Binario Sin Signo*
- b) Si comienza con 1 ($b_{n-1}=1$) entonces:
 - 1 Invertir los bits de la cadena
 - 2 Sumar 1
 - 3 Interpretar como *Binario Sin Signo*
 - 4 Multiplicar por -1

Interpretación en CA2

Ejemplo: Interpretar 1001

a) ¿Comienza con 0?

Interpretación en CA2

Ejemplo: Interpretar 1001

a) ¿Comienza con 0? No

Interpretación en CA2

Ejemplo: Interpretar 1001

- a) ¿Comienza con 0? No
- b) Si comienza con 1

Interpretación en CA2

Ejemplo: Interpretar 1001

- a) ¿Comienza con 0? No
- b) Si comienza con 1
 - 1 Invertir los bits de la cadena: $\Rightarrow$ 0110

Interpretación en CA2

Ejemplo: Interpretar 1001

- a) ¿Comienza con 0? No
- b) Si comienza con 1
 - ❶ Invertir los bits de la cadena: $\Rightarrow 0110$
 - ❷ Sumar 1 $\Rightarrow 0110 + 1 = 0111$

Interpretación en CA2

Ejemplo: Interpretar 1001

a) ¿Comienza con 0? No

b) Si comienza con 1

① Invertir los bits de la cadena: $\Rightarrow 0110$

② Sumar 1 $\Rightarrow 0110 + 1 = 0111$

③ Interpretar como *Binario Sin Signo* $\Rightarrow I(0111) = 7$

Interpretación en CA2

Ejemplo: Interpretar 1001

a) ¿Comienza con 0? No

b) Si comienza con 1

1 Invertir los bits de la cadena: $\Rightarrow 0110$

2 Sumar 1 $\Rightarrow 0110 + 1 = 0111$

3 Interpretar como *Binario Sin Signo* $\Rightarrow I(0111) = 7$

4 Multiplicar por -1 $\Rightarrow I(0111) = -7$

Comparación entre interpretaciones

Cadena de bits	Interpretación en BSS	Interpretación en CA2
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7

Comparación entre interpretaciones

Cadena de bits	Interpretación en BSS	Interpretación en CA2
1000	8	-8
1001	9	-7
1010	10	-6
1011	11	-5
1100	12	-4
1101	13	-3
1110	14	-2
1111	15	-1

Representación en CA2

Representación en CA2

Hay dos casos:

Representación en CA2

Hay dos casos:

Si $X \geq 0$ se representan como en *BSS()*

Representación en CA2

Hay dos casos:

Si $X \geq 0$ se representan como en $BSS()$

- Si $X < 0$ s
- 1 Representar $|X|$ en $BSS(n)$
 - 2 Invertir los bits de la cadena
 - 3 Sumar 1

Representación en CA2

Ejemplo: Representar -1

- $X \geq 0$ no

Representación en CA2

Ejemplo: Representar -1

- $X \geq 0$ no
- $X < 0$
 - 1 Representar $|-1|$ en $BSS(n) \Rightarrow R(1)=0001$

Representación en CA2

Ejemplo: Representar -1

- $X \geq 0$ no
- $X < 0$
 - 1 Representar $| - 1 |$ en $BSS(n) \Rightarrow R(1)=0001$
 - 2 Invertir los bits $\Rightarrow 1110$

Representación en CA2

Ejemplo: Representar -1

- $X \geq 0$ no
- $X < 0$
 - 1 Representar $| - 1 |$ en $BSS(n) \Rightarrow R(1)=0001$
 - 2 Invertir los bits $\Rightarrow 1110$
 - 3 Sumar 1 $\Rightarrow 1111$

Representación en CA2

Ejemplo: Representar -1

- $X \geq 0$ no
- $X < 0$
 - 1 Representar $| - 1 |$ en $BSS(n) \Rightarrow R(1)=0001$
 - 2 Invertir los bits $\Rightarrow 1110$
 - 3 Sumar 1 $\Rightarrow 1111$

Representación en CA2

Ejercicios en CA2(4)

- 1 Representar 0
- 2 Representar 1
- 3 Representar 6
- 4 Representar el -5
- 5 Representar el -8

Representación en CA2

Ejercicios

- 1 $R(0) \Rightarrow 0000_2$
- 2 $R(1) \Rightarrow 0001_2$
- 3 $R(6) \Rightarrow 0110_2$
- 4 $R(-5) \Rightarrow 0101_2 \Rightarrow 1010_2 + 1 \Rightarrow 1011_2$
- 5 $R(-8) \Rightarrow 1000_2 \Rightarrow 0111_2 + 1 \Rightarrow 1000_2$

Rango de CA2

- En un sistema de CA2(2) se pueden representar $2^2 = 4$ números diferentes.

Rango de CA2

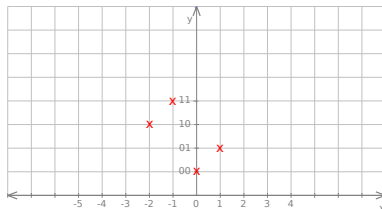
- En un sistema de CA2(2) se pueden representar $2^2 = 4$ números diferentes.
- La mitad de ellos ($4/2 = 2$) son positivos (incluyendo el cero) $\Rightarrow$ irán de 0 a 1.

Rango de CA2

- En un sistema de CA2(2) se pueden representar $2^2 = 4$ números diferentes.
- La mitad de ellos ($4/2 = 2$) son positivos (incluyendo el cero) $\Rightarrow$ irán de 0 a 1.
- La otra mitad son negativos, yendo del -2 al -1 .

Rango de CA2

- En un sistema de CA2(2) se pueden representar $2^2 = 4$ números diferentes.
- La mitad de ellos ($4/2 = 2$) son positivos (incluyendo el cero) $\Rightarrow$ irán de 0 a 1.
- La otra mitad son negativos, yendo del -2 al -1 .



Rango de CA2 con 2 bits es: $[-2, 1]$

Rango de CA2

- En un sistema de CA2 con N bits, se pueden representar 2^N números diferentes.

Rango de CA2

- En un sistema de CA2 con N bits, se pueden representar 2^N números diferentes.
- Positivos: son $2^{N-1} \Rightarrow$ irán de 0 a $2^{N-1} - 1$.

Rango de CA2

- En un sistema de CA2 con N bits, se pueden representar 2^N números diferentes.
- Positivos: son $2^{N-1} \Rightarrow$ irán de 0 a $2^{N-1} - 1$.
- La mitad de ellos (2^{N-1}) son negativos, yendo del -1 al -2^{N-1} .

Rango de CA2

- En un sistema de CA2 con N bits, se pueden representar 2^N números diferentes.
- Positivos: son $2^{N-1} \Rightarrow$ irán de 0 a $2^{N-1} - 1$.
- La mitad de ellos (2^{N-1}) son negativos, yendo del -1 al -2^{N-1} .

Rango de CA2 con N bits es: $[-2^{N-1}; 2^{N-1} - 1]$

Aritmética en CA2

- Las operaciones de suma y resta en CA2 son exactamente las mismas que las de BSS.
- Cambiando la forma de detectar que el resultado cae fuera de rango.

Suma: interpretación

Ejercicio: sumar y verificar resultados

$$\begin{array}{r} + \quad 1001 \\ \quad 0101 \\ \hline \end{array}$$

$$\begin{array}{r} + \quad 0011 \\ \quad 0010 \\ \hline \end{array}$$

$$\begin{array}{r} + \quad 1011 \\ \quad 0111 \\ \hline \end{array}$$

$$\begin{array}{r} + \quad 0011 \\ \quad 0110 \\ \hline \end{array}$$

Suma: interpretación

$$\begin{array}{r} 1 \\ 0 \ 0 \ 0 \ 1 \ - \\ + \ 1001 \\ \ 0101 \\ \hline 1110 \end{array}$$

$$\begin{array}{r} + \ -7 \\ \ 5 \\ \hline \text{¿-2?} \end{array}$$

$$\begin{array}{r} 1 \\ 0 \ 0 \ 1 \ 0 \ - \\ + \ 0011 \\ \ 0010 \\ \hline 0101 \end{array}$$

$$\begin{array}{r} + \ 3 \\ \ 2 \\ \hline 5 \end{array}$$

$$\begin{array}{r} 1 \\ 1 \ 1 \ 1 \ 1 \ - \\ + \ 1011 \\ \ 0111 \\ \hline 0010 \end{array}$$

$$\begin{array}{r} + \ -5 \\ \ 7 \\ \hline 2 \end{array}$$

$$\begin{array}{r} 1 \\ 0 \ 1 \ 1 \ 0 \ - \\ + \ 0011 \\ \ 0110 \\ \hline 1001 \end{array}$$

$$\begin{array}{r} + \ 3 \\ \ 6 \\ \hline \text{¿-7?} \end{array}$$

Resta: interpretación

Ejercicio: restar y verificar resultados

$$\begin{array}{r} - \quad 0011 \\ \quad 0110 \\ \hline \quad 1101 \end{array}$$

$$\begin{array}{r} - \quad 1001 \\ \quad 0101 \\ \hline \quad 0100 \end{array}$$

$$\begin{array}{r} - \quad 0011 \\ \quad 0010 \\ \hline \quad 0001 \end{array}$$

$$\begin{array}{r} - \quad 1011 \\ \quad 0011 \\ \hline \quad 1000 \end{array}$$

Resta: interpretación

$$\begin{array}{r} - \quad 0011 \\ \quad 0110 \\ \hline \quad 1101 \end{array}$$

$$\begin{array}{r} - \quad 3 \\ \quad 6 \\ \hline -3 \end{array}$$

$$\begin{array}{r} - \quad 1001 \\ \quad 0101 \\ \hline \quad 0100 \end{array}$$

$$\begin{array}{r} - \quad -7 \\ \quad 5 \\ \hline \quad 2 \end{array}$$

$$\begin{array}{r} - \quad 0011 \\ \quad 0010 \\ \hline \quad 0001 \end{array}$$

$$\begin{array}{r} - \quad 3 \\ \quad 2 \\ \hline \quad 1 \end{array}$$

$$\begin{array}{r} - \quad 1011 \\ \quad 0011 \\ \hline \quad 1000 \end{array}$$

$$\begin{array}{r} - \quad -5 \\ \quad 3 \\ \hline -8 \end{array}$$

Conclusiones

Conclusiones

Conclusiones

Conclusiones



- Para manejar enteros se utiliza CA2() porque se puede aprovechar la misma ALU
- Para ejecutar un programa se lo debe **Ensambla**
- Hemos hecho tareas manuales (humanas):
 - 1 **Desensamblar**
 - 2 **Representar números en binario**

Conclusiones

¿Preguntas?

Conclusiones

¿Preguntas?



Buzón de sugerencias