

Organización de computadoras

Clase 4

Universidad Nacional de Quilmes

Lic. Martínez Federico

¿Dónde estábamos?



¿Dónde estábamos?

- PC

¿Dónde estábamos?

- PC
- Limitaciones de Q2

¿Dónde estábamos?

- PC
- Limitaciones de Q2
- Flags:
 - ¿Qué?
 - ¿Cómo?
 - ¿Para qué?

¿Dónde estábamos?

- PC
- Limitaciones de Q2
- Flags:
 - ¿Qué?
 - ¿Cómo?
 - ¿Para qué?
- Saltos:
 - ¿Qué?
 - Absolutos vs relativos
 - Condicionales vs incondicionales
 - Etiquetas

¿A dónde queremos llegar?



¿A dónde queremos llegar?

- Operadores lógicos
 - Mascaras

¿A dónde queremos llegar?

- Operadores lógicos
 - Mascaras
- Pila
 - SP
 - Push y pop

¿A dónde queremos llegar?

- Operadores lógicos
 - Mascaras
- Pila
 - SP
 - Push y pop
- Modos de direccionamiento indirecto

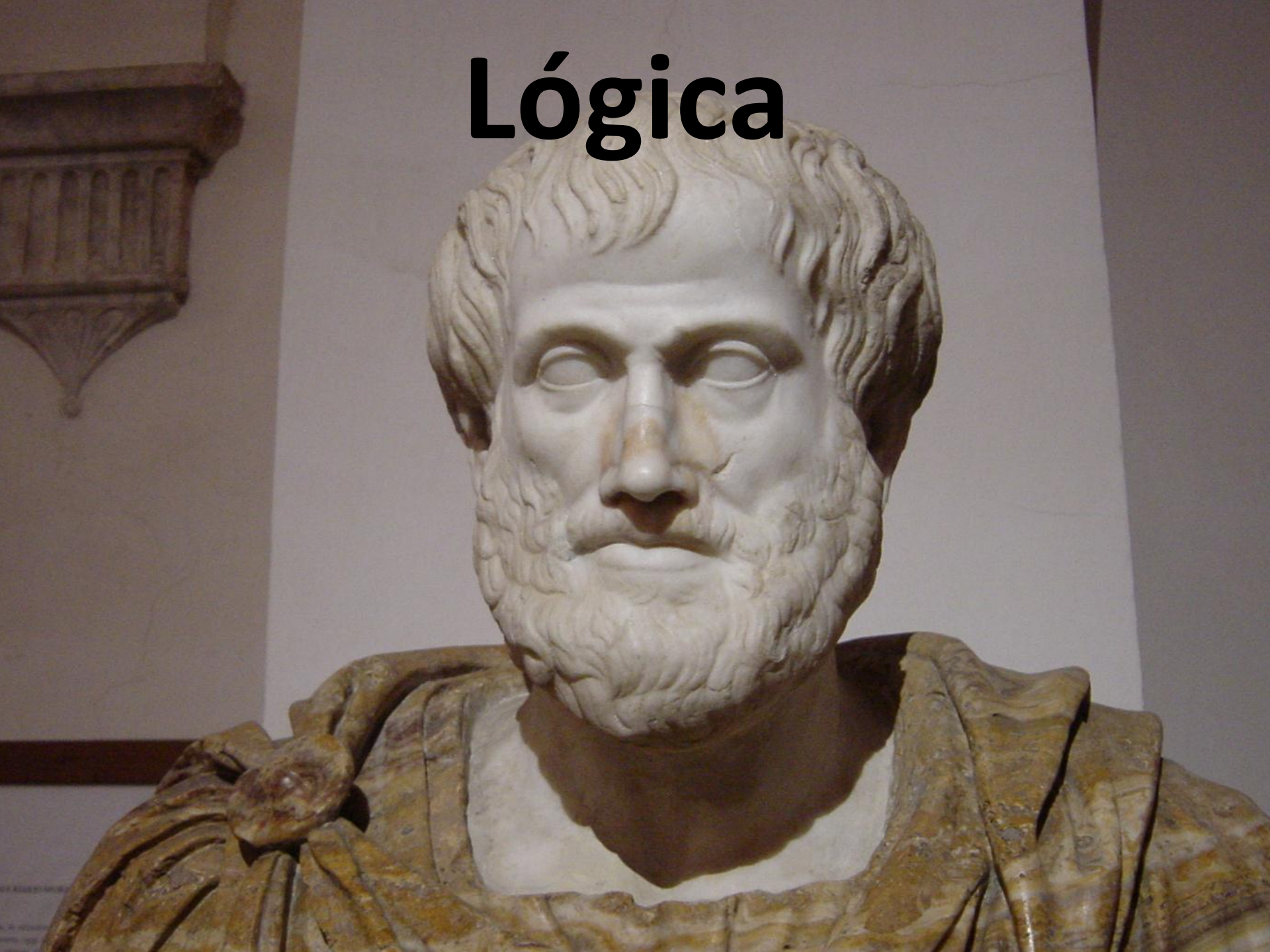
¿A dónde queremos llegar?

- Operadores lógicos
 - Mascaras
- Pila
 - SP
 - Push y pop
- Modos de direccionamiento indirecto
- Arreglos

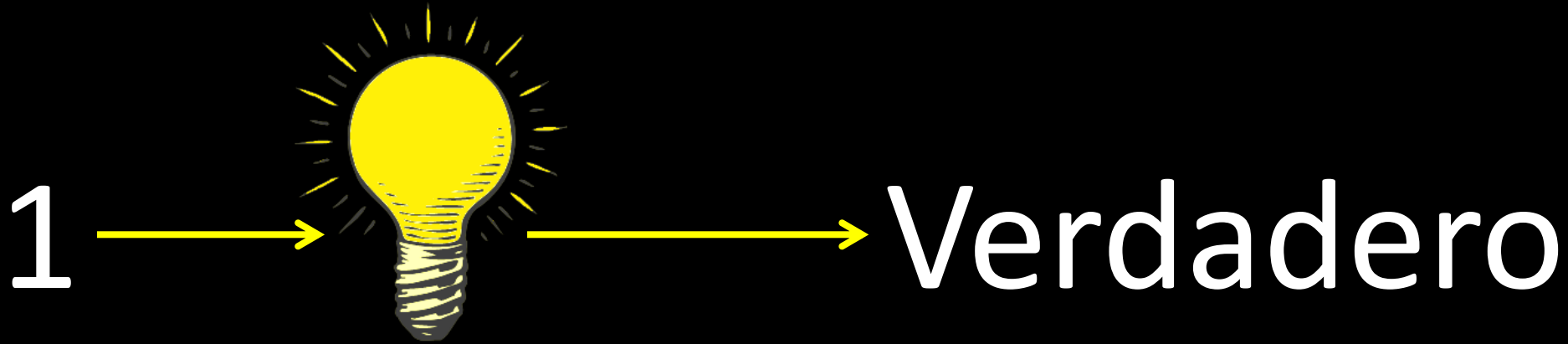
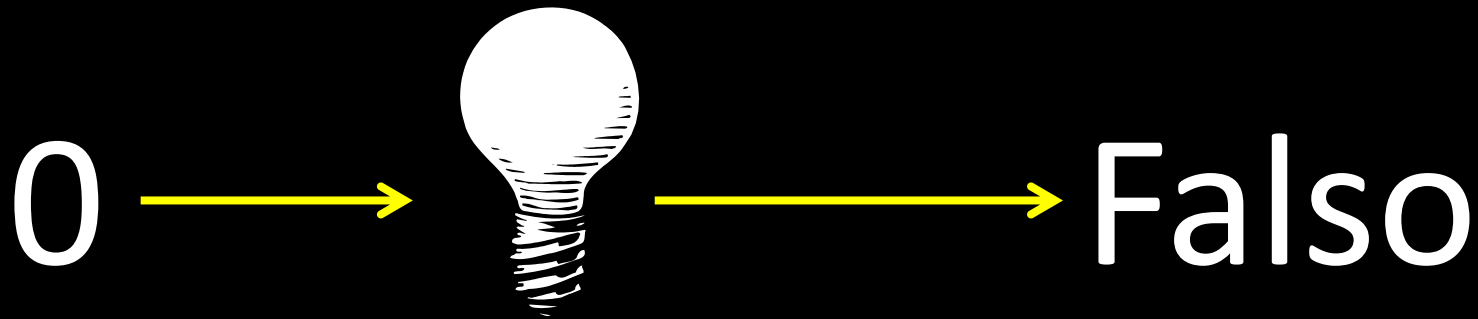
¿A dónde queremos llegar?

- Operadores lógicos
 - Mascaras
- Pila
 - SP
 - Push y pop
- Modos de direccionamiento indirecto
- Arreglos
- Q4

Lógica



Lógica



Operaciones lógicas

- Se aplican sobre cadenas
- Actúan Bit a Bit

Operaciones lógicas

- AND: Realiza el “Y” lógico entre los bits

Operaciones lógicas

- AND: Realiza el “Y” lógico entre los bits

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Operaciones lógicas

- AND: Realiza el “Y” lógico entre los bits

and 1010101010
 1100101100

Operaciones lógicas

- AND: Realiza el “Y” lógico entre los bits

and 1010101010
 1100101100

 1000101000

Operaciones lógicas

- OR: Realiza el “O” lógico entre los bits

Operaciones lógicas

- OR: Realiza el “O” lógico entre los bits

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Operaciones lógicas

- OR: Realiza el “O” lógico entre los bits

or 1010101010
 1100101100

Operaciones lógicas

- OR: Realiza el “O” lógico entre los bits

$$\begin{array}{r} \text{or} \quad 1010101010 \\ 1100101100 \\ \hline 1110101110 \end{array}$$

Operaciones lógicas

- Not: Realiza la negación de los bits

Operaciones lógicas

- Not: Realiza la negación de los bits

A	NOT A
0	1
1	0

Operaciones lógicas

- Not: Realiza la negación de los bits

not 1100101100

Operaciones lógicas

- Not: Realiza la negación de los bits

not 1100101100

0011010011

Operaciones lógicas

- XOR: Realiza el “O exclusivo” lógico entre los bits

Operaciones lógicas

- XOR: Realiza el “O exclusivo” lógico entre los bits

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Operaciones lógicas

- XOR: Realiza el “O exclusivo” lógico entre los bits

xor 1010101010
 1100101100

Operaciones lógicas

- XOR: Realiza el “O exclusivo” lógico entre los bits

```
xor 1010101010  
    1100101100  
-----  
    0110000110
```

Operaciones lógicas

- NAND: Realiza el “Y” lógico entre los bits y niega el resultado

Operaciones lógicas

- NAND: Realiza el “Y” lógico entre los bits y niega el resultado

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

Operaciones lógicas

- NAND: Realiza el “Y” lógico entre los bits y niega el resultado

Nand

1010101010
1100101100
0111010111

Operaciones lógicas

- NOR: Realiza el “O” lógico entre los bits y niega el resultado

Operaciones lógicas

- NOR: Realiza el “O” lógico entre los bits y niega el resultado

A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

Operaciones lógicas

- NOR: Realiza el “O” lógico entre los bits y niega el resultado

Nor

1010101010
1100101100
0001010001



Mascaras

- Cadenas binarias que se combinan con otras mediante operaciones lógicas

Mascaras

- Cadenas binarias que se combinan con otras mediante operaciones lógicas
- Sirven para analizar características de las cadenas

Mascaras

- Con AND:
 - Si queremos preservar el bit original: 1
 - Si queremos dejar un bit en 0: 0

Mascaras

- Con AND:
 - Si queremos preservar el bit original: 1
 - Si queremos dejar un bit en 0: 0
- Con OR:
 - Si nos interesa el bit original: 0
 - Si queremos que quede un 1: 1

Mascaras

Ej: ¿La cadena en R0 es impar?

Mascaras

Ej: ¿La cadena en R0 es impar?

```
AND R0, 0x0001
```

```
JNE TerminabaEnCero
```

Mascaras

Ej: Copiar los 3 bits mas significativos de R0 a R1

Mascaras

Ej: Copiar los 3 bits mas significativos de R0 a R1

```
MOV R1, 0xE000
```

```
AND R1, R0
```

Mascaras

Ej: ¿Son los números de R0 y R1 pares?

Mascaras

Ej: ¿Son los números de R0 y R1 pares?

```
MOV R3, 0x0000
```

```
AND R0, 0x0001
```

```
OR R3, R0
```

```
AND R1, 0x0001
```

```
OR R3, R1
```

```
CMP R3, 0
```

```
JE sonTodosPares
```


Mascaras

- Permisos de archivos en Linux:
 - 3 bits: leer (r) , escribir (w), ejecutar (x)
 - Usuario, Grupo y Otros
 - 111 111 111 le da permisos a todos para hacer cualquier cosa
 - ¿Cómo saber si otro usuario del grupo del archivo puede escribirlo?
 - AND 000 010 000

Pila





Pila

- Push: Agrega un elemento en el tope de la pila

Pila

- Push: Agrega un elemento en el tope de la pila
- Pop: Saca el elemento del tope

Stack pointer

- Registro interno

Stack pointer

- Registro interno
- Apunta al tope de la pila (Dirección)

Stack pointer

Dirección	Contenido
0xFFFF0	0x0000
0xFFFF1	0x0000
SP → 0xFFFF2	0x0000
0xFFFF3	0x0A53
0xFFFF4	0x1111

Stack pointer

SP →

Dirección	Contenido
0xFFFF0	0x0000
0xFFFF1	0x0000
0xFFFF2	0x0000
0xFFFF3	0x0A53
0xFFFF4	0x1111

Pop

Stack pointer

SP →

Dirección	Contenido
0xFFFF0	0x0000
0xFFFF1	0x0000
0xFFFF2	0x0000
0xFFFF3	0x0A53
0xFFFF4	0x1111

Pop
0x0A53

Stack pointer

SP →

Dirección	Contenido
0xFFFF0	0x0000
0xFFFF1	0x0000
0xFFFF2	0x0000
0xFFFF3	0x0A53
0xFFFF4	0x1111

Push
0xFEDE

Stack pointer

SP →

Dirección	Contenido
0xFFFF0	0x0000
0xFFFF1	0x0000
0xFFFF2	0x0000
0xFFFF3	0xFEDE
0xFFFF4	0x1111

Push
0xFEDE

Stack pointer

SP →

Dirección	Contenido
0xFFFF0	0x0000
0xFFFF1	0x0000
0xFFFF2	0x0000
0xFFFF3	0xFEDE
0xFFFF4	0x1111

Push
0x78D3

Stack pointer

SP →

Dirección	Contenido
0xFFFF0	0x0000
0xFFFF1	0x0000
0xFFFF2	0x78D3
0xFFFF3	0xFEDE
0xFFFF4	0x1111

Push
0x78D3

Pila

- La arquitectura decide el valor inicial del SP
- La memoria que ocupa la pila es memoria común y corriente
- La máquina no “válida” si al pushear no se pierden algún dato

Ejercicio

- Sumar los dos números que están en la pila y devolver el resultado en la misma

Ejercicio

- Sumar los dos números que están en la pila y devolver el resultado en la misma

POP R0

POP R1

ADD R1, R0

PUSH R1

A photograph of a person with dark hair, wearing a black jacket over a light-colored shirt, sleeping in a chair at a desk. Their head is tilted back, and their mouth is open. A large, white, cloud-shaped thought bubble is positioned above their head, containing the text '¿Había tarea de orga?'. Three small white circles lead from the bubble to the person's head. On the desk in front of them is a large, beige, multi-line keyboard. To the right of the keyboard, there are some papers and a small white cup. The background consists of vertical wooden slats.

¿Había tarea
de orga?

Ejercicio

- A partir de 0x000F hay varios números. El último de ellos es 0. Hacer un programa que sume todos esos números en R3



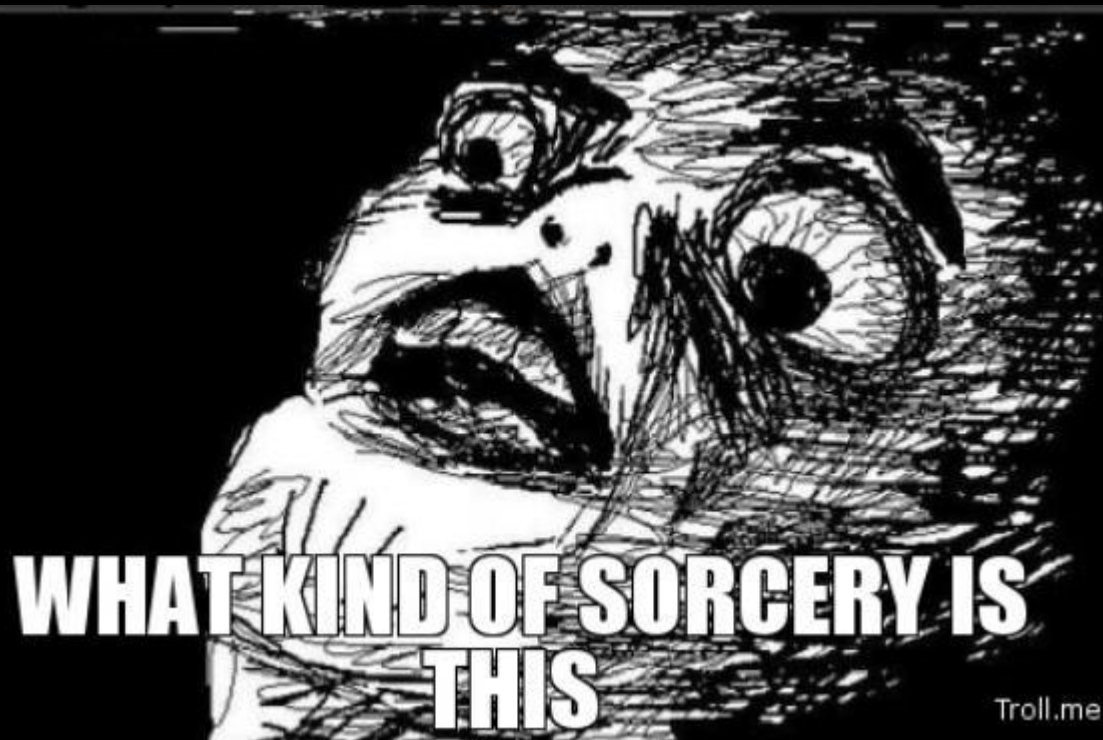
¡¡ISABEL!!

Ejercicio

- Se puede hacer si el programa modifica su propio código a medida que se ejecuta

Ejercicio

- Se puede hacer si el programa modifica su propio código a medida que se ejecuta





Ejercicio

- Nos gustaría poder referenciar a memoria sin usar una constante

Ejercicio

- Nos gustaría poder referenciar a memoria sin usar una constante
- Necesitamos un nuevo modo de direccionamiento

Ejercicio

- Nos gustaría poder referenciar a memoria sin usar una constante
- Necesitamos un nuevo modo de direccionamiento

Indirecto registro

- Se denota $[R_i]$

Indirecto registro

- Se denota $[R_i]$
- Semántica: El valor que está contenido en la celda cuya dirección está en R_i

Indirecto

- Se denota $[[Cte]]$

Indirecto

- Se denota $[[Cte]]$
- Semántica: El valor que está contenido en la celda cuya dirección está contenida en la celda Cte

Indirecto

- Requiere 2 accesos a memoria para obtener el operando
- Para guardarlo se necesita 1 solo, porque ya queda calculada la dirección de la celda

Ejercicio

- A partir de 0x000F **hay varios números**. El último de ellos es 0. Hacer un programa que sume todos esos números en R3

Arreglos

- Posiciones de memoria consecutivas
- Colección de datos
- Los elementos no necesariamente ocupan una celda
- El tamaño del arreglo es la cantidad de elementos

Arreglos

- Ejemplo:

Posición	Contenido
0xF000	0x1020
0xF001	0xDCE3
0xF002	0xE451
0xF003	0x29C8

Arreglos

- Ejemplo:

Posición	Contenido
0xF000	0x1020
0xF001	0xDCE3
0xF002	0xE451
0xF003	0x29C8

Un arreglo de 4 números de 16 bits

Arreglos

- Ejemplo:

Posición	Contenido
0xF000	0x1020
0xF001	0xDCE3
0xF002	0xE451
0xF003	0x29C8

Un arreglo de 4 números de 16 bits

Un arreglo de 8 números de 8 bits

Arreglos

- Ejemplo:

Posición	Contenido
0xF000	0x1020
0xF001	0xDCE3
0xF002	0xE451
0xF003	0x29C8

Un arreglo de 4 números de 16 bits

Un arreglo de 8 números de 8 bits

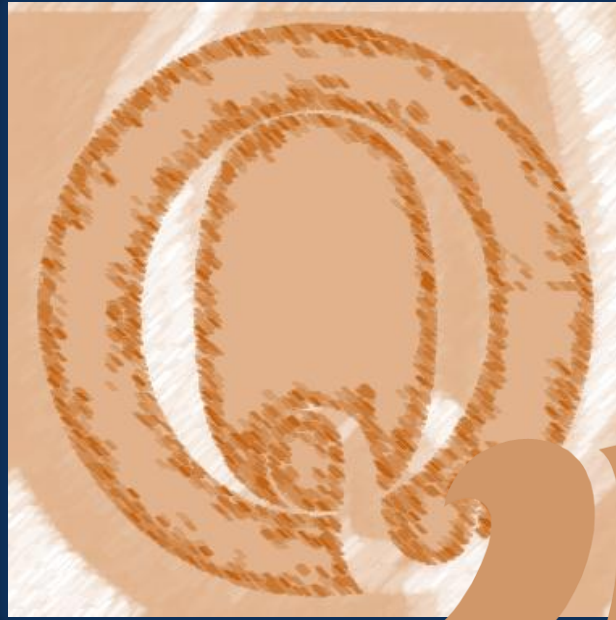
Un arreglo de 2 números de 32 bits

Arreglos

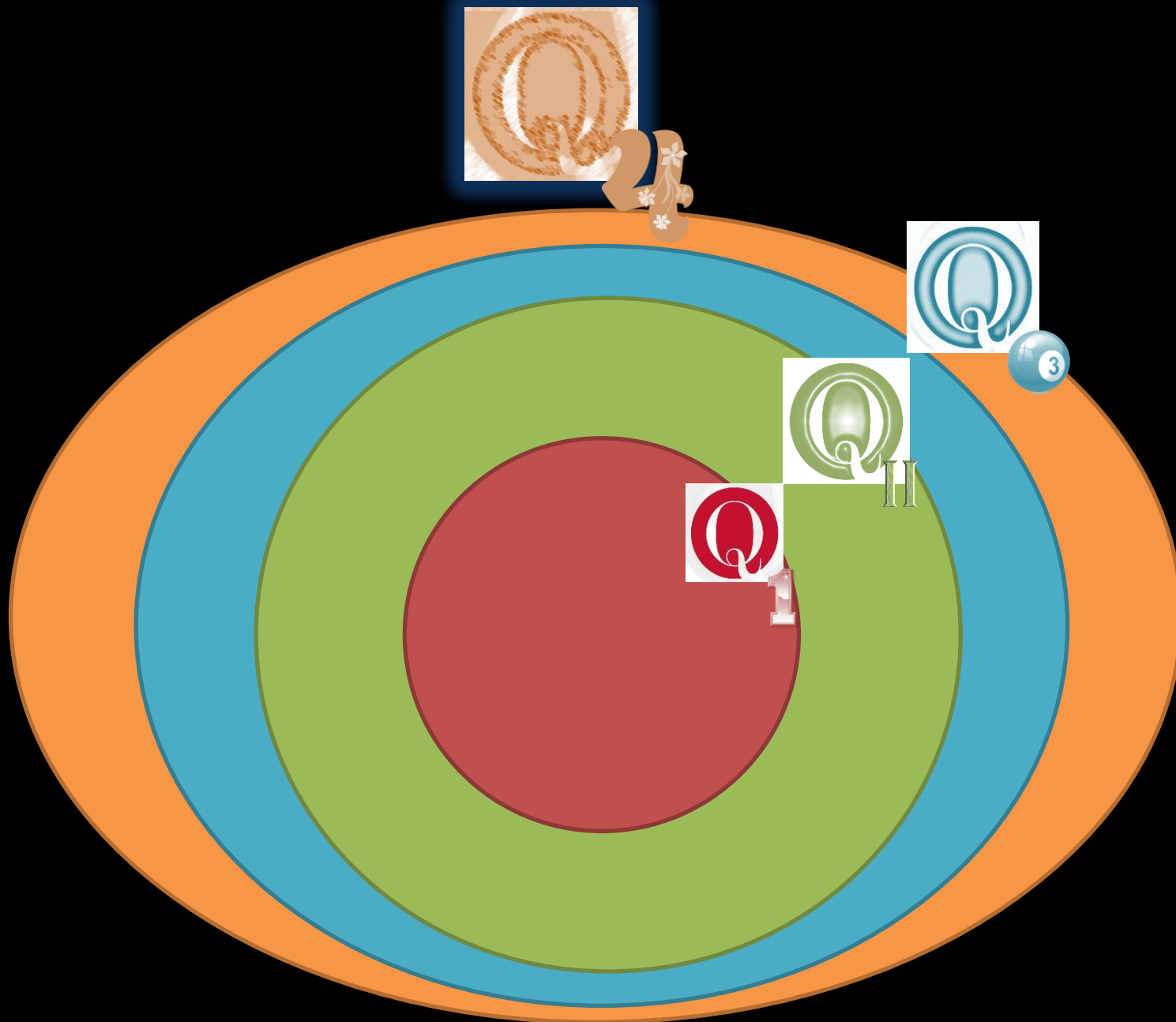
- ¿Cómo saber donde termina el arreglo?
 - Nos dicen el tamaño
 - Nos dicen que termina con algún valor especial (i.e: 0)

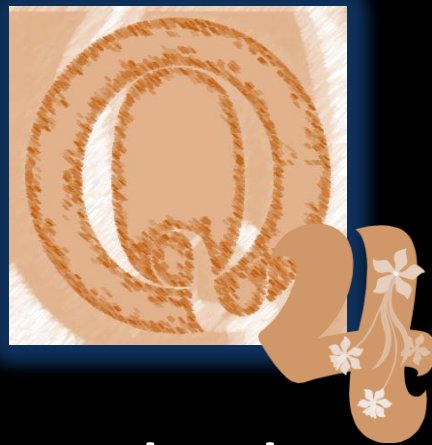
Si te gustaron





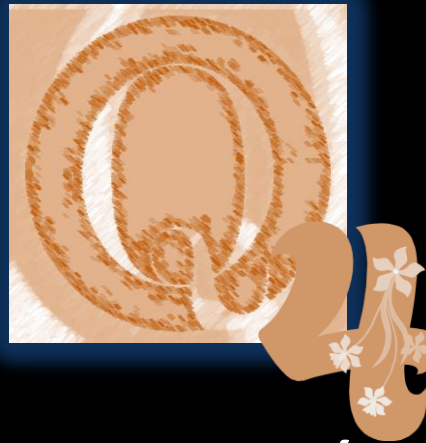
(In)Directamente te va a encantar!





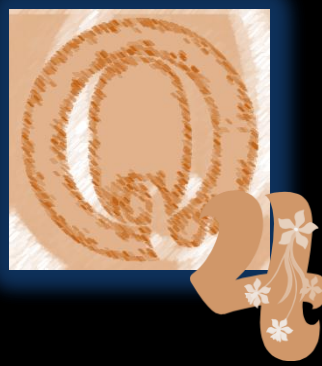
- Nuevos modos de direccionamiento

Modo	Código
Inmediato	000000
Registro	100RRR
Directo	001000
Indirecto	011000
Registro Indirecto	110RRR

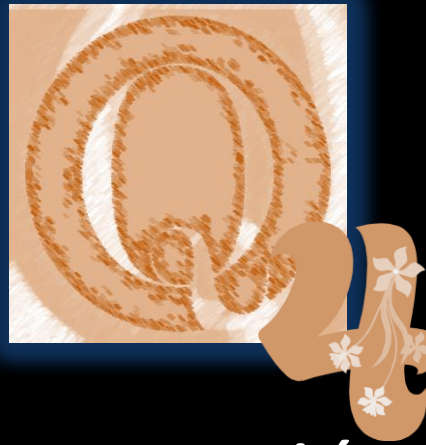


- Formato de instrucción:
 - Instrucciones tipo 1:

Cod Op (4bits)	Modo Destino (6 bits)	Modo origen (6 bits)	Destino (16 bits)	Origen (16 bits)
-------------------	--------------------------	-------------------------	----------------------	---------------------



Operación	Código	Efecto
MUL	0000	$\text{Dest} \leftarrow \text{Dest} * \text{Origen}$
MOV	0001	$\text{Dest} \leftarrow \text{Origen}$
ADD	0010	$\text{Dest} \leftarrow \text{Dest} + \text{Origen}$
SUB	0011	$\text{Dest} \leftarrow \text{Dest} - \text{Origen}$
DIV	0111	$\text{Dest} \leftarrow \text{Dest} \% \text{Origen}$
CMP	0110	Modifica los Flags según el resultado de $\text{Dest} - \text{Origen}$
AND	0100	$\text{Dest} \leftarrow \text{Dest AND Origen}$
OR	0100	$\text{Dest} \leftarrow \text{Dest OR Origen}$



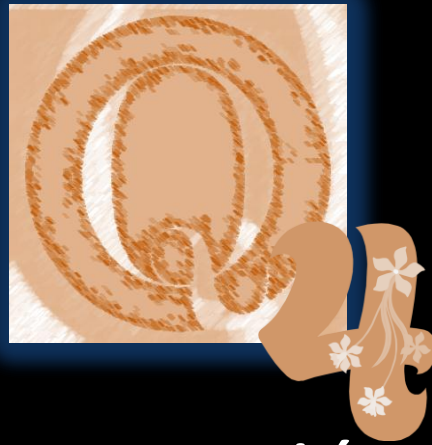
- Formato de instrucción:
 - Instrucciones tipo 2:

Cod Op (4 bits)	Relleno (000000)	Modo Origen (6 bits)	Origen (16 bits)
--------------------	---------------------	-------------------------	---------------------



- Operaciones tipo 2:

Operación	Código	Efecto
JMP	1010	$PC \leftarrow \text{Origen}$
PUSH	1110	$SP \leftarrow \text{Origen}; SP \leftarrow SP - 1$

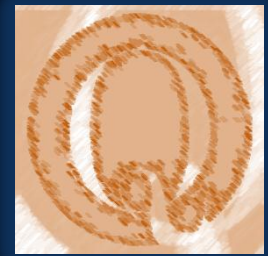


- Formato de instrucción:

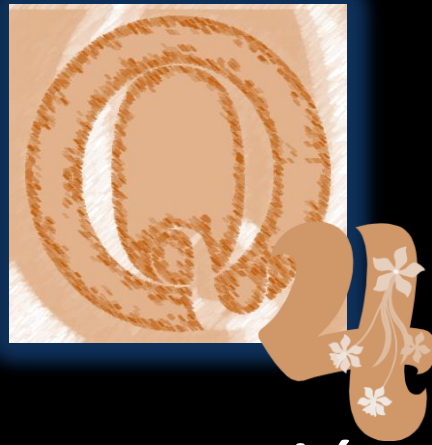
- Instrucciones tipo 3:

Prefijo (1111)	Cod Op (4 bits)	Desplazamiento (8 bits)
---------------------------------	----------------------------------	--

- Operaciones tipo 3:



Salto	Codop	Descripción	Condición
JE	0001	Igual / Cero	Z
JNE	1001	No igual	$\neg Z$
JLE	0010	Menor o igual con signo	$Z + (N \oplus V)$
JG	1010	Mayor con signo	$\neg(Z + (N \oplus V))$
JL	0011	Menor con signo	$N \oplus V$
JGE	1011	Mayor o igual con Signo	$\neg (N \oplus V)$
JLEU	0100	Menor o igual sin signo	$C + Z$
JGU	1100	Mayor sin signo	$\neg(C + Z)$
JCS	0101	Menor sin signo	C
JNEG	0110	Negativo	N
JVS	0111	Overflow	V



- Formato de instrucción:
 - Instrucciones tipo 4:

Cod Op (4 bits)	Modo Destino (6 bits)	Relleno (000000)	Destino (16 bits)
--------------------	--------------------------	---------------------	----------------------



- Operaciones tipo 4:

Operación	Código	Efecto
NOT	1001	$PC \leftarrow \text{NOT Dest}$
POP	1101	$SP \leftarrow SP + 1; \text{Dest} \leftarrow [SP]$

Ejercicio

- A partir de 0x000F hay varios números. El último de ellos es 0. Hacer un programa que sume todos esos números en R3

Ejercicio

- En 0x0010 comienza un arreglo de 14 posiciones. Buscar el valor del máximo elemento

Ejercicio

- En [R0] comienza un arreglo cuyo tamaño esta en R1. Hacer un programa que devuelva 1 en R2 si el número que está en R3 pertenece al arreglo y 0 si no.

Ejercicio

- Realizar un programa que multiplique dos números sin usar mul.

Ejercicio

- Contar los accesos a memoria del siguiente programa:

ADD [R2], 0x0001

SUB [[0x002E]], [R0]

AND [[0x002E]], [[0x00FF]]

OR [R5], R4

PUSH [R3]

POP R4

¿A dónde llegamos?



¿A dónde llegamos?

- Lógica:
 - Operadores
 - Mascaras

¿A dónde llegamos?

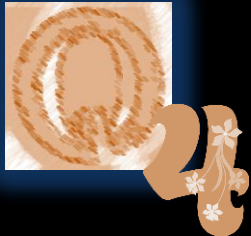
- Lógica:
 - Operadores
 - Mascaras
- Direccionamientos indirectos:
 - Motivación
 - Indirecto registro
 - Indirecto

¿A dónde llegamos?

- Lógica:
 - Operadores
 - Mascaras
- Direccionamientos indirectos:
 - Motivación
 - Indirecto registro
 - Indirecto
- Arreglos (Arrays)
 - Tamaño

¿A dónde llegamos?

- Lógica:
 - Operadores
 - Mascaras
- Direccionamientos indirectos:
 - Motivación
 - Indirecto registro
 - Indirecto
- Arreglos (Arrays)
 - Tamaño



¿Preguntas?



Graciela

