ORGA1: Ciclo de Instrucciones

Organización del Computador I

Facundo Pessacg

Departamento de Computación - FCEyN
UBA

Segundo Cuatrimestre 2018 - Turno Tarde

Conceptos claves del día de hoy

1. Tamaño de memoria

Conceptos claves del día de hoy

1. Tamaño de memoria
2. Dirección de memoria

Conceptos claves del día de hoy

1. Tamaño de memoria
2. Dirección de memoria
3. Unidad direccionable

Conceptos claves del día de hoy

1. Tamaño de memoria
2. Dirección de memoria
3. Unidad direccionable
4. Instrucción

Conceptos claves del día de hoy

1. Tamaño de memoria
2. Dirección de memoria
3. Unidad direccionable
4. Instrucción
5. Codificación de una instrucción

Tamaño, dirección, unidad direccionable

Bienvenido al Mundo Línea. Acá las personas ocupan 1m de ancho, y nada tiene altura ni profundidad.

Tamaño, dirección, unidad direccionable

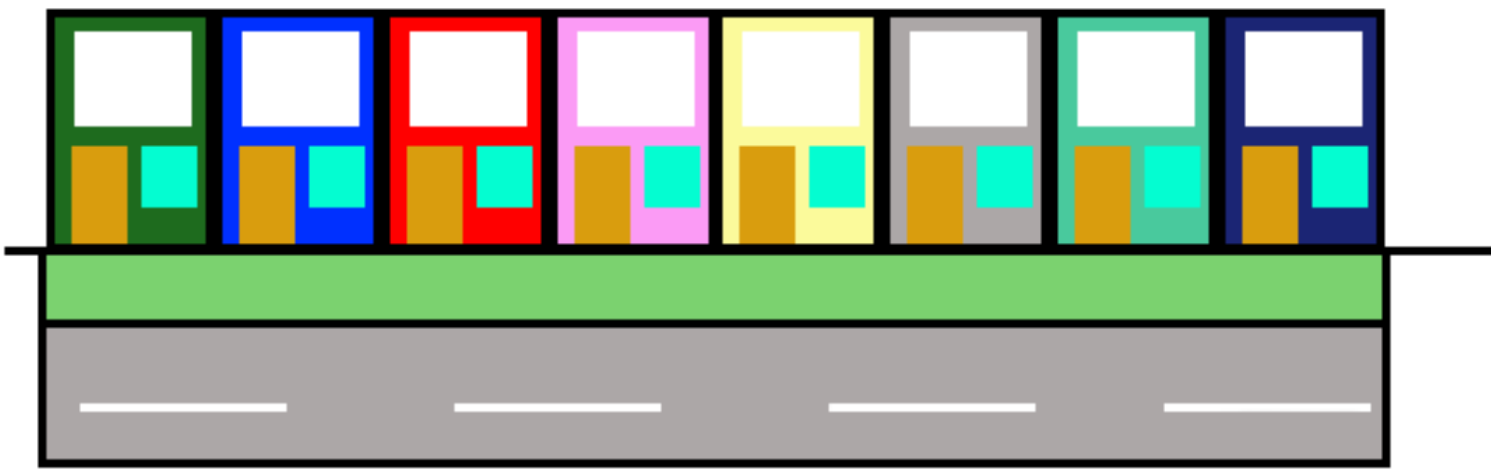
Bienvenido al Mundo Línea. Acá las personas ocupan 1m de ancho, y nada tiene altura ni profundidad. Supongamos una cuadra de unos 32m en este mundo.

Tamaño, dirección, unidad direccionable

Agreguemos casas en los 32 metros de la cuadra sabiendo que cada casa tiene 4m de ancho.

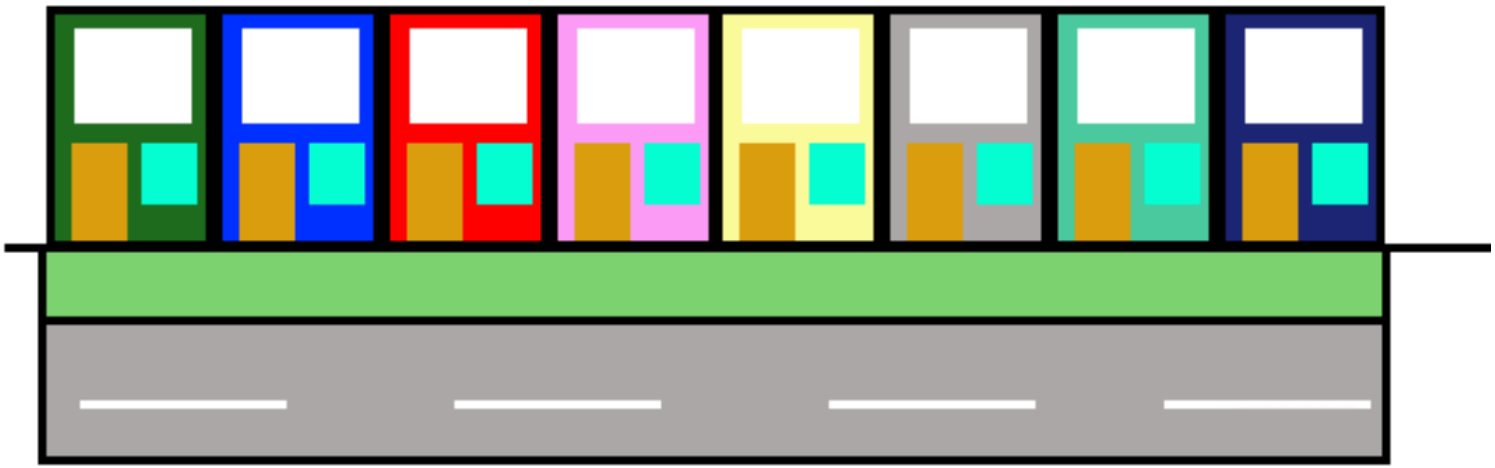
Tamaño, dirección, unidad direccionable

Agreguemos casas en los 32 metros de la cuadra sabiendo que cada casa tiene 4m de ancho.



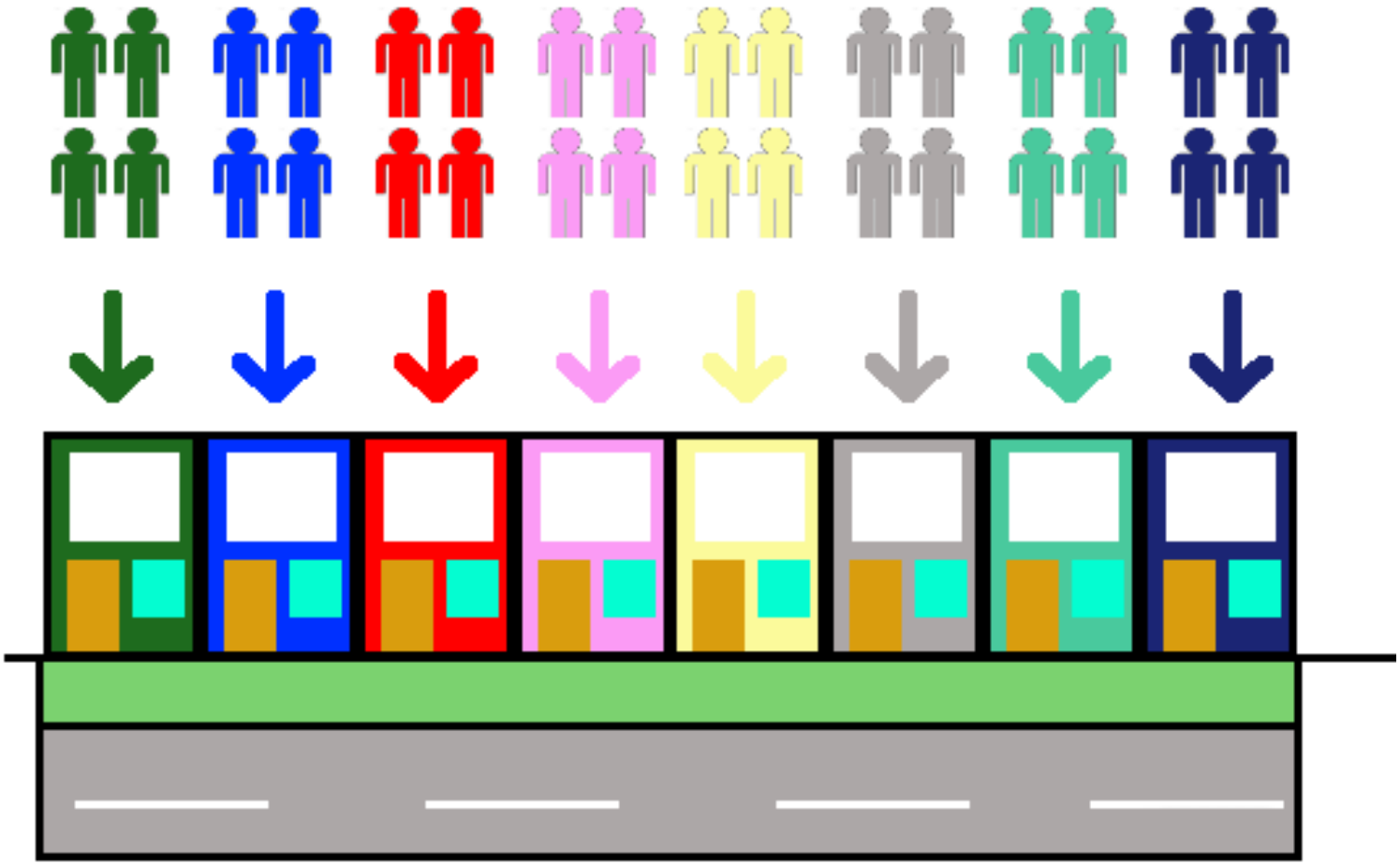
Tamaño, dirección, unidad direccionable

Como una persona ocupa 1m de ancho, y estamos en Mundo Línea donde nada tiene altura ni profundidad, la capacidad de cada casa es de 4 personas. ¿Cuántas personas entran en total en la cuadra?



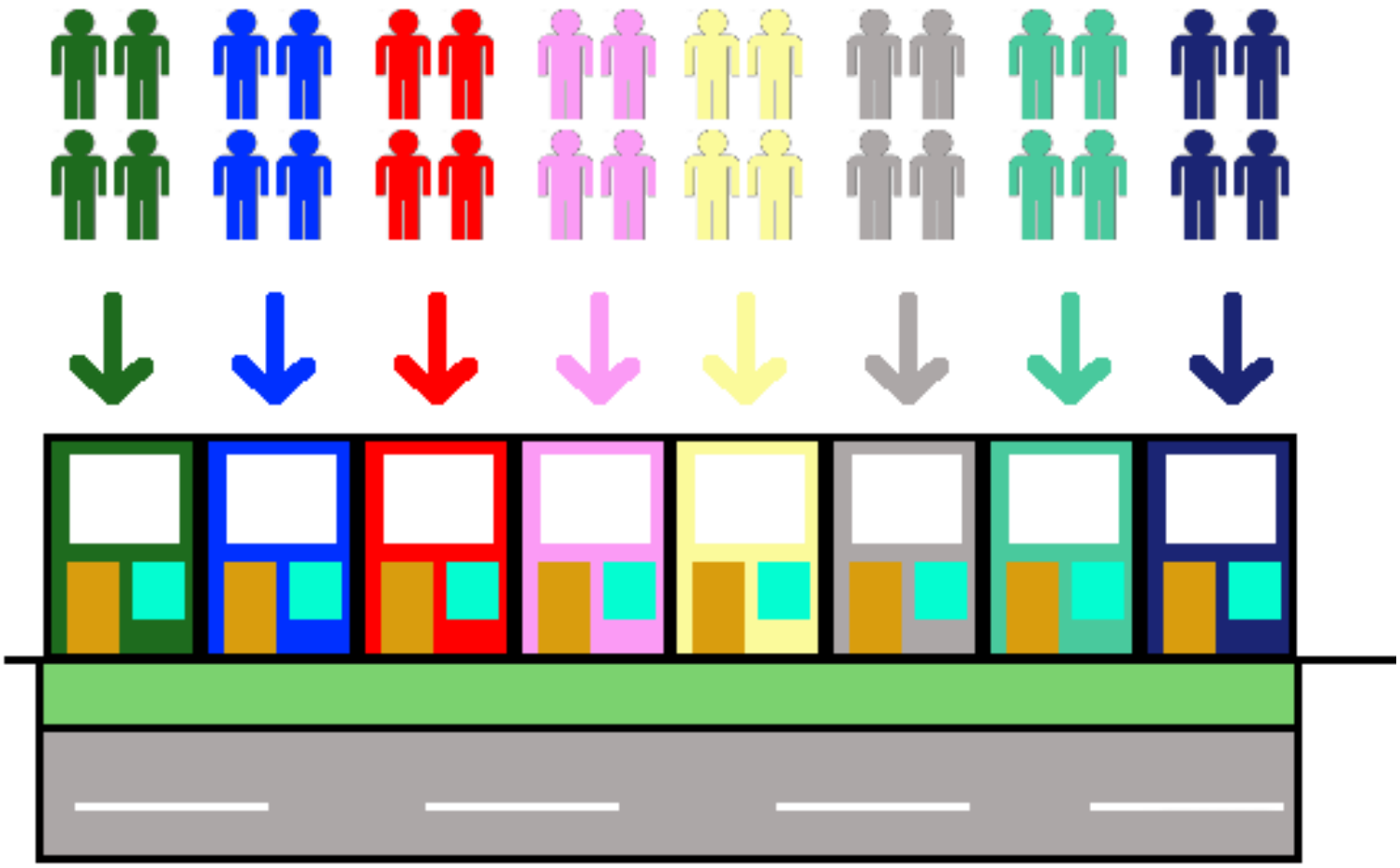
Tamaño, dirección, unidad direccionable

Como una persona ocupa 1m de ancho, y estamos en Mundo Línea donde nada tiene altura ni profundidad, la capacidad de cada casa es de 4 personas. ¿Cuántas personas entran en total en la cuadra?



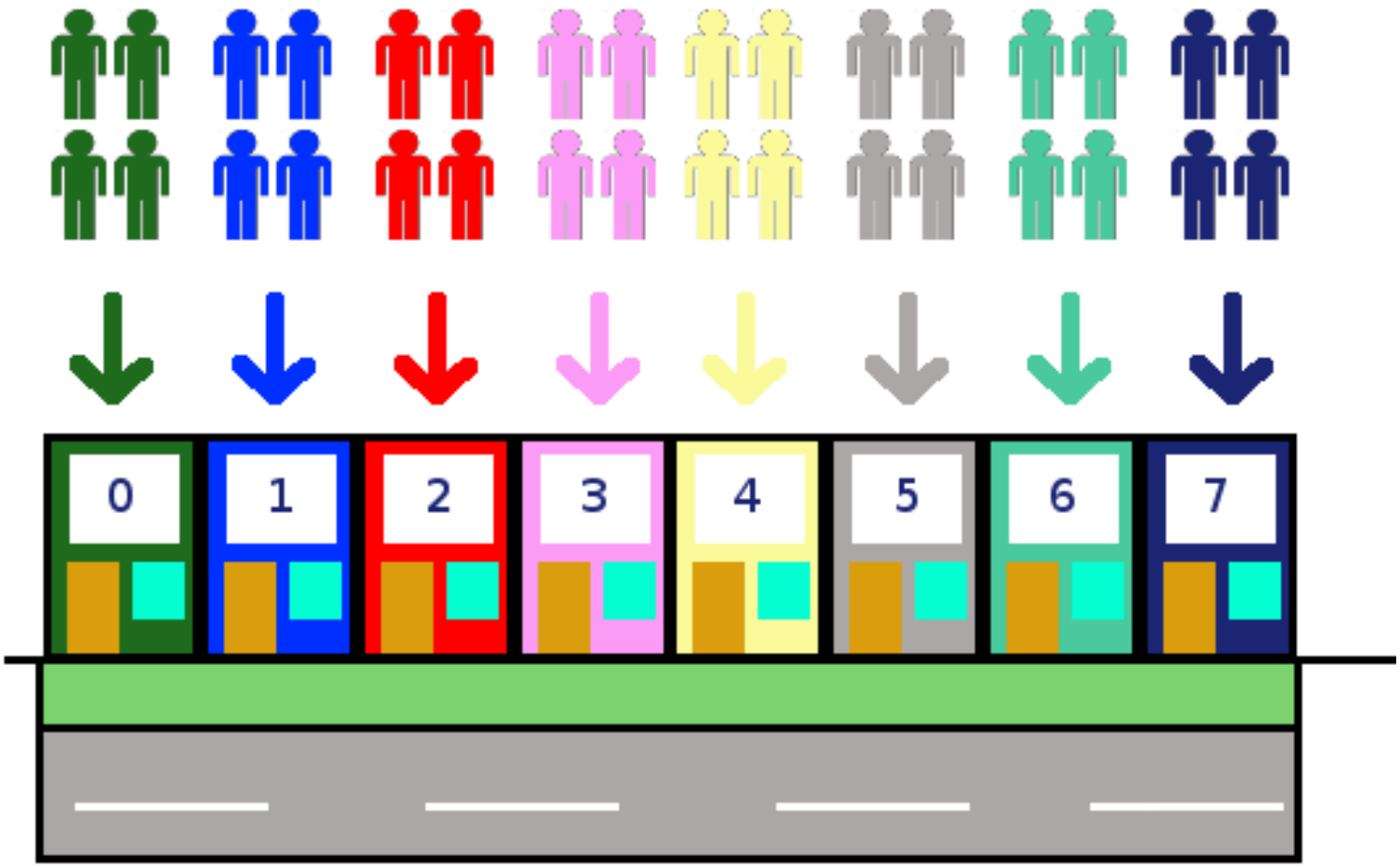
Tamaño, dirección, unidad direccionable

Numeremos las casas.



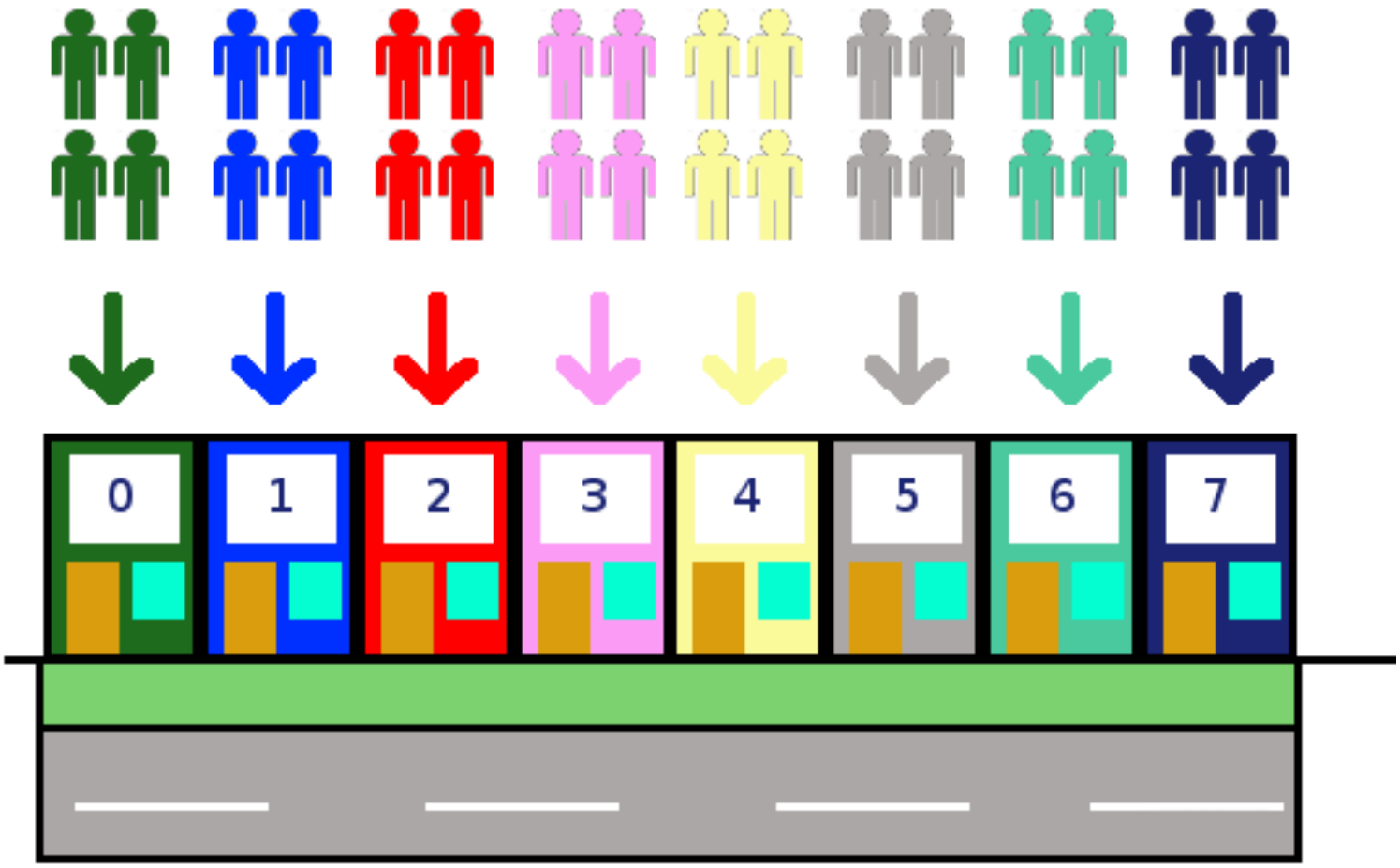
Tamaño, dirección, unidad direccionable

Numeremos las casas.



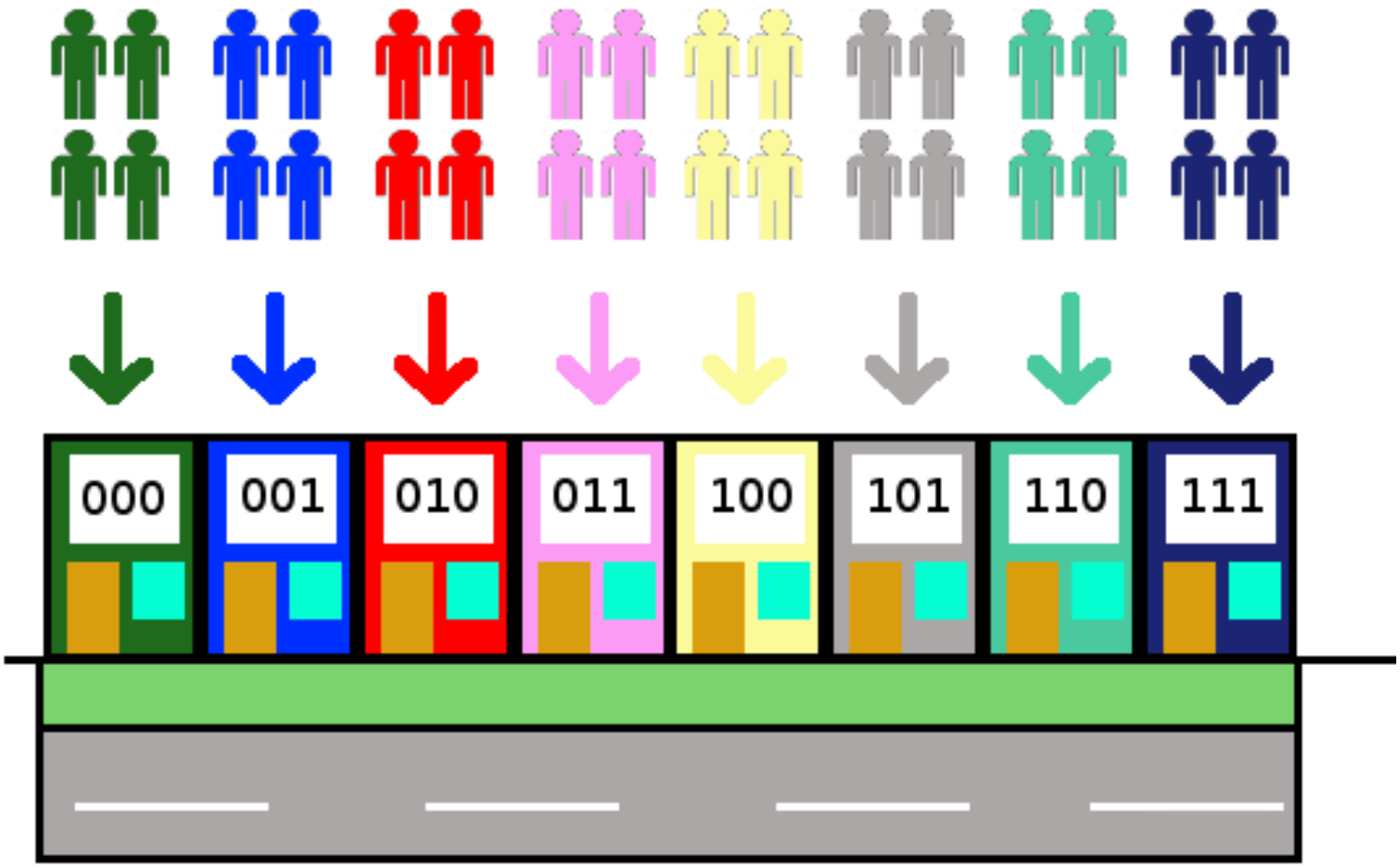
Tamaño, dirección, unidad direccionable

¿Cuántos bits necesitamos para identificar las casas?



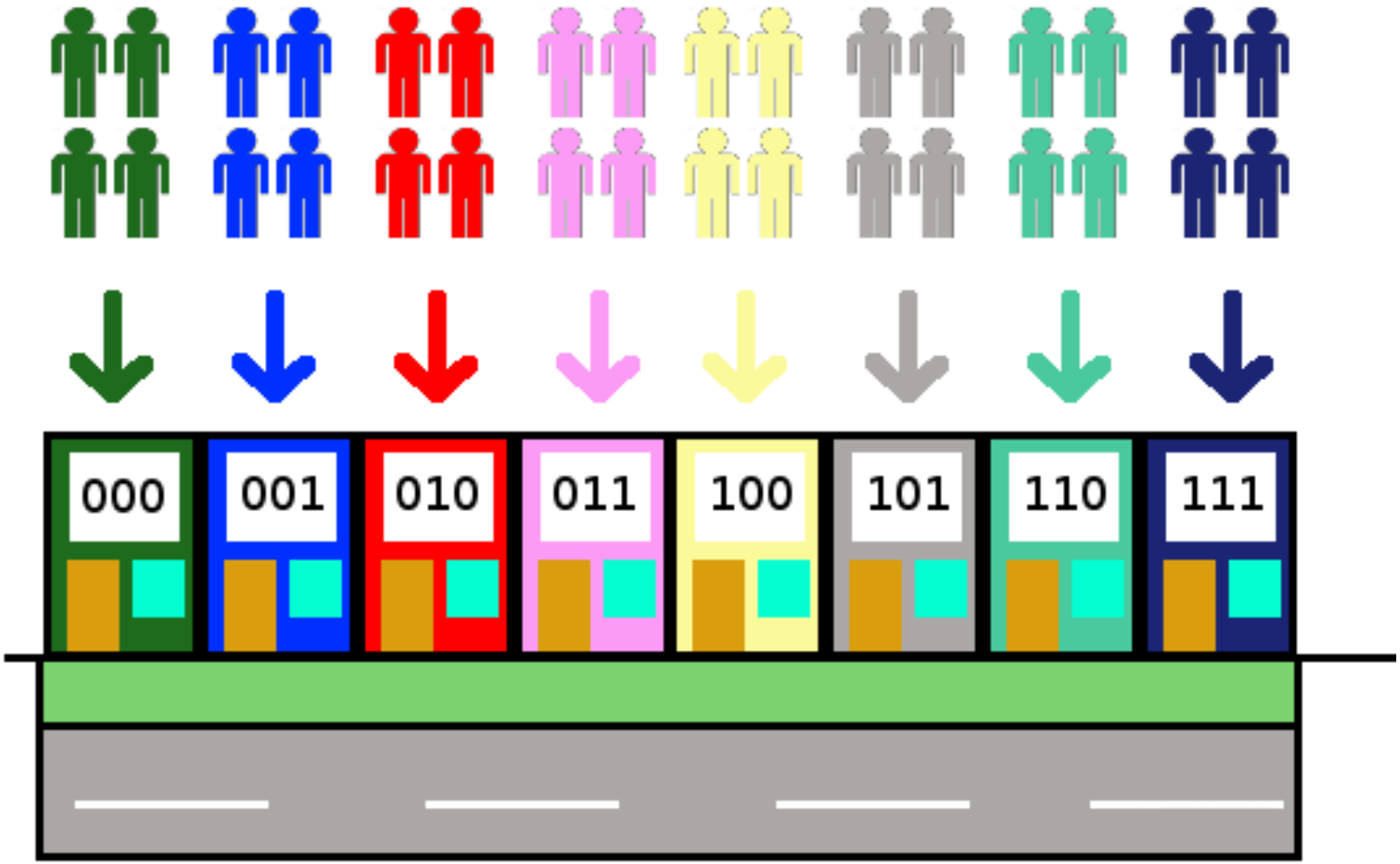
Tamaño, dirección, unidad direccionable

¿Cuántos bits necesitamos para identificar las casas?



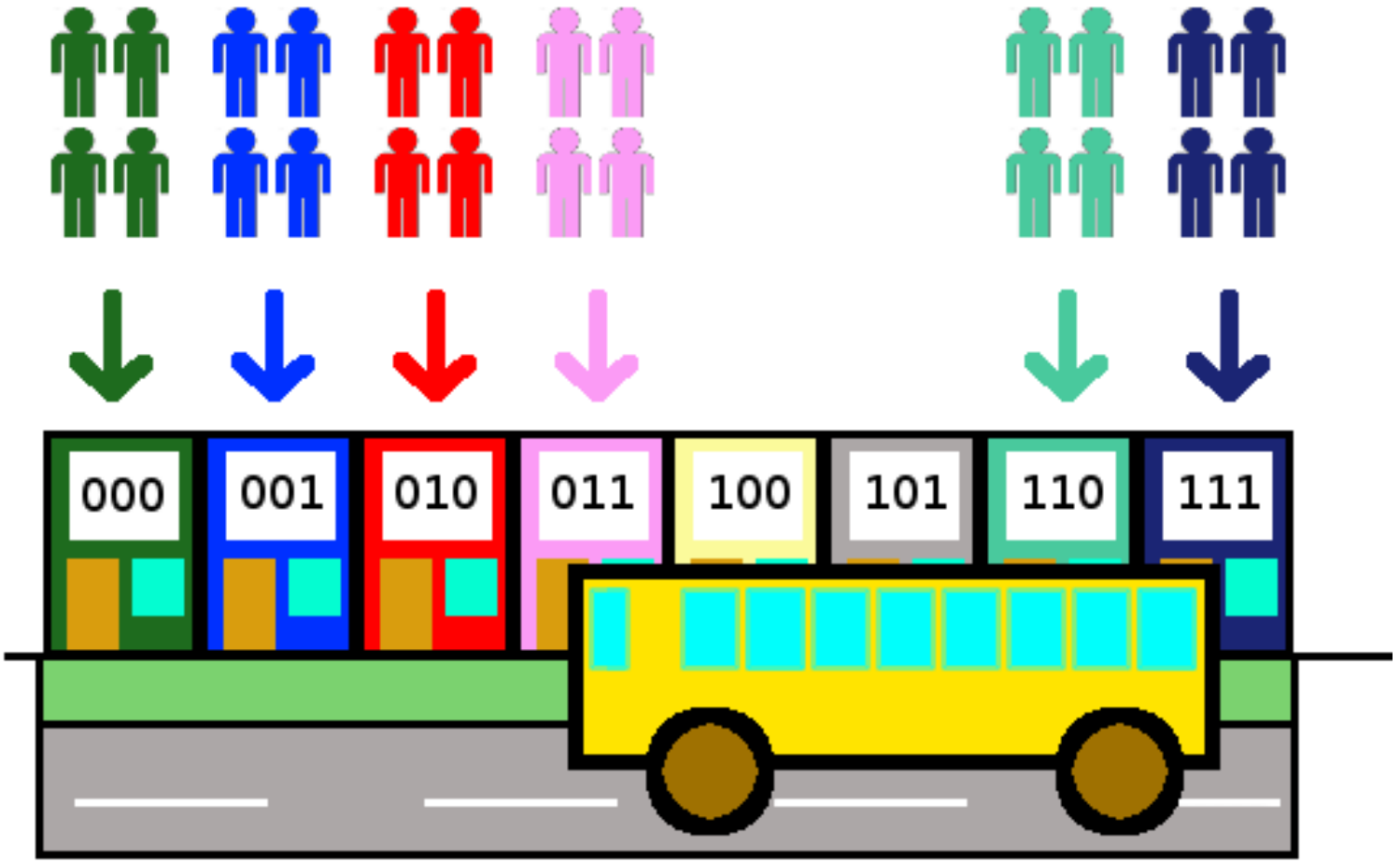
Tamaño, dirección, unidad direccionable

Para ir a trabajar hay un colectivo con lugar para 8 personas que solamente arranca cuando esta lleno. Lleva siempre dos casas contiguas completas.



Tamaño, dirección, unidad direccionable

Para ir a trabajar hay un colectivo con lugar para 8 personas que solamente arranca cuando esta lleno. Lleva siempre dos casas contiguas completas.



Tamaño, dirección, unidad direccionable

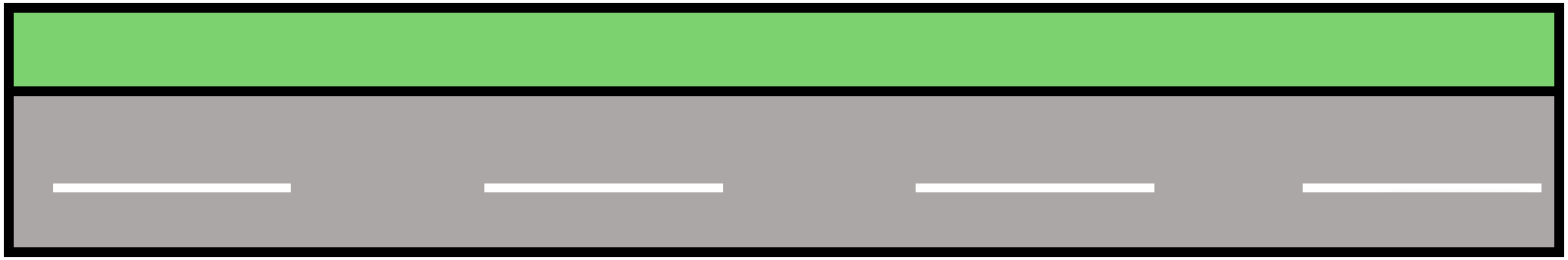
¿De qué estamos hablando? (toma 2)

Supongamos que tenemos otra cuadra de 40m.

Tamaño, dirección, unidad direccionable

¿De qué estamos hablando? (toma 2)

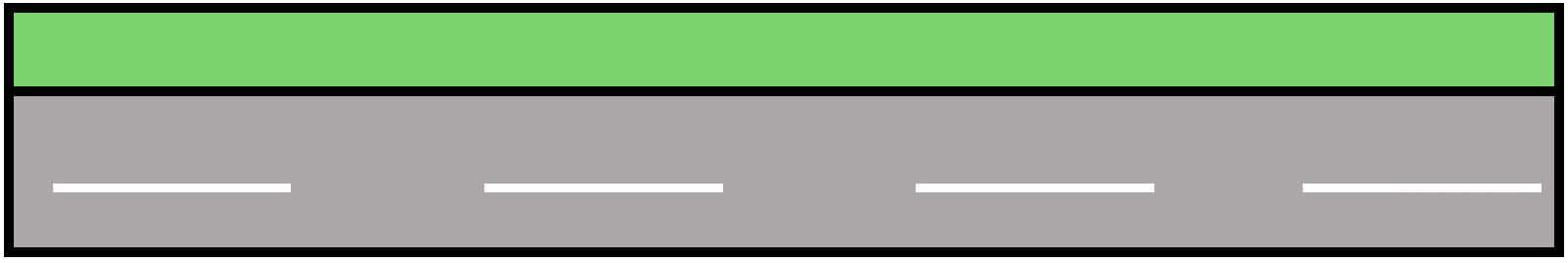
Supongamos que tenemos otra cuadra de 40m.



Tamaño, dirección, unidad direccionable

¿De qué estamos hablando? (toma 2)

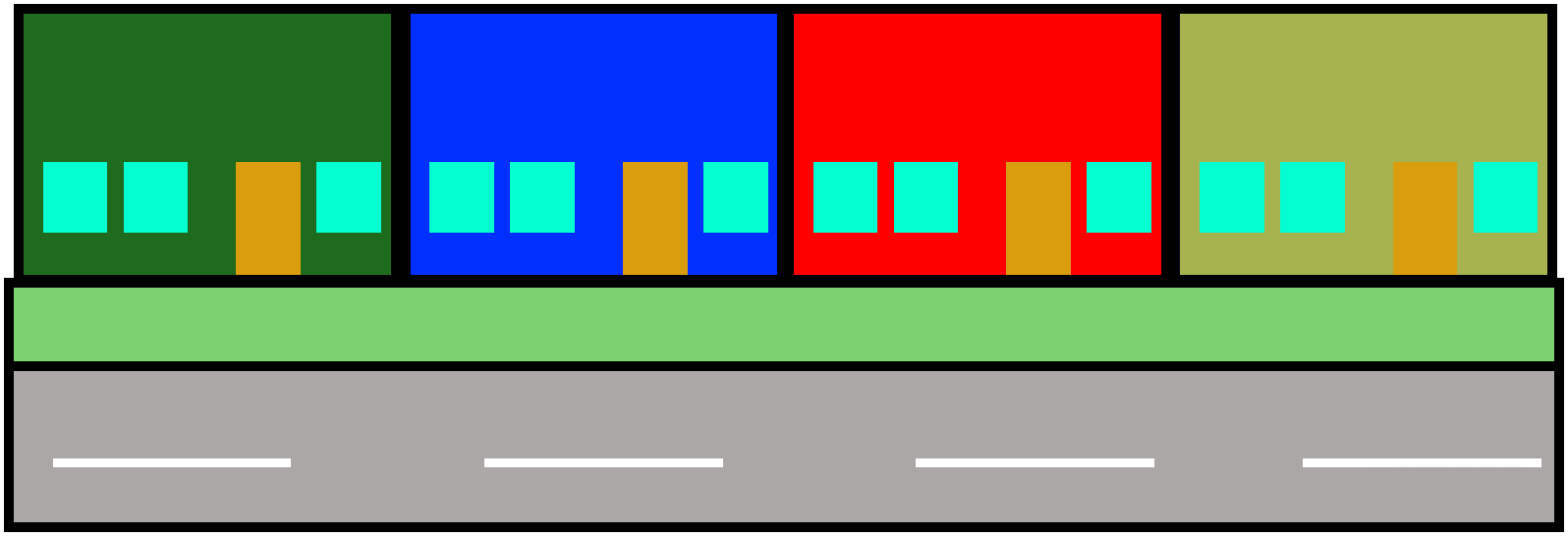
Ahora agreguemos casas a la cuadra de 40m, esta vez de las casas son de 10m de frente.



Tamaño, dirección, unidad direccionable

¿De qué estamos hablando? (toma 2)

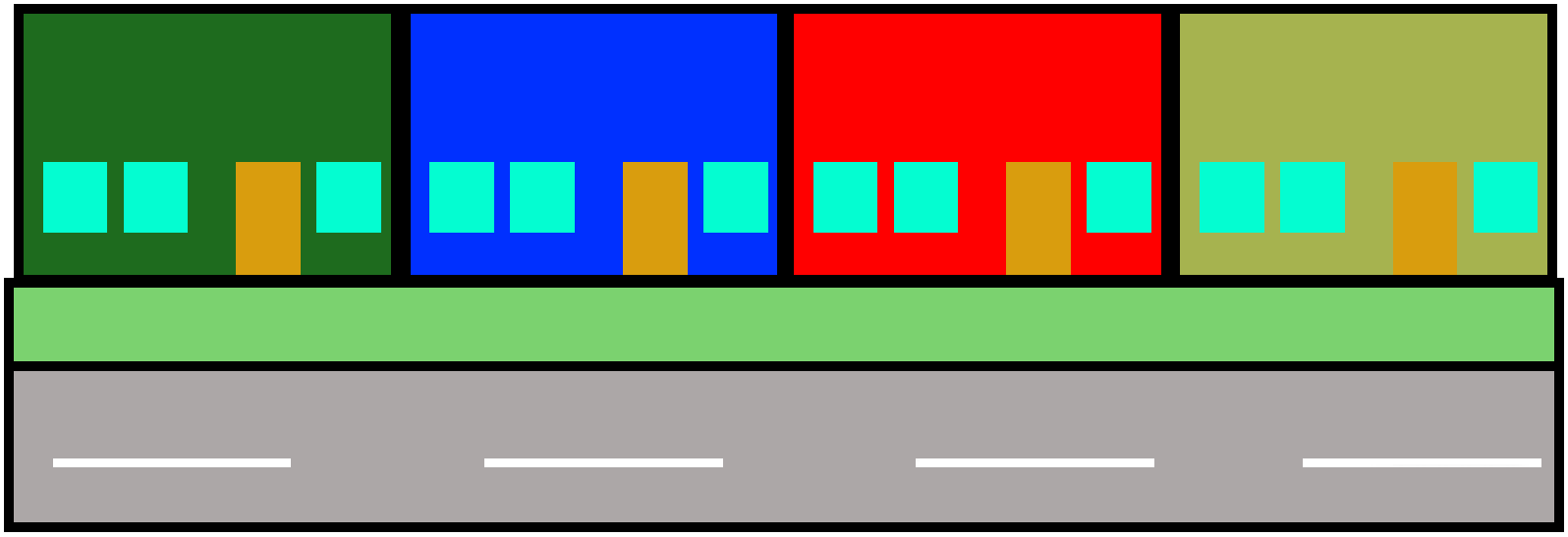
Ahora agreguemos casas a la cuadra de 40m, esta vez de las casas son de 10m de frente.



Tamaño, dirección, unidad direccionable

¿De qué estamos hablando? (toma 2)

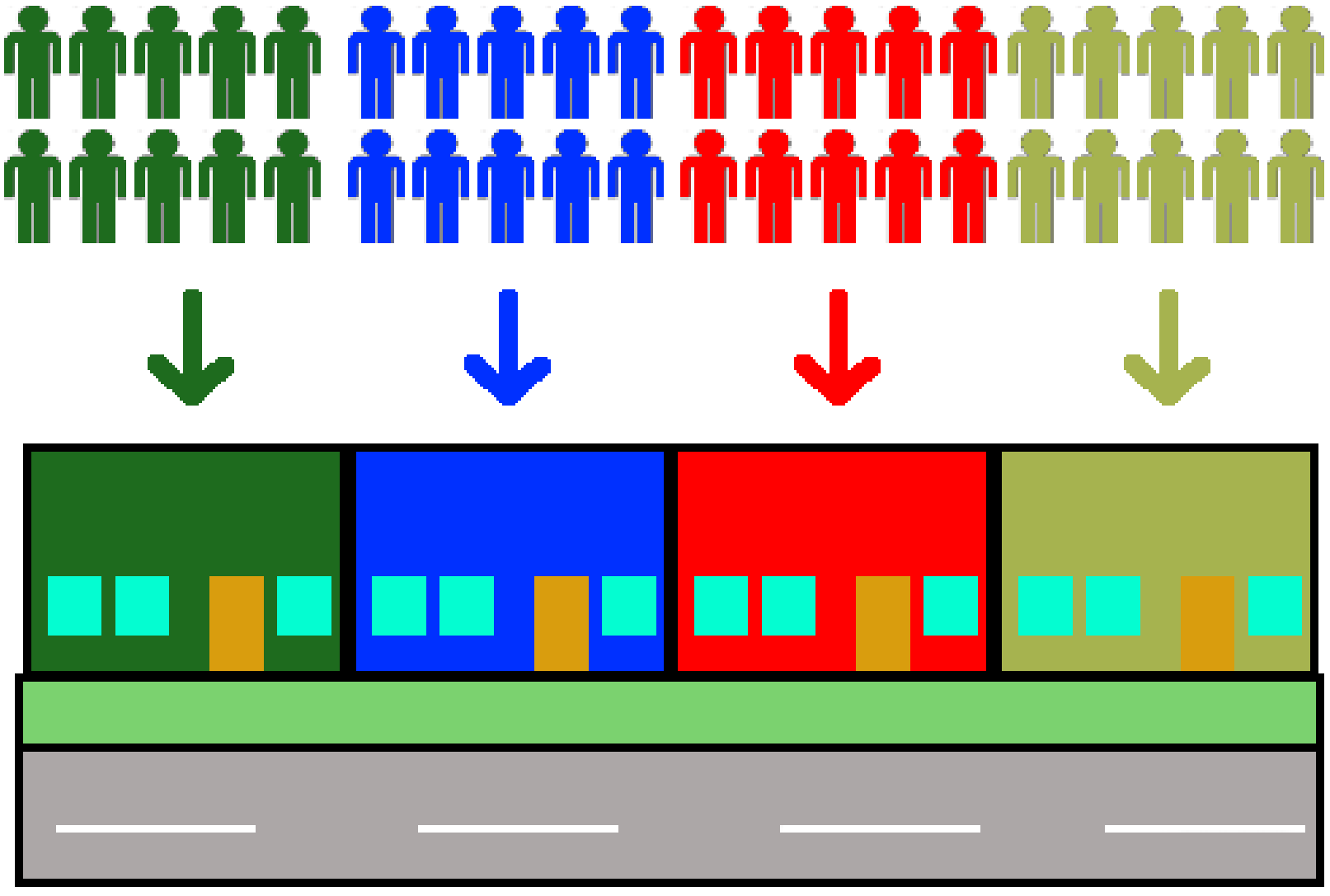
La capacidad de cada casa será de 10 personas. ¿Cuántas personas entran en la cuadra?



Tamaño, dirección, unidad direccionable

¿De qué estamos hablando? (toma 2)

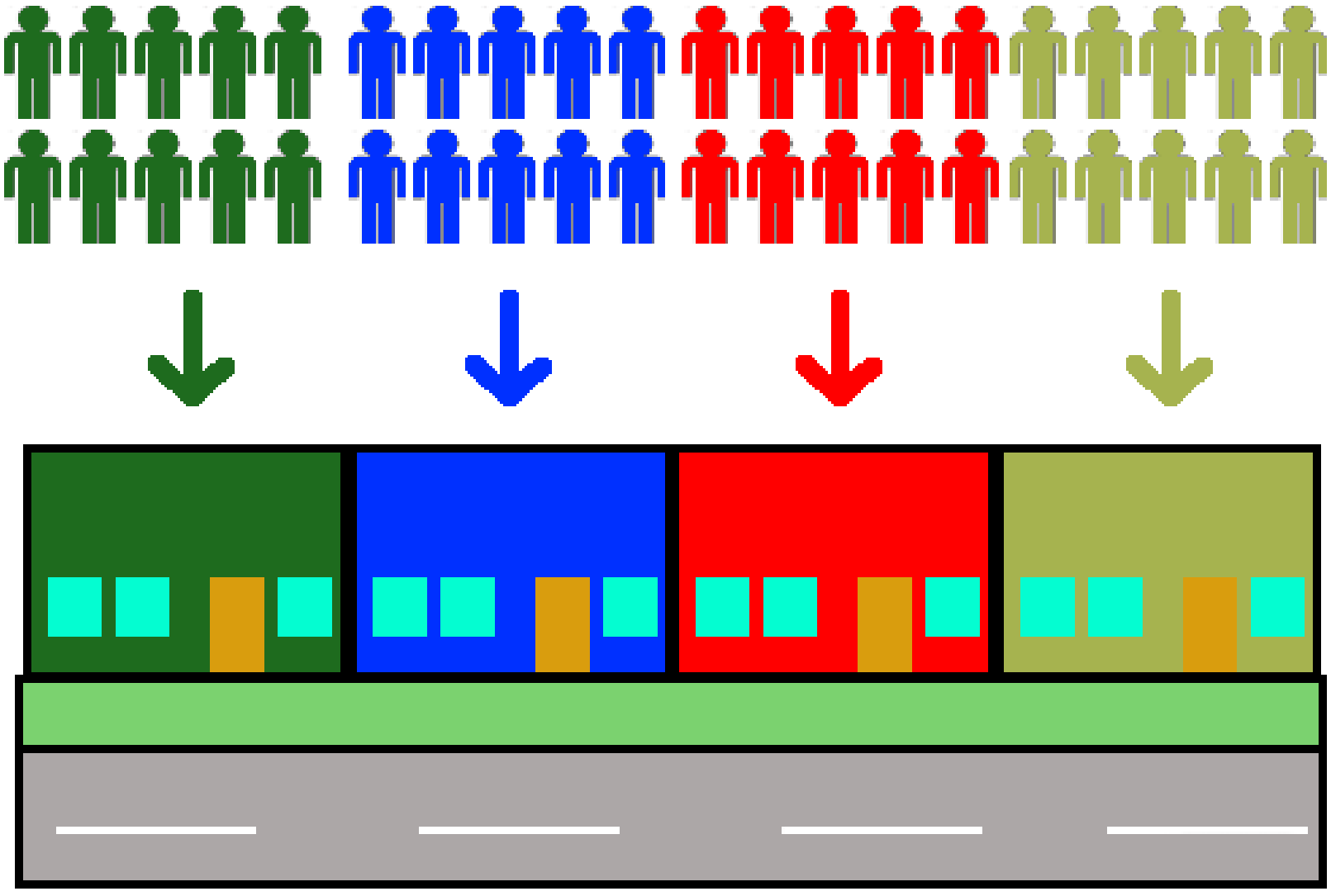
La capacidad de cada casa será de 10 personas. ¿Cuántas personas entran en la cuadra?



Tamaño, dirección, unidad direccionable

¿De qué estamos hablando? (toma 2)

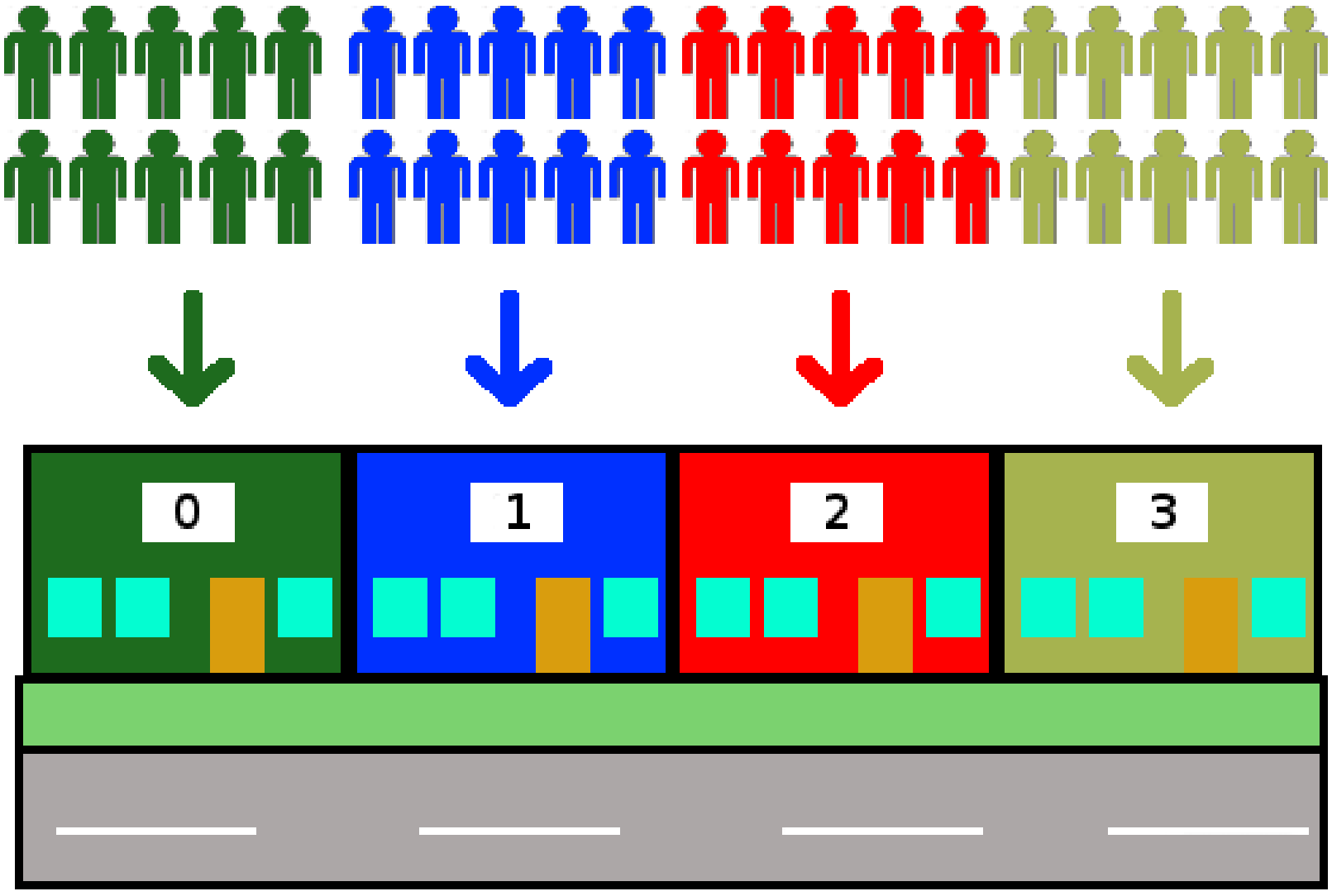
Numeremos las casas.



Tamaño, dirección, unidad direccionable

¿De qué estamos hablando? (toma 2)

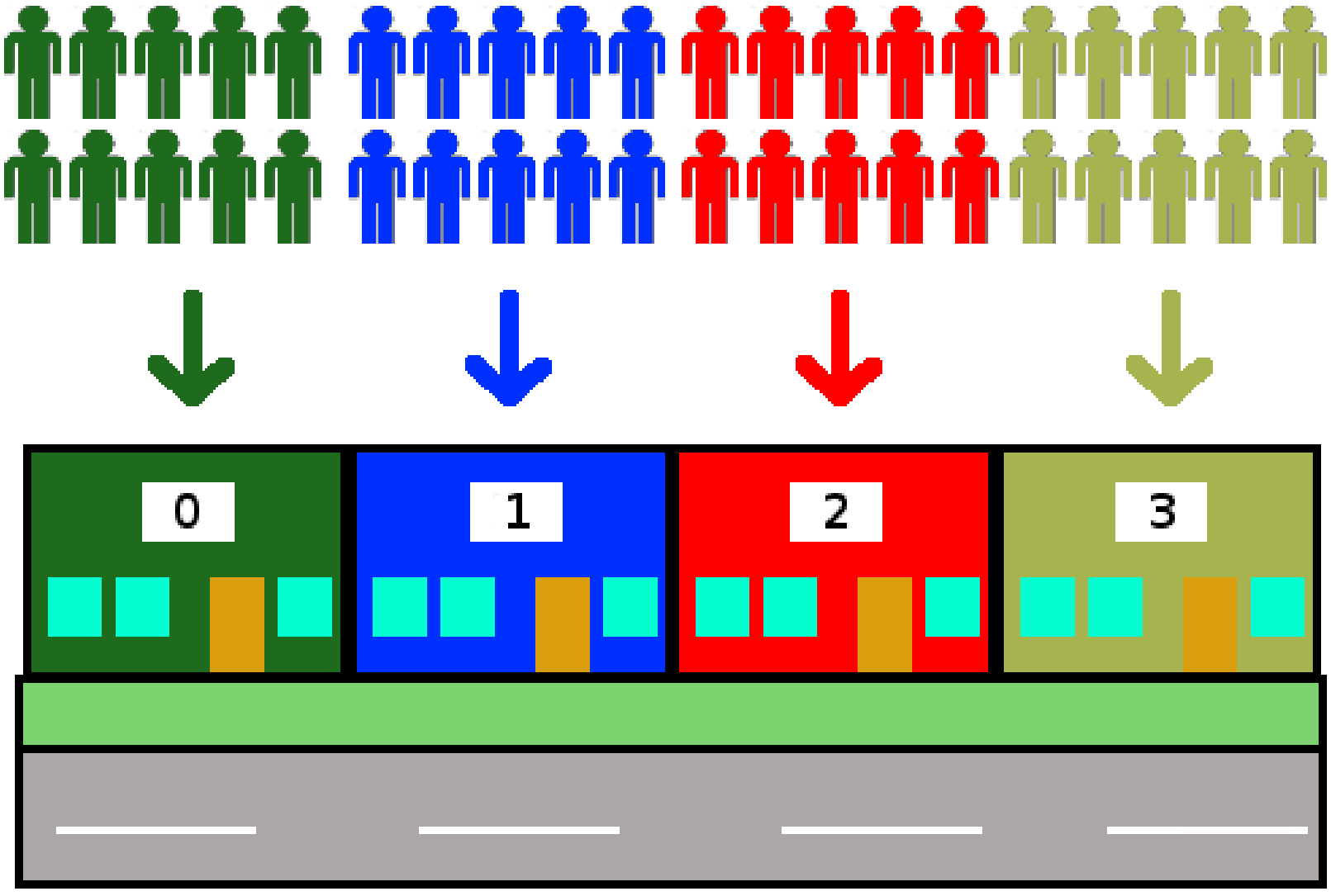
Numeremos las casas.



Tamaño, dirección, unidad direccionable

¿De qué estamos hablando? (toma 2)

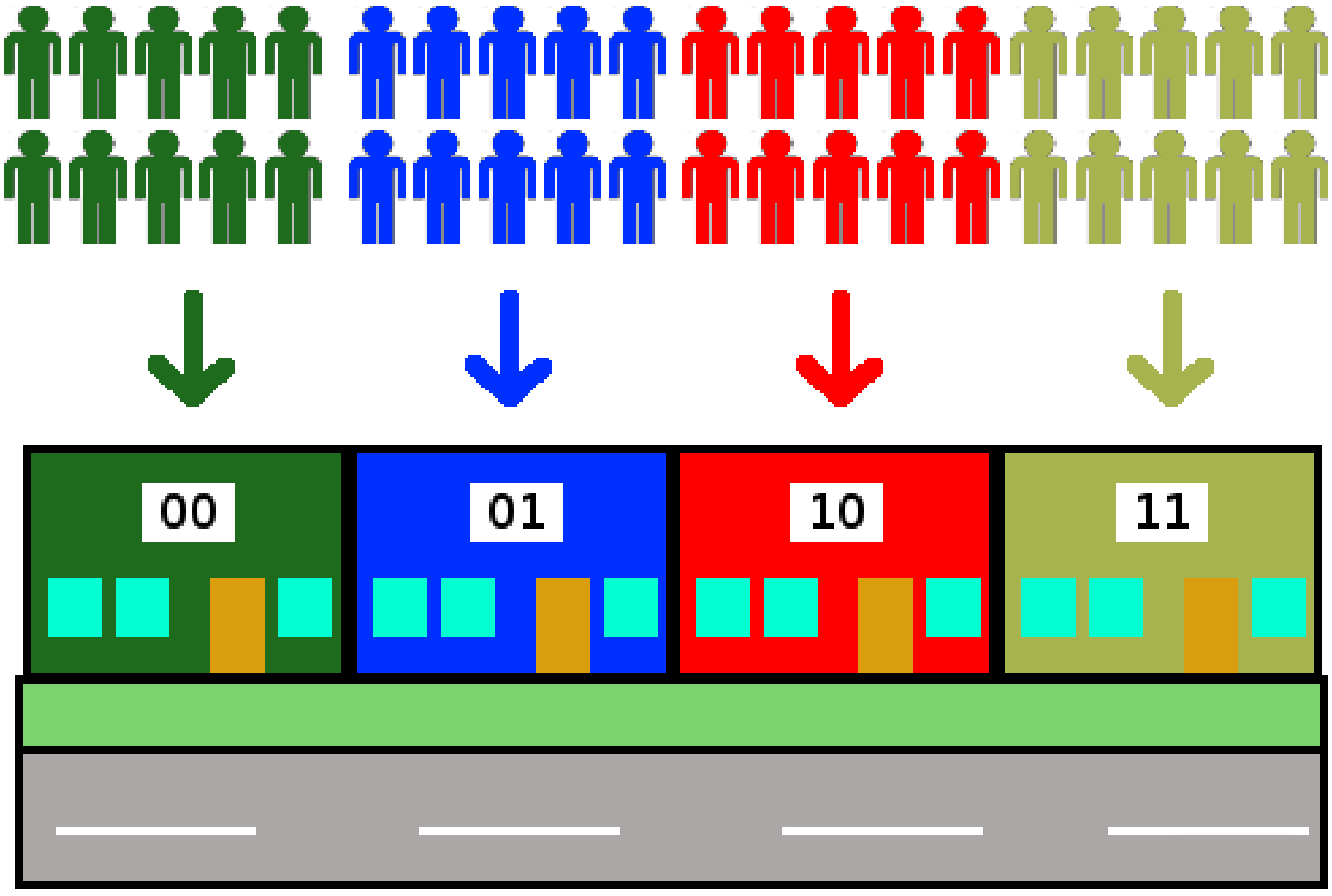
¿Cuántos bits necesitamos para identificar las casas?



Tamaño, dirección, unidad direccionable

¿De qué estamos hablando? (toma 2)

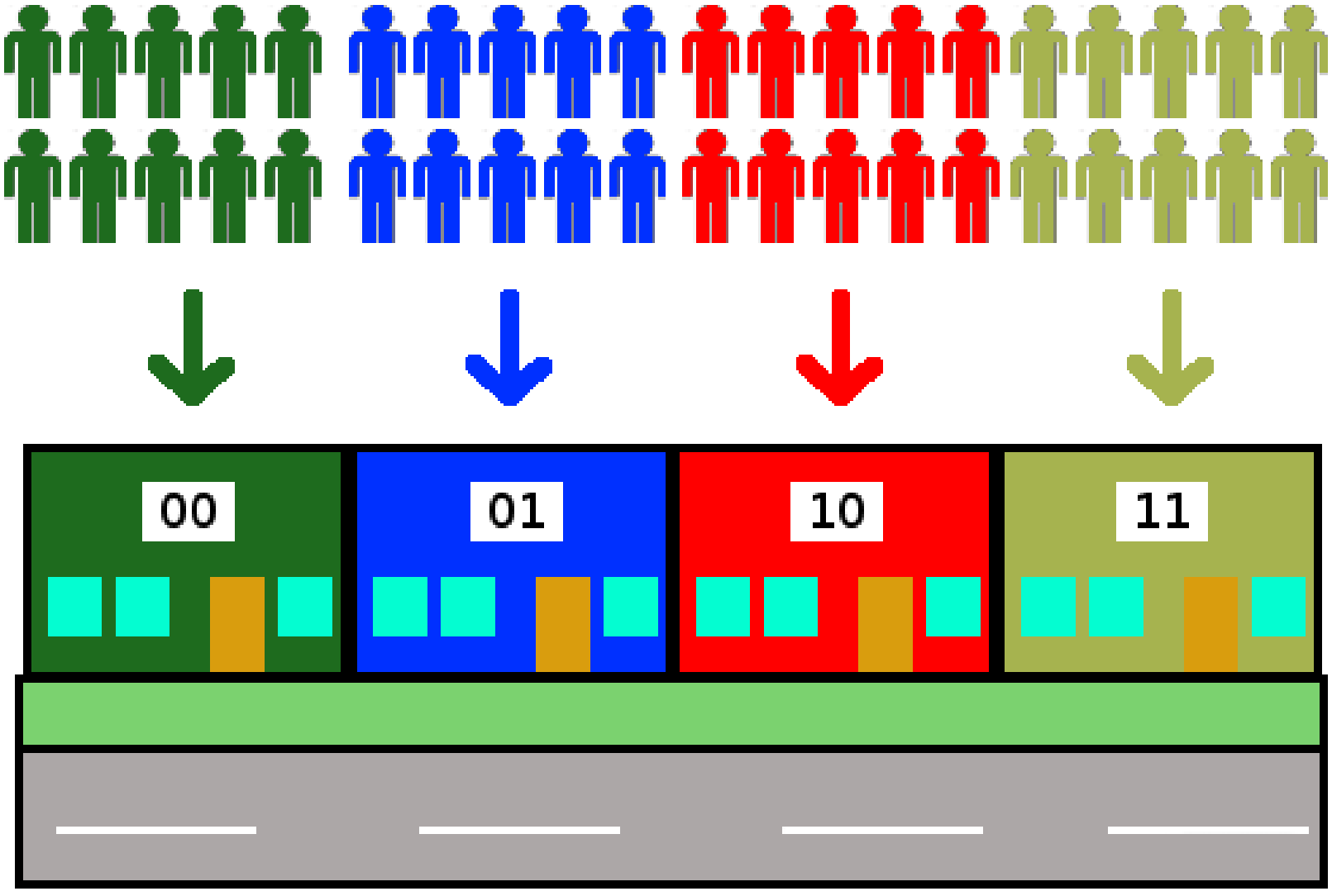
¿Cuántos bits necesitamos para identificar las casas?



Tamaño, dirección, unidad direccionable

¿De qué estamos hablando? (toma 2)

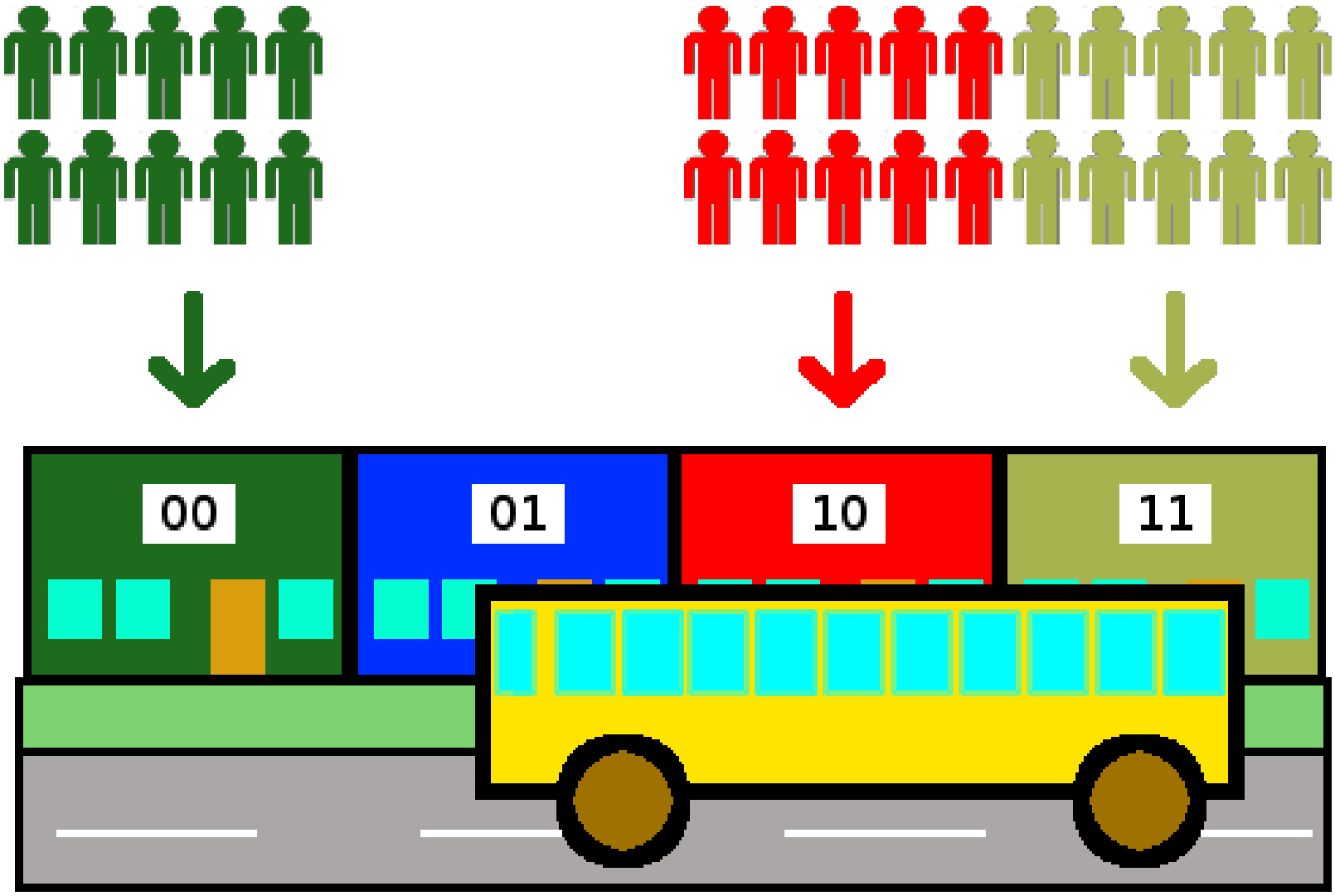
Para ir a trabajar hay un colectivo con lugar para 10 personas que solamente arranca cuando esta lleno. Lleva únicamente a una casa completa por vez.



Tamaño, dirección, unidad direccionable

¿De qué estamos hablando? (toma 2)

Para ir a trabajar hay un colectivo con lugar para 10 personas que solamente arranca cuando esta lleno. Lleva únicamente a una casa completa por vez.



¿Qué tenía esto que ver?

¿Qué tenía esto que ver?

Analogía entre la cuadra y la memoria.

¿Qué tenía esto que ver?

Analogía entre la cuadra y la memoria.

- La cantidad de personas que entra en la cuadra es el tamaño de la memoria

¿Qué tenía esto que ver?

Analogía entre la cuadra y la memoria.

- La cantidad de personas que entra en la cuadra es el tamaño de la memoria
- La cantidad de personas que entra en una casa es el tamaño de la unidad direccionable

¿Qué tenía esto que ver?

Analogía entre la cuadra y la memoria.

- La cantidad de personas que entra en la cuadra es el tamaño de la memoria
- La cantidad de personas que entra en una casa es el tamaño de la unidad direccionable
- La cantidad de casas de la cuadra es la cantidad de direcciones de la memoria

¿Qué tenía esto que ver?

Analogía entre la cuadra y la memoria.

- La cantidad de personas que entra en la cuadra es el tamaño de la memoria
- La cantidad de personas que entra en una casa es el tamaño de la unidad direccionable
- La cantidad de casas de la cuadra es la cantidad de direcciones de la memoria
- La cantidad de bits necesarios para numerar las casas es la cantidad de bits necesarios para distinguir las direcciones de memoria

¿Qué tenía esto que ver?

Analogía entre la cuadra y la memoria.

- La cantidad de personas que entra en la cuadra es el tamaño de la memoria
- La cantidad de personas que entra en una casa es el tamaño de la unidad direccionable
- La cantidad de casas de la cuadra es la cantidad de direcciones de la memoria
- La cantidad de bits necesarios para numerar las casas es la cantidad de bits necesarios para distinguir las direcciones de memoria
- La capacidad del colectivo (o bus) es el tamaño de la palabra

¡Ahora sí, arranquemos!

- ¿Cuántos bits son necesarios para poder direccionar la memoria en los siguientes casos?

- ¿Cuántos bits son necesarios para poder direccionar la memoria en los siguientes casos?
  1. Memoria de 2 GB con direccionamiento a *byte*

- ¿Cuántos bits son necesarios para poder direccionar la memoria en los siguientes casos?
  1. Memoria de 2 GB con direccionamiento a *byte*
     - $\#\text{direcciones} = \frac{2\ \text{GB}}{1\ \text{B}/dir} = \frac{2 \times 2^{30}\text{B}}{1\ \text{B}/dir} = 2^1 \times 2^{30}\ dir$

- ¿Cuántos bits son necesarios para poder direccionar la memoria en los siguientes casos?
  1. Memoria de 2 GB con direccionamiento a *byte*
     - #direcciones $= \frac{2 \text{ GB}}{1 \text{ B}/dir} = \frac{2 \times 2^{30} \text{B}}{1 \text{ B}/dir} = 2^1 \times 2^{30}\ dir$
     - #*bits* de direccionamiento $= \log_2(2^1 \times 2^{30}) = 31\ bits$

- ¿Cuántos bits son necesarios para poder direccionar la memoria en los siguientes casos?
  1. Memoria de 2 GB con direccionamiento a *byte*
     - #direcciones $= \frac{2 \text{ GB}}{1 \text{ B}/dir} = \frac{2 \times 2^{30} \text{B}}{1 \text{ B}/dir} = 2^1 \times 2^{30}\ dir$
     - #*bits* de direccionamiento $= \log_2(2^1 \times 2^{30}) = 31\ bits$
  2. Memoria de 512 MB con direccionamiento a 16 *bits*

- ¿Cuántos bits son necesarios para poder direccionar la memoria en los siguientes casos?
  1. Memoria de 2 GB con direccionamiento a *byte*
     - #direcciones $= \frac{2 \text{ GB}}{1 \text{ B}/dir} = \frac{2 \times 2^{30} \text{B}}{1 \text{ B}/dir} = 2^1 \times 2^{30}\ dir$
     - #*bits* de direccionamiento $= \log_2(2^1 \times 2^{30}) = 31\ bits$
  2. Memoria de 512 MB con direccionamiento a 16 *bits*
     - #direcciones $= \frac{512 \text{ MB}}{2 \text{ B}/dir} = \frac{2^9 \times 2^{20} \text{B}}{2 \text{ B}/dir} = 2^{28}\ dir$

- ¿Cuántos bits son necesarios para poder direccionar la memoria en los siguientes casos?
  1. Memoria de 2 GB con direccionamiento a *byte*
     - #direcciones $= \frac{2 \text{ GB}}{1 \text{ B}/dir} = \frac{2 \times 2^{30} \text{B}}{1 \text{ B}/dir} = 2^1 \times 2^{30}\ dir$
     - #*bits* de direccionamiento $= \log_2(2^1 \times 2^{30}) = 31\ bits$
  2. Memoria de 512 MB con direccionamiento a 16 *bits*
     - #direcciones $= \frac{512 \text{ MB}}{2 \text{ B}/dir} = \frac{2^9 \times 2^{20} \text{B}}{2 \text{ B}/dir} = 2^{28}\ dir$
     - #*bits* de direccionamiento $= \log_2(2^{28}) = 28\ bits$

- ¿Cuántos bits son necesarios para poder direccionar la memoria en los siguientes casos?
  1. Memoria de 2 GB con direccionamiento a *byte*
     - #direcciones $= \frac{2 \text{ GB}}{1 \text{ B}/dir} = \frac{2 \times 2^{30} \text{B}}{1 \text{ B}/dir} = 2^1 \times 2^{30}\ dir$
     - #*bits* de direccionamiento $= \log_2(2^1 \times 2^{30}) = 31\ bits$
  2. Memoria de 512 MB con direccionamiento a 16 *bits*
     - #direcciones $= \frac{512 \text{ MB}}{2 \text{ B}/dir} = \frac{2^9 \times 2^{20} \text{B}}{2 \text{ B}/dir} = 2^{28}\ dir$
     - #*bits* de direccionamiento $= \log_2(2^{28}) = 28\ bits$
- ¿Qué tamaño tiene la memoria si se usan 25 *bits* para direccionarla y tenemos direccionamiento a *nibble* (4 *bits*)?
  1. #direcciones $= 2^{25}$

- ¿Cuántos bits son necesarios para poder direccionar la memoria en los siguientes casos?
  1. Memoria de 2 GB con direccionamiento a *byte*
     - #direcciones $= \frac{2 \text{ GB}}{1 \text{ B}/dir} = \frac{2 \times 2^{30} \text{B}}{1 \text{ B}/dir} = 2^1 \times 2^{30}\ dir$
     - #*bits* de direccionamiento $= \log_2(2^1 \times 2^{30}) = 31\ bits$
  2. Memoria de 512 MB con direccionamiento a 16 *bits*
     - #direcciones $= \frac{512 \text{ MB}}{2 \text{ B}/dir} = \frac{2^9 \times 2^{20} \text{B}}{2 \text{ B}/dir} = 2^{28}\ dir$
     - #*bits* de direccionamiento $= \log_2(2^{28}) = 28\ bits$
- ¿Qué tamaño tiene la memoria si se usan 25 *bits* para direccionarla y tenemos direccionamiento a *nibble* (4 *bits*)?
  1. #direcciones $= 2^{25}$
  2. Tamaño de la Memoria $= 2^{25} dir 2^2 \frac{bits}{dir} = 2^3 \frac{B}{bits} 2^4 2^{20} bits$ $= 16$ MB

¿Qué hicimos?

¿Qué hicimos?

Para saber la cantidad de direcciones que teníamos que direccionar en cada caso, usamos la siguiente fórmula:

$$\text{cantidad de direcciones} = \frac{\text{tamaño de la memoria}}{\text{unidad de direccionamiento}}$$

¿Qué hicimos?

Para saber la cantidad de direcciones que teníamos que direccionar en cada caso, usamos la siguiente fórmula:

$$\text{cantidad de direcciones} = \frac{\text{tamaño de la memoria}}{\text{unidad de direccionamiento}}$$

Una vez encontrada la *cantidad de direcciones de memoria*, utilizamos el logaritmo en base 2 para calcular la *cantidad de bits necesarios* para poder hacer referencia a todas las direcciones de memoria.

Arquitectura

Descripción General

¿Qué es la Arquitectura de una Computadora?

Es el conjunto de reglas e instrucciones que describen su funcionalidad, así como su organización e implementación.

Descripción General

¿Qué es la Arquitectura de una Computadora?

Es el conjunto de reglas e instrucciones que describen su funcionalidad, así como su organización e implementación.

A nivel general, podemos afirmar que la máquina **ORGA1** tiene arquitectura de *von Neumann*. Esto implica que:

Descripción General

¿Qué es la Arquitectura de una Computadora?

Es el conjunto de reglas e instrucciones que describen su funcionalidad, así como su organización e implementación.

A nivel general, podemos afirmar que la máquina **ORGA1** tiene arquitectura de *von Neumann*. Esto implica que:

- Tiene un **CPU** que cuenta con una ALU y registros internos.

Descripción General

¿Qué es la Arquitectura de una Computadora?

Es el conjunto de reglas e instrucciones que describen su funcionalidad, así como su organización e implementación.

A nivel general, podemos afirmar que la máquina **ORGA1** tiene arquitectura de *von Neumann*. Esto implica que:

- Tiene un **CPU** que cuenta con una ALU y registros internos.
- Tiene una **Memoria** que guarda tanto datos como instrucciones.

Descripción General

¿Qué es la Arquitectura de una Computadora?

Es el conjunto de reglas e instrucciones que describen su funcionalidad, así como su organización e implementación.

A nivel general, podemos afirmar que la máquina **ORGA1** tiene arquitectura de *von Neumann*. Esto implica que:

- Tiene un **CPU** que cuenta con una ALU y registros internos.
- Tiene una **Memoria** que guarda tanto datos como instrucciones.
- Tiene dispositivos de **Entrada/Salida** para interactuar con el 'afuera' de la máquina.

Registros

Registros de Propósito General

- 8 registros de **propósito general** de 16 *bits*.
  - **R0**
  - ...
  - **R7**
- Se utilizan para poder traer datos desde **Memoria** hacia el **CPU** y poder operar con ellos.
- Están **disponibles para que el programador** los use a su antojo.

Registros de Propósito Específico

- 3 registros de **propósito específico** de 16 *bits*.
  - **PC** (Program Counter): Apunta a la dirección que contiene la próxima instrucción a ejecutar.
  - **SP** (Stack Pointer[1]): Apunta a la próxima posición libre de la pila.
  - **IR** (Instruction Register): Guarda la instrucción a ser inmediatamente ejecutada por el **CPU**.
- **No son directamente accesibles por el programador**. Sólo pueden ser modificados por la ejecución de instrucciones.

[1] El **Stack** es una estructura de datos en **Memoria** que veremos hacia el final de la clase.

FLAGS

- Los **FLAGS** también son registros de 1 *bit*.
- Sus valores cambian en función de la ejecución de algunas operaciones. **No todas las operaciones alteran los FLAGS**.
- Son los mismos que vimos en los *Talleres de Lógica Digital*.
  - **Z** (zero)
  - **N** (negative)
  - **C** (carry)
  - **V** (overflow)

Memoria

Memoria

- La **Memoria** tiene un tamaño fijo con 65520 posiciones ($2^{16} - 16$ posiciones).

- Cada posición de **Memoria** tiene asociada una dirección de 16 bits. Entonces:
  - Las posiciones de **Memoria** se numeran en el rango de `0x0000` a `0xFFFF`.
  - Quedan 16 direcciones libres (0xFFF0 a 0xFFFF) que no apuntan a posiciones de **Memoria**. En cambio, usan para mapear registros de E/S. [2]

- La **Memoria** tiene direccionamiento a palabra. Eso significa que:
  - Cada posición de **Memoria** puede guardar una **palabra**.
  - Las palabras en la **ORGA1** son datos de 16 *bits*.

[2] Más de esto más adelante en la materia.

Set de Instrucciones

Formato de instrucción

Tipo 1: Instrucciones de dos operandos

4 bits	*6 bits*	*6 bits*	*16 bits*	*16 bits*
cod. op.	destino	fuente	constante destino (opcional)	constante fuente (opcional)

operación	cod. op.	efecto	modifica flags
MOV d, f	0001	$d \leftarrow f$	no
ADD d, f	0010	$d \leftarrow d + f$ (suma binaria)	sí
SUB d, f	0011	$d \leftarrow d - f$ (resta binaria)	sí
AND d, f	0100	$d \leftarrow d$ and f	sí $(\star)$
OR d, f	0101	$d \leftarrow d$ or f	sí $(\star)$
CMP d, f	0110	Modifica los *flags* según el resultado de $d - f$.	sí
ADDC d, f	1101	$d \leftarrow d + f +$ *carry* (suma binaria)	sí

$(\star)$ dejan el flag de *carry* (C) y el de *overflow* (V) en cero.

Formato de instrucción

Tipo 2: Instrucciones de un operando

Tipo 2a: Instrucciones de un operando destino.

4 bits	*6 bits*	*6 bits*	*16 bits*
cod. op.	destino	000000	constante destino (opcional)

operacin	cod. op.	efecto	modifica flags
NEG d	1000	$d \leftarrow$ el inverso aditivo de d	s
NOT d	1001	$d \leftarrow$ not d (bit a bit)	s ($\star$)

($\star$) deja el *flag* de *carry* (C) y el de *overflow* (V) en cero.

Tipo 2b: Instrucciones de un operando fuente.

4 bits	*6 bits*	*6 bits*	*16 bits*
cod. op.	000000	fuente	constante fuente (opcional)

operacin	cod. op.	efecto	modifica flags
JMP f	1010	PC $\leftarrow f$	no
CALL f	1011	[SP] $\leftarrow$ PC, SP $\leftarrow$ SP $-$ 1, PC $\leftarrow f$	no

Formato de instrucción

Tipo 3: Instrucciones sin operandos

4 bits	*6 bits*	*6 bits*
cod. op.	000000	000000

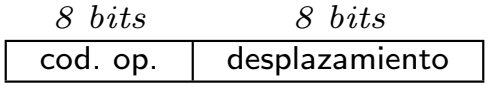
operación	cod. op.	efecto
RET	1100	PC $\leftarrow$ [SP+1], SP $\leftarrow$ SP + 1

- No modifica *flags*.

Formato de instrucción

Tipo 4: Saltos relativos condicionales

- el salto se produce si se cumple la *condición de salto* correspondiente.
- PC $\leftarrow$ PC + *desplazamiento*
- el desplazamiento se representa en *complemento a 2* de *8 bits*.
- No modifican *flags*.



Codop	Operación	Descripción	Condición de Salto
1111 0001	JE	Igual / Cero	Z
1111 1001	JNE	Distinto	not Z
1111 0010	JLE	Menor o igual	Z or (N xor V)
1111 1010	JG	Mayor	not (Z or (N xor V))
1111 0011	JL	Menor	N xor V
1111 1011	JGE	Mayor o igual	not (N xor V)
1111 0100	JLEU	Menor o igual sin signo	C or Z
1111 1100	JGU	Mayor sin signo	not (C or Z)
1111 0101	JCS	Carry / Menor sin signo	C
1111 0110	JNEG	Negativo	N
1111 0111	JVS	Overflow	V

Modos de Direccionamiento

Modos de Direccionamiento

Modo	Codificación	Resultado
Inmediato	000000	c16
Directo	001000	[c16]
Indirecto	011000	[[c16]]
Registro	100rrr	Rrrr
Indirecto registro	110rrr	[Rrrr]
Indexado	111rrr	[Rrrr + c16]

c16 es una constante de *16 bits*.

Rrrr es el registro indicado por los últimos tres *bits* del código de operando.

Las instrucciones que tienen como destino un operando de tipo *inmediato* son consideradas como inválidas por el procesador, excepto el CMP.

Ciclo de Vida de un Programa

1. Programación: Escribir un algoritmo en **lenguaje ensamblador (o assembly)**. Esto es el **código fuente**.

Ciclo de Vida de un Programa

1. Programación: Escribir un algoritmo en **lenguaje ensamblador (o assembly)**. Esto es el **código fuente**.
2. Ensamblado: Un programa llamado **Ensamblador** toma el **código fuente** y lo traduce a un **código máquina**.

Ciclo de Vida de un Programa

1. Programación: Escribir un algoritmo en **lenguaje ensamblador (o assembly)**. Esto es el **código fuente**.
2. Ensamblado: Un programa llamado **Ensamblador** toma el **código fuente** y lo traduce a un **código máquina**.
    1. Resuelve las directivas dirigidas al **Ensamblador**.
    2. Calcula el tamaño de cada instrucciones, y resuelve las etiquetas.
    3. Traduce las instrucciones a 0s y 1s.

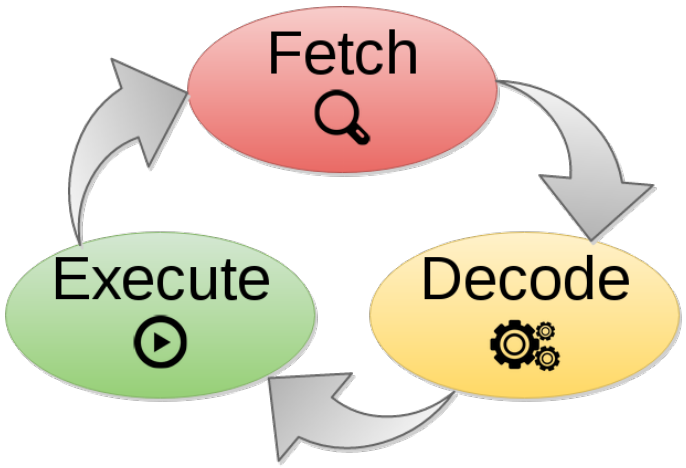
Ciclo de Vida de un Programa

1. Programación: Escribir un algoritmo en **lenguaje ensamblador (o assembly)**. Esto es el **código fuente**.
2. Ensamblado: Un programa llamado **Ensamblador** toma el **código fuente** y lo traduce a un **código máquina**.
    1. Resuelve las directivas dirigidas al **Ensamblador**.
    2. Calcula el tamaño de cada instrucciones, y resuelve las etiquetas.
    3. Traduce las instrucciones a 0s y 1s.
3. Carga: Se le indica al **Ensamblador** una dirección inicial, y éste copia el **código máquina** en la **Memoria** desde esa posición en adelante. Luego, carga esa misma dirección en el **PC (Program Counter)**.

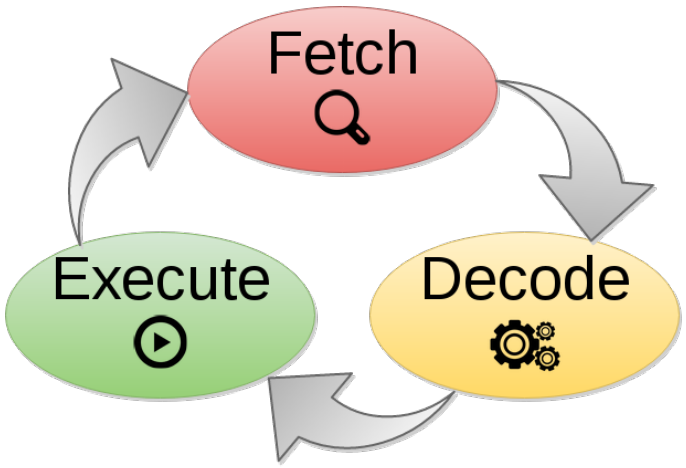
Ciclo de Vida de un Programa

1. Programación: Escribir un algoritmo en **lenguaje ensamblador (o assembly)**. Esto es el **código fuente**.
2. Ensamblado: Un programa llamado **Ensamblador** toma el **código fuente** y lo traduce a un **código máquina**.
    1. Resuelve las directivas dirigidas al **Ensamblador**.
    2. Calcula el tamaño de cada instrucciones, y resuelve las etiquetas.
    3. Traduce las instrucciones a 0s y 1s.
3. Carga: Se le indica al **Ensamblador** una dirección inicial, y éste copia el **código máquina** en la **Memoria** desde esa posición en adelante. Luego, carga esa misma dirección en el **PC (Program Counter)**.
4. Ejecución: El **CPU** da inicio a su **ciclo de ejecución** comenzando por la posición indicada en el **PC**.

Ciclo de Ejecución

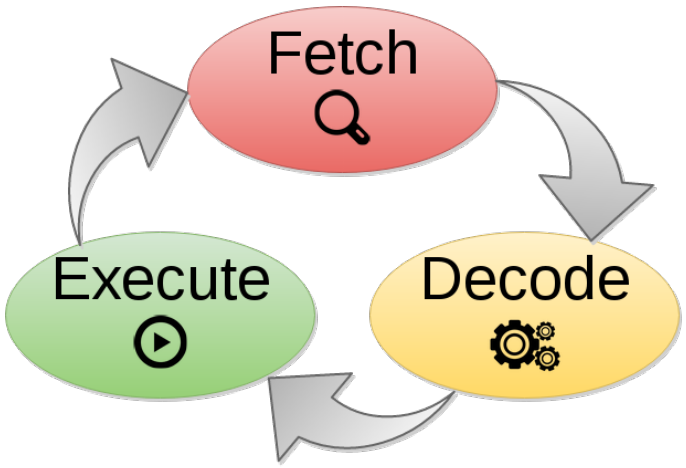


Ciclo de Ejecución



Fetch La **UC (Unidad de Control)** obtiene una instrucción de la posición de **Memoria** a la que apunta el **PC**
y lo incrementa (Si es necesario: busca más palabras de la instrucción usando el **PC** e incrementándolo cada vez.)

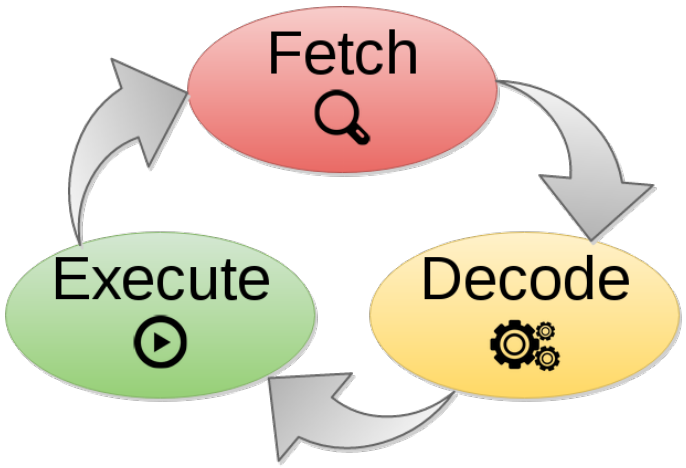
Ciclo de Ejecución



Fetch La **UC (Unidad de Control)** obtiene una instrucción de la posición de **Memoria** a la que apunta el **PC**
y lo incrementa (Si es necesario: busca más palabras de la instrucción usando el **PC** e incrementándolo cada vez.)

Decode La **UC** decodifica la instrucción.

Ciclo de Ejecución



Fetch La **UC (Unidad de Control)** obtiene una instrucción de la posición de **Memoria** a la que apunta el **PC**
y lo incrementa (Si es necesario: busca más palabras de la instrucción usando el **PC** e incrementándolo cada vez.)

Decode La **UC** decodifica la instrucción.

Execute La **UC** ejecuta la instrucción.

Ejercicio 1

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:   MOV R1,[[et1]]
        ADD [R1],0x6000
        JMP main
  et1:  DW 0x0007
        DW 0x0004
 loop:  CMP R1,0x0001
        JL fin
        JGE main
  fin:  RET
```

Ejercicio 1

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:   MOV R1,[[et1]]
        ADD [R1],0x6000
        JMP main
 et1:   DW 0x0007
        DW 0x0004
loop:   CMP R1,0x0001
        JL fin
        JGE main
 fin:   RET
```

Tenemos que:

Ejercicio 1

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:   MOV R1,[[et1]]
        ADD [R1],0x6000
        JMP main
et1:    DW 0x0007
        DW 0x0004
loop:   CMP R1,0x0001
        JL fin
        JGE main
fin:    RET
```

Tenemos que:

- ver cuántas palabras necesita cada instrucción

Ejercicio 1

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:   MOV R1,[[et1]]
        ADD [R1],0x6000
        JMP main
 et1:   DW 0x0007
        DW 0x0004
 loop:  CMP R1,0x0001
        JL fin
        JGE main
 fin:   RET
```

Tenemos que:

- ver cuántas palabras necesita cada instrucción
- calcular los valores de las etiquetas

Ejercicio 1

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:   MOV R1,[[et1]]
        ADD [R1],0x6000
        JMP main
 et1:   DW 0x0007
        DW 0x0004
 loop:  CMP R1,0x0001
        JL fin
        JGE main
 fin:   RET
```

Tenemos que:

- ver cuántas palabras necesita cada instrucción
- calcular los valores de las etiquetas

`DW` *(Define Word)*: directiva al ensamblador que provoca que en la posición de memoria que le corresponde, aparezca el valor indicado.

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

dirección ocupa

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
 loop: CMP R1,0x0001
       JL fin
       JGE main
  fin: RET
```

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

		dirección	ocupa
main:	MOV R1,[[et1]]	0x0000	
	ADD [R1],0x6000		
	JMP main		
et1:	DW 0x0007		
	DW 0x0004		
loop:	CMP R1,0x0001		
	JL fin		
	JGE main		
fin:	RET		

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
 loop: CMP R1,0x0001
       JL fin
       JGE main
 fin:  RET
```

dirección	ocupa
0x0000	dos palabras

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
 loop: CMP R1,0x0001
       JL fin
       JGE main
 fin:  RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
 loop: CMP R1,0x0001
       JL fin
       JGE main
 fin:  RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	dos palabras

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
loop:  CMP R1,0x0001
       JL fin
       JGE main
 fin:  RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	dos palabras
0x0004	

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
loop:  CMP R1,0x0001
       JL fin
       JGE main
 fin:  RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	dos palabras
0x0004	dos palabras

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
loop:  CMP R1,0x0001
       JL fin
       JGE main
 fin:  RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	dos palabras
0x0004	dos palabras
0x0006	

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
loop:  CMP R1,0x0001
       JL fin
       JGE main
 fin:  RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	dos palabras
0x0004	dos palabras
0x0006	una palabra

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
 loop: CMP R1,0x0001
       JL fin
       JGE main
  fin: RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	dos palabras
0x0004	dos palabras
0x0006	una palabra
0x0007	

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
 loop: CMP R1,0x0001
       JL fin
       JGE main
  fin: RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	dos palabras
0x0004	dos palabras
0x0006	una palabra
0x0007	una palabra

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
loop:  CMP R1,0x0001
       JL fin
       JGE main
 fin:  RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	dos palabras
0x0004	dos palabras
0x0006	una palabra
0x0007	una palabra
0x0008	

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
 loop: CMP R1,0x0001
       JL fin
       JGE main
 fin:  RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	dos palabras
0x0004	dos palabras
0x0006	una palabra
0x0007	una palabra
0x0008	dos palabras

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
 loop: CMP R1,0x0001
       JL fin
       JGE main
 fin:  RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	dos palabras
0x0004	dos palabras
0x0006	una palabra
0x0007	una palabra
0x0008	dos palabras
0x000A	

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
 loop: CMP R1,0x0001
       JL fin
       JGE main
 fin:  RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	dos palabras
0x0004	dos palabras
0x0006	una palabra
0x0007	una palabra
0x0008	dos palabras
0x000A	una palabra

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
 loop: CMP R1,0x0001
       JL fin
       JGE main
 fin:  RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	dos palabras
0x0004	dos palabras
0x0006	una palabra
0x0007	una palabra
0x0008	dos palabras
0x000A	una palabra
0x000B	

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
 loop: CMP R1,0x0001
       JL fin
       JGE main
 fin:  RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	dos palabras
0x0004	dos palabras
0x0006	una palabra
0x0007	una palabra
0x0008	dos palabras
0x000A	una palabra
0x000B	una palabra

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
 et1:  DW 0x0007
       DW 0x0004
 loop: CMP R1,0x0001
       JL fin
       JGE main
 fin:  RET
```

dirección	ocupa
0x0000	dos palabras
0x0002	dos palabras
0x0004	dos palabras
0x0006	una palabra
0x0007	una palabra
0x0008	dos palabras
0x000A	una palabra
0x000B	una palabra
0x000C	

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

```
main:  MOV R1,[[et1]]
       ADD [R1],0x6000
       JMP main
et1:   DW 0x0007
       DW 0x0004
loop:  CMP R1,0x0001
       JL fin
       JGE main
fin:   RET
```

	dirección	ocupa
MOV R1,[[et1]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP main	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabras
JL fin	0x000A	una palabra
JGE main	0x000B	una palabra
RET	0x000C	una palabra

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

		dirección	ocupa
main:	MOV R1,[[et1]]	0x0000	dos palabras
	ADD [R1],0x6000	0x0002	dos palabras
	JMP main	0x0004	dos palabras
et1:	DW 0x0007	0x0006	una palabra
	DW 0x0004	0x0007	una palabra
loop:	CMP R1,0x0001	0x0008	dos palabras
	JL fin	0x000A	una palabra
	JGE main	0x000B	una palabra
fin:	RET	0x000C	una palabra

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

		dirección	ocupa
main:	MOV R1,[[et1]]	0x0000	dos palabras
	ADD [R1],0x6000	0x0002	dos palabras
	JMP main	0x0004	dos palabras
et1:	DW 0x0007	0x0006	una palabra
	DW 0x0004	0x0007	una palabra
loop:	CMP R1,0x0001	0x0008	dos palabras
	JL fin	0x000A	una palabra
	JGE main	0x000B	una palabra
fin:	RET	0x000C	una palabra

La etiqueta main corresponde a la dirección de memoria 0x0000, et1 corresponde a 0x0006, loop corresponde a 0x0008 y fin corresponde a 0x000C.

Codifiquemos el siguiente programa y carguémoslo desde la posición de memoria 0x0000:

		dirección	ocupa
main:	MOV R1,[[0x0006]]	0x0000	dos palabras
	ADD [R1],0x6000	0x0002	dos palabras
	JMP 0x0000	0x0004	dos palabras
et1:	DW 0x0007	0x0006	una palabra
	DW 0x0004	0x0007	una palabra
loop:	CMP R1,0x0001	0x0008	dos palabras
	JL fin	0x000A	una palabra
	JGE main	0x000B	una palabra
fin:	RET	0x000C	una palabra

La etiqueta main corresponde a la dirección de memoria 0x0000, et1 corresponde a 0x0006, loop corresponde a 0x0008 y fin corresponde a 0x000C.

Reemplazando las etiqutas por valores estamos listos para codificar.

Buenísimo, tenemos las direcciones de las etiquetas, pero... *los saltos condicionales son* **relativos**.

Reemplazando las etiqutas por valores estamos listos para codificar.

Buenísimo, tenemos las direcciones de las etiquetas, pero... *los saltos condicionales son* **relativos**.

- Esto significa que las etiquetas de los saltos son reemplazadas por el desplazamiento necesario para "llegar" desde la dirección en la que estamos parados, hasta la de la etiqueta de destino.

Reemplazando las etiqutas por valores estamos listos para codificar.

Buenísimo, tenemos las direcciones de las etiquetas, pero... *los saltos condicionales son* **relativos**.

- Esto significa que las etiquetas de los saltos son reemplazadas por el desplazamiento necesario para "llegar" desde la dirección en la que estamos parados, hasta la de la etiqueta de destino.
- Vamos a tener que calcular estos desplazamientos.

```
0x0000:   MOV R1,[[0x0006]]
0x0002:   ADD [R1],0x6000
0x0004:   JMP 0x0000
0x0006:   DW 0x0007
0x0007:   DW 0x0004
0x0008:   CMP R1,0x0001
0x000A:   JL fin
0x000B:   JGE main
0x000C:   RET
```

$$\text{Despl} = \underbrace{Dir\_Etiq}_{Destino} - \underbrace{Dir\_Instr}_{Origen(postfetch)}$$

La etiqueta main corresponde a la dirección de memoria 0x0000.
La etiqueta at1 corresponde a la dirección de memoria 0x0006.
La etiqueta loop corresponde a la dirección de memoria 0x0008.
La etiqueta fin corresponde a la dirección de memoria 0x000C.

```
0x0000:   MOV R1,[[0x0006]]
0x0002:   ADD [R1],0x6000
0x0004:   JMP 0x0000
0x0006:   DW 0x0007
0x0007:   DW 0x0004
0x0008:   CMP R1,0x0001
0x000A:   JL fin
0x000B:   JGE main
0x000C:   RET
```

Despl $=$ $\underbrace{Dir\_Etiq}_{Destino} - \underbrace{Dir\_Instr}_{Origen(postfetch)}$

desplazamiento fin:

La etiqueta main corresponde a la dirección de memoria 0x0000.
La etiqueta at1 corresponde a la dirección de memoria 0x0006.
La etiqueta loop corresponde a la dirección de memoria 0x0008.
La etiqueta fin corresponde a la dirección de memoria 0x000C.

```
0x0000:   MOV R1,[[0x0006]]
0x0002:   ADD [R1],0x6000
0x0004:   JMP 0x0000
0x0006:   DW 0x0007
0x0007:   DW 0x0004
0x0008:   CMP R1,0x0001
0x000A:   JL fin
0x000B:   JGE main
0x000C:   RET
```

Despl $=$ $\underbrace{Dir\_Etiq}_{Destino} - \underbrace{Dir\_Instr}_{Origen(postfetch)}$

desplazamiento fin:

La etiqueta main corresponde a la dirección de memoria 0x0000.
La etiqueta at1 corresponde a la dirección de memoria 0x0006.
La etiqueta loop corresponde a la dirección de memoria 0x0008.
La etiqueta fin corresponde a la dirección de memoria 0x000C.

```
0x0000:   MOV R1,[[0x0006]]
0x0002:   ADD [R1],0x6000
0x0004:   JMP 0x0000
0x0006:   DW 0x0007
0x0007:   DW 0x0004
0x0008:   CMP R1,0x0001
0x000A:   JL fin
0x000B:   JGE main
0x000C:   RET
```

Despl =
$\underbrace{Dir\_Etiq}_{Destino} - \underbrace{Dir\_Instr}_{Origen(postfetch)}$

desplazamiento fin:
0x000C − 0x000B

La etiqueta main corresponde a la dirección de memoria 0x0000.
La etiqueta at1 corresponde a la dirección de memoria 0x0006.
La etiqueta loop corresponde a la dirección de memoria 0x0008.
La etiqueta fin corresponde a la dirección de memoria 0x000C.

```
0x0000:   MOV R1,[[0x0006]]
0x0002:   ADD [R1],0x6000
0x0004:   JMP 0x0000
0x0006:   DW 0x0007
0x0007:   DW 0x0004
0x0008:   CMP R1,0x0001
0x000A:   JL fin
0x000B:   JGE main
0x000C:   RET
```

Despl =
$\underbrace{Dir\_Etiq}_{Destino} - \underbrace{Dir\_Instr}_{Origen(postfetch)}$

desplazamiento fin:
0x000C − 0x000B = 1_{10}

La etiqueta main corresponde a la dirección de memoria 0x0000.
La etiqueta at1 corresponde a la dirección de memoria 0x0006.
La etiqueta loop corresponde a la dirección de memoria 0x0008.
La etiqueta fin corresponde a la dirección de memoria 0x000C.

```
0x0000:   MOV R1,[[0x0006]]
0x0002:   ADD [R1],0x6000
0x0004:   JMP 0x0000
0x0006:   DW 0x0007
0x0007:   DW 0x0004
0x0008:   CMP R1,0x0001
0x000A:   JL fin
0x000B:   JGE main
0x000C:   RET
```

Despl =
$\underbrace{Dir\_Etiq}_{Destino} - \underbrace{Dir\_Instr}_{Origen(postfetch)}$

desplazamiento fin:
0x000C − 0x000B = 1_{10} = 0x01

La etiqueta main corresponde a la dirección de memoria 0x0000.
La etiqueta at1 corresponde a la dirección de memoria 0x0006.
La etiqueta loop corresponde a la dirección de memoria 0x0008.
La etiqueta fin corresponde a la dirección de memoria 0x000C.

```
0x0000:   MOV R1,[[0x0006]]
0x0002:   ADD [R1],0x6000
0x0004:   JMP 0x0000
0x0006:   DW 0x0007
0x0007:   DW 0x0004
0x0008:   CMP R1,0x0001
0x000A:   JL fin
0x000B:   JGE main
0x000C:   RET
```

Despl = $\underbrace{Dir\_Etiq}_{Destino} - \underbrace{Dir\_Instr}_{Origen(postfetch)}$

desplazamiento fin:
0x000C − 0x000B = 1_{10} = 0x01

desplazamiento main:

La etiqueta main corresponde a la dirección de memoria 0x0000.
La etiqueta at1 corresponde a la dirección de memoria 0x0006.
La etiqueta loop corresponde a la dirección de memoria 0x0008.
La etiqueta fin corresponde a la dirección de memoria 0x000C.

```
0x0000:   MOV R1,[[0x0006]]
0x0002:   ADD [R1],0x6000
0x0004:   JMP 0x0000
0x0006:   DW 0x0007
0x0007:   DW 0x0004
0x0008:   CMP R1,0x0001
0x000A:   JL fin
0x000B:   JGE main
0x000C:   RET
```

Despl = $\underbrace{Dir\_Etiq}_{Destino} - \underbrace{Dir\_Instr}_{Origen(postfetch)}$

desplazamiento fin:
0x000C − 0x000B = 1_{10} = 0x01

desplazamiento main:

La etiqueta main corresponde a la dirección de memoria 0x0000.
La etiqueta at1 corresponde a la dirección de memoria 0x0006.
La etiqueta loop corresponde a la dirección de memoria 0x0008.
La etiqueta fin corresponde a la dirección de memoria 0x000C.

```
0x0000:   MOV R1,[[0x0006]]
0x0002:   ADD [R1],0x6000
0x0004:   JMP 0x0000
0x0006:   DW 0x0007
0x0007:   DW 0x0004
0x0008:   CMP R1,0x0001
0x000A:   JL fin
0x000B:   JGE main
0x000C:   RET
```

Despl =
$\underbrace{Dir\_Etiq}_{Destino} - \underbrace{Dir\_Instr}_{Origen(postfetch)}$

desplazamiento fin:
0x000C − 0x000B = 1_{10} =
0x01

desplazamiento main:
0x0000 − 0x000C

La etiqueta main corresponde a la dirección de memoria 0x0000.
La etiqueta at1 corresponde a la dirección de memoria 0x0006.
La etiqueta loop corresponde a la dirección de memoria 0x0008.
La etiqueta fin corresponde a la dirección de memoria 0x000C.

```
0x0000:   MOV R1,[[0x0006]]
0x0002:   ADD [R1],0x6000
0x0004:   JMP 0x0000
0x0006:   DW 0x0007
0x0007:   DW 0x0004
0x0008:   CMP R1,0x0001
0x000A:   JL fin
0x000B:   JGE main
0x000C:   RET
```

$$\text{Despl} = \underbrace{Dir\_Etiq}_{Destino} - \underbrace{Dir\_Instr}_{Origen(postfetch)}$$

desplazamiento fin:
$0x000C - 0x000B = 1_{10} = 0x01$

desplazamiento main:
$0x0000 - 0x000C = -12_{10}$

La etiqueta main corresponde a la dirección de memoria 0x0000.
La etiqueta at1 corresponde a la dirección de memoria 0x0006.
La etiqueta loop corresponde a la dirección de memoria 0x0008.
La etiqueta fin corresponde a la dirección de memoria 0x000C.

```
0x0000:   MOV R1,[[0x0006]]
0x0002:   ADD [R1],0x6000
0x0004:   JMP 0x0000
0x0006:   DW 0x0007
0x0007:   DW 0x0004
0x0008:   CMP R1,0x0001
0x000A:   JL fin
0x000B:   JGE main
0x000C:   RET
```

Despl = $\underbrace{Dir\_Etiq}_{Destino} - \underbrace{Dir\_Instr}_{Origen(postfetch)}$

desplazamiento fin:
0x000C − 0x000B = 1_{10} = 0x01

desplazamiento main:
0x0000 − 0x000C = -12_{10} = 0xF4

La etiqueta main corresponde a la dirección de memoria 0x0000.
La etiqueta at1 corresponde a la dirección de memoria 0x0006.
La etiqueta loop corresponde a la dirección de memoria 0x0008.
La etiqueta fin corresponde a la dirección de memoria 0x000C.

```
0x0000:   MOV R1,[[0x0006]]
0x0002:   ADD [R1],0x6000
0x0004:   JMP 0x0000
0x0006:   DW 0x0007
0x0007:   DW 0x0004
0x0008:   CMP R1,0x0001
0x000A:   JL 0x01
0x000B:   JGE 0xF4
0x000C:   RET
```

Despl = $\underbrace{Dir\_Etiq}_{Destino} - \underbrace{Dir\_Instr}_{Origen(postfetch)}$

desplazamiento fin:
0x000C − 0x000B = 1_{10} = 0x01

desplazamiento main:
0x0000 − 0x000C = -12_{10} = 0xF4

La etiqueta main corresponde a la dirección de memoria 0x0000.
La etiqueta at1 corresponde a la dirección de memoria 0x0006.
La etiqueta loop corresponde a la dirección de memoria 0x0008.
La etiqueta fin corresponde a la dirección de memoria 0x000C.

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
0008	0009	000A	000B	000C	000D	000E	000F

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
0008	0009	000A	000B	000C	000D	000E	000F

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
0008	0009	000A	000B	000C	000D	000E	000F

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006						
0008	0009	000A	000B	000C	000D	000E	000F

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006						
0008	0009	000A	000B	000C	000D	000E	000F

Y también cargamos la dirección inicial en el **PC**:

PC	

```
MOV R1,[[0x0006]]
ADD [R1],0x6000
JMP 0x0000
DW 0x0007
DW 0x0004
CMP R1,0x0001
JL 0x01
JGE 0xF4
RET
```

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000				
0008	0009	000A	000B	000C	000D	000E	000F

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000				
0008	0009	000A	000B	000C	000D	000E	000F

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000		
0008	0009	000A	000B	000C	000D	000E	000F

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000		
0008	0009	000A	000B	000C	000D	000E	000F

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000	0007	
0008	0009	000A	000B	000C	000D	000E	000F

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000	0007	
0008	0009	000A	000B	000C	000D	000E	000F

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000	0007	0004
0008	0009	000A	000B	000C	000D	000E	000F

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000	0007	0004
0008	0009	000A	000B	000C	000D	000E	000F

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000	0007	0004
0008	0009	000A	000B	000C	000D	000E	000F
6840	0001						

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000	0007	0004
0008	0009	000A	000B	000C	000D	000E	000F
6840	0001						

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000	0007	0004
0008	0009	000A	000B	000C	000D	000E	000F
6840	0001	F301					

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000	0007	0004
0008	0009	000A	000B	000C	000D	000E	000F
6840	0001	F301					

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000	0007	0004
0008	0009	000A	000B	000C	000D	000E	000F
6840	0001	F301	FBF4				

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000	0007	0004
0008	0009	000A	000B	000C	000D	000E	000F
6840	0001	F301	FBF4				

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000	0007	0004
0008	0009	000A	000B	000C	000D	000E	000F
6840	0001	F301	FBF4	C000			

Y también cargamos la dirección inicial en el **PC**:

PC	

	dirección	ocupa
MOV R1,[[0x0006]]	0x0000	dos palabras
ADD [R1],0x6000	0x0002	dos palabras
JMP 0x0000	0x0004	dos palabras
DW 0x0007	0x0006	una palabra
DW 0x0004	0x0007	una palabra
CMP R1,0x0001	0x0008	dos palabra
JL 0x01	0x000A	una palabra
JGE 0xF4	0x000B	una palabra
RET	0x000C	una palabra

Traducidas las etiquetas lo codificamos y cargamos en memoria:

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000	0007	0004
0008	0009	000A	000B	000C	000D	000E	000F
6840	0001	F301	FBF4	C000	0000	0000	0000

Y también cargamos la dirección inicial en el **PC**:

PC	0000

Caaaambioo Jueeeez!!!

Empezamos a hacer el ciclo **Fetch-Decode-Execute** con el **PC** = 0x0000.

0000	0001	0002	0003	0004	0005	0006	0007
1858	0006	2C40	6000	A000	0000	0007	0004
0008	0009	000A	000B	000C	000D	000E	000F
6840	0001	F301	FBF4	C000	0000	0000	0000

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1											
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000										
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000								
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	1858							
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	1858	0001 1000 0101 1000						
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001									
	Ejecución										Flags				
2															
	Ejecución										Flags				
3															
	Ejecución										Flags				

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags
2	0002										
	Ejecución										Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags
2	0002			2C40							
	Ejecución										Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags
2	0002			2C40	0010 1100 0100 0000						
	Ejecución										Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags
2	0002			2C40	0010 1100 0100 0000	0003					
	Ejecución										Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags
2	0002			2C40	0010 1100 0100 0000	0003	6000	0004			
	Ejecución										Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags
2	0002			2C40	0010 1100 0100 0000	0003	6000	0004			ADD [R1],0x6000
	Ejecución										Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags
2	0002			2C40	0010 1100 0100 0000	0003	6000	0004			ADD [R1],0x6000
	Ejecución	[0x0004]=0xA000+0x6000=0x0000									Flags
3											
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	0000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]				
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags				
2	0002			2C40	0010 1100 0100 0000	0003	6000	0004			ADD [R1],0x6000				
	Ejecución	[0x0004]=0xA000+0x6000=0x0000									Flags	1	1	0	0
3															
	Ejecución										Flags				

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	0000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]				
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags				
2	0002			2C40	0010 1100 0100 0000	0003	6000	0004			ADD [R1],0x6000				
	Ejecución	[0x0004]=0xA000+0x6000=0x0000									Flags	1	1	0	0
3	0004														
	Ejecución										Flags				

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	0000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]				
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags				
2	0002			2C40	0010 1100 0100 0000	0003	6000	0004			ADD [R1],0x6000				
	Ejecución	[0x0004]=0xA000+0x6000=0x0000									Flags	1	1	0	0
3	0004			0000											
	Ejecución										Flags				

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	0000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]				
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags				
2	0002			2C40	0010 1100 0100 0000	0003	6000	0004			ADD [R1],0x6000				
	Ejecución	[0x0004]=0xA000+0x6000=0x0000									Flags	1	1	0	0
3	0004			0000	0000 0000 0000 0000										
	Ejecución										Flags				

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	0000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]				
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags				
2	0002			2C40	0010 1100 0100 0000	0003	6000	0004			ADD [R1],0x6000				
	Ejecución	[0x0004]=0xA000+0x6000=0x0000									Flags	1	1	0	0
3	0004			0000	0000 0000 0000 0000										
	Ejecución										Flags				

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	0000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]				
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags				
2	0002			2C40	0010 1100 0100 0000	0003	6000	0004			ADD [R1],0x6000				
	Ejecución	[0x0004]=0xA000+0x6000=0x0000									Flags	1	1	0	0
3	0004			0000	0000 0000 0000 0000	0005									
	Ejecución										Flags				

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	0000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags
2	0002			2C40	0010 1100 0100 0000	0003	6000	0004			ADD [R1],0x6000
	Ejecución	[0x0004]=0xA000+0x6000=0x0000									Flags 1 1 0 0
3	0004			0000	0000 0000 0000 0000	0005					Instrucción Inválida
	Ejecución										Flags

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	0000	0000	0007	0004	6840	0001	F301	FBF4	C000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]
	Ejecución	R1=[[0x0006]]=[0x0007]=4									Flags
2	0002			2C40	0010 1100 0100 0000	0003	6000	0004			ADD [R1],0x6000
	Ejecución	[0x0004]=0xA000+0x6000=0x0000									Flags 1 1 0 0
3	0004			0000	0000 0000 0000 0000	0005					Instrucción Inválida
	Ejecución	FIN DE LA EJECUCIÓN									Flags

Ejercicio 1 - Solución

0000

Planilla de Seguimiento

Valores iniciales:

PC	SP
0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

Memoria:

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	1858	0006	2C40	6000	A000	0000	0007	0004	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1^{er} palabra (en bits)	PC	2^{da} palabra	PC	3^{era} palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	1858	0001 1000 0101 1000	0001	0006	0002			MOV R1, [[0x0006]]				
	Ejecución		R1 = [[0x0006]] = [0x0007] = 4								Flags[1]	Z	C	V	N
2	0002			2C40	0010 1100 0100 0000	0003	6000	0004			ADD [R1], 0x6000				
	Ejecución		[0x0004] = 0xA000 + 0x6000 = 0x0000								Flags[1]	Z 1	C 1	V 0	N 0
3	0004			0000	0000 0000 0000 0000	0005					Instrucción inválida				
	Ejecución		FIN DE LA EJECUCIÓN								Flags[1]	Z	C	V	N

Ejercicio 1 - Observaciones

¿Qué pasó con el programa anterior?

Ejercicio 1 - Observaciones

¿Qué pasó con el programa anterior?

En la ejecución del programa cambió una posición de memoria donde estaba cargado... **el mismo programa!**

Ejercicio 1 - Observaciones

¿Qué pasó con el programa anterior?

En la ejecución del programa cambió una posición de memoria donde estaba cargado... **el mismo programa!**

Eso hizo que nuestro programa se modificara, haciendo que la máquina llegase a una instrucción inválida.

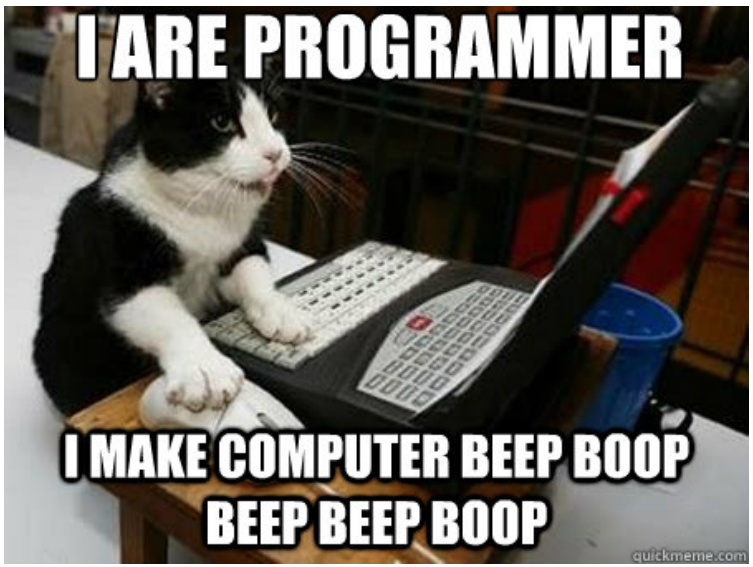
Ejercicio 1 - Observaciones

¿Qué pasó con el programa anterior?

En la ejecución del programa cambió una posición de memoria donde estaba cargado... **el mismo programa!**

Eso hizo que nuestro programa se modificara, haciendo que la máquina llegase a una instrucción inválida.

Moraleja:



Programen con cuidado y atención.

Ejercicio 2

Realizar el seguimiento de la ejecución del siguiente programa cargado en memoria a partir de la dirección 0x0000.

0000	0001	0002	0003	0004	0005	0006	0007
19C8	0004	B000	0005	0FE0	39A7	C000	0000

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1											
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000										
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000								
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB							
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000						
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001					
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución										Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP	R0	R1	R2	R3	R4	R5	R6	R7	Z	C	V	N
	0000	FFEF	0000	0000	0000	0000	0000	0000	0000	0000	0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2											
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002										
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000							
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000						
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP	R0	R1	R2	R3	R4	R5	R6	R7	Z	C	V	N
	0000	FFEF	0000	0000	0000	0000	0000	0000	0000	0000	0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003					
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución										Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3											
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3	0005										
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP	R0	R1	R2	R3	R4	R5	R6	R7	Z	C	V	N
	0000	FFEF	0000	0000	0000	0000	0000	0000	0000	0000	0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3	0005	FFEE									
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3	0005	FFEE	0004	39A7							
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3	0005	FFEE	0004	39A7	0011 1001 1010 0111						
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7
	Ejecución										Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	F020	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags 0 1 0 1
4											
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	F020	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags 0 1 0 1
4	0006										
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	F020	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]				
	Ejecución	R7=[0x0004]=0x0FE0									Flags				
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005				
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags				
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7				
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags	0	1	0	1
4	0006			C000											
	Ejecución										Flags				
5															
	Ejecución										Flags				

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	F020	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags 0 1 0 1
4	0006			C000	1100 0000 0000 0000						
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	F020	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags 0 1 0 1
4	0006			C000	1100 0000 0000 0000	0007					
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	F020	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags 0 1 0 1
4	0006			C000	1100 0000 0000 0000	0007					RET
	Ejecución										Flags
5											
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	F020	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]				
	Ejecución	R7=[0x0004]=0x0FE0									Flags				
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005				
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags				
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7				
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags	0	1	0	1
4	0006			C000	1100 0000 0000 0000	0007					RET				
	Ejecución	PC=[0xFFEF]=0x0004 SP=0xFFEF									Flags				
5															
	Ejecución										Flags				

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	F020	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]				
	Ejecución	R7=[0x0004]=0x0FE0									Flags				
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005				
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags				
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7				
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags	0	1	0	1
4	0006			C000	1100 0000 0000 0000	0007					RET				
	Ejecución	PC=[0xFFEF]=0x0004 SP=0xFFEF									Flags				
5	0004														
	Ejecución										Flags				

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	F020	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]				
	Ejecución	R7=[0x0004]=0x0FE0									Flags				
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005				
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags				
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7				
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags	0	1	0	1
4	0006			C000	1100 0000 0000 0000	0007					RET				
	Ejecución	PC=[0xFFEF]=0x0004 SP=0xFFEF									Flags				
5	0004	FFEF	0000												
	Ejecución										Flags				

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	F020	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]				
	Ejecución	R7=[0x0004]=0x0FE0									Flags				
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005				
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags				
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7				
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags	0	1	0	1
4	0006			C000	1100 0000 0000 0000	0007					RET				
	Ejecución	PC=[0xFFEF]=0x0004 SP=0xFFEF									Flags				
5	0004	FFEF	0000	0FE0											
	Ejecución										Flags				

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	F020	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]				
	Ejecución	R7=[0x0004]=0x0FE0									Flags				
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005				
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags				
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7				
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags	0	1	0	1
4	0006			C000	1100 0000 0000 0000	0007					RET				
	Ejecución	PC=[0xFFEF]=0x0004 SP=0xFFEF									Flags				
5	0004	FFEF	0000	0FE0	0000 1111 1110 0000										
	Ejecución										Flags				

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	F020	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]				
	Ejecución	R7=[0x0004]=0x0FE0									Flags				
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005				
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags				
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7				
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags	0	1	0	1
4	0006			C000	1100 0000 0000 0000	0007					RET				
	Ejecución	PC=[0xFFEF]=0x0004 SP=0xFFEF									Flags				
5	0004	FFEF	0000	0FE0	0000 1111 1110 0000	0005									
	Ejecución										Flags				

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	F020	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags 0 1 0 1
4	0006			C000	1100 0000 0000 0000	0007					RET
	Ejecución	PC=[0xFFEF]=0x0004 SP=0xFFEF									Flags
5	0004	FFEF	0000	0FE0	0000 1111 1110 0000	0005					Instrucción Inválida
	Ejecución										Flags

Ejercicio 2 - Seguimiento

Valores iniciales	PC	SP
	0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	F020	0000

Z	C	V	N
0	0	0	0

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19CB	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0010	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1er palabra	PC	2da palabra	PC	3ra palabra	PC	Instrucción decodificada
1	0000	FFEF	0000	19CB	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]
	Ejecución	R7=[0x0004]=0x0FE0									Flags
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005
	Ejecución	[0xFFEF]=0x0004 SP=0xFFEE PC=0x0005									Flags
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7
	Ejecución	R6=0x0000-0x0FE0=0xF020									Flags 0 1 0 1
4	0006			C000	1100 0000 0000 0000	0007					RET
	Ejecución	PC=[0xFFEF]=0x0004 SP=0xFFEF									Flags
5	0004	FFEF	0000	0FE0	0000 1111 1110 0000	0005					Instrucción Inválida
	Ejecución	FIN DE LA EJECUCIÓN									Flags

Ejercicio 2 - Solución

Planilla de Seguimiento

F020

Valores iniciales:

PC	SP
0000	FFEF

R0	R1	R2	R3	R4	R5	R6	R7
0000	0000	0000	0000	0000	0000	0000	0000

Z	C	V	N
0	0	0	0

Memoria:

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	19C8	0004	B000	0005	0FE0	39A7	C000	0000	0000	0000	0000	0000	0000	0000	0000	0000
0001	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

	PC	SP	[SP+1]	IR	Instrucción - 1^{er} palabra (en bits)	PC	2^{da} palabra	PC	3^{era} palabra	PC	Instrucción decodificada				
1	0000	FFEF	0000	19C8	0001 1001 1100 1000	0001	0004	0002			MOV R7, [0x0004]				
	Ejecución		R7 = [0x0004] = 0x0FE0								Flags[1]	Z	C	V	N
2	0002			B000	1011 0000 0000 0000	0003	0005	0004			CALL 0x0005				
	Ejecución		[0xFFEF] = 0x0004 SP = 0xFFEE PC = 0x0005								Flags[1]	Z	C	V	N
3	0005	FFEE	0004	39A7	0011 1001 1010 0111	0006					SUB R6, R7				
	Ejecución		R6 = 0x0000 - 0x0FE0 = 0xF020								Flags[1]	Z 0	C 1	V 0	N 1
4	0006			C000	1100 0000 0000 0000	0007					RET				
	Ejecución		PC = [0xFFEF] = 0x0004 SP = 0xFFEF								Flags[1]	Z	C	V	N
5	0004	FFEF	0000	0FE0	0000 1111 1110 0000	0005					Instrucción Inválida				
	Ejecución		FIN DE LA EJECUCIÓN.								Flags[1]	Z	C	V	N

Ejercicio 2 - Observaciones

¿Qué pasó con el programa anterior?

Ejercicio 2 - Observaciones

¿Qué pasó con el programa anterior?

- Utilizamos la instrucción CALL.
- El valor del PC de retorno se almacena en la pila durante de la ejecución de la instrucción.
- Recuerden que la pila crece en el sentido de las direcciones de memoria menores.
- Al regresar con RET, se "saca" la dirección de retorno de allí.

Programación en Asembler

Acordando pautas...

- Se utilizará el lenguaje Assembler de la máquina 'Orga 1'
- Es un lenguaje ensamblador particular de esta arquitectura, que es una simplificación de una arquitectura Intel x86
- Posee instrucciones con 2 operandos, 1 operando, 0 operandos y saltos
- Posee diferentes modos de direccionamiento

Enunciado Ejercicio Nro 1

Estaría muy bueno manejar algunas operaciones en Assembler. Y la división tiene sus vericuetos....Por lo tanto vamos a escribir un programa que calcule la división entera entre dos enteros sin signo de 16 bits.

- R1 contiene la dirección de memoria donde se aloja el dividendo.
- R2 contiene la dirección de memoria donde se aloja el divisor.
- R3 debe ser el registro en el que se devuelva el cociente (el resultado de la división).

Como somos muy buenos...En caso de que el divisor sea 0, basta con que se retorne 0.

Pseudocódigo

```
resultado = 0
si divisor == 0 entonces
    listo
sino
    mientras dividendo >= divisor hacer
        dividendo = dividendo - divisor
        resultado = resultado + 1
    fin mientras
fin si
```

Assembler!

```
; R1 --> puntero al dividendo
; R2 --> puntero al divisor
; R3 --> cociente
; R4 --> dividendo
; R5 --> divisor
  inicio: MOV R3, 0x0000   ; R3 = 0
          MOV R4, [R1]     ; R4 = dividendo
          MOV R5, [R2]     ; R5 = divisor
          CMP R5, 0x0000   ; divisor == 0?
          JE fin
  ciclo:  CMP R4, R5       ; dividendo < divisor?
          JCS fin ; uso JCS en vez de JL porque son enteros sin signo
          SUB R4, R5       ; R4 = R4-R5
          ADD R3, 0x0001   ; R3 = R3+1
          JMP ciclo
  fin:    (...)
```

Tarea: Retornar en R0 el resto de la division!

Enunciado Ejercicio Nro 2

Los docentes del turno tarde precisamos ingresos extras... compramos una pareja de conejos y queremos saber cuántos se podrían reproducir en un año a partir de la pareja inicial, teniendo en cuenta que de forma natural tienen una pareja en un mes, y que a partir del segundo mes se empiezan a reproducir.

Sucesión de Fibonacci

$$f_n = f_{n-1} + f_{n-2} \text{ con } f_0{=}1 \text{ y } f_1{=}1$$

Pseudocódigo

```
meses = 10; //(12-2) me da el término 12 de la suc.
anteUltMes = 1;
ultMes = 1;
cantParejas = ultMes;
mientras meses > 0
    cantParejas = cantParejas + anteUltMes;
    anteUltMes = ultMes;
    ultMes = cantParejas;
    meses = meses - 1;
fin mientras
```

Assembler!

```
; R0 --> meses
; R1 --> anteUltMes
; R2 --> ultMes
; R3 --> cantParejas
inicio:   MOV R0, 0x000A        ; meses = 10
          MOV R1, 0x0001        ; anteUltMes = 1
          MOV R2, 0x0001        ; ultMes = 1
          MOV R3, R2            ; cantParejas = ultMes;
ciclo:    ADD R3, R1            ; cantParejas += anteUltMes;
          MOV R1, R2            ; anteUltMes = ultMes;
          MOV R2, R3            ; ultMes = cantParejas;
          SUB R0, 0x0001        ; meses = meses - 1;
          JG ciclo              ;
fin:      (...)
```

Resumen de la clase

Hoy repasamos los siguiente conceptos:

- Tamaño de memoria
- Dirección de memoria
- Unidad direccionable
- Instrucción
- Codificación de una instrucción

Resumen de la clase

Hoy también vimos:

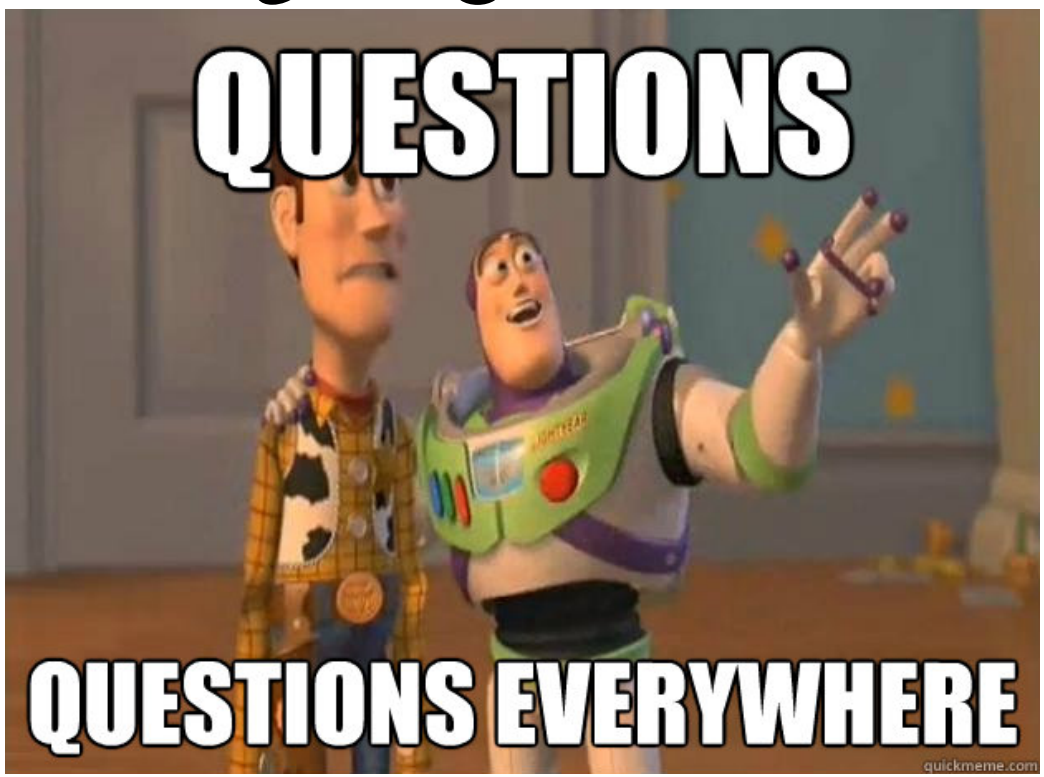
- Máquina Orga 1 y su arquitectura
  - ISA: formato de instrucción, modos de direccionamiento
  - ciclo de ejecución
- Ciclo de vida de un programa
  - uso de etiquetas
  - directivas
- Ensamblamos programas
- Seguimos programas dentro de la máquina Orga1

Resumen de la clase

Por último:

- Vimos como realizar pequeños programas en Assembler utilizando la Arquitectura 'Orga 1'
- Es importante utilizar pseudocódigo para resolver el problema y verificar que el algoritmo funcione
- Pueden utilizar un pseudocódigo informal, que pueda comprenderse y no sea ambiguo
- Una vez verificada la solución, realizar el código en el lenguaje Ensamblador de la materia

¿Preguntas?



Avisos

Martes 2/10: Taller de Ciclo de Instrucción .

IMPORTANTE: Traer Cartilla Orga 1 y planillas de Seguimiento

IMPORTANTE 2: Hacer la guia de ejercicios!!!!

¡Eso es todo amigos!