UPMDIE
INDUSTRIALES

3 - Arquitectura interna de un uP

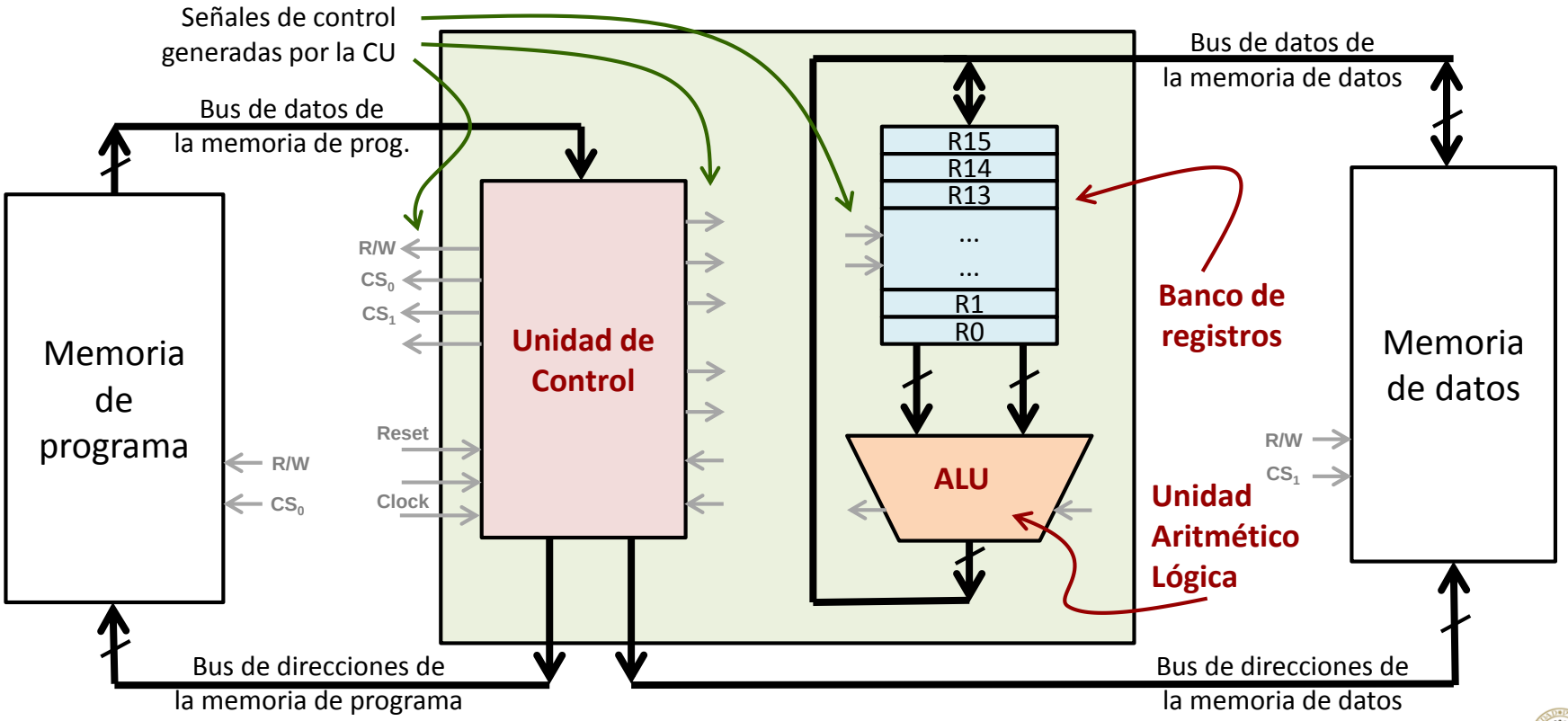
- *Componentes básicos*
- *Lenguaje ensamblador y código máquina*
- *Ciclo básico de ejecución de una instrucción*
- *Algunos ejemplos*

cei@upm.es

POLITÉCNICA

Componentes básicos de la CPU: Registros, ALU y CU

- CPU: Es la encargada de todas las operaciones y movimiento de datos.
  - ***Banco de registros:*** *almacena datos y resultados de operaciones con la ALU. El tamaño de los registros suele coincidir con el de la memoria de datos, y define el tipo de CPU (8 bits, 32 bits,…)*
  - ***Unidad Aritmético Lógica (ALU):*** *Realiza operaciones entre registros (+, -, and, or…)*
  - ***Unidad de control (CU):*** *Secuencia las acciones en función de la instrucción a ejecutar*



Lenguaje ensamblador y código máquina (I)

- ¿Qué tipo de instrucciones ejecuta una CPU?
  En principio son instrucciones muy básicas:

  - *Respecto a los datos, puede*:
    - **Cargar** un registro con un dato de la memoria o una constante
      - Ejemplos: **LD R0, 1000** == *R0 ← M(1000)* **LDI R0, 5** == *R0 ← 5*
    - **Operar** dos registros con la ALU (guardando el resultado el uno de ellos)
      - Ejemplos: **ADD R0, R3** == *R0 ← R0 + R3* **NOT R4** == *R4 ← $\underline{R4}$*
    - **Guardar** el contenido de un registro en la memoria o en otro registro
      - Ejemplos: **ST R3, FC00** == *M(FC00) ← R3* **MOV R5, R4** == *R4 ← R5*

  - *Respecto a las instrucciones, puede*:
    - **Ejecutar secuencialmente** una instrucción detrás de otra (según están en la memoria)
    - **Saltar** de una parte a otra del programa de forma **incondicional**
      - Ejemplo: **JMP 2000** == saltar a la instrucción que está en la posición 2000 y continuar la ejecución desde allí
    - **Saltar** unas cuantas instrucciones de forma **condicional** en función de los resultados de la última operación que hizo la ALU
      - Ejemplo: **BREQ 100** == Salta 100 instrucciones más adelante si el resultado de la última operación con la ALU fue 0
    - Saltar a una parte del programa en función de alguna señal externa (se verá)

Lenguaje ensamblador y código máquina (II)

- ¿Cómo se ejecutaría un programa escrito, por ejemplo, en C?
  - El compilador transforma el código en C para realizar la funcionalidad descrita a través de ese conjunto de instrucciones básicas
  - Ejemplo: *(versión algo simplificada respecto a cómo resultaría en la realidad):*

Código en C

```
// Declaración variables
byte a = 0;
byte i = 0;


// Bucle for
for(i = 0; i < 20; i++) {


  a = a + i;


}

...
...
```

Código en «ensamblador»

```
LDI r0, 0       ; cargamos r0 con 0, y lo almacenamos en la
ST  r0, 200     ; posición de memoria 200, que albergaría el valor
                ; de la variable 'a' (lo decide el compilador)
LDI r0, 0       ; idem con la posición 204,
ST  r0, 204     ; que albergaría la variable 'i'

LDI r0, 0       ; inicializamos la variable 'i' con 0
ST  r0, 204     ; lo guardo en su sitio

LD  r3, 200     ; cargo r3 con el valor de la variable 'a'
ADD r3, r0      ; le sumo el valor de 'i'
ST  r3, 200     ; y guardo el valor en su lugar (o sea, a = a + i)

INC r0          ; incremento el valor de 'i'
ST  r0, 204     ; lo guardo en su sitio para que el valor en
                ; la memoria y en los registros sea coherente
LDI r1, 20      ; cargamos r1 con 20 (el valor del final del bucle)
SUB r1, r0      ; le restamos r0 ('i'), para comparar si i < 20
BRGT -7         ; si el resultado es mayor que 0 entonces i < 20 y
                ; (BRanch if Greater Than) salto 7 posiciones
                ; hacia atrás. Si no, continuaría con la siguiente
...             ; instrucción que hubiese después
```

Lenguaje ensamblador y código máquina (III)

- ¿Cómo se codifican y almacenan las instrucciones en la memoria?
  - Las instrucciones en «lenguaje ensamblador» se codifican en un código binario. Según el micro, esta codificación podría hacerse en una o varias palabras por instrucción.
  - Para realizar la codificación, se usa un «programa ensamblador», que se encarga de traducir las instrucciones en «lenguaje ensamblador» a «código máquina»
  - Normalmente se utiliza una codificación por «campos»
    Ejemplo: Cómo quedarían algunas instrucciones del anterior programa en la memoria

Código ensamblador

```
...   ...
...   ...
3FF   ...
400   LDI r0, 0
401   ST  r0, 200
403   LDI r0, 0
404   ST  r0, 204
406   LD  r0, 204
408   LD  r3, 200
40A   ADD r3, r0
40B   ST  r3, 200
40D   INC r0
40E   ST  r0, 204
410   LDI r1, 20
411   SUB r1, r0
412   BRGT -7
413   ...
...   ...
```

El código ensamblador es una representación «legible» del código máquina

ADD r3, r0

Cod. op.	Reg dest.	Reg orig.	Type
Operación con la ALU	3	0	Suma
1001	0011	0000	0001

LDI r1, 20

Cod. op.	Reg dest.	val
Cargar constante	1	20
0001	0001	00010100

ST r3, 200

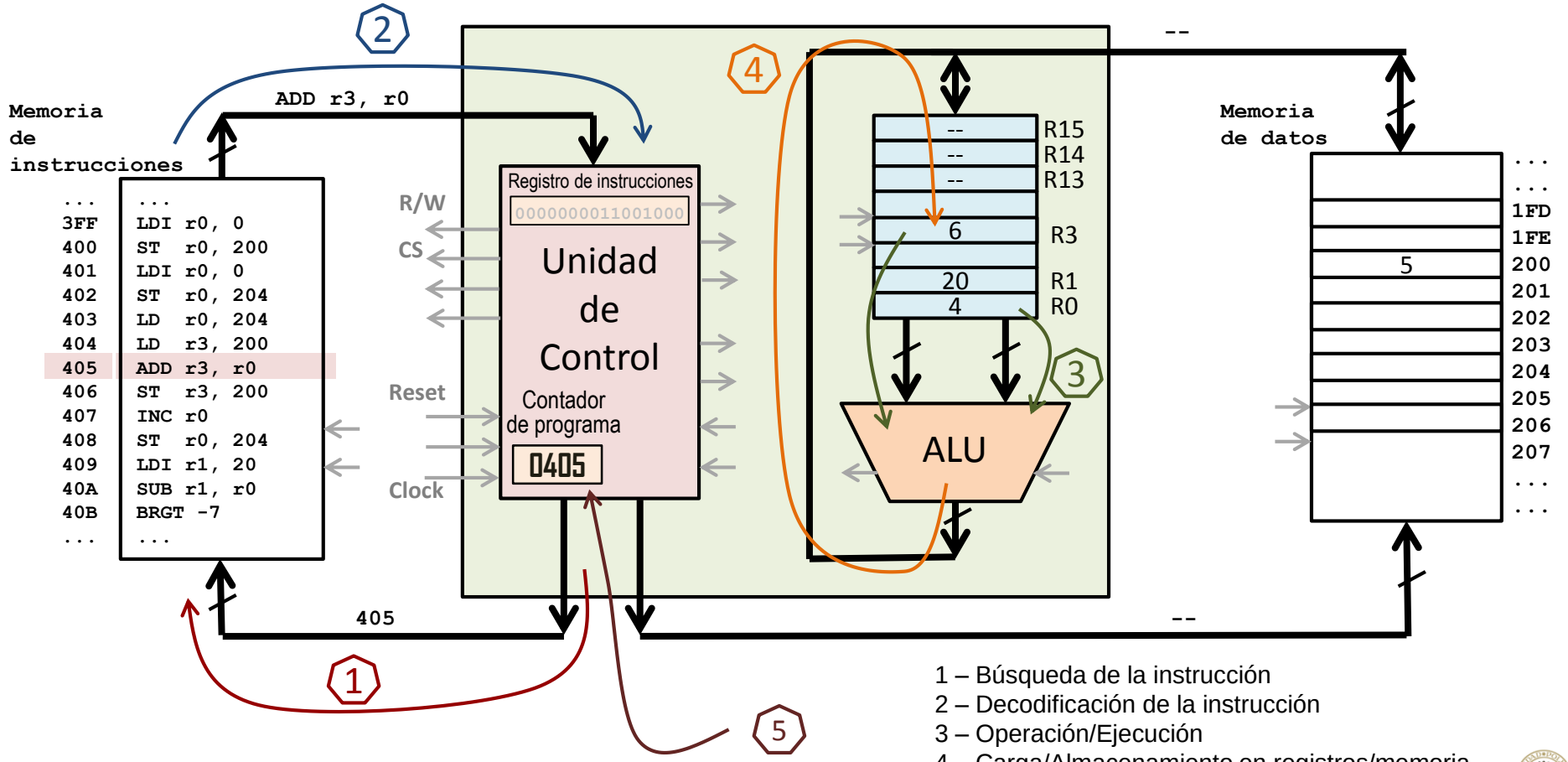
Cod. op.	Reg dest.	
val		
Guardar en memoria	3	no importa el valor
200		
0010	0011	00000000
0000000011001000		

```
...   ...
...   ...
3FF   ...
400   0001000000000000
401   0010000000000000
      0000000011001000
403   0001000100000001
404   0010000000000000
      0000000011001100
406   0011000000000000
      0000000011001100
408   0011001100000000
      0000000011001000
40A   1001001100000001
40B   0010001100000000
      0000000011001000
40D   etc
40E   etc
410   ...
...
...
...
```

Código máquina

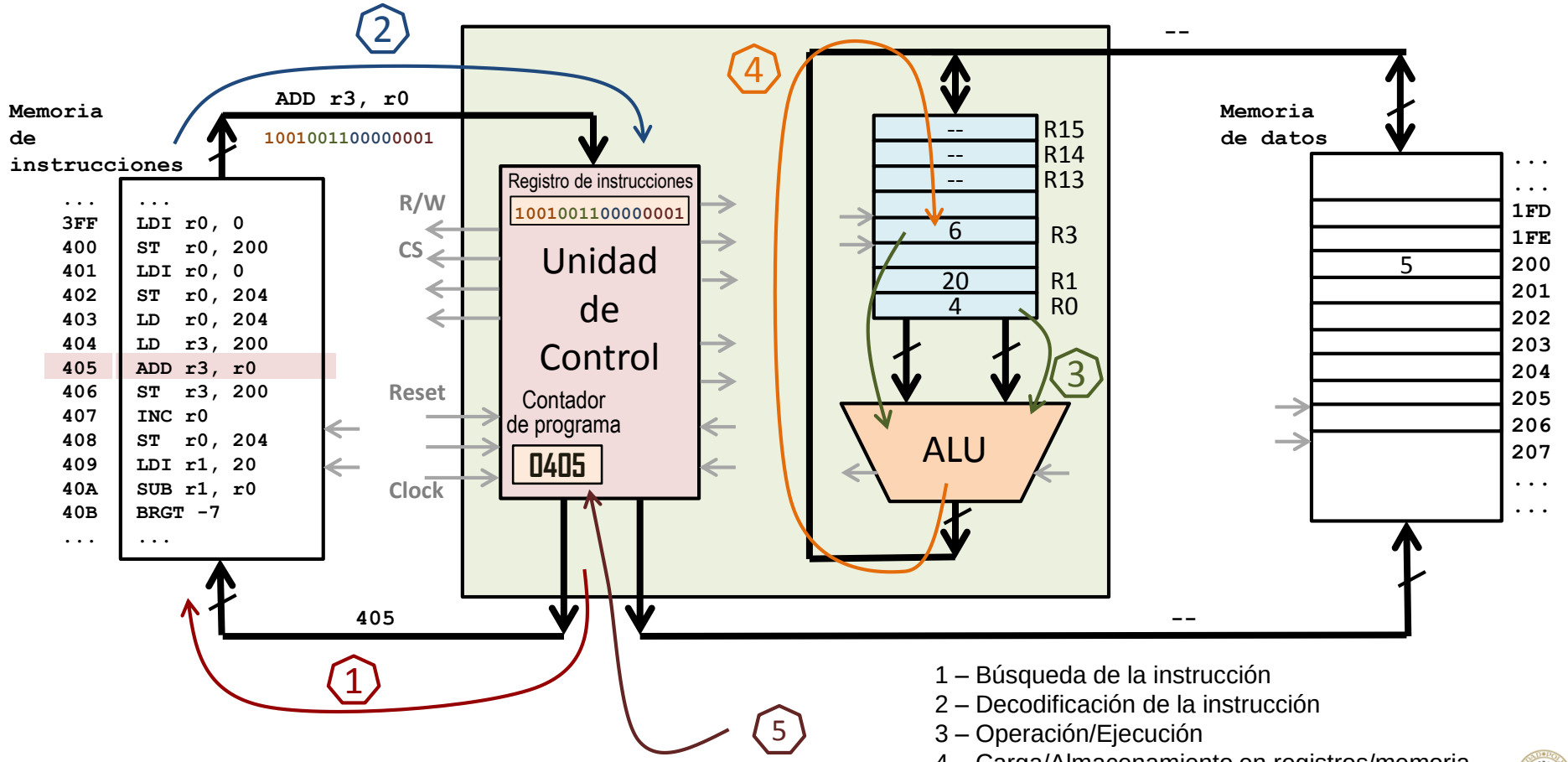
Ciclo básico de ejecución de una instrucción (III)

- ¿Cómo se ejecutan las instrucciones?
  - La CPU va leyendo cíclicamente instrucciones de la memoria de programa, las interpreta (en función de su codificación en binario), y la unidad de control (máquina de estados compleja) activa las señales de control correspondientes para ejecutarlas.



Ciclo básico de ejecución de una instrucción (III)

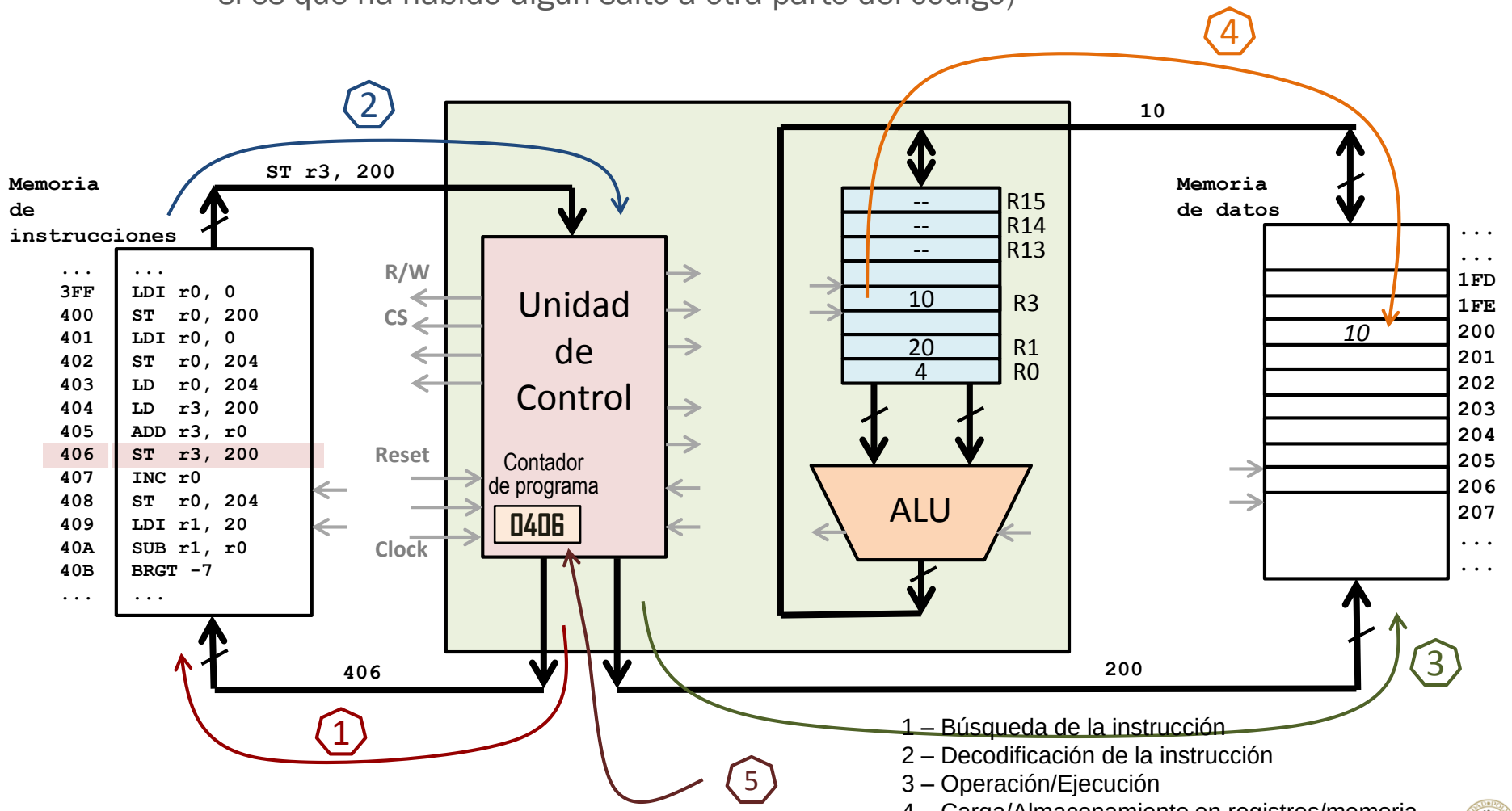
- ¿Cómo se ejecutan las instrucciones?
  - La CPU va leyendo cíclicamente instrucciones de la memoria de programa, las interpreta (en función de su codificación en binario), y la unidad de control (máquina de estados compleja) activa las señales de control correspondientes para ejecutarlas.



1 – Búsqueda de la instrucción
2 – Decodificación de la instrucción
3 – Operación/Ejecución
4 – Carga/Almacenamiento en registros/memoria
5 – Incremento del contador de programa

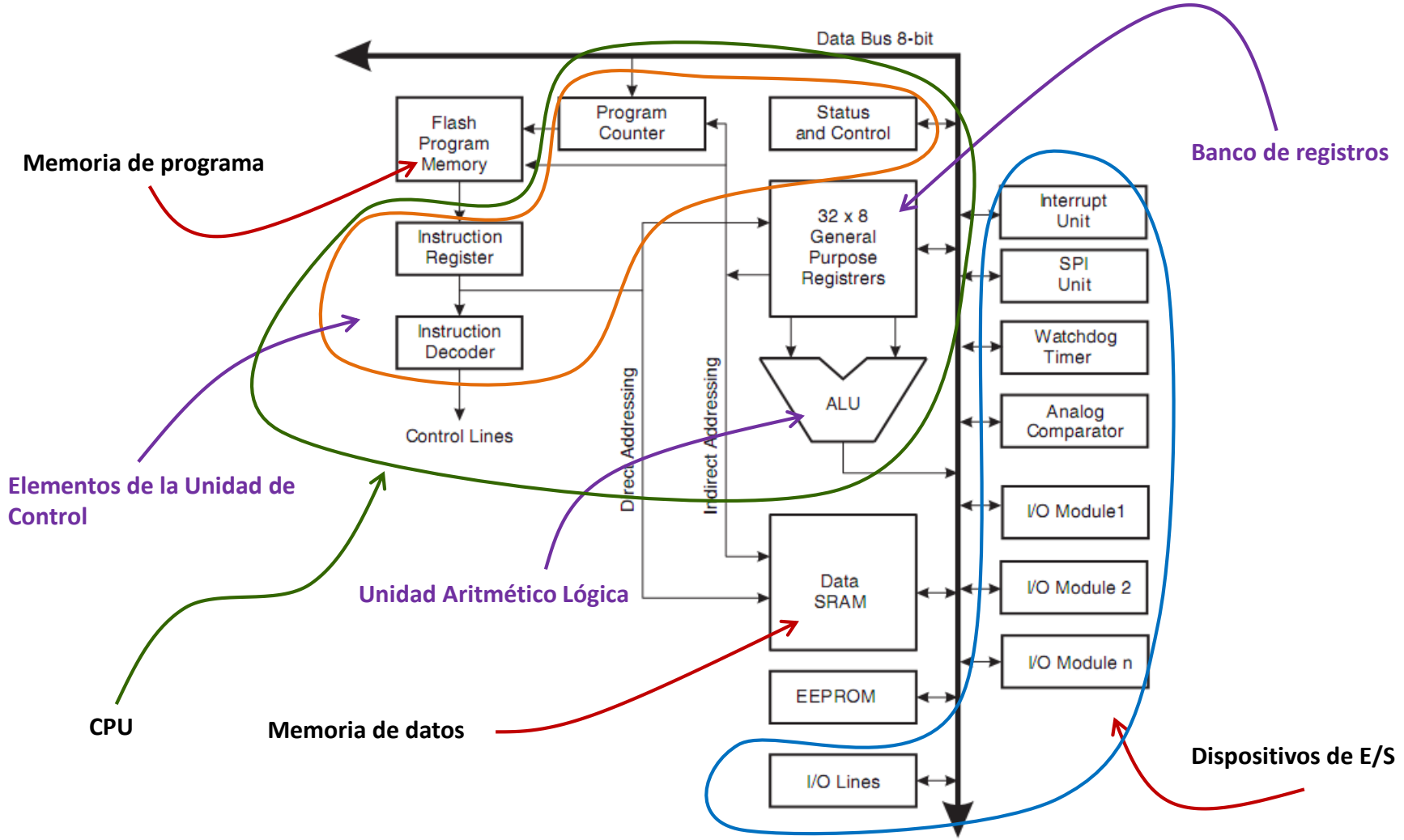
Ciclo básico de ejecución de una instrucción (IV)

- ¿Cómo se ejecutan las instrucciones?
  - Terminada una instrucción, se repite el ciclo con la siguiente (o con la que corresponda si es que ha habido algún salto a otra parte del código)



1 – Búsqueda de la instrucción
2 – Decodificación de la instrucción
3 – Operación/Ejecución
4 – Carga/Almacenamiento en registros/memoria
5 – Incremento del contador de programa

Ejemplo: arquitectura AVR

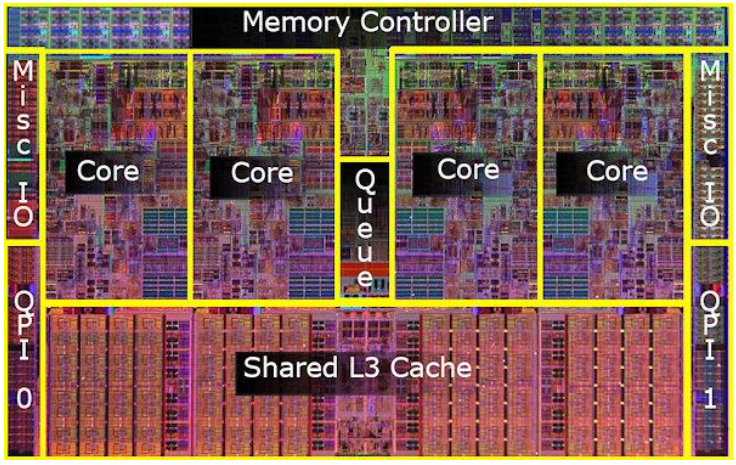


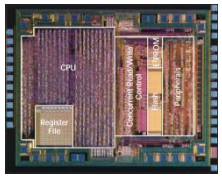
Extraído de la hoja de características de los microcontroladores AtMega de Atmel

Comparando: Un Intel I7 frente a un AtMega168

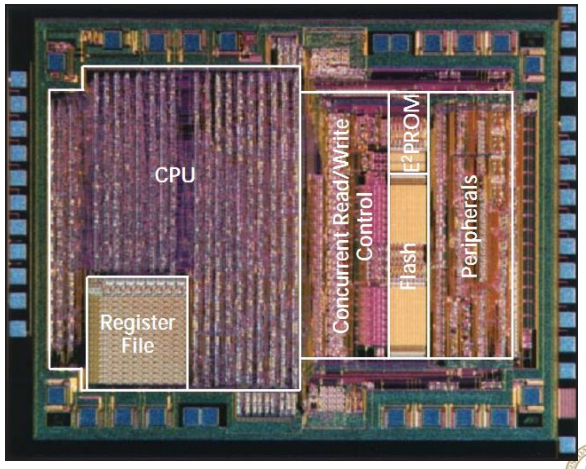
Cracterística	AtMega168	Intel Core I7 (Quad)	Ratio (aprox)
Ancho de bus	8	64	x8
Nº de transistores	< 100.000	~731.000.000	x10.000
Nº de pines	28	~1155	x40
Frecuencia de trabajo	16MHz	~3.2GHz	x200
Memoria de trabajo típica	16KB prog. / 1KB datos	~4-8GBytes	x1.000.000
MIPS/MFLOPS	16/0,5	~75.000/60.000	x5.000/x120.000
Consumo	20mW	~100W	x5.000
Precio	< 2$	~350$	x200

Intel Core i7 - 263 mm^2





AtMega - 24 mm^2



http://blog.ioactive.com/2008_01_01_archive.html http://blog.sina.com.cn/s/blog_54f48a4b01009bdf.html