Arquitectura de Computadores

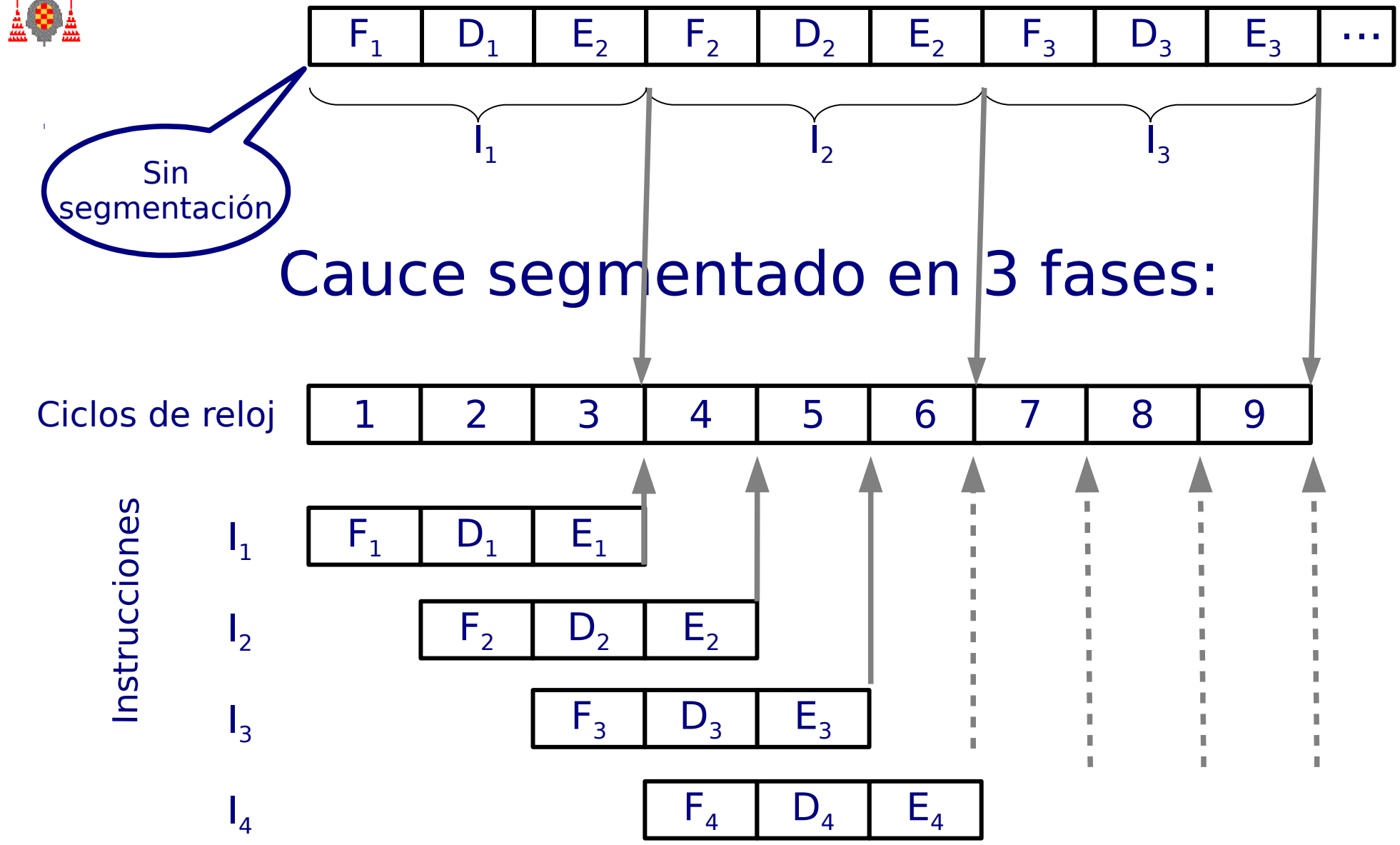
SEGMENTACIÓN DE CAUCE

Bibliografía

- Arquitectura de Computadores. Un enfoque cuantitativo" de Hennessy- Patterson, Mc Graw Hill, 1ª Edición 1993.
  - Cap.4, p. 172 (arquitec DLX), and Cap 6, p.273 en adelante (segmentación básica DLX)
- Arquitectura de Computadores, JULIO ORTEGA, MANCIA ANGUITA, ALBERTO PRIETO. Thomson-Paraninfo, 2005
  - Capítulo 2. Secciones: 2.1 a 2.3

Segmentación de cauce (pipelining)

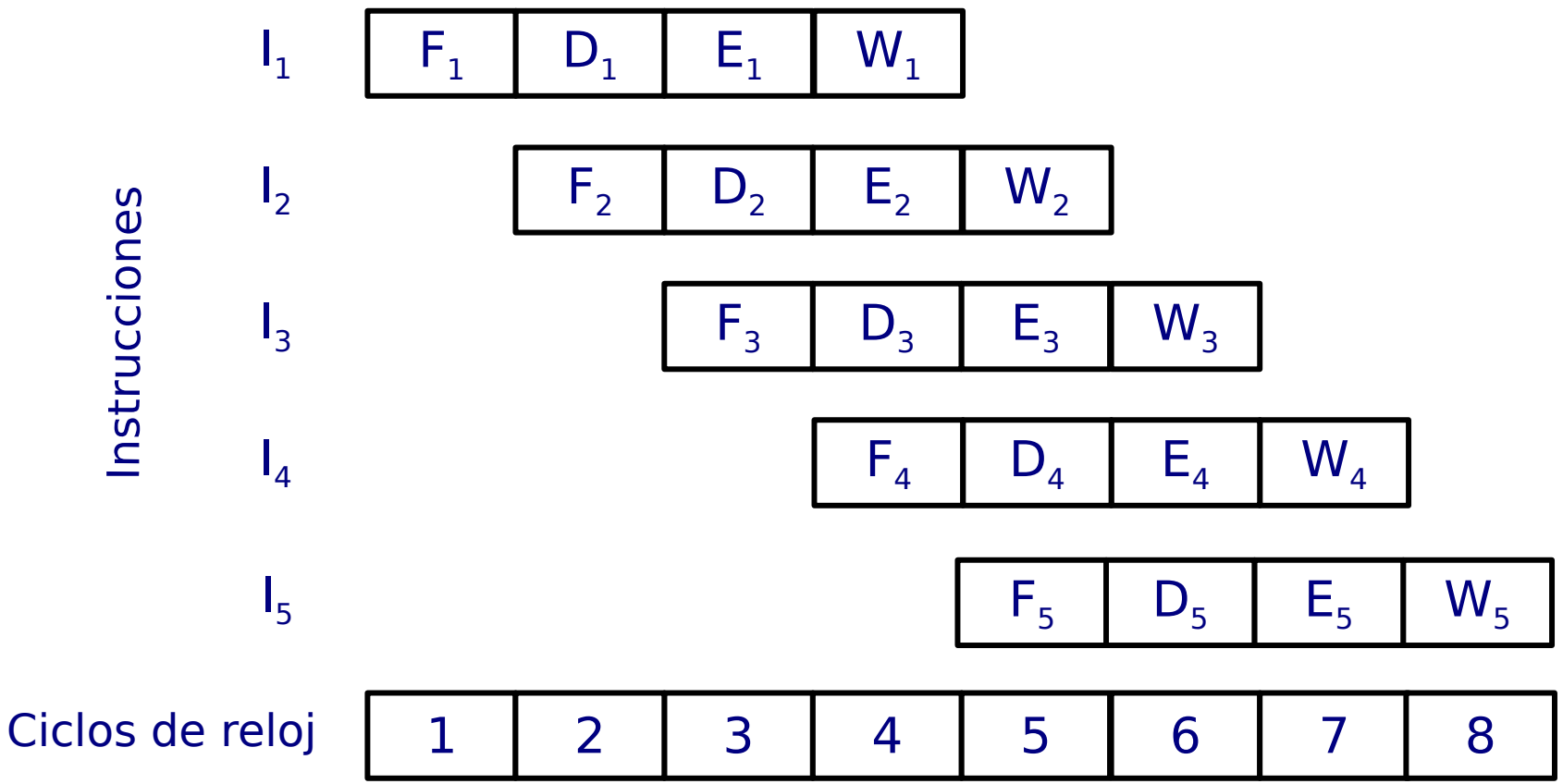
- Es una forma efectiva de organizar el ***hardware*** del computador para realizar más de una operación a la vez.
  - Es invisible al programador
- Consiste en descomponer el proceso de ejecución de las instrucciones en fases o etapas que permitan una ejecución simultánea de varias instrucciones.
- Cuando una instrucción ha completado una etapa, pasa a la siguiente etapa de modo que una nueva instrucción puede ocupar el sitio dejado por ésta (segmentación).
  - Ejemplo de similitud: cadena de montaje.


F1 D1 E2 F2 D2 E2 F3 D3 E3 ...
I1
I2
I3
Sin segmentación
Cauce segmentado en 3 fases:
Ciclos de reloj
1 2 3 4 5 6 7 8 9
Instrucciones
I1 F1 D1 E1
I2 F2 D2 E2
I3 F3 D3 E3
I4 F4 D4 E4

Características

- Necesidad de uniformizar las etapas
  - Al tiempo de la más lenta
  - Al número de etapas máximo (instrucción más larga)
- Fases de ejecución de un solo ciclo de reloj.
- Si el cauce es de profundidad 4 significa que:
  - Una instrucción tarda 4 ciclos en completarse.
  - Se ha dividido la ejecución de una instrucción en 4 etapas.
  - Las etapas o fases son sucesivas y son mono-ciclo.
  - Puede haber hasta 4 instrucciones distintas simultáneamente en ejecución pero en **diferente fase** de ejecución

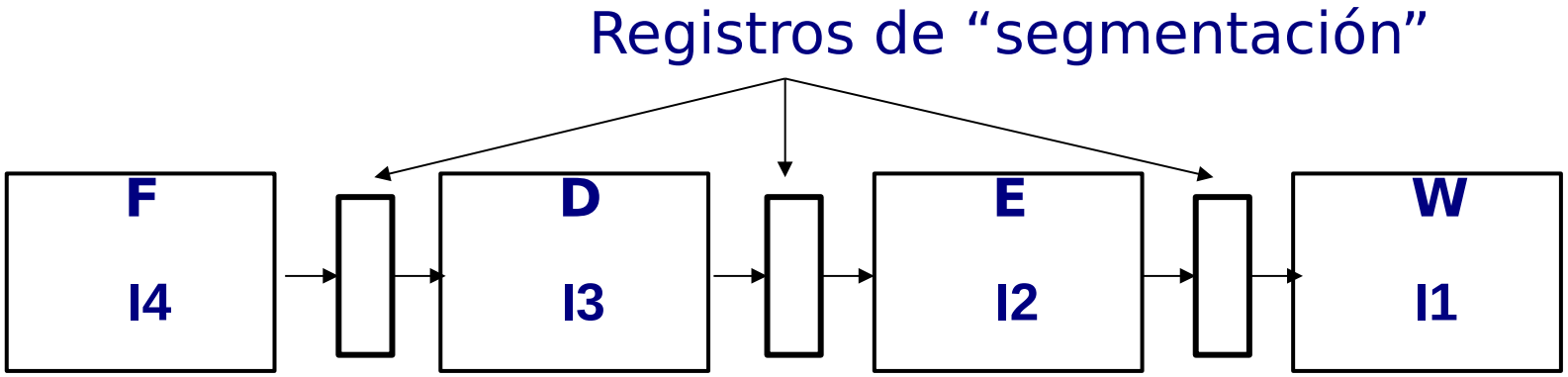
Segmentación de cauce de profundidad 4



Etapas del cauce seg. profundidad 4

- Búsqueda (F) Fetch o Captación
  - Acceso a memoria cache (búsqueda de la instrucción)
  - Incremento del CP
- Decodificación (D)
  - Decodificación de la instrucción
  - Obtención de operandos
- Ejecución (E)
  - Procesamiento: ejecución en la ALU
  - Carga/Almacenamiento: Acceso a memoria (R/W)
  - Salto: cálculo del destino y decisión de salto (si/no)
- Actualización (W)
  - Almacenamiento del resultado (si lo hay)

Organización del Hardware



Los registros de segmentación contienen los datos de la instrucción que se encuentra en cada etapa

Ejemplo: **Segmentación de un procesador lineal**

- Sea una máquina serie cuya ejecución de instrucciones tiene 5 fases que pueden segmentarse, convirtiéndose entonces en una máquina segmentada de 5 etapas.
- La duración de cada etapa es de 50ns., excepto la de la tercera que tarda 60ns.
- Suponer que el tiempo que es necesario añadir para implementar la segmentación (registros de segmentación etc.) es de 5 ns. en cada etapa.
- Calcular la Ganancia obtenida en la ejecución de instrucciones.

$$G = 260 / 65 = 4$$

Ejemplo: **Segmentación de un procesador lineal**

- Sea una máquina serie cuya ejecución de instrucciones tiene 5 fases que pueden segmentarse, convirtiéndose entonces en una máquina segmentada de 5 etapas.
- La duración de cada etapa es de 50ns., excepto la de la tercera que tarda 60ns.
- Suponer que el tiempo que es necesario añadir para implementar la segmentación (registros de segmentación etc.) es de 5 ns. en cada etapa.
- Calcular la Ganancia obtenida en la ejecución de instrucciones.

$$G = 260 / 65 = 4$$

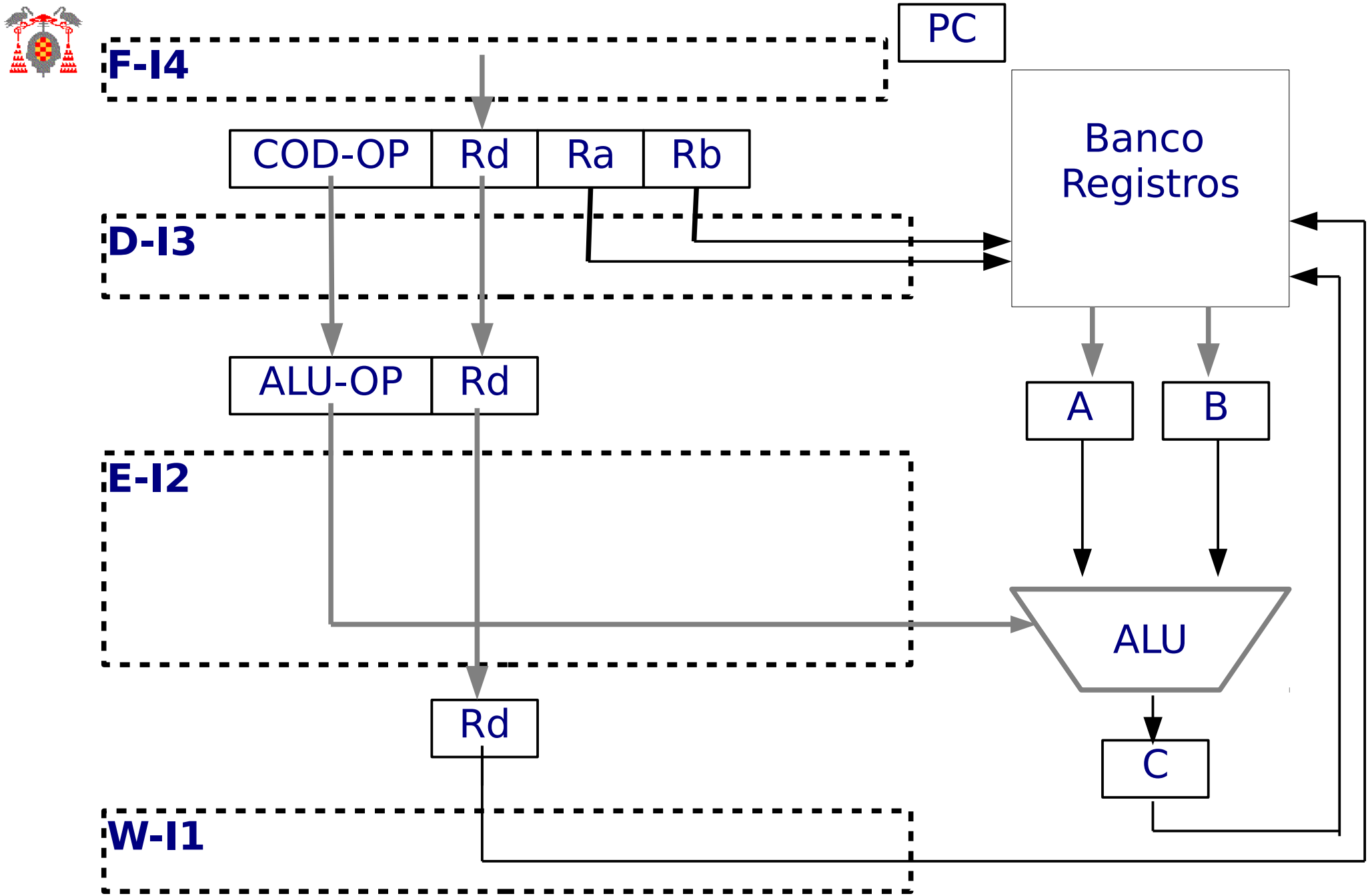
Etapa más lenta
Tiempo extra para cargar Registros de Segmentación
--> Reloj más lento
T = I *CPI*/f

65	65	65	65	65	65	65	
50+5	50+5	60+5	50+5	50+5			Instrucción i
	50+5	50+5	60+5	50+5	50+5		Instrucción i+1
		50+5	50+5	60+5	50+5	50+5	Instrucción i+2

Arquitectura e Ingeniería de Computadores

SEGMENTACIÓN DE CAUCE

Funcionamiento


PC
F-I4
COD-OP
Rd
Ra
Rb
Banco
Registros
D-I3
ALU-OP
Rd
A
B
E-I2
ALU
Rd
C
W-I1

Ejemplo

Supongamos que R1=1, R2=2, R3=3...

Veamos cómo se ejecuta en el cauce

I1: ADD R1,R2,R3 (R1<--R2+R3)

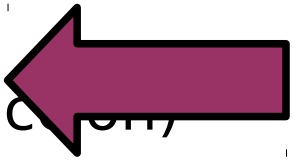
I2: sub r5,r0,r9

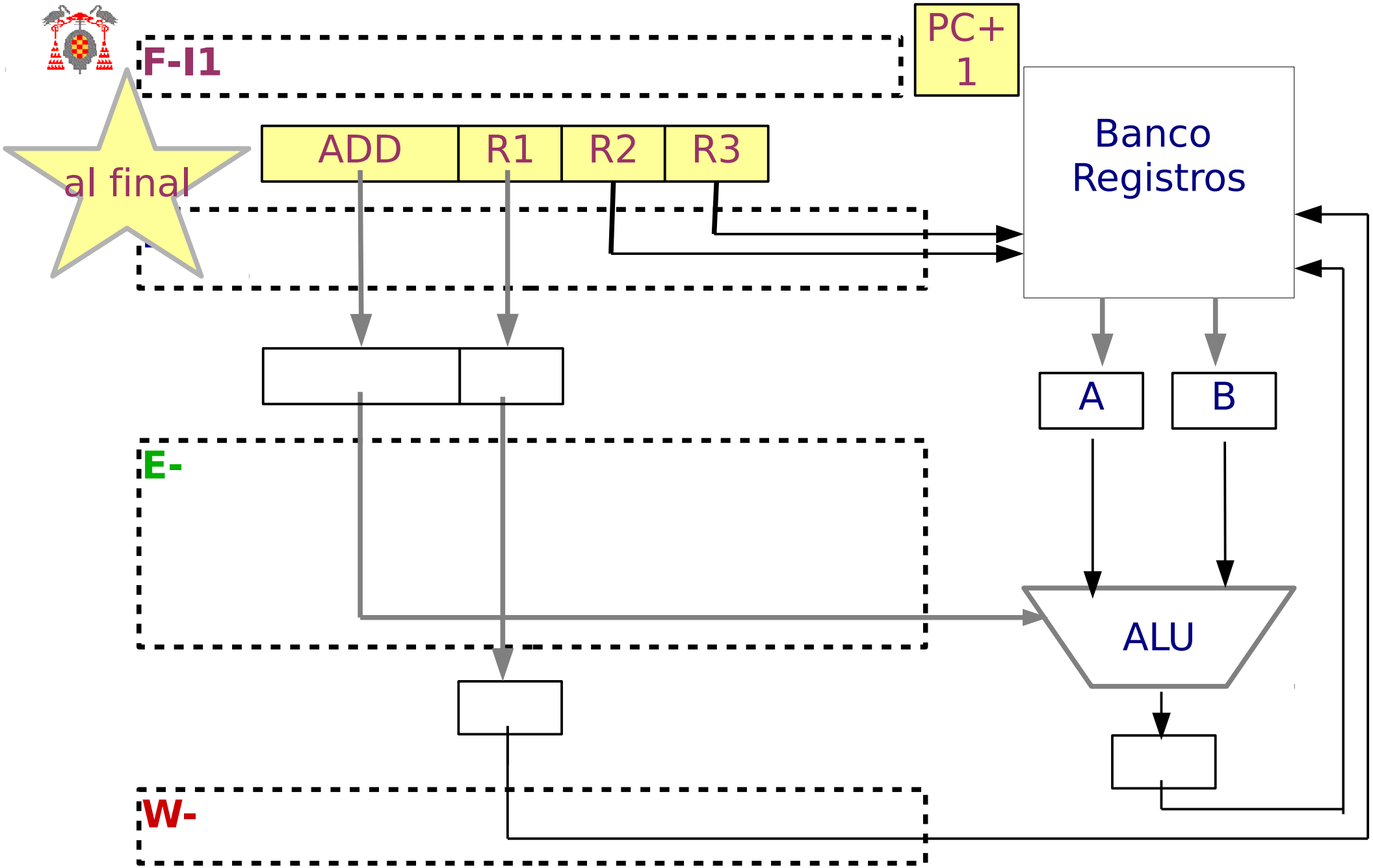
I3: and r2,r3,r4

I4: ...

Etapas del cauce seg. profundidad 4

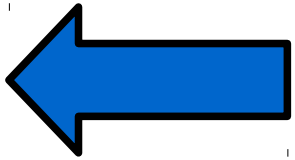
- Búsqueda (F) Fetch o Captación
  - Acceso a memoria cache (búsqueda de la instrucción)
  - Incremento del CP
- Decodificación (D)
  - Decodificación de la instrucción
  - Obtención de operandos
- Ejecución (E)
  - Procesamiento: ejecución en la ALU
  - Carga/Almacenamiento: Acceso a memoria (R/W)
  - Salto: cálculo del destino y decisión de salto (si/no)
- Actualización (W)
  - Almacenamiento del resultado (si lo hay)

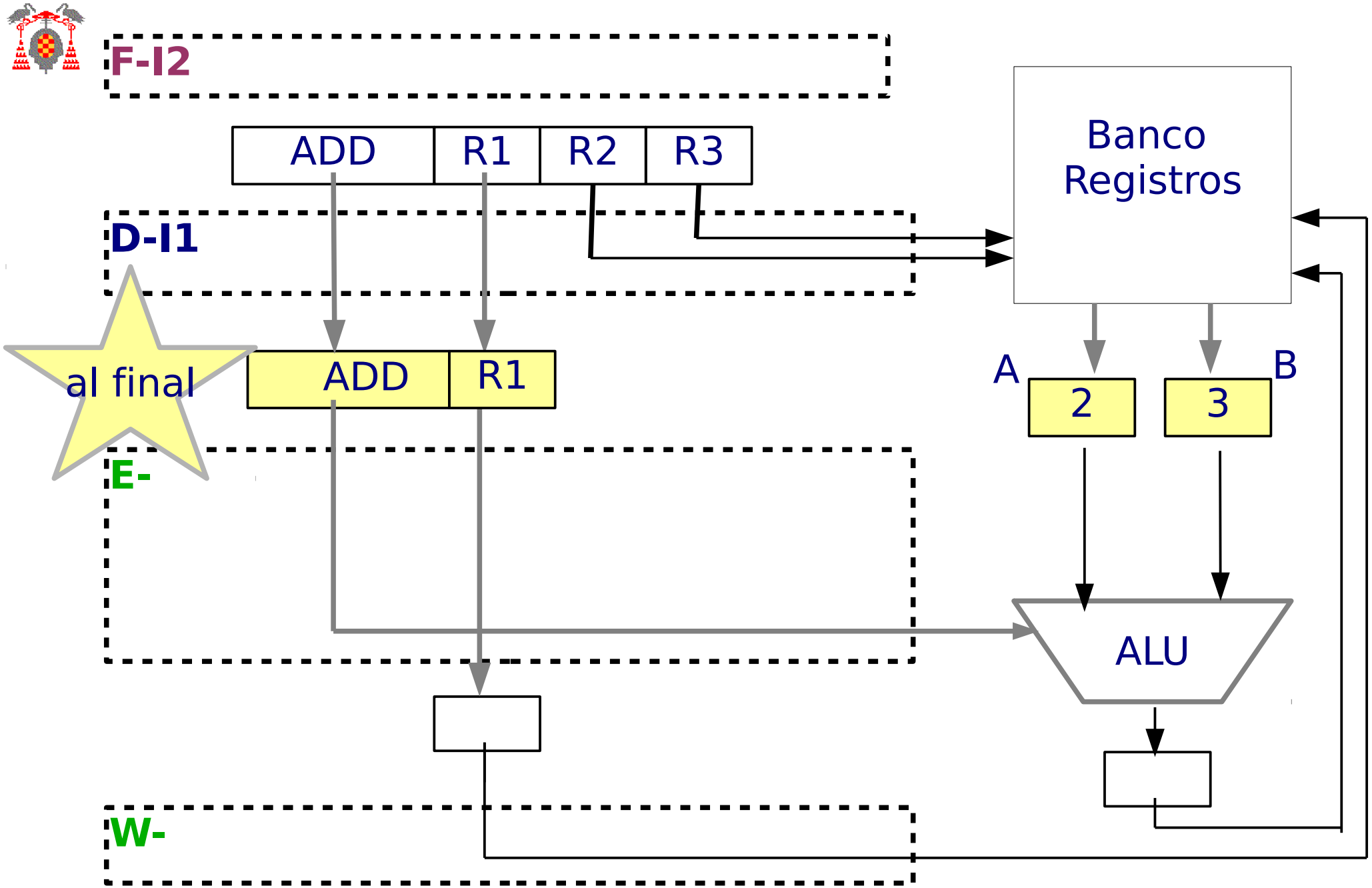



F-I1
PC+
1
al final
ADD
R1
R2
R3
Banco
Registros
A
B
E-
ALU
W-

Etapas del cauce seg. profundidad 4

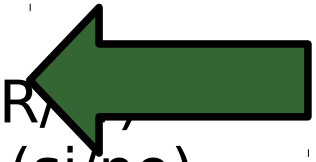
- Búsqueda (F) Fetch o Captación
  - Acceso a memoria cache (búsqueda de la instrucción)
  - Incremento del CP
- Decodificación (D)
  - Decodificación de la instrucción
  - Obtención de operandos
- Ejecución (E)
  - Procesamiento: ejecución en la ALU
  - Carga/Almacenamiento: Acceso a memoria (R/W)
  - Salto: cálculo del destino y decisión de salto (si/no)
- Actualización (W)
  - Almacenamiento del resultado (si lo hay)

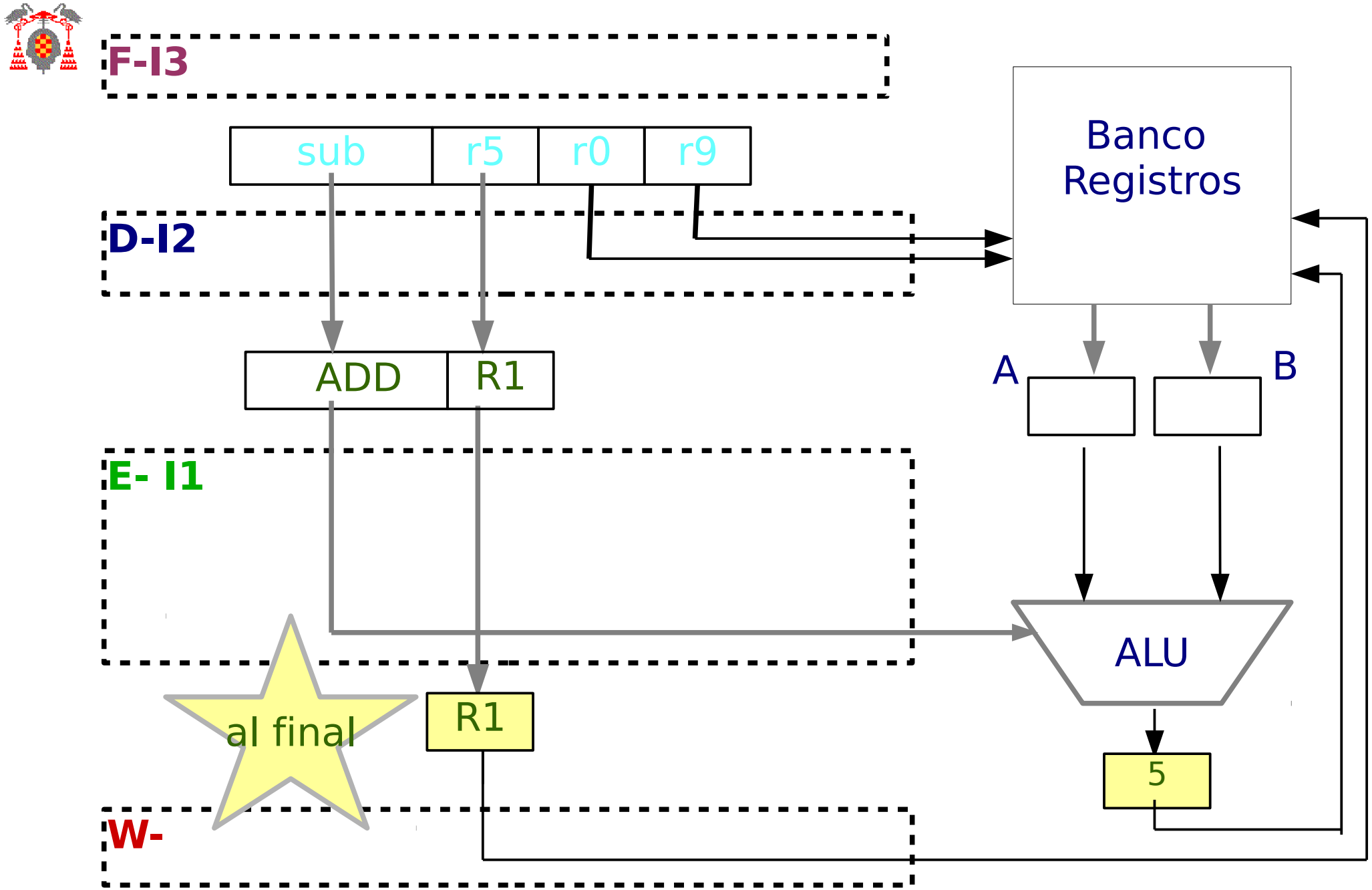



F-I2
ADD
R1
R2
R3
Banco Registros
D-I1
al final
ADD
R1
A
B
2
3
E-
ALU
W-

Etapas del cauce seg. profundidad 4

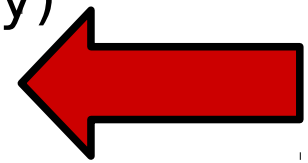
- Búsqueda (F) Fetch o Captación
  - Acceso a memoria cache (búsqueda de la instrucción)
  - Incremento del CP
- Decodificación (D)
  - Decodificación de la instrucción
  - Obtención de operandos
- Ejecución (E)
  - Procesamiento: ejecución en la ALU
  - Carga/Almacenamiento: Acceso a memoria (R/W)
  - Salto: cálculo del destino y decisión de salto (si/no)
- Actualización (W)
  - Almacenamiento del resultado (si lo hay)

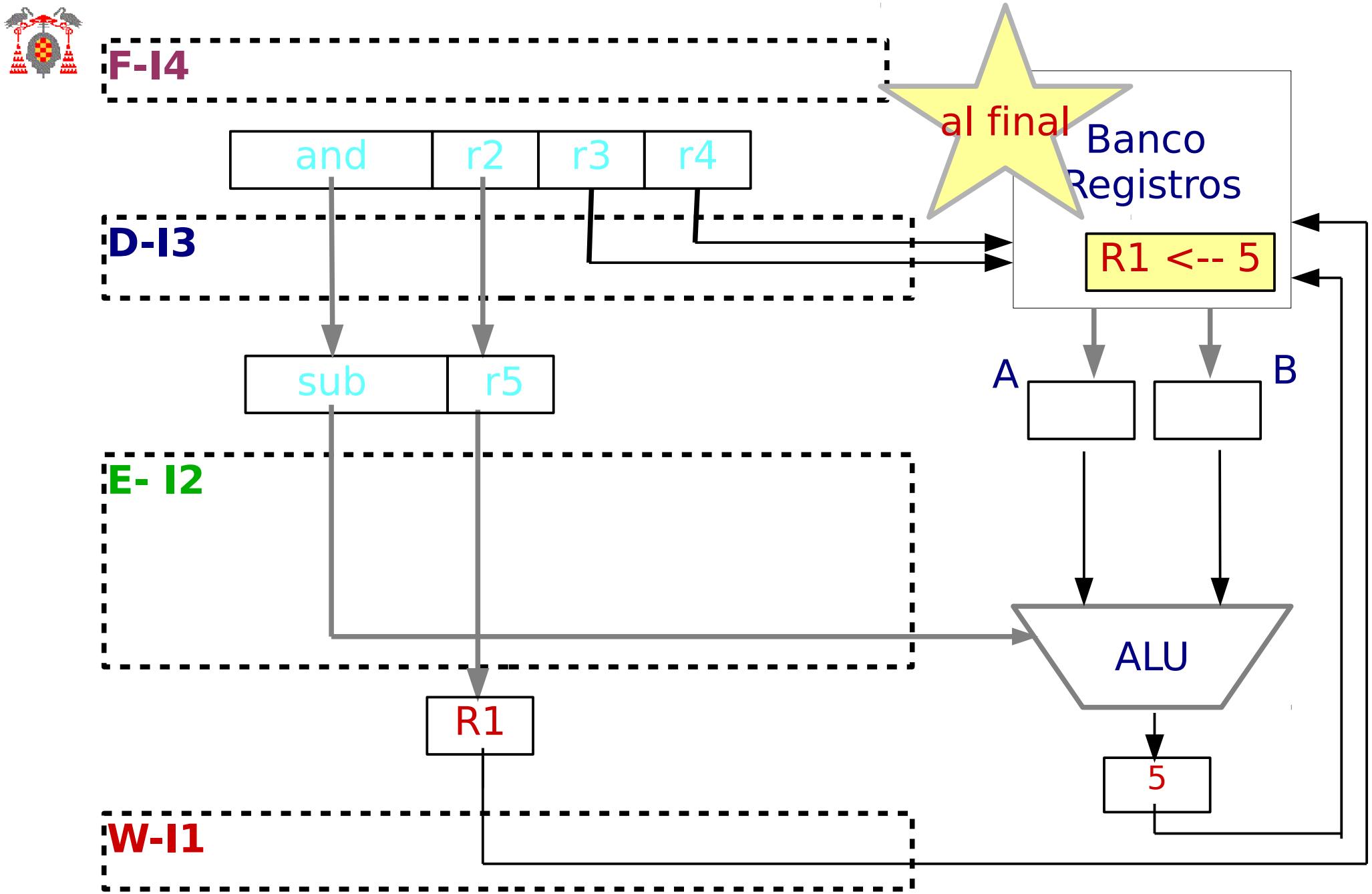


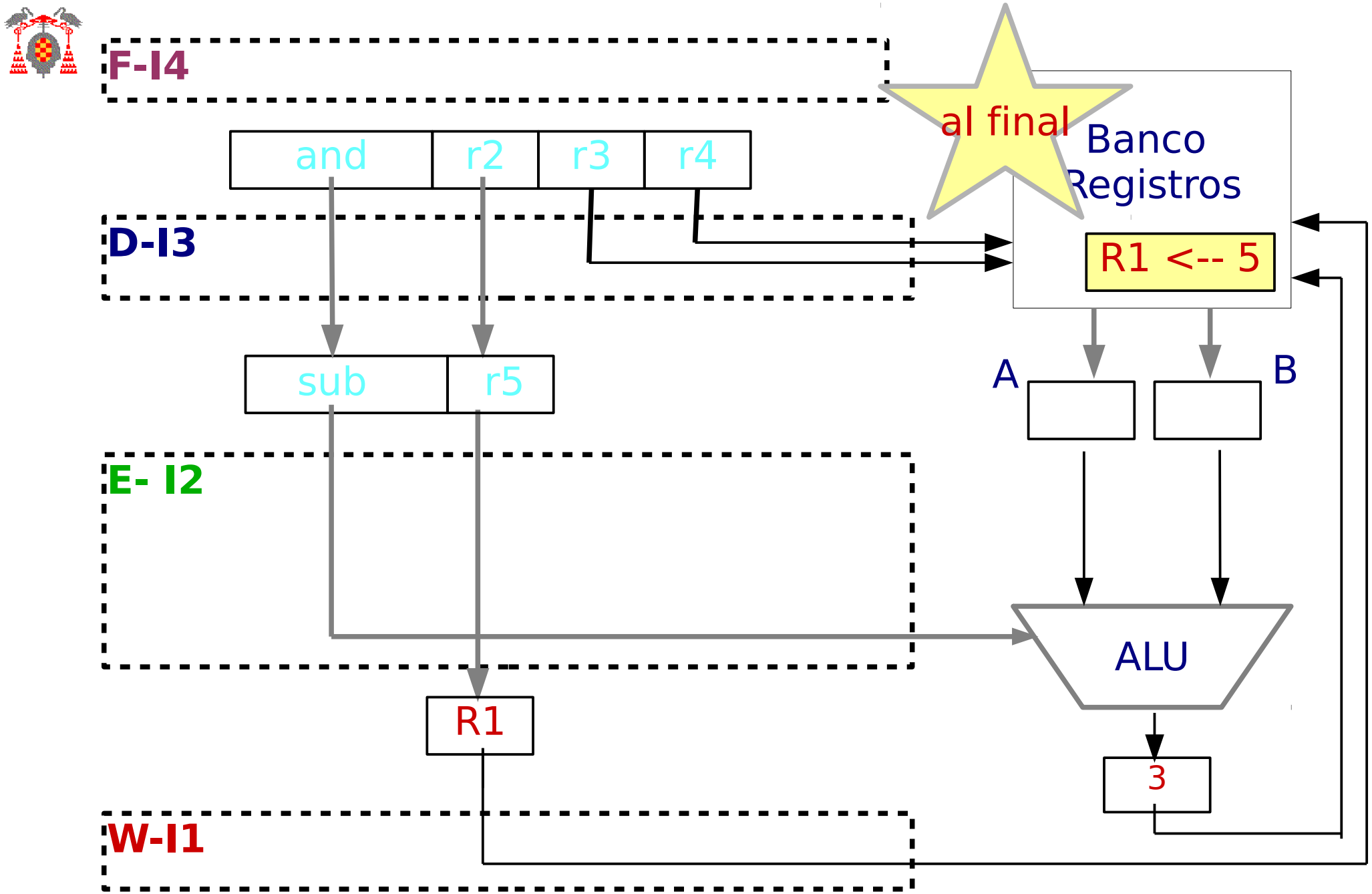

F-I3
sub
r5
r0
r9
Banco
Registros
D-I2
ADD
R1
A
B
E- I1
ALU
al final
R1
5
W-

Etapas del cauce seg. profundidad 4

- Búsqueda (F) Fetch o Captación
  - Acceso a memoria cache (búsqueda de la instrucción)
  - Incremento del CP
- Decodificación (D)
  - Decodificación de la instrucción
  - Obtención de operandos
- Ejecución (E)
  - Procesamiento: ejecución en la ALU
  - Carga/Almacenamiento: Acceso a memoria (R/W)
  - Salto: cálculo del destino y decisión de salto (si/no)
- Actualización (W)
  - Almacenamiento del resultado (si lo hay)




F-I4
and
r2
r3
r4
D-I3
sub
r5
E- I2
R1
W-I1
al final
Banco
Registros
R1 <-- 5
A
B
ALU
5


F-I4
and
r2
r3
r4
al final
Banco
Registros
D-I3
R1 <-- 5
A
B
sub
r5
E- I2
ALU
R1
3
W-I1

Rendimiento

- Todas las unidades funcionales del computador están activas a la vez y se mejora el rendimiento.
- El máximo rendimiento teórico es completar una instrucción cada ciclo de reloj
  - Máximo rendimiento teórico 1 CPI
- Si N es el nº de etapas del cauce, idealmente:
  - Velocidad proc. segmentado = Velocidad secuencial x N
  - ACELERACIÓN = ciclos no-segm / ciclos segm ~ N (para la ejecución de las mismas instrucciones)
- En la práctica, existen RIESGOS de que esto no sea así

Riesgos en Segmentación:

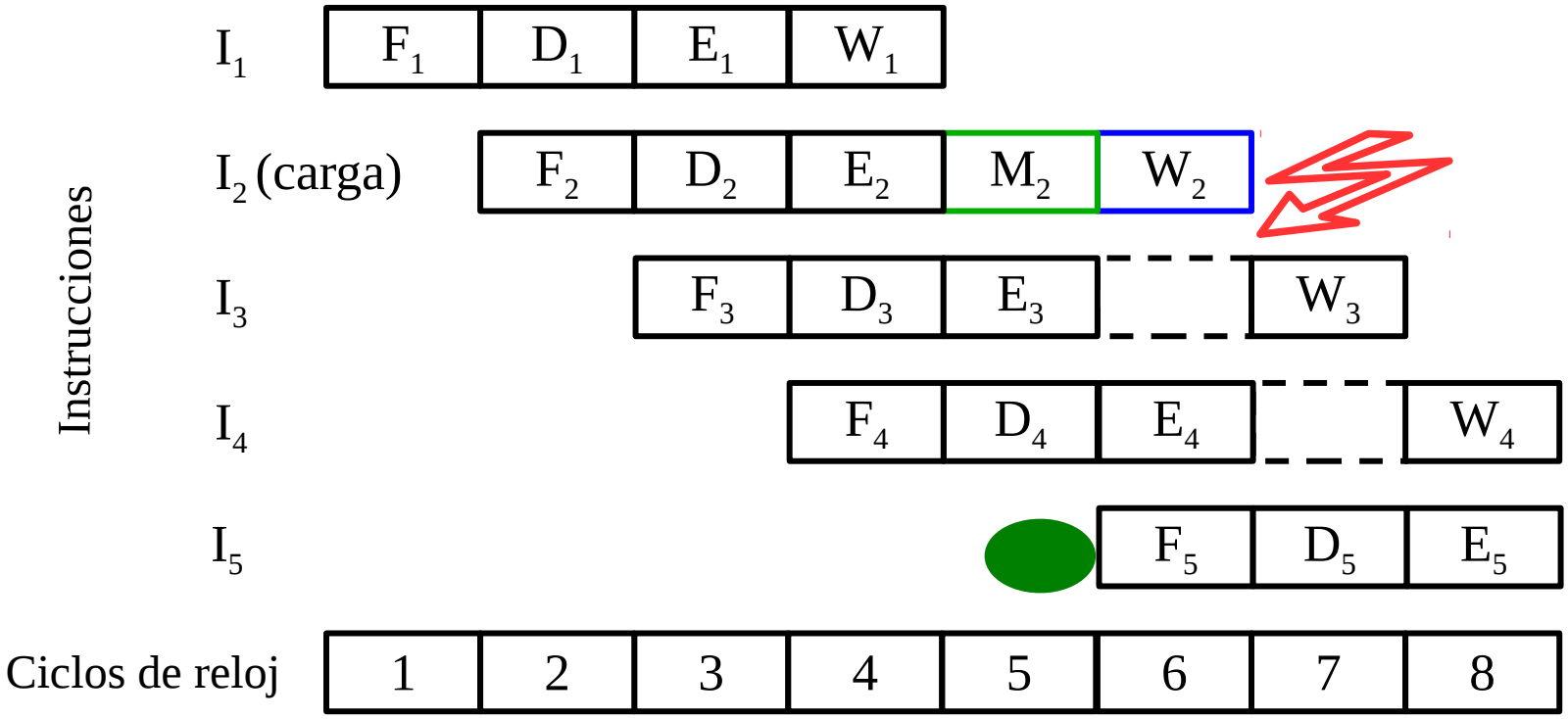
- El diseño de procesadores segmentados tiene gran dependencia del repertorio de instrucciones (RISC / CISC)
- Riesgos de Datos
  - Debidos a dependencias que existen entre los datos que manipulan las instrucciones
- Riesgos de Instrucciones o de Control
  - Debidos a cambios en el flujo de los programas
- Riesgos Estructurales
  - Debidos a conflictos de utilización de los elementos de la estructura de la máquina

Riesgos Estructurales

Dos instrucciones necesitan utilizar el mismo recurso

I_2 e I_5 acceden a memoria en el ciclo 5 (sin caches separadas)

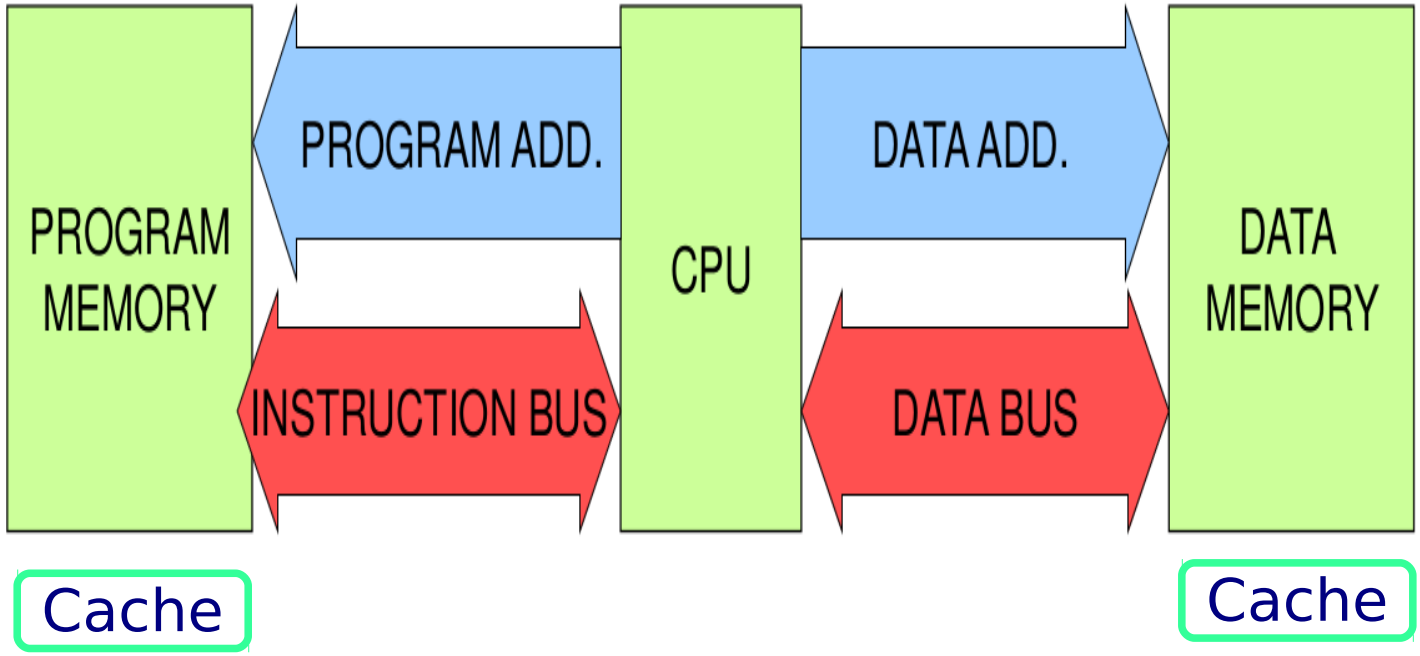
I_2 e I_3 acceden al Banco de Registros en ciclo 6 (en cauces no homogéneos)

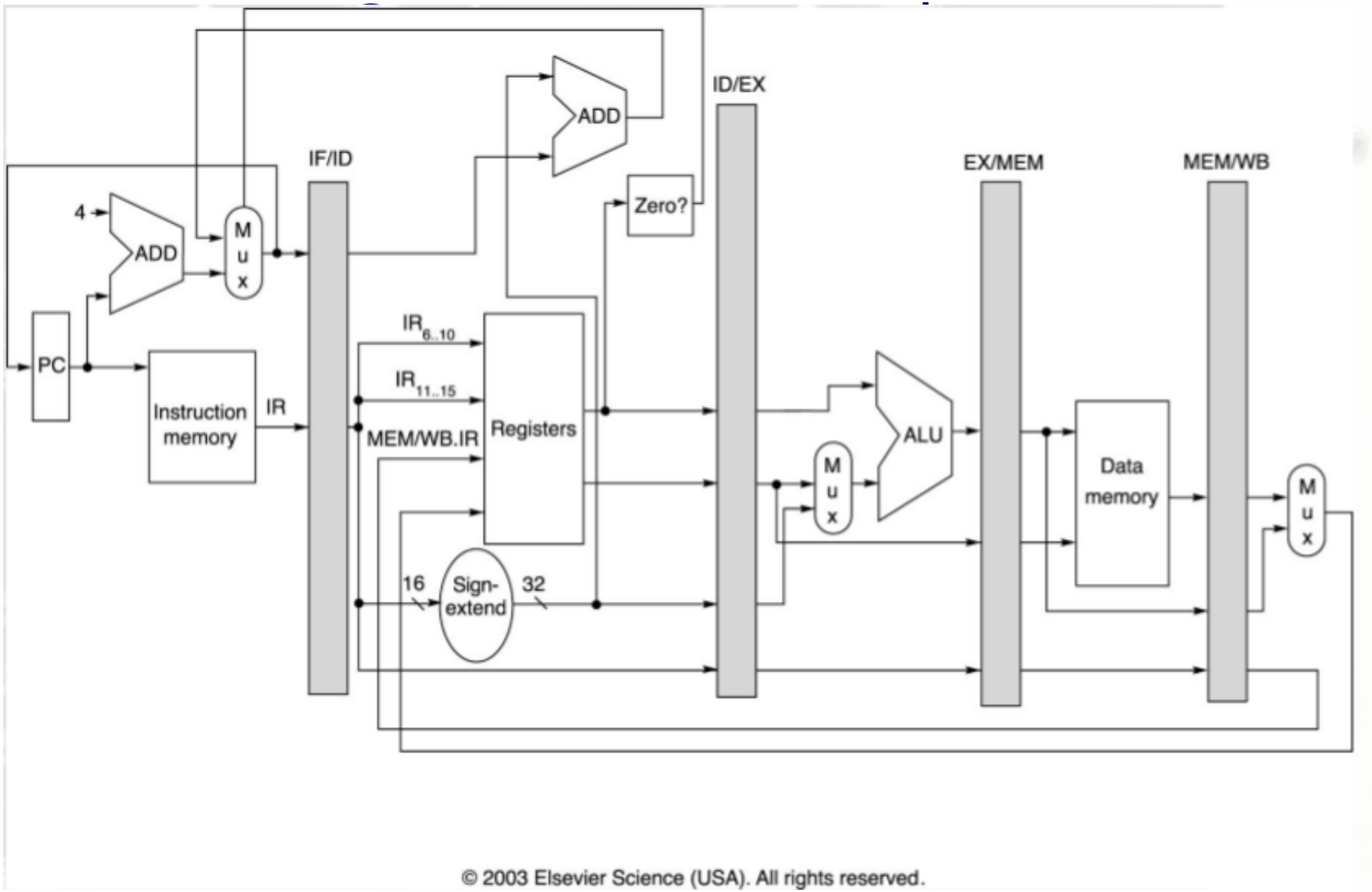


Conclusión: Diseño estructura

- Cachés diferenciadas de Datos e Instrucciones
  - Obvio...
- Más de un puerto de acceso al banco de registros
  - para obtener los dos operandos de una inst. proceso
- Cauce homogéneo
  - Todas las instrucciones pasen las mismas fases para evitar coincidir en el uso de recursos
  - Aún a costa de que en algunos casos, para una instrucción no se haga nada en una fase
  - Depende del Repertorio también...

Arquitectura Harvard





Influencia Repertorio: Modos Dir.

- Modos de direccionamiento ... a memoria, claro
  - Número de ciclos (accesos a memoria y cálculos)
  - Efectos colaterales (auto-incremento/decremento)
- Direccionamientos complejos requieren más ciclos para obtener la dirección efectiva o para completar la instrucción.
  - lw R6, [n(R1)],R2 R6 <-- M(M(R1+n)) 2 accesos a memoria
  - lw R3, (R2)R1++ R5 <-- M(R2+R1) ; R1<--R1+4 post-incremento, un nuevo uso de la ALU
  - Propio de CISC
- Direccionamientos sencillos requieren más instrucciones
  - Pero se pueden ejecutar en un cauce homogéneo
  - Propio de RISC

Conclusiones Modos Dir.

- Son convenientes las siguientes características:
  - Que sólo las instrucciones de Load/Store accedan a memoria
  - Obtención del operando con un sólo acceso a memoria
  - Modos de direccionamiento sin efectos colaterales
- Direccionamientos “recomendados” (RISC)
  - Direccionamiento a registro. No necesita cálculo (en CPU)
  - Direccionamiento Inmediato. En la propia instrucción (en CPU)
  - Direccionamiento indirecto con registro. No necesita cálculo.
  - Direccionamiento indexado (o “relativo a Registro”)
    - Cálculo de la indexación 1 ciclo: suma de R+R o R+Inm.

Códigos de Condición

- Modificados por muchas instrucciones
- Consultados para saltos condicionales
- Fuerzan a que el salto se ubique inmediatamente después de la instrucción en cuestión
  - Ej: contador de un bucle o la operación de un “if”
  - ¡¡Codigo “acoplado”!!
- Reducen la flexibilidad del compilador para “reordenar” las instrucciones y aprovechar mejor el flujo en el cauce sin paradas como veremos en breve
  - Opción 1: el compilador indica cuándo han de afectarse (Alpha)
  - Opción 2: la condición es el valor de un Registro GPR (DLX)