

Práctica 2.
Determinación de la
microarquitectura de la
memoria cache a partir
de la evaluación de
prestaciones de un
computador real

Arquitectura de Computadores
2º Grado Ingeniería Informática



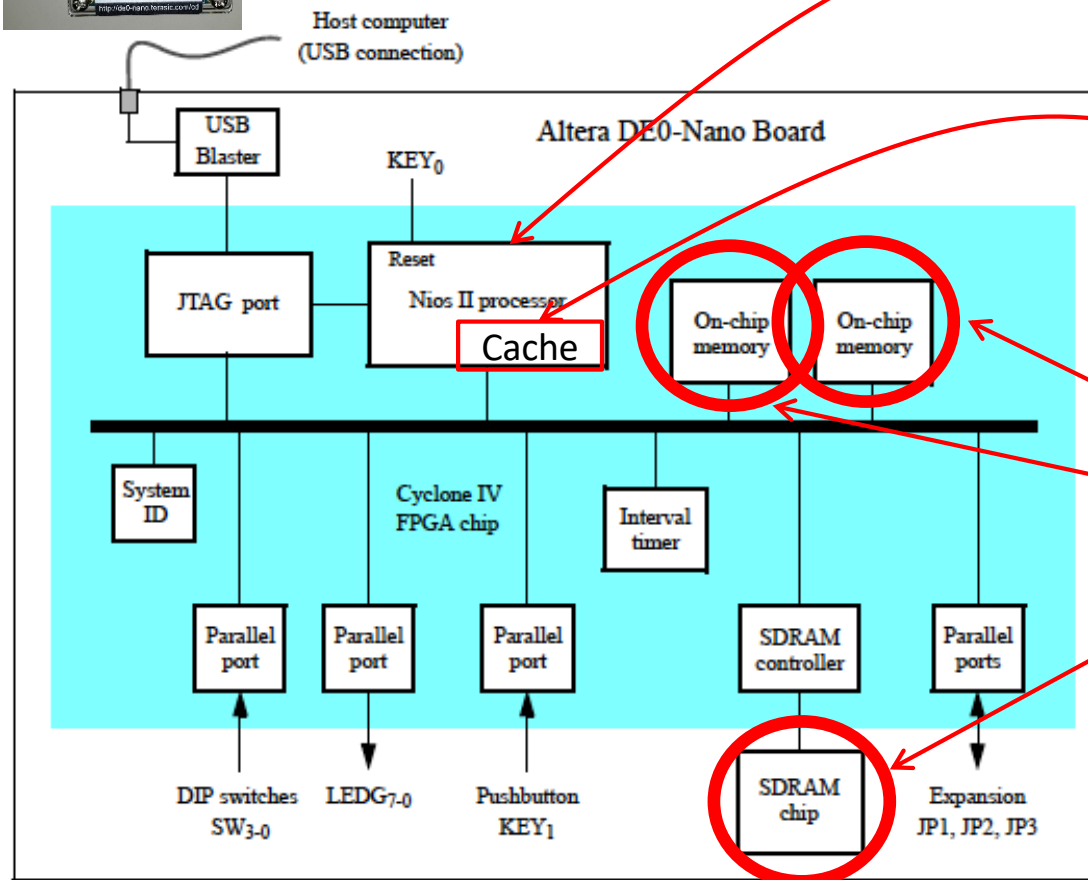
Programación académica: 4 Semanas

- S1: Actividades 1,2,3; 2 horas
- S2: Actividad 4 ; 2 horas
- S3: Descubrimiento de una configuración prueba; 2 horas
- S4: Examen; 1,5 horas

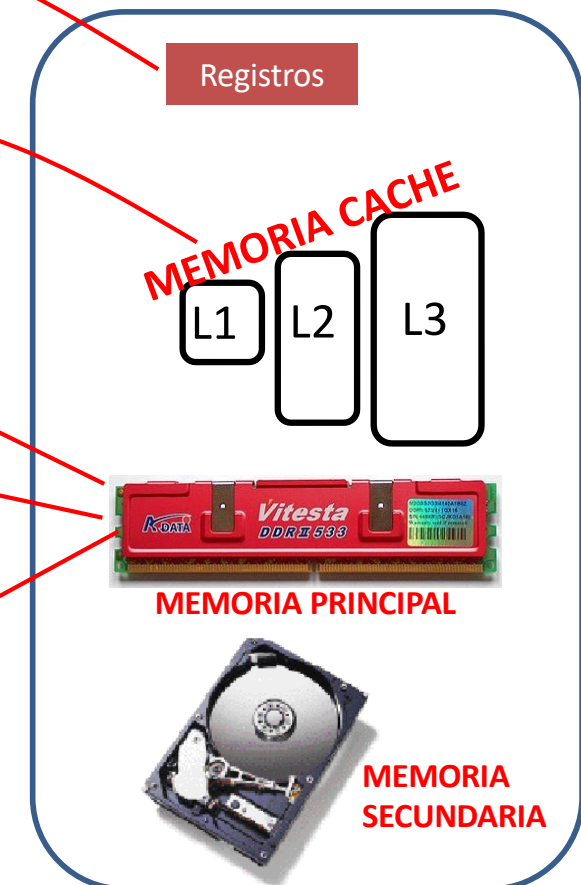
Implementación de los niveles de la Jerarquía de Memoria de la estructura del Computador Básico de la placa DE0-Nano



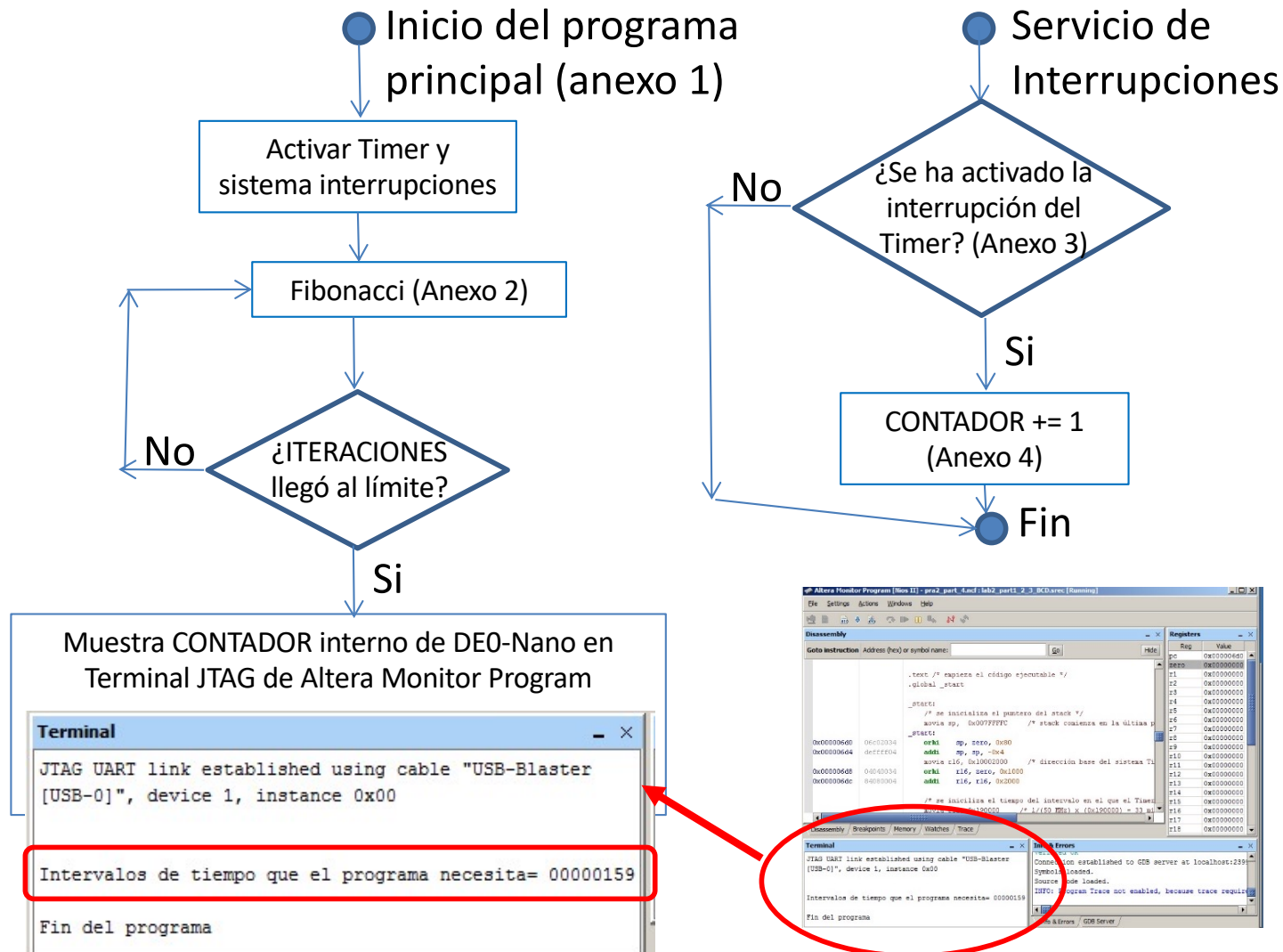
DE0-Nano



Implementación de los niveles de la jerarquía de memoria

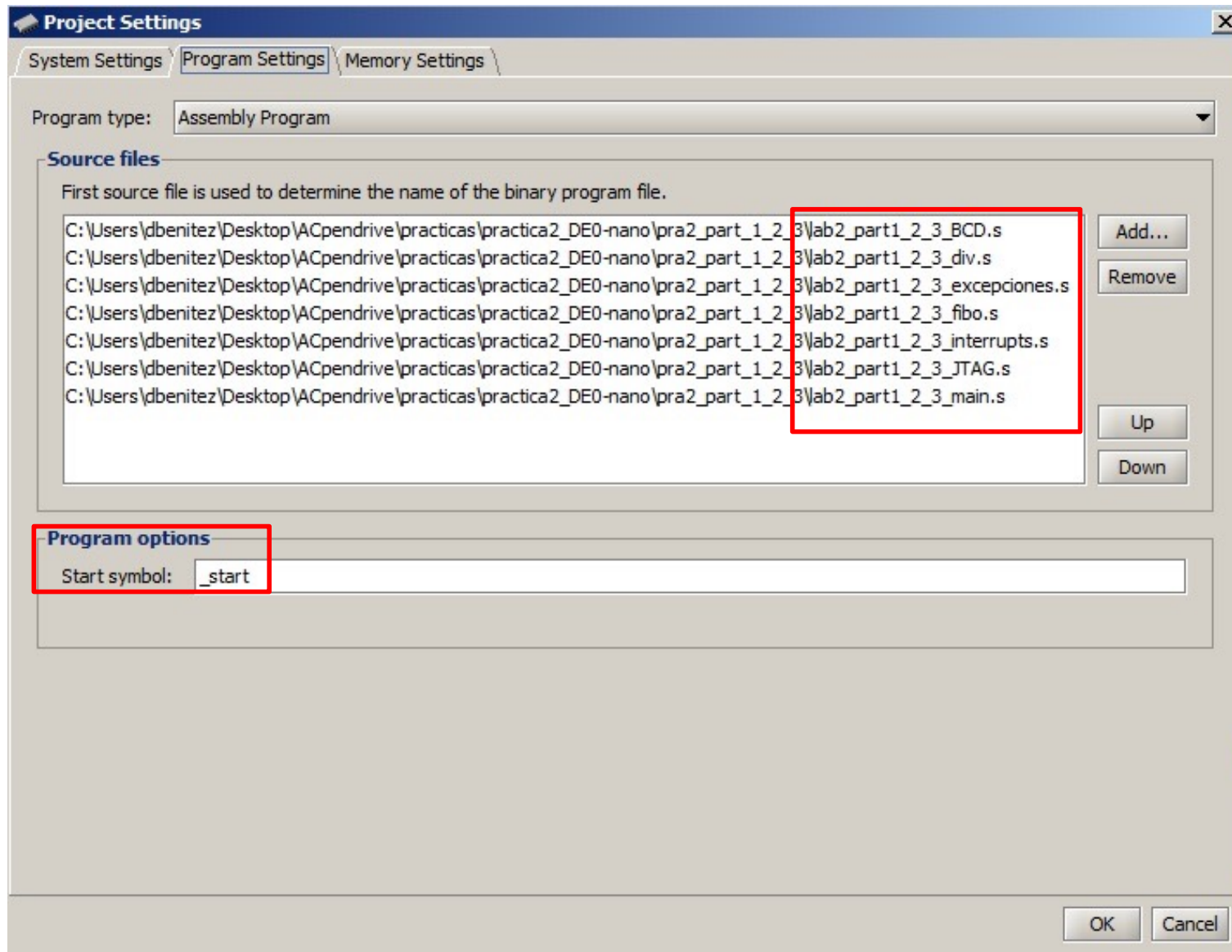


Actividad 1: diagrama de flujo del programa benchmark



Altera
Monitor
Program

Actividad 1: configuración del proyecto AMP



Actividad 1: configuración del proyecto AMP

Project Settings

System Settings | Program Settings | **Memory Settings**

Processor's reset and exception vectors (read-only)

Reset vector address: 0x00000000

Exception vector address: 0x00000020

Memory options

Here you can specify the starting addresses of sections identified by .text and .data assembler directives. These addresses can be in the same or in different memories (on-chip, SDRAM, ...). They can be used to ensure that the .text and .data sections do not overlap with other sections, such as .reset and .exceptions. If .text and .data are specified to have the same address, the .data section will be placed right after the .text section by the linker.

.text section

Memory device: SDRAM/s1 (0x0 - 0x1ffffff)

Start offset in device (hex): 400

.data section

Memory device: SDRAM/s1 (0x0 - 0x1ffffff)

Start offset in device (hex): 400

OK Cancel

Actividad 1: Observación del tiempo de ejecución

Versión de procesador + tecnología memoria	Tiempo de ejecución	Speed-up
Nios II/e + memoria SDRAM (Actividad 1)		1X
Nios II/e + memoria on-chip (Actividad 2)		
Nios II/f + memoria SDRAM (Actividad 3)		
Nios II/f + memoria on-chip (Actividad 3)		

Pregunta 1

- Cambiar ITERACIONES de 500.000 a 100.000 en el fichero lab2_part1_2_3_main.s, compilar de nuevo y cargar el programa en la placa. A continuación, ejecutar el programa y apuntar el correspondiente valor que se muestra por el terminal de AMP.
- ¿Este nuevo resultado de prestaciones temporales es razonable?
- Razónalo y justifica la respuesta.

Actividad 2: Acceso a la memoria “on-chip” con el procesador Nios II/e

configuración del proyecto AMP

The screenshot shows the 'Project Settings' dialog box with the 'Memory Settings' tab selected. The 'Processor's reset and exception vectors (read-only)' section shows the reset vector address as 0x00000000 and the exception vector address as 0x00000020. The 'Memory options' section contains a text box explaining that users can specify starting addresses for .text and .data sections. Below this, the '.text section' and '.data section' are configured. For the '.text section', the 'Memory device' is 'Onchip_memory/s1 (0x9000000 - 0x9001fff)' and the 'Start offset in device (hex)' is 400. For the '.data section', the 'Memory device' is 'Onchip_memory/s2 (0x9000000 - 0x9001fff)' and the 'Start offset in device (hex)' is 400. Red and blue boxes highlight these configuration fields.

Project Settings

System Settings | Program Settings | **Memory Settings**

Processor's reset and exception vectors (read-only)

Reset vector address: 0x00000000

Exception vector address: 0x00000020

Memory options

Here you can specify the starting addresses of sections identified by .text and .data assembler directives. These addresses can be in the same or in different memories (on-chip, SDRAM, ...). They can be used to ensure that the .text and .data sections do not overlap with other sections, such as .reset and .exceptions. If .text and .data are specified to have the same address, the .data section will be placed right after the .text section by the linker.

.text section

Memory device: Onchip_memory/s1 (0x9000000 - 0x9001fff)

Start offset in device (hex): 400

.data section

Memory device: Onchip_memory/s2 (0x9000000 - 0x9001fff)

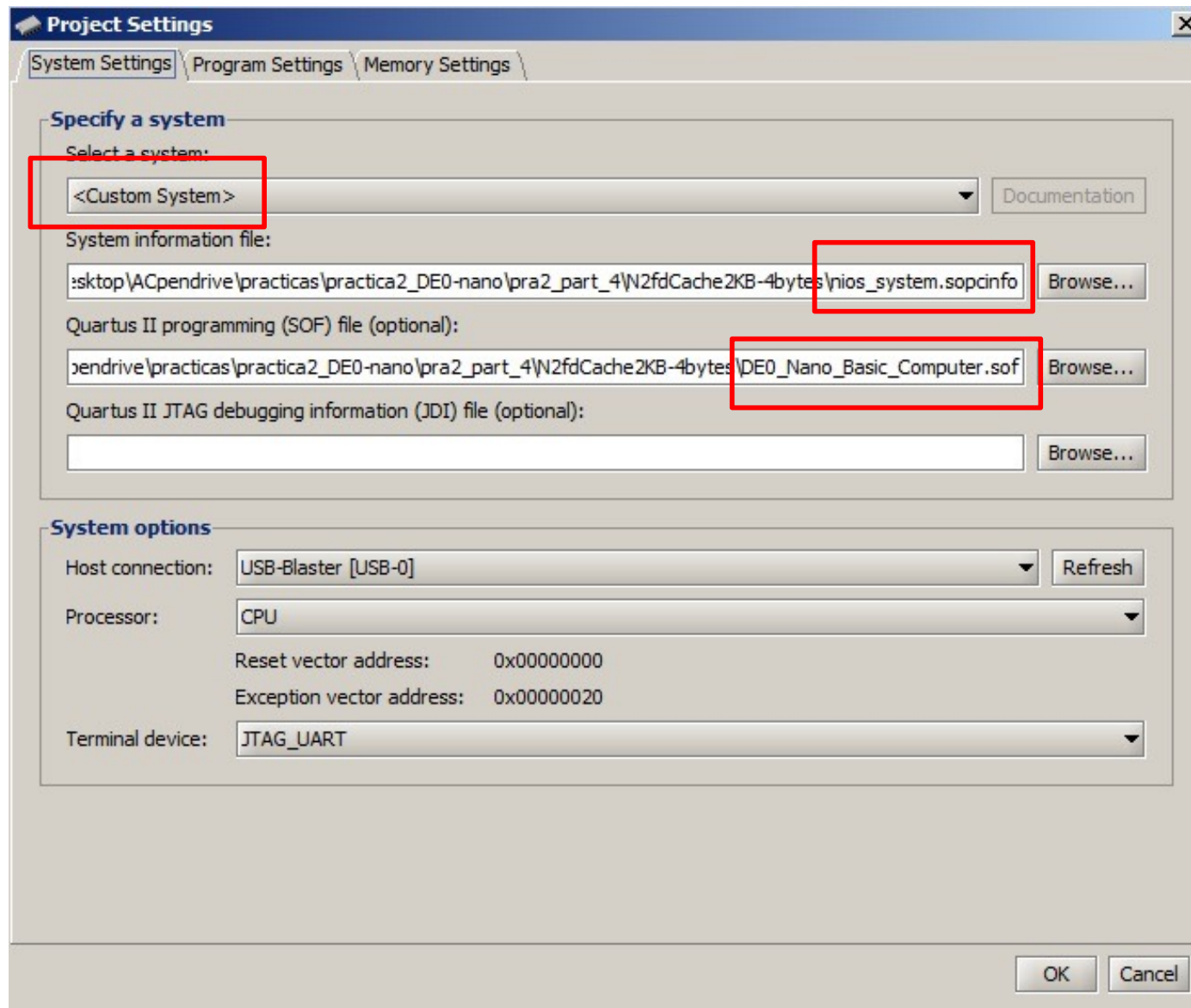
Start offset in device (hex): 400

OK Cancel

Pregunta 2

- Cambiar al dispositivo de memoria SRAM (denominada “on-chip”), la cual se encuentra dentro del chip.
- ¿Este nuevo resultado de prestaciones temporales es razonable?
- Razónalo y justifica la respuesta.

Actividad 3: Configuración de Nios II / f



Preguntas de la Actividad 3

Pregunta 3

- ¿Cuál es la razón por la que las distintas versiones del procesador Nios II/e proporcionan tal variación de prestaciones?

Pregunta 4

- ¿Cuál es la razón por la que las dos versiones del procesador Nios II/f coinciden en tiempo de ejecución?

Pregunta 5

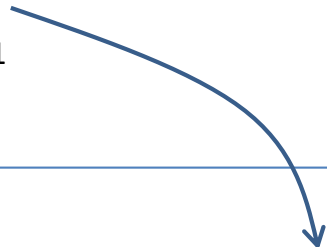
- ¿Cuál es la razón por la que la versión Nios II/e proporciona peores prestaciones que Nios II/f?

Actividad 4: Descubrimiento de la arquitectura interna de la memoria cache de datos

Código fuente del programa principal:

lab2_part1_2_3_main.s

```
...  
movia r14, ITERACIONES /* inicializa el contador de iteraciones LOOP, cada una de ellas ejecuta un bucle Fibonacci */  
addi r17, r0, 0 /* inicializa el contador de intervalos del programa "r17" */  
  
LOOP: beq r14, r0, END /* se ejecuta el bucle Fibonacci */  
      call FIBONACCI  
      addi r14, r14, -1  
      br LOOP
```



Código fuente de la subrutina FIBONACCI:

lab2_part1_2_3_fibo.s

```
...  
      movi r4, 0  
      movi r5, X  
LOOP: bge r4, r5, END  
      ldb r0, V(r4)  
      addi r4, r4, P  
      br LOOP  
END:  
  
...  
      .data  
V:  
      .skip 65536  
...
```

Modificación que se tiene que realizar en el fichero: lab2_part1_2_3_fibo.s

Actividad 4

```

...
movi r4, 0
movi r5, X
LOOP: bge r4, r5, END
      ldb r0, V(r4)
      addi r4, r4, P
      br LOOP
END:
...
.data
V:
.skip 65536
...

```

X: volumen de datos que maneja el programa:

$$X = P \times E$$

E: número de accesos a memoria del programa

Tabla 3. Tabla que se utiliza en la Actividad 4 para recoger las medidas de tiempos de ejecución.

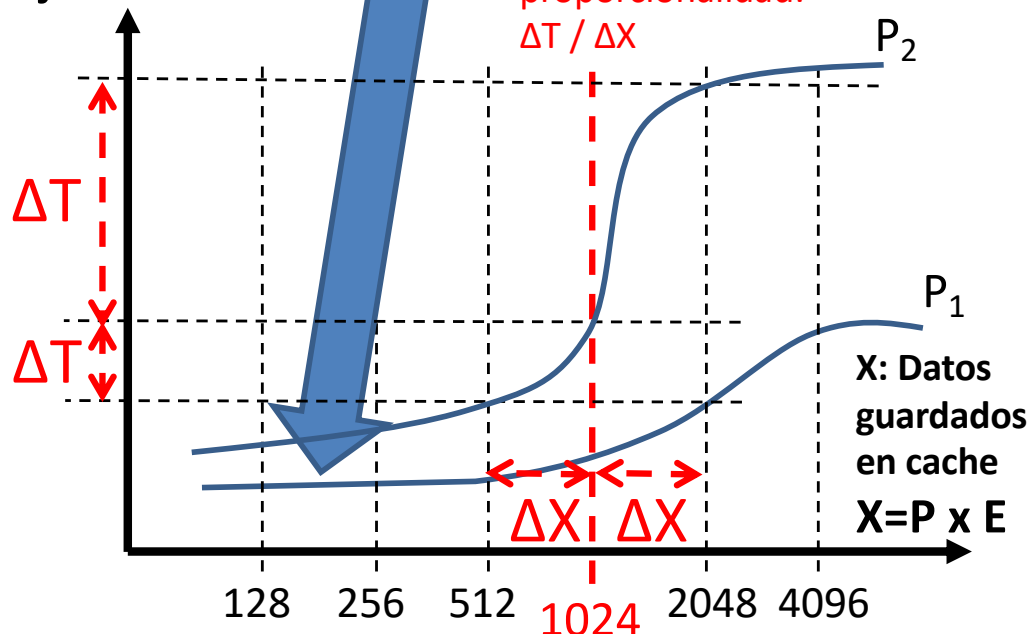
	E: Número de bytes del vector V realmente accedidos					
P: Pauta de salto	128	256	512	1024	2048	4096
P = 1: de 1 en 1						
P = 2: de 2 en 2						
P = 4: de 4 en 4						
P = 8: de 8 en 8						

Actividad 4

Tabla 3. Tabla que se utiliza en la Actividad 4 para recoger las medidas de tiempos de ejecución.

P: Pauta de salto	E: Número de bytes del vector V realmente accedidos					
	128	256	512	1024	2048	4096
P = 1: de 1 en 1						
P = 2: de 2 en 2						
P = 4: de 4 en 4						
P = 8: de 8 en 8						

T: tiempo de ejecución



Procedimiento para rellenar la Tabla 3

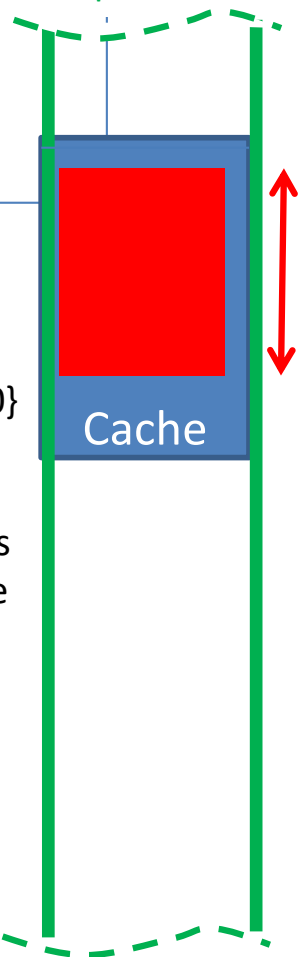
- 1) Ejecutar AMP
 - 2) New project
- CAMBIA-CACHE:**
- 3) Settings > System Settings > "Custom System" + browse > <file>.sopcinfo (System information file)
 - 4) Settings > System Settings > "Custom System" + browse > <file>.sof (Quartus II Programming File)
 - 5) Memory Settings > Memory device > SDRAM
 - 6) Actions > Download System
- CAMBIA-DATOS:**
- 7) Modificar fichero del programa: lab2_part1_2_3_fibo.s para establecer **X** y **P**
 - 8) Compile
 - 9) Load
 - 10) Ejecutar el programa y esperar a que salga el tiempo en el terminal de AMP y apuntarlo en la Tabla 3
- SEGUIR CAMBIA-DATOS**
- SEGUIR CAMBIA-CACHE**

¿Cómo se descubre la capacidad de la cache?: P=1

```
...  
movi r4, 0  
movi r5, X  
LOOP: bge r4, r5, END  
ldb r0, V(r4)  
addi r4, r4, 1  
br LOOP  
END:  
...  
.data  
V:  
.skip 65536  
...
```

- **X sí** cabe en la cache.
- Sólo existen fallos en ITERACION=1.
- En ITERACION={2,...,50000} los accesos aciertan en la cache.
- Tiempo de ejecución (T) es proporcional al número de accesos a la memoria (E).

Espacio de direccionamiento del procesador

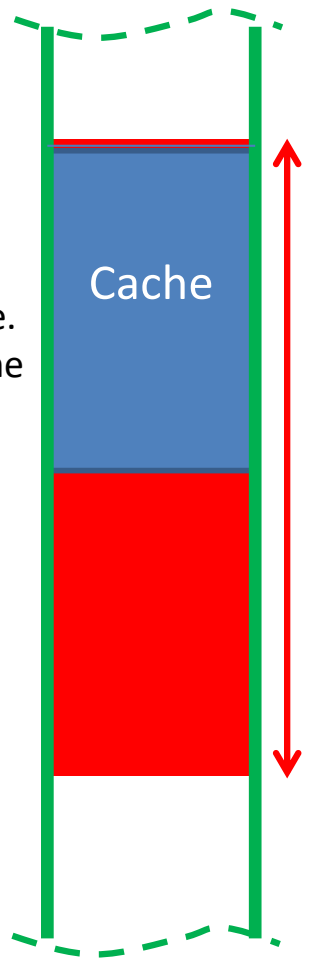


Situación Tipo-1:
NO-desbordamiento

X: volúmenes de datos que maneja el programa; los bytes del vector V caben en la memoria cache de datos.

Situación Tipo-2:
SI-desbordamiento

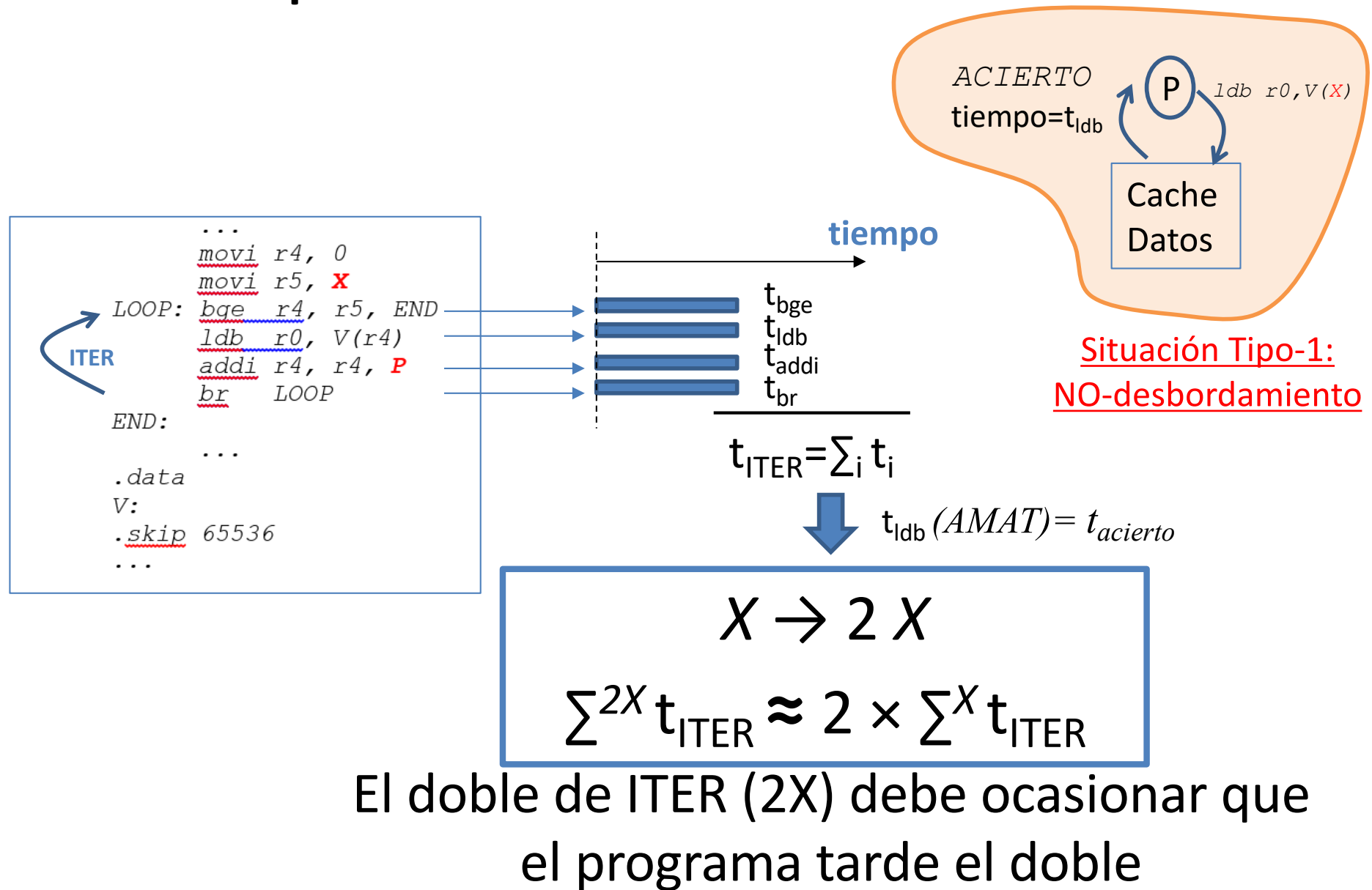
Espacio de direccionamiento del procesador



- **X' no** cabe en la cache.
- Muchos fallos en cache por remplazamientos en todas las ITERACIONES.
- Tiempo de ejecución NO tiene la misma proporción que en X inferiores.

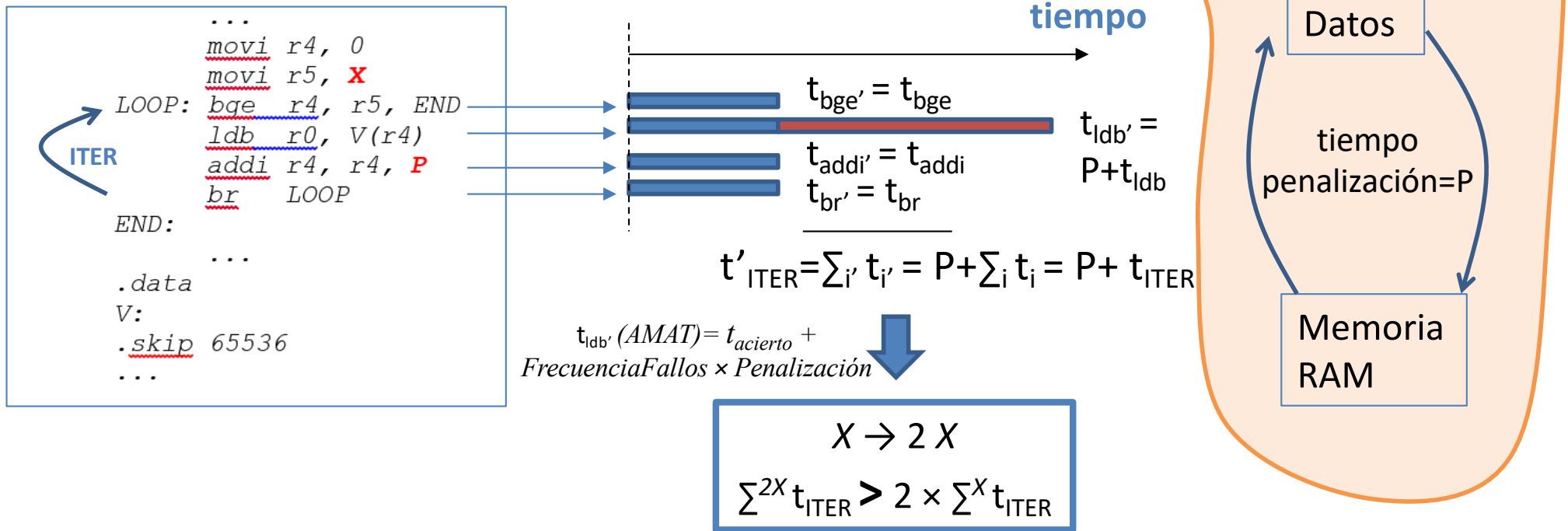
- **CLAVE:** encontrar X' para $P=1$ que hace que el tiempo de ejecución no guarda la proporción con el número de accesos E que en X inferiores; la capacidad es $X'/2$

Tiempo de ejecución **SIN** fallos de cache después de la 1ª ITERACIÓN



Tiempo de ejecución **CON** fallos de cache

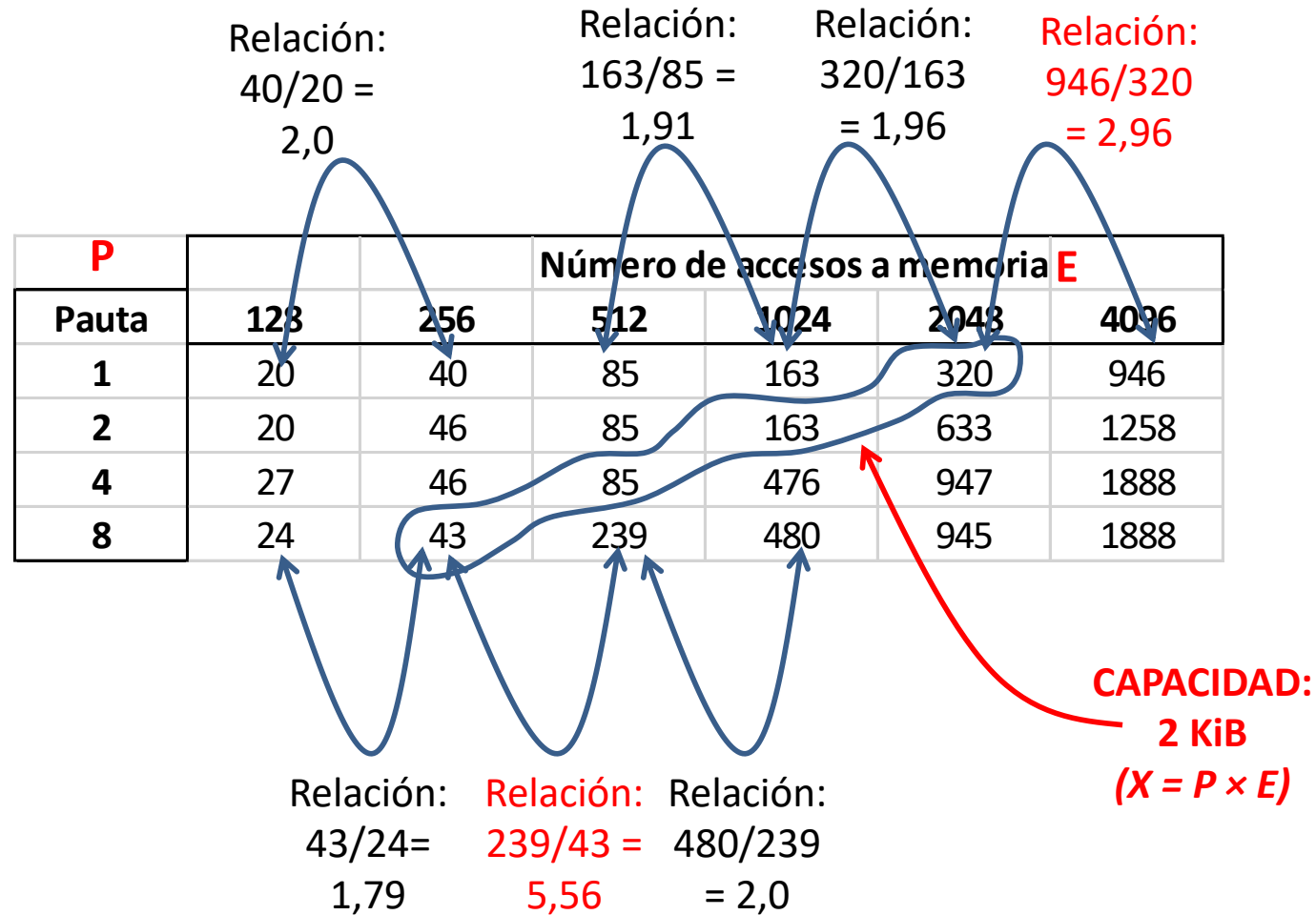
Situación Tipo-2: SI-desbordamiento



El doble de ITER ($2X$) debe ocasionar que el programa tarde **más** que el doble del tiempo con la mitad de iteraciones (X)

¿Cómo se descubre la capacidad de la cache?: P=1

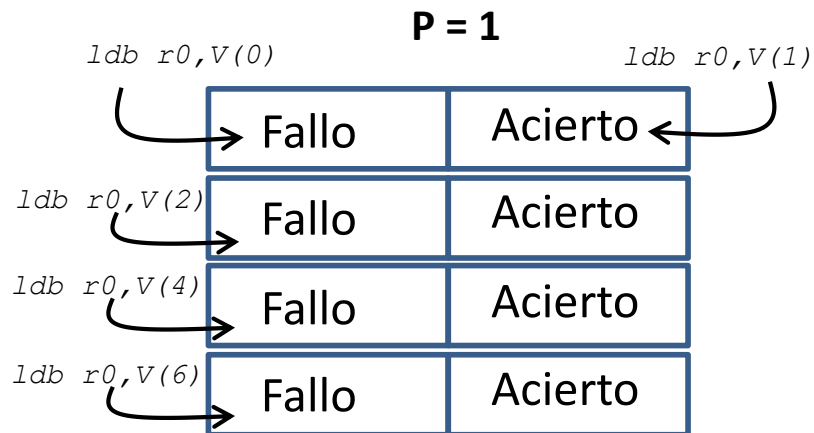
Placa
DE0-Nano



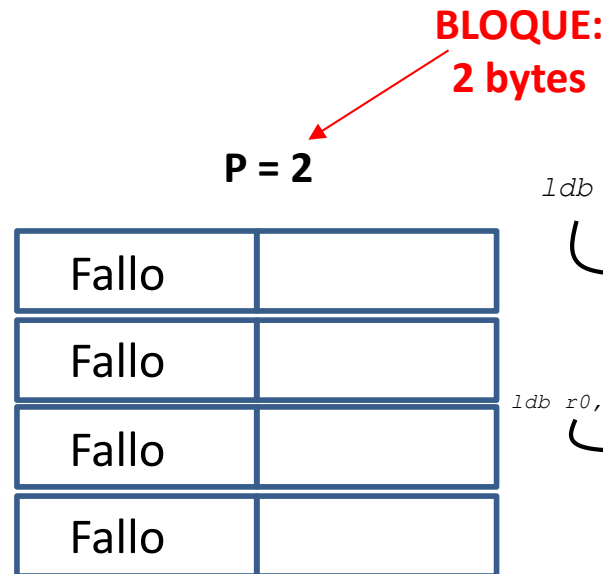
¿Cómo se descubre el tamaño de bloque de la cache?

```
...  
movi r4, 0  
movi r5, X  
LOOP: bge r4, r5, END  
      ldb r0, V(r4)  
      addi r4, r4, P  
      br LOOP  
END:  
...  
.data  
V:  
  .skip 65536  
...
```

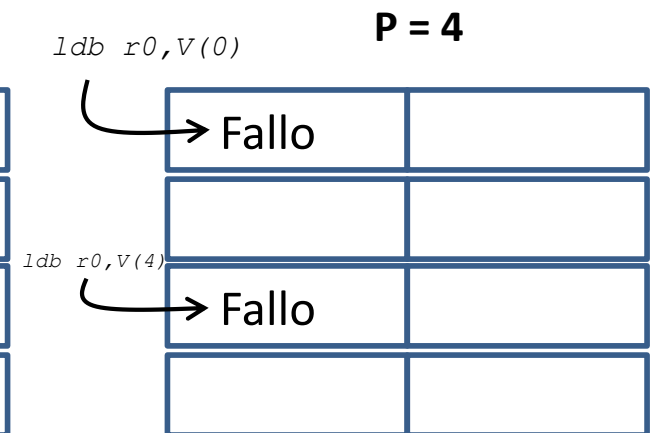
Suponer el caso particular en el que el tamaño de bloque es de 2 bytes



Algunos accesos
aciertan en la cache →
Tiempo de ejecución
inferior a P=2,4



Todos los accesos
fallan en la cache →
Tiempo de ejecución
superior a P=1



Todos los accesos
fallan en la cache →
Tiempo de ejecución
similar a P=2

¿Cómo se descubre el tamaño de bloque de la cache?

Fijarse en una misma columna de accesos, y que la cache se desborda para asegurar que al menos se producen fallos de capacidad

Placa
DE0-Nano

BLOQUE:
4 B

P	Número de accesos a memoria					E
Pauta	128	256	512	1024	2048	4096
1	20	40	85	163	320	946
2	20	46	85	163	633	1258
4	27	46	85	476	947	1888
8	24	43	239	480	945	1888

Relación:
 $480/476$
 $= 1,0$

Relación:
 $476/163$
 $= 2,9$