Práctica 3: Evaluación de prestaciones de los procesadores segmentados

Arquitectura de Computadores

Escuela de Ingeniería Informática
Universidad de Las Palmas de Gran Canaria

Objetivos de la Práctica 3

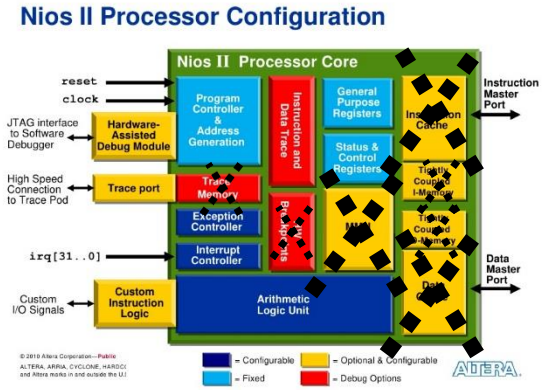
- Evaluación de prestaciones del procesador segmentado Nios II/f y compararlo con el procesador multiciclo Nios II/e
- Analizar el efecto sobre las prestaciones de Nios II/f de la técnica software de reordenación de instrucciones máquina
- Proponer la realización de un ejercicio teórico en el que se evalúa un posible cambio de la microarquitectura de Nios II/f

Programación académica: 4 Semanas

- S1: Parte 1; 2 horas
- S2: Parte 2; 2 horas
- S3: Partes 3 y 4; 2 horas
- S4: Examen; 1,5 horas

Procesadores Software Nios II/{e,f}

Nios II/e



Multiciclo: mínimo 6 clk/inst

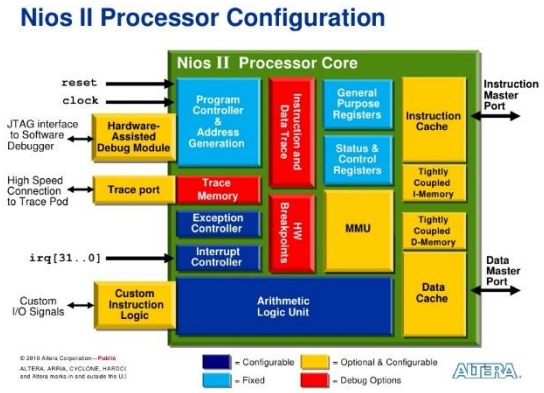
SIN cache de instrucciones

SIN cache de datos

f = 50 MHz

Ficheros configuración DE0-Nano: configuración “DE0-Nano Basic Computer” de Altera Monitor Program 13.1

Nios II/f



Segmentado: 6 segmentos

CON cache de instrucciones

CON cache de datos

CON predicción dinámica de saltos

f = 50 MHz

Ficheros configuración DE0-Nano: N2fdCache512B-4bytes/<…>.{sopcinfo, sof}

Etapas de segmentación de Nios II/f

Table 8: Implementation Pipeline Stages for Nios II/f Core

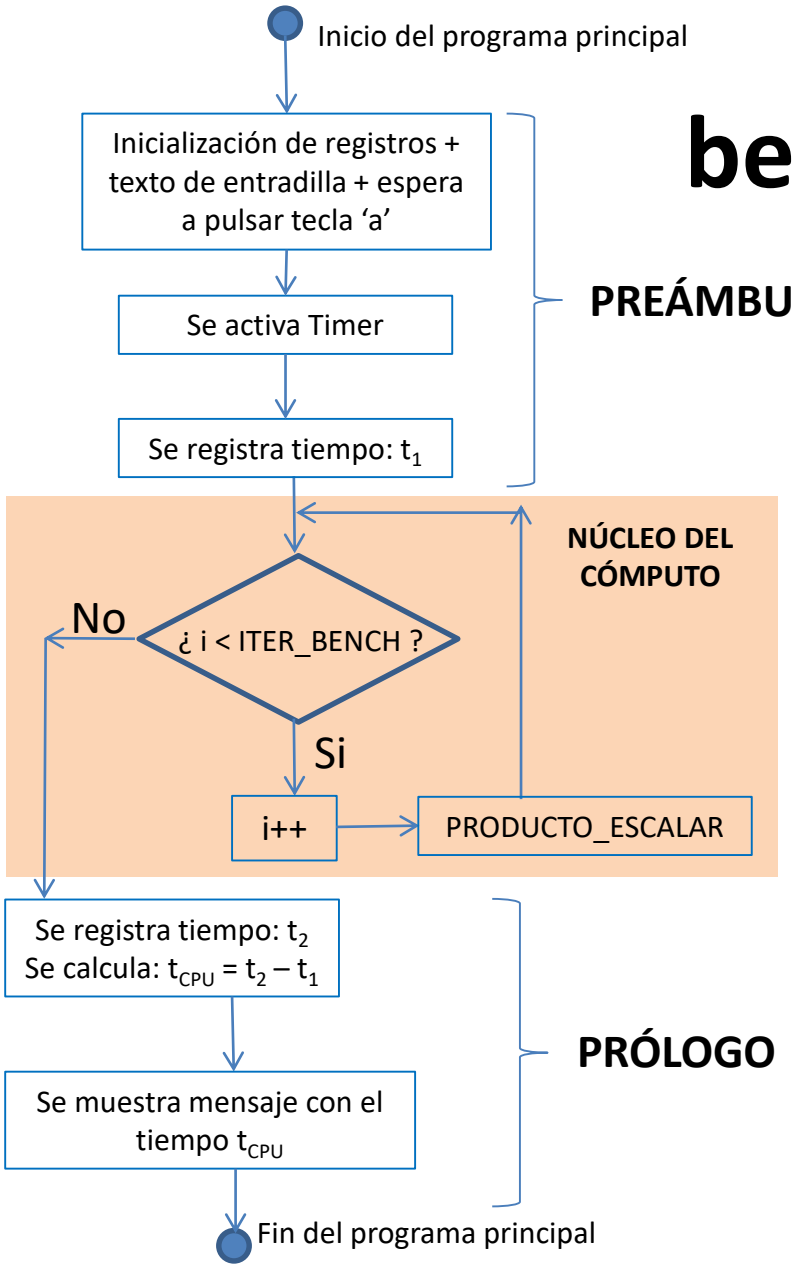
Stage Letter	Stage Name
F	Fetch
D	Decode
E	Execute
M	Memory
A	Align
W	Writeback

- Envío a ejecutar: 1 instrucción / ciclo
- Retirado: 1 instrucción / ciclo
- Predicción de saltos: BHT 2-bits
- CPI base = 1 ciclo
- Paradas del flujo de instrucciones: instrucciones multiciclo, dependencia de datos

Parte 1: Análisis de la mezcla de tipos de instrucciones en un programa benchmark y del CPI de los procesadores Nios II/{e,f}

Descripción general. Se utilizará un programa benchmark sintético que denominamos *benchNIOSII2021_Parte1* para analizar la mezcla de instrucciones del repertorio de instrucciones Nios II de 32 bits. Este programa realiza el producto escalar de dos vectores repetidas veces, tantas como indica la constante del programa `ITER_BENCH`. Adicionalmente, se calculará el CPI de los procesadores Nios II/{e,f}.

PARTE 1: benchNIOSII2021_Parte1



Parte 1: Objetivos

- Objetivo 1: Clasificar las instrucciones, contando el número de veces que se ejecuta cada una de ellas en la subrutina `PRODUCTO_ESCALAR` del código fuente que se encuentra en el fichero: `producto_escalar.s`. En la transparencia anterior se muestra un diagrama de flujo de las principales actividades que se realizan en el benchmark.
- Objetivo 2: A continuación, calcular el número total de instrucciones ejecutadas y los porcentajes de cada tipo de instrucción (ALU, MEMORIA, SALTOS, OTRAS) rellenando la Tabla 1.
- Objetivo 3: registrar el número total de ciclos de reloj en los que se ejecuta el benchmark, tanto para el procesador Nios II/e como Nios II/f, y calcular el CPI del programa. Importante: considerar que la parte del benchmark que no es el núcleo del cómputo no aporta un número significativo de instrucciones al cálculo del CPI.

Parte 1: Metodología práctica con los procesadores Nios II

1. Crear un proyecto nuevo en AMP utilizando la configuración *DE0-Nano Basic Computer* (procesador Nios II/e) o *Custom System* (procesador Nios II/f).
2. Seleccionar SDRAM como dispositivo de memoria principal y establecer el desplazamiento de las secciones `.text` y `.data` dentro del espacio de direccionamiento de memoria en `0x400`.
3. Compilar y cargar el programa benchmark (carpeta: benchNIOSII_Parte1) en la placa DE0-Nano.
4. Ejecutar paso a paso para contar los tipos de instrucciones que se ejecutan dentro del núcleo de cómputo del benchmark utilizando breakpoints en la zona correspondiente del programa ejecutable.

Parte 1: Tabla 1

Instrucción ALU	**Número de ejecuciones**	**Instrucción MEMORIA**	**Número de ejecuciones**	**Instrucción SALTOS**	**Número de ejecuciones**	**Instrucciones OTRAS**	**Número de ejecuciones**
`addi`		`ldw`		`beq`		`nop`	
…		…		…		…	
…		…		…		…	
Total instrucciones ALU		**Total instrucciones MEMORIA**		**Total instrucciones SALTOS**		**Total instrucciones OTRAS**	
N (instrucciones totales ejecutadas)							
% ALU		**% MEMORIA**		**% SALTOS**		**% OTRAS**	
Ciclos Nios II/e				**Ciclos Nios II/f**			
CPI Total del programa Nios II/e				**CPI Total del programa Nios II/f**			

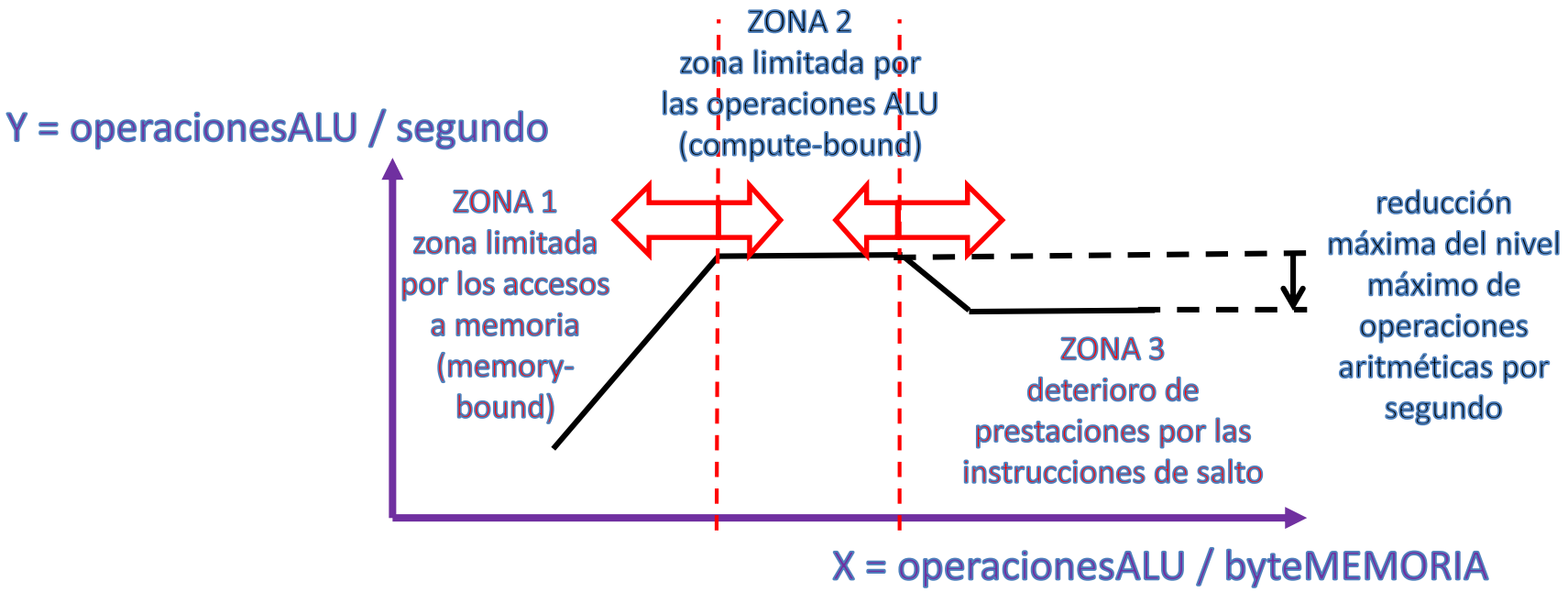
Parte 2. Análisis de las limitaciones de la relación “operacionesALU/segundo” de un programa benchmark en los procesadores multiciclo Nios II/e y segmentado Nios II/f

Descripción general. En esta parte se evalúan los límites de cada procesador software Nios II{e,f} en cuanto al número de operaciones ALU que pueden realizar en cada unidad de tiempo. Concretamente, se analizarán los límites medidos en “operacionesALU/segundo” impuestos por la unidad funcional ALU, la jerarquía de memoria y las instrucciones de salto.

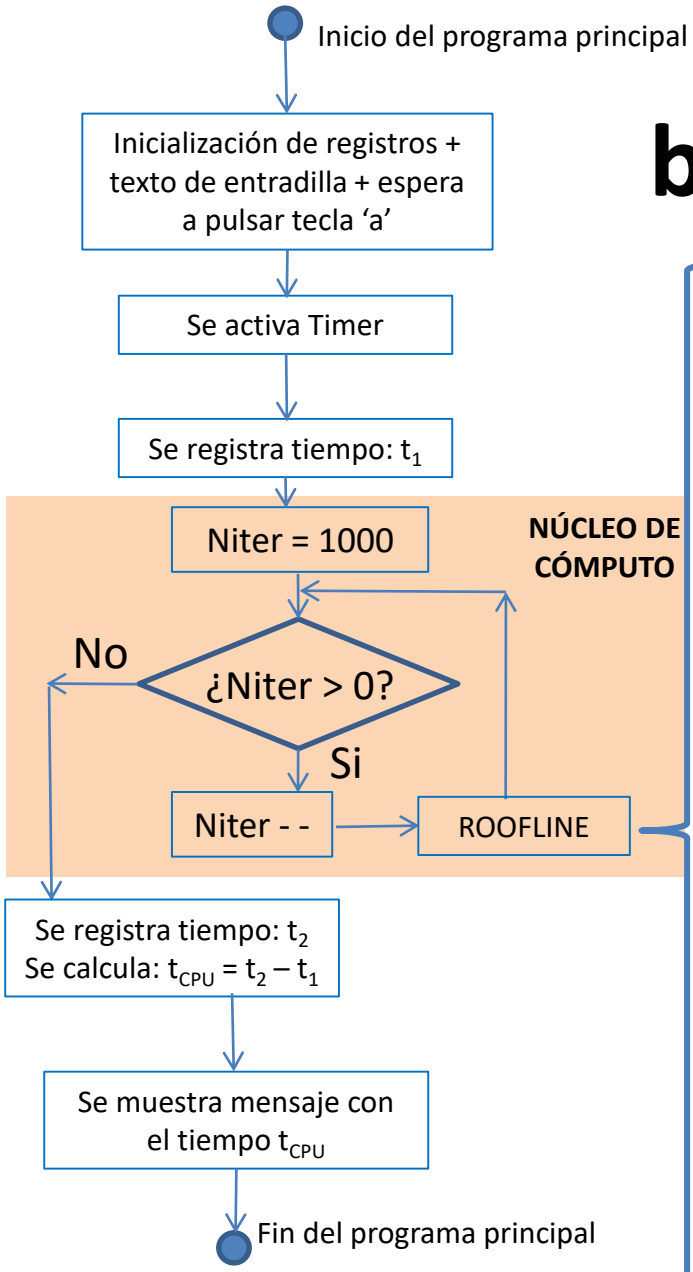
Parte 2: Objetivos

- Objetivo 1: Obtener la curva "*operacionesALU/segundo"* versus "*operacionesALU/byteMEMORIA*" (ver Figura 2). Esta curva representa el nivel de prestaciones del procesador Nios II medido en número de operaciones ALU realizadas por unidad de tiempo.

Figura 2: “operacionesALU/segundo” vs. “operacionesALU/byteMEMORIA”



PARTE 2: benchNIOSII2021_Parte2.s



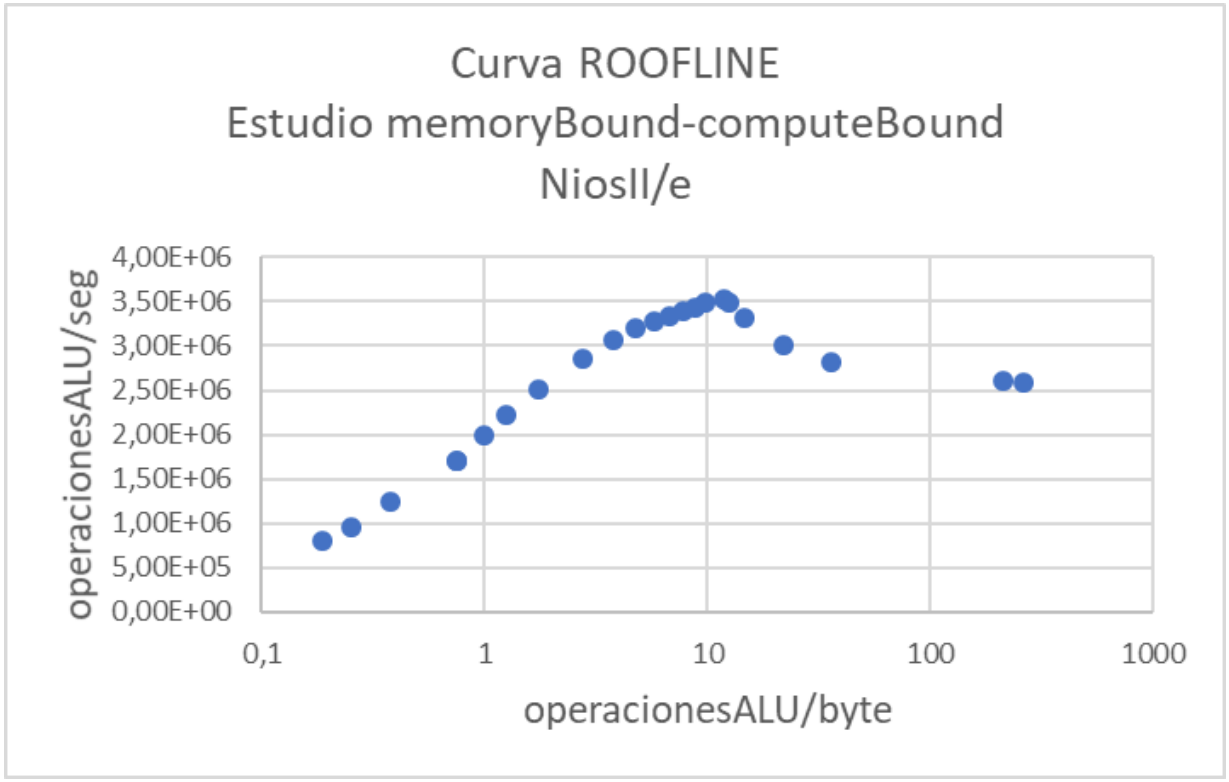

```
ldw             r4, 0(r4)   /* r4 = Niter, numero de iteraciones a realizar */
LOOP:
/* Begin: ZONA 1 de accesos a memoria */
                ldw             r6, 0(r2)   /* carga A */
/* End: Zona de accesos a memoria */

/* Begin: ZONA 2 de operaciones ALU */
                add             r6, r6, r6
/* End: Zona de operaciones ALU */

/* 2 instrucciones más de tipo ALU, nunca se desactivan */
                addi            r7, r7, 1   /* contador_iteraciones_realizadas++ */
                subi            r4, r4, 1   /* Niter-- */

/* Begin: ZONA 3 de bucle interno para forzar la ejecución de múltiples saltos */
                addi            r5,r0,NiterInternas /* NiterInternas= 1,5,20,47,... */
bucleInterno:
                add             r6, r6, r6
                subi            r5, r5, 1
                bgt             r5, r0, bucleInterno
/* End: ZONA 3 de bucle interno para forzar la ejecución de múltiples saltos */

/* Fin del bucle  LOOP */
                bgt             r4, r0, LOOP
```

ID del punto de la curva	kernel: `roofline.s` número instrucciones por iteración			**Coordenada X** operacionesALU/ byteMEM (ALU / 4 * ldw)	Iteraciones (N_{iter})	N (instrucciones ejecutadas, N_{iter} * [ldw+ALU+br])	Nios II/{e,f}, f=50 MHz			
	ldw	**ALU**	**br**				ciclos	t_{CPU} (seg=ciclos / f)	**Coordenada Y** operacionesALU/seg (N_{iter} * ALU / t_{CPU})	CPI (ciclos / N)
1	4	3	1		1000					
2	3	3	1		1000					
3	2	3	1		1000					
4	1	3	1		1000					
5	1	4	1		1000					
6	1	5	1		1000					
7	1	7	1		1000					
8	1	11	1		1000					
9	1	15	1		1000					
10	1	19	1		1000					
11	1	23	1		1000					
12	1	27	1		1000					
13	1	31	1		1000					
14	1	35	1		1000					
15	1	39	1		1000					
16	1	47	1		1000					
17	1	50	2		1000					
18	1	58	6		1000					
19	1	88	21		1000					
20	1	142	48		1000					
21	1	848	401		1000					
22	1	1048	501		1000					

Parte 2: Tabla 2

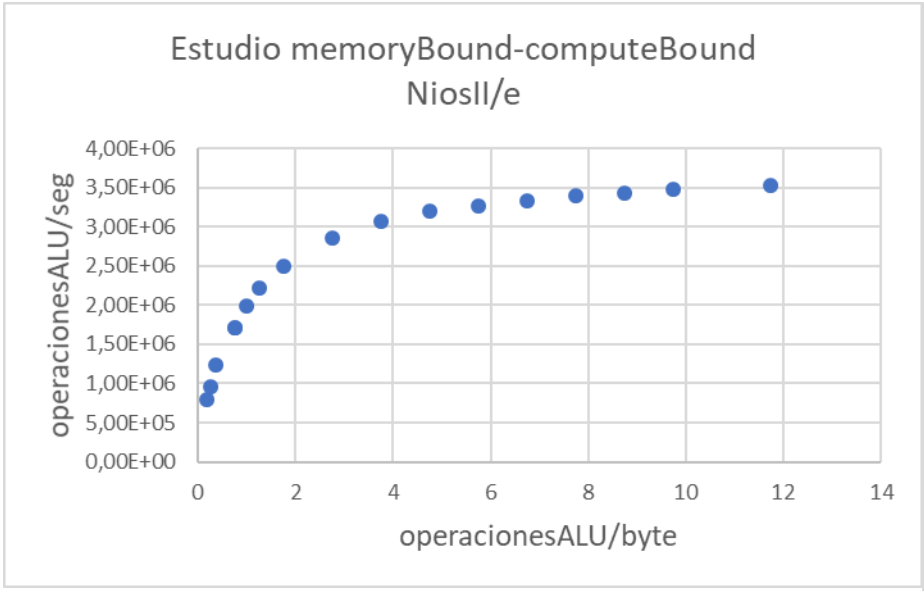
(una para Nios II/e y otra para Nios II/f)

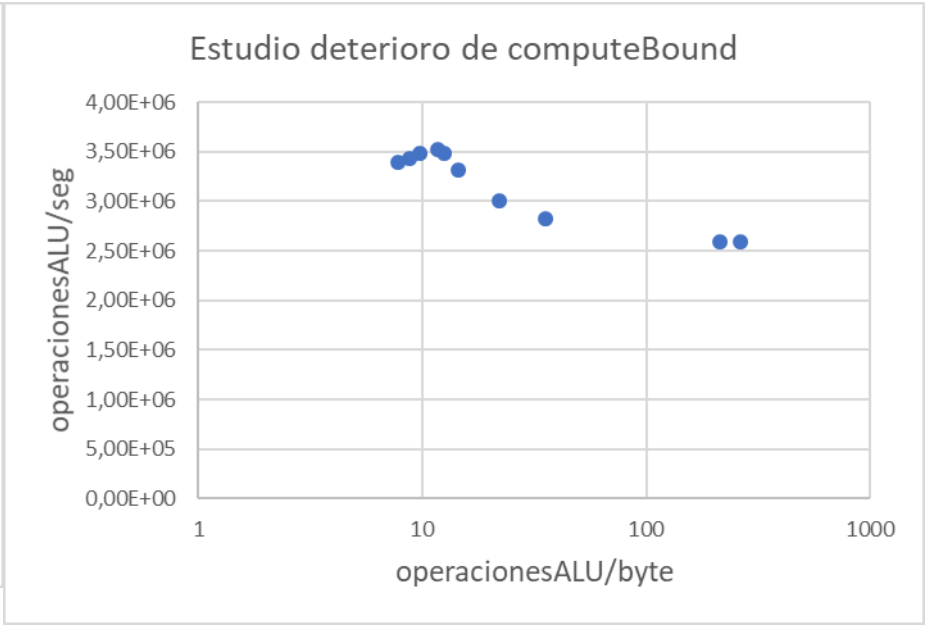
Resultados Parte 2



Nios II/e

Resultados Parte 2





Nios II/f

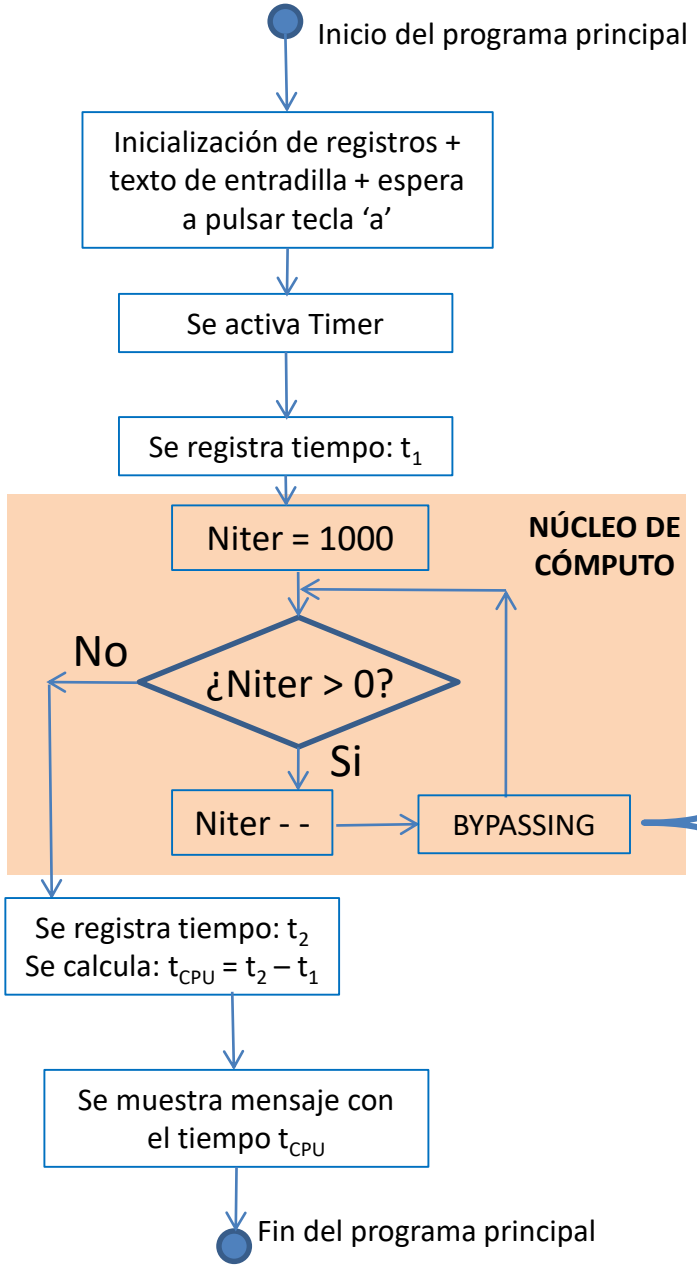
Preguntas

- ¿A partir de qué número de operacionesALU por byte accedido a memoria los procesadores Nios II/{e,f} están limitados por las operaciones ALU? ¿Cuál es el número máximo de operaciones ALU por segundo que puede alcanzar el procesador Nios II/e? ¿Y el Nios II/f?
- ¿Cuál es el porcentaje máximo de deterioro de las prestaciones de los procesadores Nios II/{e,f} cuando se ejecutan instrucciones de salto?
- ¿El programa benchmark de la Parte 1 (benchNIOSII2021_Parte1/producto_escalar) está limitado por memoria o limitado por las operaciones ALU?

Parte 3. Análisis de los efectos que ocasiona la reordenación de instrucciones en procesadores segmentados Nios II/f

Descripción general. Se utilizará un nuevo programa benchmark sintético que denominamos *benchNIOSII2021_Parte3* para evaluar el efecto de las dependencias de datos verdaderas RAW entre instrucciones de carga e instrucciones ALU. Este programa benchmark es similar al de las partes 1 y 2 de esta práctica excepto que se modifica el núcleo de cómputo que ahora se encuentra en un fichero llamado `bypassing.s`. Posteriormente, se propone aplicar la técnica de reordenación de instrucciones para reducir el tiempo de ejecución del programa benchmark usando el procesador Nios II/f.

PARTE 3: benchNIOSII2021_Parte3.s



VERSIÓN 1

```
LOOP:
        ldw     r6, 0(r2)   /* carga A */

/* ZONA de dependencia de datos */
        /* suma CON dependencia de datos con ldw r6, 0(r2) */
        add     r6, r6, r6

/* 2 instrucciones mas de tipo ALU, nunca se comentan */
        addi    r7, r7, 1   /* Niter++ */
        subi    r4, r4, 1   /* N-- */
/* fin del bucle LOOP*/
        bgt     r4, r0, LOOP
```

VERSIÓN 2

```
LOOP:
        ldw     r6, 0(r2)   /* carga A */

/* ZONA de dependencia de datos */
        /* suma SIN dependencia de datos con ldw r6, 0(r2) */
        add     r6, r4, r4

/* 2 instrucciones mas de tipo ALU, nunca se comentan */
        addi    r7, r7, 1   /* Niter++ */
        subi    r4, r4, 1   /* N-- */
/* fin del bucle LOOP*/
        bgt     r4, r0, LOOP
```

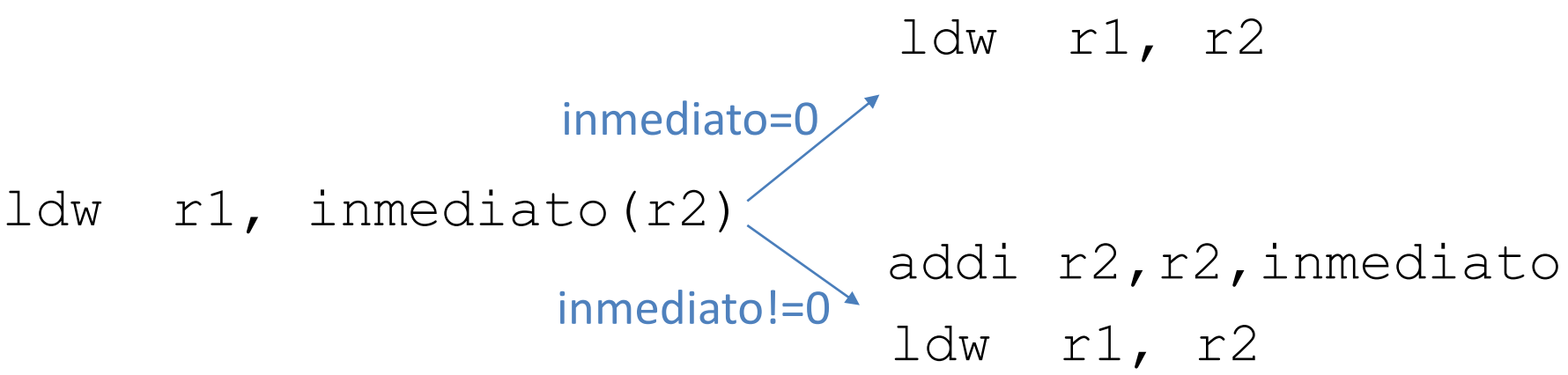
Ejercicio Parte 3: VERSIÓN 3

- Reordenar las instrucciones del bucle `LOOP` de la Versión 1 para reducir el tiempo de cómputo total del programa (t_{CPU})
- Medir el tiempo de ejecución (t_{CPU}) en Nios II/f
- Medir el speed-up
- Medir el CPI total
- Medir el CPI de penalización ocasionado por la dependencia verdadera RAW de la pareja de instrucciones `ldw -> add`

PARTE 4: NUEVO DISEÑO DE PROCESADOR SEGMENTADO

Procesador original 6 segmentos

Nuevo Procesador 5 segmentos



¿Cuál es el porcentaje adicional de instrucciones que se tienen que ejecutar en el nuevo procesador suponiendo que se ejecuta la subrutina PRODUCTO_ESCALAR de la Parte 1 de esta práctica?