# Práctica 3
# Evaluación de prestaciones de los procesadores segmentados

## Introducción

El principal objetivo de esta Práctica 3 consiste en la evaluación de prestaciones del procesador segmentado Nios II/f y compararlo con el procesador multiciclo Nios II/e. Adicionalmente, se analizará el efecto sobre las prestaciones de Nios II/f de la técnica software de reordenación de instrucciones máquina. Por último, se propone la realización de un ejercicio teórico en el que se evalúa un posible cambio de la microarquitectura de Nios II/f usando datos obtenidos durante las actividades prácticas.

Esta práctica consta de cuatro partes que se resumen a continuación.

Parte 1. Se realiza un análisis de las instrucciones ejecutadas de un programa benchmark para conocer los porcentajes de cada uno de los tipos de instrucciones: ALU, memoria, saltos.

Parte 2. Se analizan las prestaciones de los procesadores multiciclo Nios II/e y segmentado Nios II/f para conocer en qué circunstancias la ejecución de un programa está limitada por los accesos a memoria o por las operaciones ALU. Adicionalmente, se analiza el deterioro sobre las prestaciones de estos procesadores que introducen las instrucciones de salto.

Parte 3. Se comparan los efectos que sobre las prestaciones del procesador segmentado Nios II/f ocasiona la técnica software de reordenación de instrucciones.

Parte 4. Se propone evaluar un nuevo diseño de procesador segmentado.

El material informático de esta práctica se encuentra en las siguientes carpetas:

- benchNIOSII2021_Parte1
- benchNIOSII2021_Parte2
- benchNIOSII2021_Parte3
- N2fdCache512B-4bytes
- guion

Se utilizará una placa Intel DE0-Nano que se encuentra en uno de los laboratorios de la ULPGC (ver Figura 1). Para ello, cada estudiante se conectará a un ordenador del laboratorio y ejecutará la máquina virtual “Altera” donde se encuentra la herramienta Altera Monitor Program (AMP) que permite interactuar con la placa.

**Figura 1.** Placa DE0-Nano.


**Parte 1. Análisis de la mezcla de tipos de instrucciones en un programa benchmark**

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` (ver la subrutina `PRODUCTO_ESCALAR` en el fichero `producto_escalar.s`).

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 Figura 2 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 para cada procesador. 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.

Metodología práctica con el procesador multiciclo Nios II/e:

1. Crear un proyecto nuevo en AMP utilizando la configuración *D0-Nano Basic Computer*.
2. 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 que se identifica en la Figura 1 utilizando breakpoints en la zona correspondiente del programa ejecutable.

Metodología práctica con el procesador multiciclo Nios II/f:

1. Crear un proyecto nuevo en AMP utilizando la configuración *Custom System* con el fichero sopcinfo: `nios_system.sopcinfo` y el fichero sof: `DE0_Nano_Basic_Computer.sof` (ambos están en la carpeta: N2fdCache512B-4bytes).

2. 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 el programa hasta recibir el mensaje "fin del programa".

Ficheros de la práctica:

- Programa principal del benchmark: `benchNIOSII2021_Parte1.s`
- Subrutinas: `DIV.s`, `JTAG2021.s`, `productoEscalar.s`, `BCD.s`, `nios_macros.s` (carpeta: benchNIOSII_Parte1).
- Ficheros de configuración de NIOSII/f: `nios_system.sopcinfo`, `DE0_Nano_Basic_Computer.sof` (carpeta: N2fdCache512B-4bytes).

Preguntas:

1. ¿En qué tipo de programa clasificarías al programa benchNIOSII2021_Parte1 (aritmético, dominado por los accesos a memoria o dominado por las instrucciones de salto)? Justifica y razona la respuesta.
2. Conociendo que las instrucciones tardan en promedio en ejecutarse en el procesador Nios II/e un total de 6 ciclos [1], y que en el procesador Nios II/f [1]: 1 ciclo (ALU), 1 ciclo (MEMORIA), 2 ciclos (SALTOS), obtener el CPI teórico del programa en ambos procesadores.
3. ¿Qué diferencia encuentras con los valores obtenidos de los CPIs respecto a los medidos en el Objetivo 3 de esta Parte 1 en la placa DE0-Nano? ¿A qué causa crees que es debida tales diferencias? Justifica y razona las respuestas.

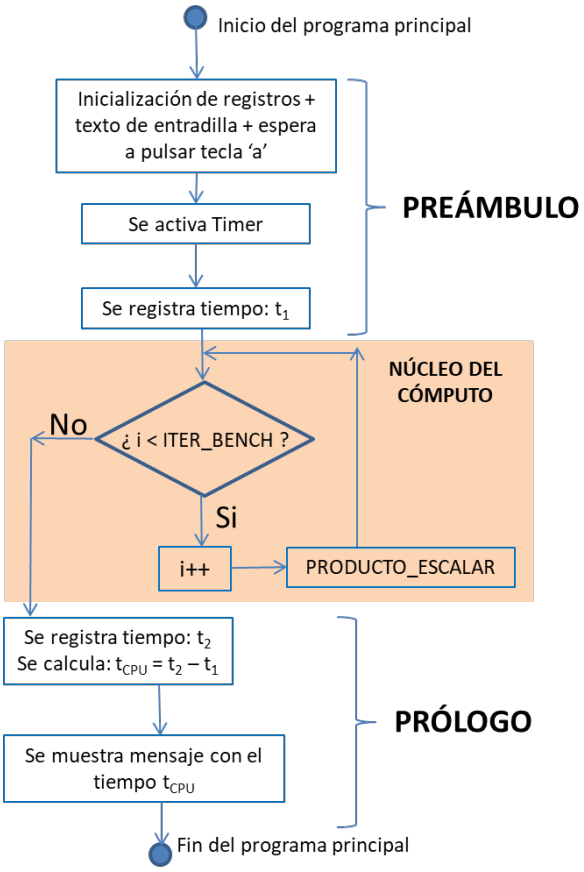


**Figura 2.** Diagrama de flujo del programa benchmark cuyo programa principal se encuentra en el fichero `benchNIOSII2021_Parte1.s`. El benchmark se divide en tres partes: preámbulo, núcleo del cómputo y prólogo. En el preámbulo se espera la introducción de la tecla 'a' a través del teclado del ordenador, se configura el dispositivo de entrada/salida denomina Timer de la placa DE0-Nano y se registra la primera marca de tiempo $t_1$. En el núcleo del cómputo se realiza repetidas veces el producto escalar de dos vectores de seis componentes cada vector. Y en el prólogo se registra la segunda marca de tiempo $t_2$, se calcula el tiempo entre las dos marcas de tiempos y se muestra este valor por el terminal de AMP.

**Tabla 1.** Obtención de los porcentajes de instrucciones ejecutadas por los procesadores Nios II/{e,f} usando la subrutina `PRODUCTO_ESCALAR` que se encuentra en el fichero `producto_escalar.s` (carpeta: benchNIOSII_Parte1)

| **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 "operaciones ALU/segundo" de la ejecución 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 aritméticas 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. Para ello se utilizará un segundo benchmark denominado *benchNIOSII2021_Parte2*.

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 aritméticas realizadas por unidad de tiempo. En la curva se pueden distinguir tres partes (ver Figura 2):

1. *Zona 1* en la que las prestaciones del procesador medidas en "operacionesALU/segundo" están limitadas por los accesos a memoria (en inglés se dice que las prestaciones del procesador son de tipo *memory-bound*). Esta zona se caracteriza porque a medida que aumenta el número de operaciones aritmética (operacionesALU) por byte de memoria accedido, aumenta el número de operaciones aritméticas por segundo que el procesador realiza.
2. *Zona 2* en la que las prestaciones del procesador medidas en "operacionesALU/segundo" están limitadas por el máximo número de operaciones aritméticas que puede realizar el procesador por segundo (en inglés se dice que las prestaciones del procesador son de tipo *compute-bound*). Esta zona se caracteriza porque a medida que aumenta el número de operacionesALU por byte de memoria accedido,

el número de operaciones ALU por segundo se mantiene constante en el nivel máximo que es posible para el procesador.

3. *Zona 3* en la que las prestaciones del procesador medidas en operacionesALU/segundo están deterioradas por las instrucciones de salto respecto al valor máximo medido en la Zona 2. Esta zona se caracteriza porque a medida que aumenta el número de operacionesALU por byte de memoria accedido, el valor de "operacionesALU/segundo" se mantiene constante, pero en un nivel inferior al proporcionado en la Zona 2.

Metodología común a los procesadores Nios II/{e,f} en las experiencias prácticas de laboratorio: se modifica el programa fuente que se encuentra en el fichero `roofline.s` de las maneras que se indican a continuación. Este código pertenece al benchmark *benchNIOSII2021_Parte2* (ver Figura 3). Seguidamente, el programa se ejecuta usando ambos procesadores Nios II varias veces, tantas como líneas aparecen en la Tabla 2:

- Cada línea de la Tabla 2 tiene asociada la ejecución del programa benchmark *benchNIOSII2021_Parte2* en la que previamente se debe modificar cada una de las zonas marcadas con comentarios en el propio fichero `roofline.s`.
    - Modificación de la Zona 1: se activarán/descomentarán tantas instrucciones `ldw` como se indica en la columna "ldw" de la Tabla 2.
    - Modificación de la Zona 2: se activarán/descomentarán tantas instrucciones `add` como se indica en la columna "ALU" de la Tabla 2.
    - Modificación de la Zona 3: se activarán/descomentarán tantas iteraciones como se indica en la columna "br" de la Tabla 2.
- Rellenar la Tabla 2 para cada procesador Nios II{e,f}.
- Representar las tuplas de valores (operacionesALU/segundo, operacionesALU/byteMEMORIA) en dos gráficos X-Y como el de la Figura 2, uno para cada procesador, utilizando los valores obtenidos en la Tabla 2.

Metodología propia del procesador multiciclo Nios II/e:

1. Crear un proyecto nuevo en AMP, utilizar la configuración "DE0-Nano Basic Computer".
2. 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_Parte2) en la placa DE0-Nano.
4. Ejecutar hasta el final del programa, momento en el que aparecerá en el terminal de AMP el mensaje "fin del programa".

Metodología propia del procesador segmentado Nios II/f:

1. Crear un proyecto nuevo en AMP, utilizar la configuración "Custom system" con fichero sopcinfo: nios_system.sopcinfo y fichero sof: `DE0_Nano_Basic_Computer.sof`.
2. 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_Parte2) en la placa DE0-Nano.
4. Ejecutar hasta recibir el mensaje "fin del programa".

Ficheros de la práctica:

- Programa benchmark: `benchNIOSII2021_Parte2.s`, `DIV.s`, `JTAG2021.s`, `roofline.s`, `BCD.s`, `nios_macros.s` (carpeta: bench-NIOSII_Parte2).
- Ficheros de configuración de Nios II/f: `nios_system.sopcinfo`, `DE0_Nano_Basic_Computer.sof` (carpeta: N2fdCache512B-4bytes).

Preguntas:

4. ¿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?
5. ¿Cuál es el porcentaje máximo de deterioro de las prestaciones de los procesadores Nios II/{e,f} cuando se ejecutan instrucciones de salto?
6. ¿El programa benchmark de la Parte 1 (benchNIOSII2021_Parte1/producto_escalar) está limitado por memoria o limitado por las operaciones ALU?

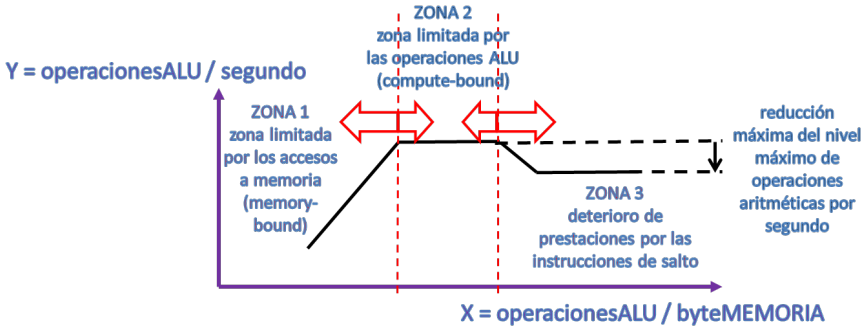


**Figura 2.** Curva "roofline" para la determinación de las zonas de limitación de las operaciones ALU por unidad de tiempo.

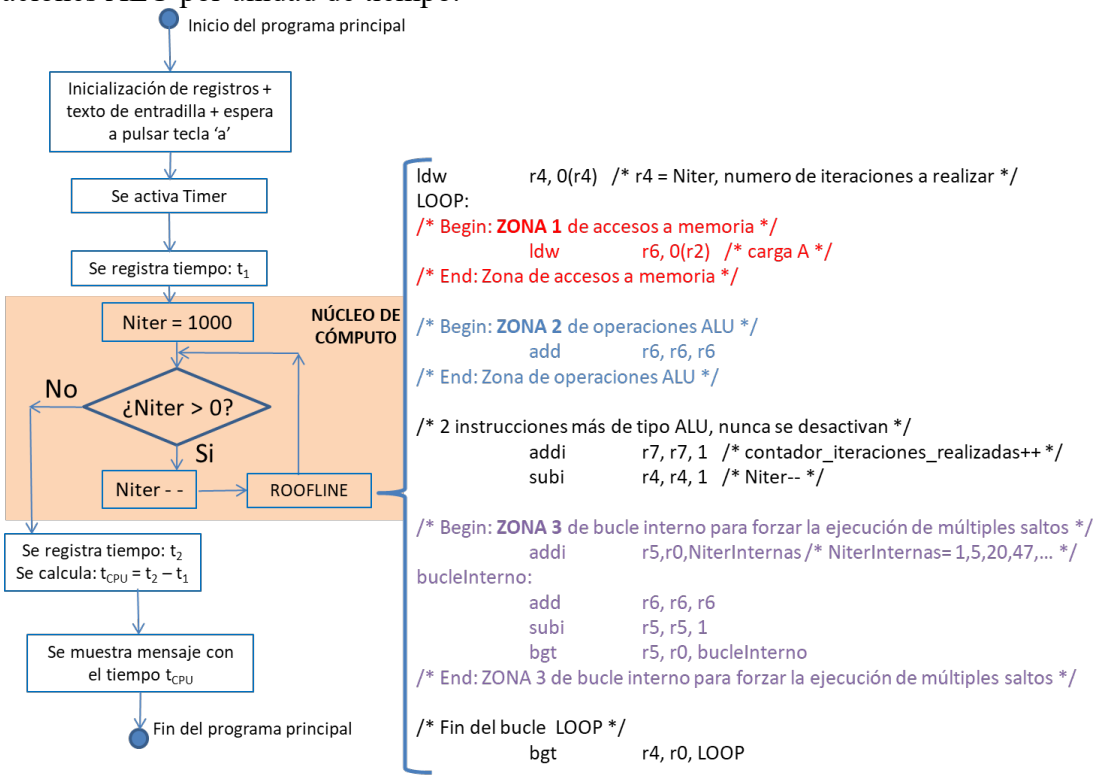


**Figura 3.** Diagrama de flujo del programa benchmark benchNIOSII2021_Parte2 y bucle interno de la subrutina `ROOFLINE` que se encuentra en el fichero `roofline.s`.

**Tabla 2.** Obtención de la curva "roofline" de limitación de las prestaciones de los procesadores Nios II/{e,f} (ver Figura 2). La coordenada X de la curva corresponde a la medida "operacionesALU/byteMEM", y la coordenada Y corresponde a la medida "operacionesALU/seg".

Leyendas.

- ID: número de identificación del experimento práctico para obtener un punto de la curva.
- ldw: número de instrucciones `ldw` del bucle LOOP del código fuente `roofline.s`.
- ALU: número de instrucciones ALU del bucle LOOP del código fuente `roofline.s`.
- br: número de instrucciones de salto del bucle LOOP del código fuente `roofline.s`.
- operacionesALU/byteMEM: número de instrucciones ALU que se ejecutan en cada iteración del bucle por cada byte que es accedido a la memoria principal. Su valor viene determinado por los valores de las correspondientes columnas: ALU / 4 * ldw.
- $N_{iter}$: número de iteraciones del bucle LOOP del fichero `roofline.s`.
- N: número total de instrucciones ejecutadas por la subrutina `ROOFLINE` que se encuentra en el fichero `roofline.s`. Su valor se obtiene combinando los valores de las columnas: $N_{iter}$ x [ldw + ALU + br].
- ciclos: medida obtenida por el terminal de AMP que se corresponde con el tiempo total de ejecución del programa benchNIOSII2021_Parte2.
- $t_{CPU}$: tiempo de ejecución del programa benchNIOSII2021_Parte2 y que se obtiene con la fórmula: ciclos/f, donde f=50 MHz=50 $10^6$ Hz. $t_{CPU}$ se mide en segundos.
- operacionesALU/seg: número de operaciones realizadas en la unidad funcional ALU por unidad de tiempo. Su valor se obtiene combinando los valores de las columnas: $N_{iter}$ * ALU / $t_{CPU}$
- CPI: número promedio en el que cada instrucción del programa se ejecuta. Su valor se obtiene combinando los valores de las columnas: ciclos / N.

| 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 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.

Objetivo 1: Editar el fichero `bypassing.s` y colocar una instrucción `add` que SÍ sea dependiente en datos con la instrucción `ldw` (ver Versión 1 en la Figura 4). Posteriormente, medir y anotar el tiempo de ejecución del programa benchmark que se obtiene en la terminal de AMP usando el procesador Nios II/f.

Objetivo 2: Editar el fichero `bypassing.s` y colocar una instrucción `add` que NO sea dependiente en datos con la instrucción `ldw` (ver Versión 2 en la Figura 4). Posteriormente, medir y anotar el tiempo de ejecución del programa benchmark que se obtiene en la terminal de AMP usando el procesador Nios II/f.

Preguntas:

7. Cuantificar e indicar el incremento del tiempo de ejecución del programa ($t_{CPU}$) en porcentaje (%) cuando existen dependencias de datos `ldw -> add` respecto a la no existencia de dependencias de datos `ldw -> add`.
8. Calcular el CPI de las dos versiones del programa benchmark benchNIOSII2021_Parte3: Versión 1 y 2. Para ello, cuenta sólo las instrucciones ejecutadas en el bucle LOOP, observando que existe una constante denominada $N_{iter}$=1000 en el código fuente de `bypassing.s`.
9. A partir de los dos valores de CPI de la Pregunta 8, calcular el valor del CPI de penalización debido sólo a la dependencia `ldw -> add` en la Versión 1 (con penalizaciones por dependencias verdaderas RAW) y su porcentaje respecto al CPI de la Versión 2 (sin penalizaciones RAW). Recordar que: $CPI = CPI_{base} + CPI_{penalizacion}$

10. Proponer una nueva versión (Versión 3) de la Versión 1 del benchmark utilizando la técnica de *reordenación de instrucciones*. Observar que no tiene que cambiar el número de instrucciones que se ejecutan dentro del bucle `LOOP`.
11. Crear la Versión 3 del programa benchmark y medir el tiempo de ejecución en AMP usando el procesador Nios II/f.
12. Calcular el speed-up de la nueva Versión 3 respecto a la Versión 1.
13. Calcular el speed-up de la nueva Versión 3 respecto a la Versión 2.
14. A partir de los resultados anteriores, ¿qué conclusión relativa a las prestaciones del procesador segmentado Nios II/f puedes obtener de la aplicación de la técnica de reordenación de instrucciones?

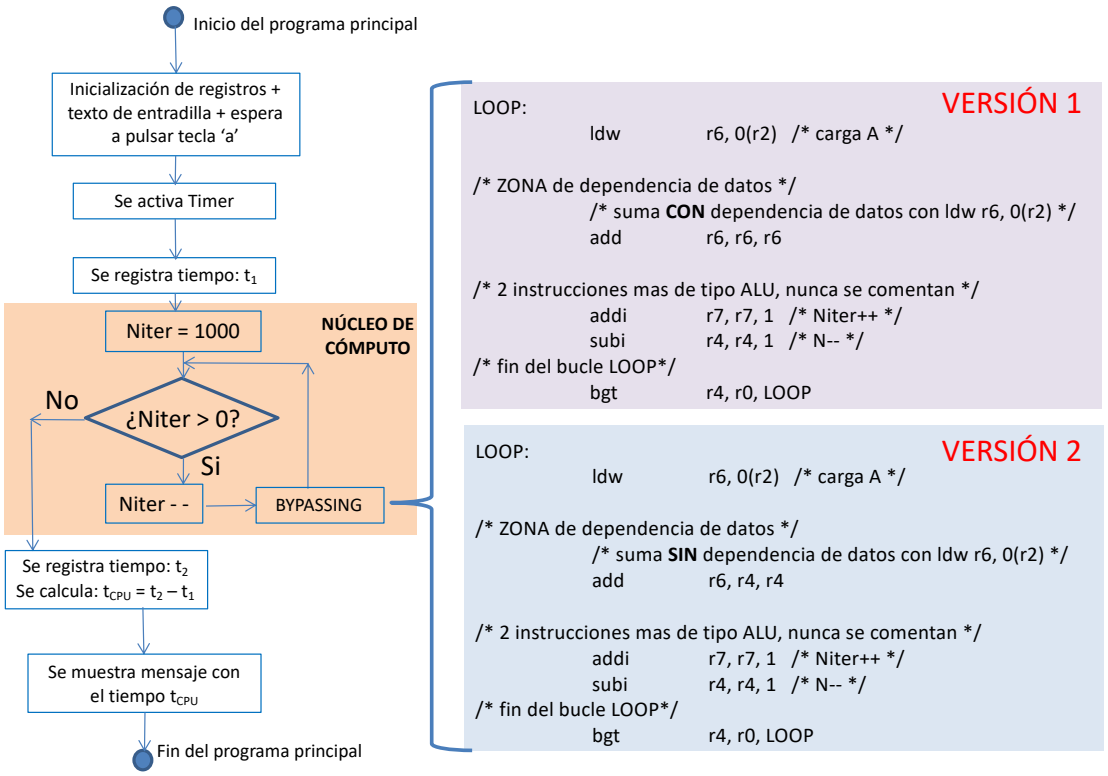


**Figura 4.** Diagrama de flujo del programa benchmark benchNIOSII2021_Parte3 y bucle interno `LOOP` de la subrutina `BYPASSING` que se encuentra en el fichero `bypassing.s`.

## Parte 4. Problema de diseño de un nuevo procesador segmentado

Imagina que tu jefe del departamento de arquitectura de computadores en Intel te pide que evalúes una posible modificación del diseño del procesador segmentado Nios II/f de 6 etapas. La modificación propuesta es que la etapa "Ejecución / Cálculo de direcciones (E)" y la etapa de "Acceso a Memoria (M)" se fusionen en una sola etapa [2]. En esta etapa combinada, la ALU y la memoria funcionarán en paralelo. Las instrucciones de acceso a los datos utilizarán la memoria dejando la ALU inactiva, y las instrucciones aritmético-lógicas utilizarán la ALU dejando la memoria inactiva. Estos cambios son beneficiosos en términos de área y eficiencia energética.

En la arquitectura de repertorio de instrucciones Nios II, la dirección efectiva de una instrucción de carga o almacenamiento se calcula sumando el contenido de un registro (`rs1`) con un valor inmediato (`imm`). El problema con el nuevo diseño es que ahora no se realiza ningún cálculo de dirección en las instrucciones de carga o almacenamiento, ya que estas instrucciones no llegan a acceder a la ALU.

Los defensores del nuevo diseño del procesador Nios II abogan por cambiar la arquitectura del repertorio de instrucciones para permitir un solo modo de direccionamiento, el denominado *direccionamiento directo de registro*. En este modo de direccionamiento, solo se utiliza un registro fuente, y el valor que contiene es la dirección de memoria a la que se accede. No se puede especificar ningún desplazamiento con inmediatos.

En Nios II, la única forma de realizar el direccionamiento directo de registro es especificar la dirección en un registro y asignar el valor 0 al inmediato, `imm` = 0. Con el diseño propuesto, cualquier instrucción de carga o almacenamiento que utilice el direccionamiento registro-inmediato con `imm` != 0, necesitará dos instrucciones. En primer lugar, los valores del registro y del inmediato deben sumarse con una instrucción `addi`, y luego esta dirección calculada puede ser cargada o almacenada en la siguiente instrucción. Las instrucciones de carga y almacenamiento que actualmente utilizan un `inm`=0 no requerirán instrucciones adicionales en el nuevo diseño.

Preguntas:

15. Tu trabajo consiste en determinar el porcentaje de aumento en el número total de instrucciones que tendrían que ser ejecutadas bajo el nuevo diseño de cinco etapas. Esto requerirá un análisis más detallado de los diferentes tipos de cargas y almacenamientos ejecutados por la subrutina `PRODUCTO_ESCALAR` del programa benchmark benchNIOSII2021_Parte1 y que se analizó en la Parte 1 de esta Práctica 3.
16. Evalúa el nuevo diseño en función del aumento porcentual del número de instrucciones que tendrán que ser ejecutadas.
17. ¿Qué diseño aconsejarías a tu jefe de Intel que adopte? Justifica tu respuesta cuantitativamente.

**Referencias bibliográficas**

[1] Altera. Nios II Core Implementation Details. Altera, 2015. https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/nios2/n2cpu_nii51015.pdf

[2] Intel. Nios II Processor Reference Guide, Intel, 2020. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/nios2/n2cpu-nii5v1gen2.pdf