

PRÁCTICA DLXV-2: ACCESO A DATOS NO CONSECUTIVOS

Objetivos de la práctica



Asimilar el concepto de *strip-mining*, desarrollando por cuenta propia un método para implementarlo.

Observar el deterioro del rendimiento en los accesos a memoria cuando éstos no se realizan de forma secuencial. Poner estas observaciones en correspondencia con el funcionamiento teórico de la memoria en un procesador vectorial.

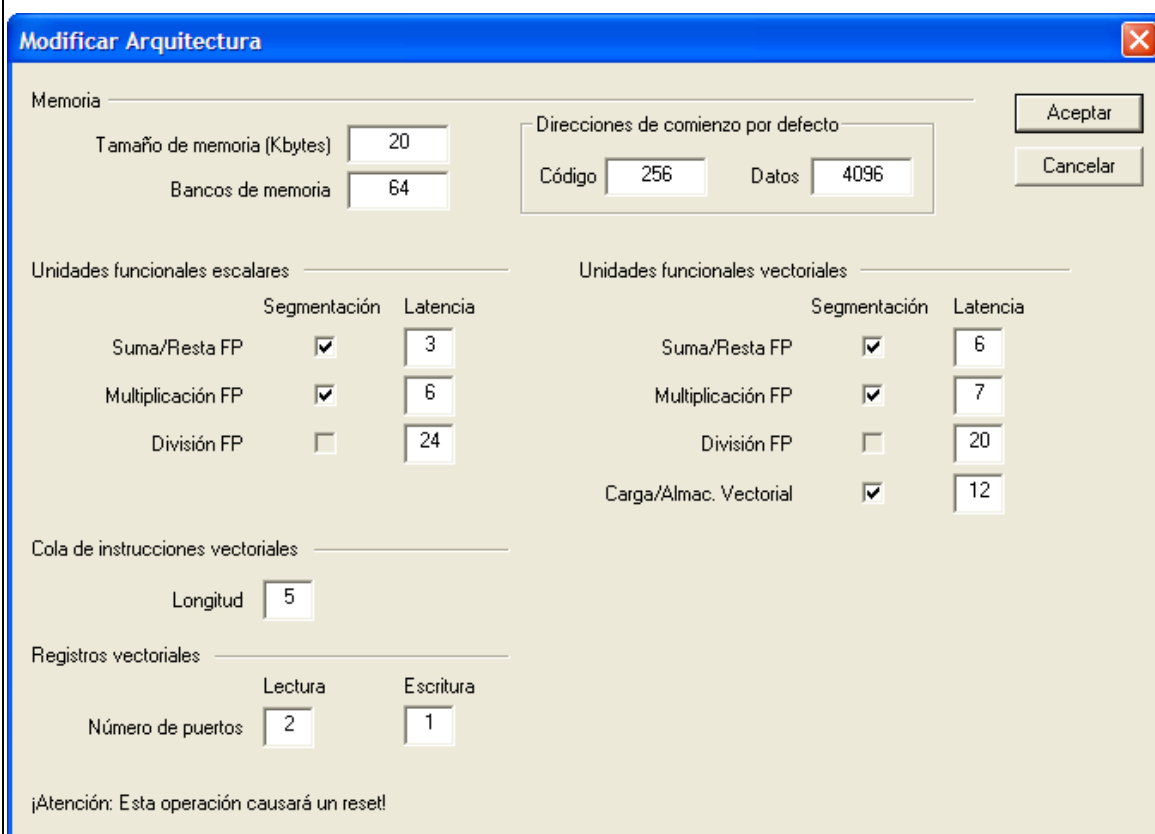
Arrancar y configurar el simulador



Arranca el simulador WinDLXV



Ve al menú Configuración/Arquitectura: Comprueba que sean estos los ajustes:



Modificar Arquitectura

Memoria

Tamaño de memoria (Kbytes)

Bancos de memoria

Direcciones de comienzo por defecto

Código Datos

Aceptar

Cancelar

Unidades funcionales escalares

	Segmentación	Latencia
Suma/Resta FP	<input checked="" type="checkbox"/>	<input type="text" value="3"/>
Multiplicación FP	<input checked="" type="checkbox"/>	<input type="text" value="6"/>
División FP	<input type="checkbox"/>	<input type="text" value="24"/>

Unidades funcionales vectoriales

	Segmentación	Latencia
Suma/Resta FP	<input checked="" type="checkbox"/>	<input type="text" value="6"/>
Multiplicación FP	<input checked="" type="checkbox"/>	<input type="text" value="7"/>
División FP	<input type="checkbox"/>	<input type="text" value="20"/>
Carga/Álmac. Vectorial	<input checked="" type="checkbox"/>	<input type="text" value="12"/>

Cola de instrucciones vectoriales

Longitud

Registros vectoriales





	Lectura	Escritura
Número de puertos	<input type="text" value="2"/>	<input type="text" value="1"/>

¡Atención: Esta operación causará un reset!






Activa en el menú Configuración las opciones “adelanto de resultados” y “encadenamiento vectorial”





Ejercicio 1: Procesado escalar en bucle

 <p>1 PASO</p>	<p>Abre en un editor el fichero edaxpy.s proporcionado con el simulador. Para evitar modificar el fichero, guárdalo de nuevo como edaxpy_strip.s y edita esta copia dejando intacto el fichero original.</p>
 <p>2 PASO</p>	<p>Utilizando el resumen de instrucciones dado en este guión (y los anexos y manuales anteriores, si fuera necesario) edita el programa para que procese 150 elementos, en vez de los 64 del programa original. Esto solo debe suponer cambios triviales en el programa.</p>
 <p>3 PASO</p>	<p>Ejecuta tu programa y comprueba su funcionamiento. Si no hace lo que se pide, reedita y prueba hasta que lo haga. Cuando lo consigas, guárdalo como edaxpy_strip.s y resérvalo para el tercer ejercicio de esta práctica</p>
	<p>Explica qué haces para conseguir el funcionamiento deseado. Le ponemos 150 elementos a cada vector para que, en vez de 64, trabaje con 150 elementos, lo que son 1200 bytes de memoria.</p>

Ejercicio 2: Acceso a una columna de una matriz

	<p>Partiendo de nuevo de stride.s edita el programa para que haga lo siguiente: Para evitar modificar el fichero, guárdalo de nuevo como stride_ej3.s y edita esta copia dejando intacto el fichero original.</p> <ul style="list-style-type: none"> • Tomando y como una matriz 4x4, guarda la primera columna en el registro vectorial v1 (piensa qué <i>stride</i> tienes que usar para acceder a los elementos de una columna). • Este resultado guárdalo en posiciones consecutivas en memoria, a partir de la posición dada por z.
	<p>Ejecuta tu programa y comprueba su funcionamiento. Si no hace lo que se pide, reedita y prueba hasta que lo haga. Cuando lo consigas, guárdalo como stride_ej3.s</p>
	<p>Explica qué haces para conseguir el funcionamiento deseado.</p> <p>Establecemos el stride a 32 para que podamos coger todos los elementos de la primera columna. Luego, copiamos el vector que contiene los elementos de la primera columna en la otra matriz, de manera que tengamos la primera columna de la primera matriz como primera fila de esta matriz.</p>

Ejercicio 3: Acceso todas las columnas de una matriz

 <p>1 PASO</p>	<p>Partiendo del programa del ejercicio 1, edítalo para que haga lo siguiente:</p> <ul style="list-style-type: none"> • Tomando y como una matriz 4x4, haz un bucle que guarde la primera columna en el registro vectorial v1 (piensa qué <i>stride</i> tienes que usar para acceder a los elementos de una columna). • En cada ejecución del bucle, guarda los datos descargados en posiciones consecutivas en memoria, a partir de la posición dada por z • Al final de la ejecución del programa tienen que estar en z todos los elementos de y, pero en el orden en que se han descargado. 						
 <p>2 PASO</p>	<p>Ejecuta tu programa y comprueba su funcionamiento. Si no hace lo que se pide, reedita y prueba hasta que lo haga. Cuando lo consigas, guárdalo como stride_ej4.s</p>						
	<p>Explica qué haces para conseguir el funcionamiento deseado.</p> <p>Cargamos los registros iniciales con los valores convenientes. A continuación, establecemos el stride a 32 y luego, tenemos la etiqueta "loop" que iterará sobre los elementos de la matriz de forma que irá cogiendo todos los elementos de la primera columna de la matriz y dejándolos en la primera fila de la otra matriz. Básicamente estamos haciendo la traspuesta de dicha matriz. Con el bucle (etiqueta "loop") vamos cogiendo columna a columna y como tenemos el stride establecido a 32, conseguimos copiar las columnas completas a la vez.</p>						
	<p>Anota las estadísticas:</p> <table border="1" data-bbox="284 1585 992 1706"> <tr> <td>Instrucciones</td><td>32</td></tr> <tr> <td>CPI</td><td>5344</td></tr> <tr> <td>Riesgos de Datos (RAW/WAW/WAR)</td><td>200</td></tr> </table>	Instrucciones	32	CPI	5344	Riesgos de Datos (RAW/WAW/WAR)	200
Instrucciones	32						
CPI	5344						
Riesgos de Datos (RAW/WAW/WAR)	200						