Laboratorio de Arquitectura e Ingeniería de Computadores

PRÁCTICA I

COMPUTADORES SEGMENTADOS (DLX)

Ingeniería de Computadores

Pedro Barquín Ayuso Miguel Ballesteros García 1.-Realizar un programa para el DLX que sume dos vectores de 10 componentes, sin usar bucles. Analizar los resultados estadísticos y las dependencias que producen detenciones en el cauce.

- Código:

```
data
                   .word 10,20,30,40,50,60,70,80,90,100
.word 100,90,80,70,60,50,40,30,20,10
.word 0,0,0,0,0,0,0,0,0
vector1:
vector2:
resultado:
.text
ini:
          addi r1,r0,vector1
          addi r2,r0,vector2
          addi r3,r0,resultado
          lw r4.0(r1)
                                        ;posición inicial (0) de r1 a r4
                                        ;posición inicial (0) de r2 a r5
          lw r5,0(r2)
                                       ;suma en r6 los registros r4 y r5
;r6 a la posición inicial (0) de r3
          add r6,r4,r5
          sw 0(r3),r6
          lw r4,4(r1)
lw r5,4(r2)
add r6,r4,r5
                                       ;posición siguiente (+4) de r1
;posición siguiente (+4) de r2
                                        ;suma
          sw 4(r3),r6
                                        ;r6 a la siguiente posicion de r3
          lw r4,8(r1)
lw r5,8(r2)
                                       ;posición siguiente (+4) de r1
;posición siguiente (+4) de r2
          add r6,r4,r5
                                        ;suma
          sw 8(r3),r6
                                        r6 a la siguiente posicion de r3
          lw r4,12(r1)
                                        ;posición siguiente (+4) de r1
          lw r5,12(r2)
                                        ;posición siguiente (+4) de r2
          add r6,r4,r5
                                        ;suma
          sw 12(r3),r6
                                        r6 a la siguiente posicion de r3
          lw r4,16(r1)
lw r5,16(r2)
                                        ;posición siguiente (+4) de r1
                                        ;posición siguiente (+4) de r2
                                         ;suma
          add r6, r4, r5
          sw 16(r3),r6
                                        r6 a la siguiente posicion de r3
          lw r4,20(r1)
lw r5,20(r2)
                                        ;posición siguiente (+4) de r1;posición siguiente (+4) de r2
          add r6,r4,r5
                                         suma
          sw 20(r3),r6
                                         ;r6 a la siguiente posicion de r3
```

```
lw r4,24(r1)
lw r5,24(r2)
                          ;posición siguiente (+4) de r1
                           ;posición siguiente (+4) de r2
add r6,r4,r5
                           ;suma
sw 24(r3),r6
                           r6 a la siguiente posicion de r3
lw r4,28(r1)
                           ;posición siguiente (+4) de r1
                          posición siguiente (+4) de r2
lw r5,28(r2)
                           ;suma
add r6,r4,r5
sw 28(r3),r6
                           r6 a la siguiente posicion de r3
lw r4,32(r1)
lw r5,32(r2)
add r6,r4,r5
                          ;posición siguiente (+4) de r1
                           ;posición siguiente (+4) de r2
                           ;suma
sw 32(r3),r6
                           r6 a la siguiente posicion de r3
                          ;posición siguiente (+4) de r1
;posición siguiente (+4) de r2
lw r4,36(r1)
lw r5,36(r2)
add r6,r4,r5
                           suma
sw 36(r3),r6
                           r6 a la siguiente posicion de r3
```

trap #6

-Estadisticas:

HARDWARE:

Memoria: 65536 bytes, distribuida en 16 bancos.

Unidades escalares en Coma Flotante

1 unidad(es) de suma/resta, latencia = 2 ciclos 1 unidad(es) de división, latencia = 19 ciclos 1 unidad(es) de multiplicación, latencia = 5 ciclos

Unidades vectoriales

1 unidad(es) de suma/resta, latencia = 6 ciclos 1 unidad(es) de división, latencia = 20 ciclos 1 unidad(es) de multiplicación, latencia = 7 ciclos 1 unidad(es) de carga/almac., latencia = 12 ciclos 1 unidad(es) logica(s), latencia = 1 ciclos

2/1 puertos de lectura/escritura en los regs. vectoriales

CONTADORES DE CICLOS:

Ciclos de parada por dependencias:

Entre instr. escalares = 10 Entre instr. de coma flotante = 0 Entre instr. vectoriales = 0

Tiempo de ejecución del programa = 54 ciclos

INSTRUCCIONES DE SALTO:

No se han ejecutado instrucciones de salto.

OPERACIONES PENDIENTES:

ninguna.

Registro vectorial: 0 1 2 3 4 5 6 7 puertos Lectura: 0 0 0 0 0 0 0 0 0 0 0 puertos Escritura: 0 0 0 0 0 0 0 0 0

INSTRUCCIONES EJECUTADAS:

OPERACI	ONES	ENTERAS

			===:	====	====	===	===				
ADD AND BFPT JALR LD LHU MOVFP2I OR SEQUI SGEU SGEU SGEU SLEI SLLI SLTUI SNEUI SRLI SUBUI	10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		ADD: AND: BNEZ JR LF LW MOV: SEQ SET! SCE! SCE! SIE! SNE SRA SUB SW	I Z I2FP JI JI	3 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0	===	===	ADDU BEQZ J LB LH MOVD MOVI2S RFE SEQI SGE SGT SH SLEUI SLTI SNEI SRAI SUBI TRAP	000000000000000000000000000000000000000	ADDUI BFPF JAL LBU LHI MOVF MOVS2I SB SEQU SGTI SGTI SLE SLTU SNEU SRL SUBU XOR	000000000000000000000000000000000000000
XORI	Ŏ.		NOP		0				_	MOR	٠
Número	total	de	op.	ente	eras	=	44	ŀ			

OPERACIONES EN COMA FLOTANTE

ADDD	0	ADDF	0	CVTD2F	0	CVTD2I	0
CVTF2D	0	CVTF2I	0	CVTI2D	0	CVTI2F	0
DIA	0	DIVD	0	DIVF	0	DIVU	0
EQD	0	EQF	0	GED	0	GEF	0
GTD	0	GTF	0	LED	0	LEF	0
LTD	0	LTF	0	MULT	0	MULTD	0
MULTF	0	\mathtt{MULTU}	0	NED	0	NEF	0
SHBD	Π	SHRF	Π				

Número total de op. en coma flotante = 0

OPERACIONES VECTORIALES

ADDSV	0	ADDV	0	CVI	0	CVM	0
DIVSV	0	DIVV	0	DIVVS	0	ΓΛ	0
LVI	0	LVWS	0	MOVF2S	0	MOVS2F	0
MULTSV	0	MULTV	0	POP	0	SEQSV	0
SEQV	0	SGESV	0	SGEV	0	SGTSV	0
SGTV	0	SLESV	0	SLEV	0	SLTSV	0
SLTV	0	SNESV	0	SNEV	0	SUBSV	0
SUBV	0	SUBVS	0	SV	0	SVI	0
SVUS	ñ		_		_		_

Número total de op. vectoriales = 0

Número total de operaciones = 44 Número total de ciclos = 54 Como hemos visto en la práctica, la operación suma debe esperar a que el dato sea leído de memoria. Debido a esto podemos explicar los 10 parones que vemos reflejados en las estadísticas porque realizamos 10 sumas, una por cada componente del vector.

2.- Modificar el programa anterior para evitar las detenciones, comprobar los resultados.

-Código:

```
.data
                  .word 10,20,30,40,50,60,70,80,90,100
vector1:
                   .word 100,90,80,70,60,50,40,30,20,10
vector2:
                  .word 0,0,0,0,0,0,0,0,0,0
resultado:
.text
ini:
         addi r1,r0,vector1
         addi r2,r0,vector2
         addi r3,r0,resultado
                                      posición inicial (0) de r1 a r4 posición inicial (0) de r2 a r5
         lw r4.0(r1)
         lw r5.0(r2)
lw r7.4(r1)
                                      ;posición siguiente (+4) de r1
;posición siguiente (+4) de r2
         lw r8,4(r2)
         add r6,r4,r5
                                      ;suma en r6 los registros r4 y r5
         sw 0(r3),r6
                                      ;r6 a la posición inicial (0) de r3
         add r6,r7,r8
                                       ;suma
         sw 4(r3),r6
                                      ;r6 a la siguiente posicion de r3
                                    ;posición siguiente (+4) de r1
;posición siguiente (+4) de r2
;posición siguiente (+4) de r1
         lw r4,8(r1)
lw r5,8(r2)
         lw r7,12(r1)
                                      ;posición siguiente (+4) de r2
         lw r8,12(r2)
                                      ;suma
         add r6,r4,r5
         sw 8(r3),r6
                                      r6 a la siguiente posicion de r3
         add r6,r7,r8
                                      ;suma
         sw 12(r3),r6
                                      ;r6 a la siguiente posicion de r3
         lw r4,16(r1)
                                     posición siguiente (+4) de r1
         lw r5,16(r2)
                                      posición siguiente (+4) de r2
         lw r7,20(r1)
lw r8,20(r2)
                                      ;posición siguiente (+4) de r1
;posición siguiente (+4) de r2
         add r6,r4,r5
                                      suma
         sw 16(r3), r6
                                      r6 a la siguiente posicion de r3
         add r6,r7,r8
                                      ;suma
         sw 20(r3),r6
                                      r6 a la siguiente posicion de r3
```

```
lw r4,24(r1)
lw r5,24(r2)
lw r7,28(r1)
lw r8,28(r2)
                              ;posición siguiente (+4) de r1
                              posición siguiente (+4) de r2
                              ;posición siguiente (+4) de r1
;posición siguiente (+4) de r2
add r6,r4,r5
                               ;suma
                               ;r6 a la siguiente posicion de r3
sw 24(r3),r6
add r6,r7,r8
                                ;suma
sw 28(r3),r6
                               ;r6 a la siguiente posicion de r3
                              ;posición siguiente (+4) de r1
;posición siguiente (+4) de r2
;posición siguiente (+4) de r1
lw r4,32(r1)
lw r5,32(r2)
lw r7,36(r1)
lw r8,36(r2)
                               ;posición siguiente (+4) de r2
                               ;suma
add r6,r4,r5
sw 32(r3),r6
                               ;r6 a la siguiente posicion de r3
add r6,r7,r8
                               :suma
sw 36(r3),r6
                               r6 a la siguiente posicion de r3
```

trap #6

-Estadisticas:

HARDWARE:

Memoria: 65536 bytes, distribuida en 16 bancos.

Unidades escalares en Coma Flotante

1 unidad(es) de suma/resta, latencia = 2 ciclos 1 unidad(es) de división, latencia = 19 ciclos 1 unidad(es) de multiplicación, latencia = 5 ciclos

Unidades vectoriales

1 unidad(es) de suma/resta, latencia = 6 ciclos 1 unidad(es) de división, latencia = 20 ciclos 1 unidad(es) de multiplicación, latencia = 7 ciclos 1 unidad(es) de carga/almac., latencia = 12 ciclos 1 unidad(es) logica(s), latencia = 1 ciclos

2/1 puertos de lectura/escritura en los regs. vectoriales

CONTADORES DE CICLOS:

Ciclos de parada por dependencias:

```
Entre instr. escalares = 0
Entre instr. de coma flotante = 0
Entre instr. vectoriales = 0
```

Tiempo de ejecución del programa = 44 ciclos

INSTRUCCIONES DE SALTO:

No se han ejecutado instrucciones de salto.

OPERACIONES PENDIENTES:

ninguna.

Registro vectorial: 0 1 2 3 4 5 6 puertos Lectura: 0 0 0 0 0 0 0 0 puertos Escritura: 0 0 0 0 0 0 0 Registro vectorial: puertos Lectura: Ö

INSTRUCCIONES EJECUTADAS:

O	PERACIONES.	ENTERAS

ADD AND BFPT JALR LD LHU MOVFP2I OR SEQUI SGEU SGEU SGEU SLEI SLEI SLII SNEUI SRLI	10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	ADDI ANDI BNEZ JR LF LW MOVI2FP ORI SEQ SF SGEUI SGTUI SLEU SLEU SIT SNE SRA SUB	3 0 0 0 0 2 0 0 0 0 0 0 0	ADDU BEQZ J LB LH MOVD MOVI2S RFE SEQI SGE SGT SH SLEUI SITI SNEI SRAI SUBI	000000000000000000000000000000000000000	ADDUI BFPF JAL LBU LHI MOVF MOVS2I SB SEQU SGEI SGEI SLE SLE SLE SLE SLE SLE SLE SLE SLE SLE	000000000000000000000000000000000000000
SUBUI XORI	0 0	SW NOP	10 0	TRAP	1	XOR	0

Número total de op. enteras = 44

OPERACIONES EN COMA FLOTANTE

______ ADDD ADDF CVTD2F 0 0

CVTD2I CVTF2D 0 CVTF2I 0 CVTI2D 0 CVTI2F 0 DIVF DIVU DIVD 0 DIV 0 0 0 GED LED GEF LEF EQD 0 EQF 0 0 0 GTF GTD 0 0 0 0 LTF LTD 0 MULT 0 MULTD 0 0 MULTU MULTF 0 0 0 NED 0 NEF SUBD SUBF 0

Número total de op. en coma flotante = 0

OPERACIONES VECTORIALES ______

ADDSV DIVSV	0 0	ADDV DIVV	0 0	CVI DIVVS	0 0	CVM LV	0
LVI	Ō	LVWS	Ō	MOVF2S	Ō	MOVS2F	Ō
MULTSV	0	MULTV	0	POP	0	SEQSV	0
SEQV	0	SGESV	0	SGEV	0	SGTSV	0
SGTV	0	SLESV	0	SLEV	0	SLTSV	0
SLTV	0	SNESV	0	SNEV	0	SUBSV	0
SUBV	0	SUBVS	0	SV	0	SVI	0
SMMS	Ω						

Número total de op. vectoriales = 0

Número total de operaciones = 44 = 44 Número total de ciclos

Los parones se efectuaban porque no dábamos suficiente tiempo para que el dato fuera leído de memoria, lo que hacemos es traer más datos para que la primera suma tenga tiempo suficiente para tener los datos que necesita. De este modo organizamos el código en pares de sumas y evitamos todos los parones.

3.-Repetir la operación usando un bucle para la suma de componentes.

-Código:

```
.word 10,20,30,40,50,60,70,80,90,100
.word 100,90,80,70,60,50,40,30,20,10
.word 0,0,0,0,0,0,0,0,0
vector1:
vector2
resultado:
 .text
          addi r1.r0,#0   ;registro usado para controlar las posiciones a operar en cada ciclo addi r2.r0,#10   ;contador bucle
bucle:
          lw r4.vector1(r1)
          lw r5, vector2(r1)
          add r6,r4,r5
          sw resultado(r1),r6
          addi r1,r1,#4
subi r2,r2,#1
bnez r2,bucle
                            ; disminuimos el número de iteraciones del bucle ;
                             ;aumentamos el PC
          nop
trap #6
-Estadisticas:
HARDWARE:
Memoria: 65536 bytes, distribuida en 16 bancos.
```

```
Unidades escalares en Coma Flotante
1 unidad(es) de suma/resta,
                                    latencia = 2 ciclos
```

1 unidad(es) de división, latencia = 19 ciclos 1 unidad(es) de multiplicación, latencia = 5 ciclos

Unidades vectoriales

```
latencia = 6 ciclos
1 unidad(es) de suma/resta,
1 unidad(es) de división,
1 unidad(es) de multiplicación,
1 unidad(es) de carga/almac.,
                                               latencia = 20 ciclos
                                                               7 ciclos
                                                latencia =
                                               latencia = 12 ciclos
1 unidad(es) logica(s),
                                               latencia = 1 ciclos
```

2/1 puertos de lectura/escritura en los regs. vectoriales

CONTADORES DE CICLOS:

Ciclos de parada por dependencias:

```
= 20
Entre instr. escalares
Entre instr. de coma flotante
Entre instr. vectoriales
```

Tiempo de ejecución del programa = 103 ciclos

INSTRUCCIONES DE SALTO:

Total 10, tomados 9 (90.00%), no tomados 1 (10.00%)

OPERACIONES PENDIENTES:

ninguna.

Registro vectorial: 0 1 2 3 4 5 6 7 puertos Lectura: 0 0 0 0 0 0 0 0 0 0 0 puertos Escritura: 0 0 0 0 0 0 0 0 0

INSTRUCCIONES EJECUTADAS:

OPERACIONES ENTERAS

		OPERACIO	ONES ENT				
ADD	10	ADDI	12	ADDU	0	ADDUI	0
AND	0	ANDI	0	BEQZ	0	BFPF	0
BFPT	0	BNEZ	10	J	0	JAL	0
JALR	0	JR	0	LB	0	LBU	0
LD	0	LF	0	LH	0	LHI	0
LHU	0	LW	20	MOVD	0	MOVF	0
MOVFP2I	0	MOVI2FP	0	MOVI2S	0	MOVS2I	0
ΩR	0	ORI	0	RFE	0	SB	0
SD	0	SEQ	0	SEQI	0	SEQU	0
SEQUI	0	SF	0	SGE	0	SGEI	0
BGEU BOTH	0	SGEUI	0	SGT	0	SGTI	0
SGTU Strt	0	SGTUI	0	SH	0 0	SLE	0
SLEI SLLI	0	SLEU SLT	0	SLEUI SLTI	0	SLL SLTU	0
SLTUI	0	SNE	0	SNEI	0	SNEU	0
SNEUI	Ö	SRA	0	SRAI	Ö	SRL	ŏ
SRLI	ŏ	SUB	Õ	SUBI	10	SUBU	ŏ
SUBUI	ň	SW	10	TRAP	1	XOR	ŏ
XORI	ŏ	NOP	10		_		-
	total de			13			

OPE	CRA	C	Ι	O.	NΙ	35	•	Ε	Ν		C	0	M	A		F	L	0	T	A	Ν	T	Ε
===	===	=	=	=	= =	= =	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=

Número total de op. en coma flotante = 0

OPERACIONES VECTORIALES

ADDSV	0	ADDV	0	CAI	0	CVM	0
DIVSV	0	DIVV	0	DIVVS	0	LV	0
LVI	0	LVWS	0	MOVF2S	0	MOVS2F	0
MULTSV	0	MULTV	0	POP	0	SEQSV	0
SEQV	0	SGESV	0	SGEV	0	SGTSV	0
SGTV	0	SLESV	0	SLEV	0	SLTSV	0
SLTV	0	SNESV	0	SNEV	0	SUBSV	0
SUBV	0	SUBVS	0	SV	0	SVI	0
SVWS	0						

Número total de op. vectoriales = 0

Número total de operaciones = 83 Número total de ciclos = 103 Utilizamos dos registros uno para controlar las iteraciones del bucle (r2) y otro para controlar la posición del vector(r1). Sumamos un elemento de cada vector como en los ejercicios anteriores, almacenándolo en resultado, y después aumentamos la posición de r1 y restamos una iteración del bucle. Aunque en la implementación hemos utilizado un vector para controlar el bucle no sería necesario, ya que se podría realizar utilizando el vector r1 para controlar el bucle con lo que el numero de ciclos de reduciría en 9.

4.- Optimizar el programa del apartado anterior, comprobar resultados.

-Código:

```
data
                  .word 10,20,30,40,50,60,70,80,90,100
.word 100,90,80,70,60,50,40,30,20,10
.word 0,0,0,0,0,0,0,0,0
vector1:
vector2:
resultado:
.text
ini:
           addi r1,r0,#0
                                           ;vector que controla la posicion dentro de un vector
;vector que controla la posicion dentro de un vector
;control de bucle
           addi r3,r0,#4
bucle:
           lw r4, vector1(r1)
           lw r5, vector2(r1)
           lw r6, vector1(r2)
           lw r7, vector2(r2)
           add r8,r4,r5
           add r9,r6,r7
           sw resultado(r1).r8
sw resultado(r2).r9
           addi r1,r1,#8
           addi r2,r2,#8
           bnez r3, bucle
           subi r3,r3,#1
trap #6
```

-Estadisticas:

HARDWARE:

Memoria: 65536 bytes, distribuida en 16 bancos.

Unidades escalares en Coma Flotante

1 unidad(es) de suma/resta, latencia = 2 ciclos 1 unidad(es) de división, latencia = 19 ciclos 1 unidad(es) de multiplicación, latencia = 5 ciclos

Unidades vectoriales

1 unidad(es) de suma/resta, latencia = 6 ciclos 1 unidad(es) de división, latencia = 20 ciclos 1 unidad(es) de multiplicación, latencia = 7 ciclos 1 unidad(es) de carga/almac., latencia = 12 ciclos 1 unidad(es) logica(s), latencia = 1 ciclos

2/1 puertos de lectura/escritura en los regs. vectoriales

CONTADORES DE CICLOS:

Ciclos de parada por dependencias:

Entre instr. escalares = 0 Entre instr. de coma flotante = 0 Entre instr. vectoriales = 0

Tiempo de ejecución del programa = 64 ciclos

OPERACIONES PENDIENTES: ninguna. 0 3 Registro vectorial: 1 0 0 puertos Lectura: 0 puertos Escritura: INSTRUCCIONES EJECUTADAS: OPERACIONES ENTERAS -----ADD 13 10 ADDI ADDU AND 0 0 ANDI BEQZ 5 BFPT 0 BNEZ J JALR 0 JR 0 LB LD 0 LF 0 LH LHU 0 LW 20 MOVD MOVI2S MOVFP2I 0 MOVI2FP 0 0 ORI 0 OR RFE SD 0 SEQ 0 SEQI SEQUI SF SGE 0 0 SGEUI SGT SGEU 0 0 SGTU 0 SGTUI 0 SH SLEI SLEU SLEUI 0 0 SLLI 0 SLT 0 SLTI SLTUI 0 SNE 0 SNEI SNEUI 0 SRA 0 SRAI SRLI SUB SUBI 0 0 SUBUI SW 10 TRAP 0 NOP 0 XORI Número total de op. enteras = 64

			OPERAC	ONES E	N C	OMA FLO	OTANTE		
ADDD CVTF2D DIV EQD GTD LTD MULTF SUBD Número	0 0 0 0 0 0 0 0 total	de	ADDF CVTF2I DIVD EQF GTF LTF MULTU SUBF Op. en	0 0 0 0 0 0 0 0 0) (]]	CVTD2F CVTI2D DIVF GED LED MULT NED	0 0 0 0 0 0	CVTD2I CVTI2F DIVU GEF LEF MULTD NEF	0 0 0 0 0
				ONES V			_		
ADDSV DIVSV LVI MULTSV SEQV SGTV SLTV SUBV SVWS Número	0 0 0 0 0 0 0 0 0 total	de	ADDV DIVV LVWS MULTV SGESV SLESV SNESV SUBVS	0 0 0 0 0 0 0 0 0		CVI DIVVS MOVF2S POP BGEV BLEV BNEV BV	0 0 0 0 0 0	CVM LV MOVS2F SEQSV SGTSV SLTSV SUBSV SVI	0 0 0 0 0 0
Número Número			operac: ciclos	iones	:	= 64 = 64			

7

Ò

0

ADDUI

BFPF

JAL

LBU

LHI

SB

MOVF

SEQU

SGEI

SGTI

SLE

SLL

SLTU

SNEU

SUBU

SRL

XOR

MOVS2I

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

5

Para optimizar el ejercicio anterior utilizamos un nuevo registro para controlar la posición del vector y guardar en una determinada posición con un registro y en la

siguiente con el otro, pudiendo hacer así dos sumas seguidas. También hacemos lo mismo que en apartados anteriores cargando más datos en memoria y utilizamos el el hueco de retardo vacío que hay después de la instrucción de salto. Aunque en la implementación hemos utilizado un vector para controlar el bucle no sería necesario, ya que se podría realizar utilizando el vector r1 o r2 indistintamente.