

Laboratorio de Arquitectura e Ingeniería de
Computadores

PRÁCTICA II

Planificación Estática de Instrucciones

Ingeniería de Computadores

Pedro Barquín Ayuso
Miguel Ballesteros García

[illegible]

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	= 240
Entre instr. vectoriales	= 0

Tiempo de ejecución del programa = 565 ciclos

INSTRUCCIONES DE SALTO:

Total 16, tomados 15 (93.75%), no tomados 1 (6.25%)

OPERACIONES PENDIENTES:

ninguna.

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

INSTRUCCIONES EJECUTADAS:

OPERACIONES ENTERAS =====

ADD	1	ADDI	17	ADDU	0	ADDUI	0
AND	0	ANDI	0	BEQZ	0	BFPF	0
BFPT	0	BNEZ	16	J	0	JAL	0
JALR	0	JR	0	LB	0	LBU	0
LD	129	LF	0	LH	0	LHI	0
LHU	0	LW	0	MOVD	0	MOVF	0
MOVFP2I	0	MOVI2FP	0	MOVI2S	0	MOVS2I	0
OR	0	ORI	0	RFE	0	SB	0
SD	16	SEQ	0	SEI	0	SEQU	0
SEQUI	0	SF	0	SGE	0	SGEI	0
SGEU	0	SGEUI	0	SGT	0	SGTI	0
SGTU	0	SGTUI	0	SH	0	SLE	0
SLEI	0	SLEU	0	SLEUI	0	SLL	0
SLI	0	SLT	0	SLTI	0	SLTU	0
SLTUI	0	SNE	0	SNEI	0	SNEU	0
SNEUI	0	SRA	0	SRAI	0	SRL	0
SRLI	0	SUB	1	SUBI	16	SUBU	0
SUBUI	0	SW	0	TRAP	1	XOR	0
XORI	0	NOP	0				

Número total de op. enteras = 197

OPERACIONES EN COMA FLOTANTE

=====

ADDD	64	ADDF	0	CVTD2F	0	CVTD2I	0
CVTF2D	0	CVTF2I	0	CVTI2D	0	CVTI2F	0
DIV	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	64
MULTF	0	MULTU	0	NED	0	NEF	0
SUBD	0	SUBF	0				

Número total de op. en coma flotante = 128

OPERACIONES VECTORIALES

=====

ADDSV	0	ADDV	0	CVI	0	CVM	0
DIVSV	0	DIVV	0	DIVVS	0	LV	0
LVI	0	LVWS	0	MOVFS2S	0	MOVFS2F	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 = 325

Número total de ciclos = 565

Sabiendo que el sistema DLX tiene 32 registros, por lo tanto al trabajar en doble precisión (8bytes) solo disponemos de 16 registros dobles. Al trabajar con vectores de 64 componentes y tener dos vectores (Vector1 y Vector2) tenemos un total de 124 componentes por lo que el mejor factor de desenrollado del código seria de 16, lo cual es $124/8=16$ iteraciones. Usamos 8 registros en coma flotante, doble, para cargar 4 datos de cada uno de los vectores, utilizamos 4 registros mas para guardar los resultados de las multiplicaciones y un último registro para guardar el resultado final dado que el enunciado pide el producto escalar de los dos vectores.

[illegible]

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 = 278 ciclos

INSTRUCCIONES DE SALTO:

Total 11, tomados 10 (90.91%), no tomados 1 (9.09%)

vectorb:	0	vectorb+0x9c:	1
vectorb+0x4:	0	vectorb+0xa0:	0
vectorb+0x8:	0	vectorb+0xa4:	0
vectorb+0xc:	0	vectorb+0xa8:	0
vectorb+0x10:	1	vectorb+0xac:	0
vectorb+0x14:	0	vectorb+0xb0:	1
vectorb+0x18:	0	vectorb+0xb4:	0
vectorb+0x1c:	0	vectorb+0xb8:	0
vectorb+0x20:	0	vectorb+0xbc:	0
vectorb+0x24:	1	vectorb+0xc0:	0
vectorb+0x28:	0	vectorb+0xc4:	1
vectorb+0x2c:	0	vectorb+0xc8:	0
vectorb+0x30:	0	vectorb+0xcc:	0
vectorb+0x34:	0	vectorb+0xd0:	0
vectorb+0x38:	1	vectorb+0xd4:	0
vectorb+0x3c:	0	vectorb+0xd8:	1
vectorb+0x40:	0	vectorb+0xdc:	0
vectorb+0x44:	0	vectorb+0xe0:	0
vectorb+0x48:	0	vectorb+0xe4:	0
vectorb+0x4c:	1	vectorb+0xe8:	0
vectorb+0x50:	0	vectorb+0xec:	1
vectorb+0x54:	0	vectorb+0xf0:	0
vectorb+0x58:	0	vectorb+0xf4:	0
vectorb+0x5c:	0	vectorb+0xf8:	0
vectorb+0x60:	1	vectorb+0xfc:	0
vectorb+0x64:	0	vectorb+0x100:	1
vectorb+0x68:	0	vectorb+0x104:	0
vectorb+0x6c:	0	vectorb+0x108:	0
vectorb+0x70:	0	vectorb+0x10c:	0
vectorb+0x74:	1	vectorb+0x110:	0
vectorb+0x78:	0	vectorb+0x114:	0
vectorb+0x7c:	0	vectorb+0x118:	0
vectorb+0x80:	0	vectorb+0x11c:	0
vectorb+0x84:	0	vectorb+0x120:	0
vectorb+0x88:	0	vectorb+0x124:	0
vectorb+0x8c:	0	vectorb+0x128:	1
vectorb+0x90:	0	vectorb+0x12c:	0
vectorb+0x94:	0	vectorb+0x130:	0
vectorb+0x98:	0		

En este caso el factor de desenrollado del bucle es de 7, $77/7=11$ iteraciones, utilizamos 7 registros para almacenar 7 valores del vector, utilizamos el comando **seq** para poner a 1 los componentes del vector b si la componentes correspondiente de a tiene valor 0, y 0 en el caso contrario. Guardamos los resultados obtenidos en el vector b.

3.- Aplicar desenrollado y reordenación a un bucle que calcula la suma de las componentes de un vector A de 100 componentes.

```
.data
vector1:      .word 1,1,1,1,1,1,1,1,1,1,1
               .word 1,1,1,1,1,1,1,1,1,1,1
               .word 1,1,1,1,1,1,1,1,1,1,1
               .word 1,1,1,1,1,1,1,1,1,1,1
               .word 1,1,1,1,1,1,1,1,1,1,1
               .word 1,1,1,1,1,1,1,1,1,1,1
               .word 1,1,1,1,1,1,1,1,1,1,1
               .word 1,1,1,1,1,1,1,1,1,1,1
               .word 1,1,1,1,1,1,1,1,1,1,1
               .word 1,1,1,1,1,1,1,1,1,1,1
resultado:    .word 0

.text
ini:          addi r1, r0, #10
               addi r2, r0, #0

bucle:        lw r12, vector1(r2)
               lw r13, vector1+4(r2)
               lw r14, vector1+8(r2)
               lw r15, vector1+12(r2)
               lw r16, vector1+16(r2)
               lw r17, vector1+20(r2)
               lw r18, vector1+24(r2)
               lw r19, vector1+28(r2)
               lw r20, vector1+32(r2)
               lw r21, vector1+36(r2)
               lw r22, vector1+40(r2)

               subi r1, r1, #1

               add r23, r23, r12
               add r23, r23, r13
               add r23, r23, r14
               add r23, r23, r15
               add r23, r23, r16
               add r23, r23, r17
               add r23, r23, r18
               add r23, r23, r19
               add r23, r23, r20
               add r23, r23, r21

               bnez r1, bucle
               sw resultado(r0), r23
               addi r2, r2, #40

trap #6
```

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 = 244 ciclos

INSTRUCCIONES DE SALTO:

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

 **ExaminarOT(resultado 1d) - Exaot.tmp**
resultado: 100

Para este ejercicio el factor de desenrollado es de 10 , lo que es $100/10=10$, almacenamos las 10 primeras componentes en los registros, para después ir realizando la suma de una en una hasta completar el ciclo y guardar el resultado en el vector resultado.