UNIVERSIDAD CARLOS III DE MADRID

Tema 10. Optimización

Departamento de Ingeniería de Sistemas y Automática

RAÚL PÉRULA MARTÍNEZ
LUIS ENRIQUE MORENO LORENTE
ALBERTO BRUNETE GONZALEZ
CESAR AUGUSTO ARISMENDI GUTIERREZ
DOMINGO MIGUEL GUINEA GARCIA ALEGRE
JOSÉ CARLOS CASTILLO MONTOYA

Universidad Carlos III de Madrid

Ejercicio 1

Suponiendo que se tiene el siguiente código en un cierto lenguaje ensamblador:

```
          ld    f2, A
          add   r2, r0, #4
          add   r1, r0, DIR_X
1   L1:   ld    f0, 0(r1)
2         ld    f6, -8(r1)
3         multd f4, f6, f2
4         addd  f6, f4, f2
5         addd  f4, f0, f2
6         sd    0(r1), f4
7         sd    -8(r1), f6
8         add   r1, r1, #16
9         sub   r2, r2, #16
10        bnez  r2, L1
```

Donde se lista en primer lugar el operando de destino y a continuación los operandos fuente, y en el caso de las instrucciones de salto se listan primero los operandos a ser comparados y después la dirección de salto.

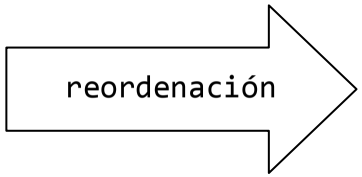
Se pide:

1. Realice un desenrollado del ciclo de forma que 2 iteraciones del ciclo actual correspondan a 1 en el código desenrollado, reordénelo y muestre el cronograma.
2. Suponiendo que el procesador utiliza la técnica de salto retardado, reordene el código y muestre el cronograma de ejecución.

Solución

1. Desenrollado y reordenación:

```
ld      f0, 0(r1)
ld      f6, -8(r1)
multd   f4, f6, f2
addd    f6, f4, f2
addd    f4, f0, f2
sd      0(r1), f4
sd      -8(r1), f6
ld      f8, 16(r1)
ld      f10, 8(r1)
multd   f12, f10, f2
addd    f10, f12, f2
addd    f2, f8, f2
sd      16(r1), f12
sd      8(r1), f10
add     r1, r1, #32
sub     r2, r2, #32
bnez    r2, L1
```




```
ld      f0, 0(r1)
ld      f6, -8(r1)
ld      f8, 16(r1)
ld      f10, 8(r1)
multd   f4, f6, f2
addd    f6, f4, f2
addd    f4, f0, f2
multd   f12, f10, f2
addd    f10, f12, f2
addd    f2, f8, f2
sd      0(r1), f4
sd      -8(r1), f6
sd      16(r1), f12
sd      8(r1), f10
add     r1, r1, #32
sub     r2, r2, #32
bnez    r2, L1
```

Reorganizando el código anterior, el cronograma será el siguiente:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
ld	IF	ID	EX	MEM	WB																						
ld		IF	ID	EX	MEM	WB																					
ld			IF	ID	EX	MEM	WB																				
ld				IF	ID	EX	MEM	WB																			
multd					IF	ID	EX	MEM	WB																		
addd						IF	•	•	ID	EX	MEM	WB															
addd									IF	ID	EX	MEM	WB														
multd										IF	ID	EX	MEM	WB													
add											IF	•	•	ID	EX	MEM	WB										
add														IF	ID	EX	MEM	WB									
sd															IF	ID	EX	MEM	WB								
sd																IF	ID	EX	MEM	WB							
sd																	IF	ID	EX	MEM	WB						
sd																		IF	ID	EX	MEM	WB					
add																			IF	ID	EX	MEM	WB				
sub																				IF	ID	EX	MEM	WB			
bnez																					IF	•	•	ID	EX	MEM	WB

2. Salto retardado (ventana de retardo de 2 instrucciones):

Reordenación.

```
ld      f0, 0(r1)
ld      f6, -8(r1)
multd   f4, f6, f2
addd    f6, f4, f2
addd    f4, f0, f2
sd      0(r1), f4
sub     r2, r2, #16
bnez    r2, L1
--------------------
sd      -8(r1), f6
add     r1, r1, #16
```

ventana

retardo

Cronograma.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
ld	IF	ID	EX	MEM	WB														
ld		IF	ID	EX	MEM	WB													
multd			IF	•	•	ID	EX	MEM	WB										
addd						IF	•	•	ID	EX	MEM	WB							
addd									IF	ID	EX	MEM	WB						
sub										IF	ID	EX	MEM	WB					
sd											IF	•	ID	EX	MEM	WB			
bnez													IF	ID	EX	MEM	WB		
sd														IF	ID	EX	MEM	WB	
add															IF	ID	EX	MEM	WB

Ejercicio 2

El siguiente fragmento de código se ejecuta en un DLX dotado de planificación dinámica de instrucciones:

```
1  loop: ld    f2, 0(r1)
2        multd f4, f2, f0
3        ld    f6, 0(r2)
4        addd  f6, f4, f6
5        sd    0(r2), f6
6        addi  r1, r1, #8
7        addi  r2, r2, #8
8        sgti  r3, r1, done
9        beqz  r3, loop
10       nop
```

1. Suponiendo que:

   - La planificación dinámica emplea el **método del Scoreboard**.
   - Un dato no se puede escribir en un registro y leer su valor en el mismo ciclo.
   - Los saltos se resuelven en la etapa de ejecución.
   - Las operaciones de carga/almacenamiento, saltos y *nops* se lleva a cabo en las unidades funcionales enteras.
   - Se emplea un *branchdelay slot* de una instrucción.
   - Las unidades funcionales tienen las siguientes características:

UF	Cantidad	Latencia	Segmentación
FP ADD	1	4	NO
FP MUL	1	5	NO
INT ALU	2	1	NO

   a. Mostrar el cronograma de ejecución. Calcular el CPI.
   b. Planificar el código para que el CPI sea mínimo.

2. Suponiendo que:

   - La planificación dinámica emplea el **método de Tomasulo**.
   - Los saltos se resuelven en la etapa de ejecución.
   - Las operaciones de carga/almacenamiento, saltos y *nops* se llevan a cabo en las estaciones de reserva enteras.
   - Se emplea un *branchdelay slot* de una instrucción.
   - Las unidades funcionales tienen las siguientes características:

RS	Cantidad	Latencia	Segmentación
FP ADD	2	4	NO
FP MUL	2	5	NO
INT ALU	2	1	NO
LOAD/STORE	2	2	NO

a. Mostrar el cronograma de ejecución. Calcular el CPI.
b. Desenrollar el bucle para obtener 3 copias del cuerpo y planificarlo asumiendo las latencias anteriores (emplear renombrado explícito de registros). ¿Cuántos ciclos por iteración se obtendrán?

Solución

1. Condición primera.
    a. Cronograma.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
ld	IS	RO	EX	WB																							
multd		IS	-(1)	-	RO	EX	EX	EX	EX	EX	WB																
ld			IS	RO	EX	WB																					
addd				-(2)	-	-	IS	-(1)	-	-	-	RO	EX	EX	EX	EX	WB										
sd								IS	-(1)	-	-	-	-	-	-	-	-	RO	EX	WB							
addi									IS	RO	EX	WB															
addi										-(3)	-	-	IS	RO	EX	-(4)	-	-	WB								
sgti												-(3)	-	-	-	-	-	-	-	IS	RO	EX	WB				
beqz																					IS	-(1)	-	RO	ES	WB	
NOP																						-(3)	-	IS	RO	EX	WB
ld																									-(3)	-	IS

CPI = 2,7

    b. Planificación.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld	IS	RO	EX	WB																
ld		IS	RO	EX	WB															
multd			IS	-(1)	RO	EX	EX	EX	EX	EX	WB									
addi				-(3)	IS	RO	EX	WB												
addd						IS	-(1)	-	-	-	-	RO	EX	EX	EX	EX	WB			
addi							IS	RO	EX	WB										
sgti								-(3)	IS	RO	EX	WB								
beqz									-(3)	-	IS	-(1)	RO	EX	WB					
sd												-(3)	IS	-(1)	-	-	-	RO	EX	WB
ld														-(3)	-	IS	RO	EX	WB	

CPI = 2

(1) Parada por riesgo RAW
(2) Parada por riesgo WAW
(3) Parada por falta de UF
(4) Parada por riesgo WAR

2. Condición segunda.
    a. Cronograma.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ld	IS	EX	EX	WB												
multd		IS	-(1)	EX	EX	EX	EX	EX	WB							
ld			IS	EX	EX	WB										
addd				IS	-(1)	-	-	-	EX	EX	EX	EX	WB			
sd					IS	-(1)	-	-	-	-	-	-	EX	EX	WB	
addi						IS	EX	WB								
addi							IS	EX	-(2)	WB						
sgti								IS	EX	-(2)	WB					
beqz									-(3)	IS	EX	WB				
NOP											IS	EX	-(2)	WB		
ld												IS	EX	-(2)	-	WB

CPI = 1,36

(1) Parada por riesgo RAW
(2) Parada por escritura única en CDB
(3) Parada por falta de RS

b. Desenrollado.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
ld	IS	EX	EX	WB																										
multd		IS	-	EX	EX	EX	EX	EX	WB																					
ld			IS	EX	EX	WB																								
addd				IS	-	-	-	-	EX	EX	EX	EX	WB																	
sd					IS	-	-	-	-	-	-	-	EX	EX	WB															
ld						IS	EX	EX	WB																					
multd							IS	-	EX	EX	EX	EX	EX	WB																
ld								-	IS	EX	EX	WB																		
addd										IS	-	-	-	EX	EX	EX	EX	WB												
sd											-	IS	-	-	-	-	-	EX	EX	WB										
ld													-	-	IS	EX	EX	-	WB											
multd																IS	-	-	EX	EX	EX	EX	EX	WB						
ld																	-	-	IS	EX	EX	WB								
addd																				IS	-	-	-	EX	EX	EX	EX	WB		
sd																					IS	-	-	-	-	-	-	EX	EX	WB
addi																						IS	EX	-	WB					
addi																							IS	EX	-	WB				
sgti																								-	IS	EX	WB			
beqz																										IS	EX	-	WB	

CPI = 2

Ejercicio 3

Suponiendo que se tiene el siguiente código en un cierto lenguaje ensamblador:

```
1  loop: ld    f2, 0(r1)
2        multd f4, f2, f0
3        multd f8, f2, f2
4        ld    f6, 0(r2)
5        addd  f6, f4, f6
6        subd  f8, f8, f6
7        sd    0(r2), f8
8        addi  r1, r1, #8
9        addi  r2, r2, #8
10       subi  r3, r3, #1
11       beqz  r3, loop
```

Donde se lista en primer lugar el operando de destino y a continuación los operandos fuente, y en el caso de las instrucciones de salto se listan primero los operandos a ser comparados y después la dirección de salto.

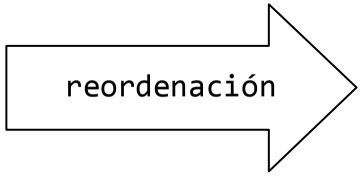
Se pide:

1. Realice un desenrollado del ciclo de forma que 2 iteraciones del ciclo actual correspondan a 1 en el código desenrollado, reordénelo y muestre el cronograma.

Solución

1. Desenrollado, reordenación:

```
ld    f2, 0(r1)
multd f4, f2, f0
multd f8, f2, f2
ld    f6, 0(r2)
addd  f6, f4, f6
subd  f8, f8, f6
sd    0(r2), f8
ld    f10, 8(r1)
multd f12, f10, f0
multd f16, f10, f10
ld    f14, 8(r2)
addd  f14, f12, f14
subd  f16, f16, f14
sd    8(r2), f16
addi  r1, r1, #16
addi  r2, r2, #16
subi  r3, r3, #2
beqz  r3, loop
```




```
ld    f2, 0(r1)
ld    f10, 8(r1)
addi  r1, r1, #8
multd f4, f2, f0
multd f12, f10, f0
ld    f6, 0(r2)
ld    f14, 8(r2)
multd f8, f2, f2
multd f16, f10, f10
addd  f6, f4, f6
addd  f14, f12, f14
subd  f8, f8, f6
subd  f16, f16, f14
subi  r3, r3, #1
sd    0(r2), f8
sd    8(r2), f16
addi  r2, r2, #8
beqz  r3, loop
```

Cronograma.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
ld	IF	ID	EX	MEM	WB																		
ld		IF	ID	EX	MEM	WB																	
add			IF	ID	EX	MEM	WB																
multd				IF	ID	EX	MEM	WB															
multd					IF	ID	EX	MEM	WB														
ld						IF	ID	EX	MEM	WB													
ld							IF	ID	EX	MEM	WB												
multd								IF	ID	EX	MEM	WB											
multd									IF	ID	EX	MEM	WB										
addd										IF	ID	EX	MEM	WB									
addd											IF	ID	EX	MEM	WB								
subd												IF	•	ID	EX	MEM	WB						
subd														IF	ID	EX	MEM	WB					
subi															IF	ID	EX	MEM	WB				
sd																IF	ID	EX	MEM	WB			
sd																	IF	ID	EX	MEM	WB		
addi																		IF	ID	EX	MEM	WB	
beqz																			IF	ID	EX	MEM	WB

Ejercicio 4

Suponiendo que el siguiente código en C:

```
for(i=0; i<100; i++)
      if(X[i] > 0)
            Y[i] = X[i] + Y[i] + Y[i+1];
      else
            Y[i] = X[i];
```

Se tiene en un cierto lenguaje ensamblador:

```
     L0:   ld    r3, #100
           ld    r1, DIR_X
           ld    r2, DIR_Y
  1  L1:   ld    f2, 0(r1)
  2        blt   f2, 0, L3
  3  L2:   ld    f4, 0(r2)
  4        ld    f6, 4(r2)
  5        add   f8, f2, f4
  6        add   f8, f8, f6
  7        store 0(r2), f8
  8        br    L4
  9  L3:   store 0(r2), f2
  10 L4:   add   r1, r1, #4
  11       add   r2, r2, #4
  12       sub   r3, r3, #-4
  13       blt   r3, 0, L1
```

Donde se lista en primer lugar el operando de destino y a continuación los operandos fuente, y en el caso de las instrucciones de salto se listan primero los operandos a ser comparados y después la dirección de salto.

Se pide:

1. Realice un desenrollado del ciclo de forma que 2 iteraciones del ciclo actual correspondan a 1 en el código desenrollado, reordénelo y muestre el cronograma.

Solución

1. Desenrrolado y reordenación:

```
L1:   ld    f2, 0(r1)
      add   r1, r1, #4
      sub   r3, r3, #-4
      blt   f2, 0, L3
L2:   ld    f4, 0(r2)
      ld    f6, 4(r2)
      add   f8, f2, f4
      add   f8, f8, f6
      store 0(r2), f8
      br    L4
L3:   store 0(r2), f2
L4:   add   r2, r2, #4
      blt   r3, 0, L1
```

Cronograma.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
ld	IF	ID	EX	MEM	WB																
add		IF	ID	EX	MEM	WB															
sub			IF	ID	EX	MEM	WB														
blt				IF	ID	EX	MEM	WB													
ld					IF	ID	EX	MEM	WB												
ld						IF	ID	EX	MEM	WB											
add							IF	•	ID	EX	MEM	WB									
add									IF	•	•	ID	EX	MEM	WB						
store												IF	•	•	ID	EX	MEM	WB			
br															IF	ID	EX	MEM	WB		
add																IF	ID	EX	MEM	WB	
blt																	IF	ID	EX	MEM	WB