Trabajo Práctico 2: MIPS Datapath

66.20 Organización de las Computadoras

Nicolás Calvo, *Padrón Nro. 78.914*
nicolas.g.calvo@gmail.com
Celeste Maldonado, *Padrón Nro. 85.630*
maldonado.celeste@gmail.com
Matias Acosta, *Padrón Nro. 88.590*
matiasja@gmail.com

2do. Cuatrimestre de 2011

Facultad de Ingeniería, Universidad de Buenos Aires

Índice

1. Introducción

2. Desarrollo

1.
2.
3.

Ejecución del programa sin forwarding

Se ejecutan 129 ciclos y 64 instrucciones , con 2 instrucciones en el pipeline al finalizar.

4. $CPI = 129 ciclos / 64 instrucciones = 2{,}02$

Se contaron un total de 64 stalls, divididos en las siguientes categorÃas:

- 10 stalls de control (7.75
- 2 stalls correspondientes a la instruccion trap (1.55
- 52 stalls RAW (Read After Write) (40.31

Ejecución del programa con forwarding:

Se ejecutan 98 ciclos y 64 instrucciones , con 2 instrucciones en el pipeline al finalizar.

$$CPI = 98 ciclos / 64 instrucciones = 1,53$$

Se logró un $SpeedUp = 1{,}32$

Se contaron un total de 33 stalls, divididos en las siguientes categorÃas:

- 10 stalls de control (10.20
- 2 stalls correspondientes a la instruccion trap (2.04 % de todos los ciclos)
- 21 stalls RAW (Read After Write) (21.43 % de todos los ciclos), de los cuales los 21 corresponden a stalls de branch, es decir, que dichos branches tienen por argumentos registros escritos en la instruccion que los precede.

Para el codigo presentado Branch Delay Slot no podrÃa usarse para lograr una mejora significativa en tiempo de ejecución debido a las dependencias de los branches respecto a los argumentos de las instrucciones que se ejecutan antes y despues de ellos.

Por ejemplo, para el segmento:

```
andi r3,r2,#1
bnez r3,Modulo
add r1,r1,r2
Modulo: sgt r8,r1,r6
```

La instruccion andi se usa en el branch, por lo que no puede moverse al delay slot y sgt requiere de un registro que es modificado en caso de no tomar el branch, por lo que mover esta instruccion al delay slot modificarÃa la logica del programa.

Se reordenó el codigo moviendo a la instrucción

5. `andi r3,r2,#1`

de la siguiente forma:

Codigo original:

```
sge r8,r2,r5
bnez r8,Fin
andi r3,r2,#1
bnez r3,Modulo
add r1,r1,r2
```

Codigo reordenado:

```
sge r8,r2,r5
andi r3,r2,#1
bnez r8,Fin
bnez r3,Modulo
add r1,r1,r2
```

De esta forma se espera reducir los ciclos de stall RAW para que bnez r3, Modulo tenga disponible el valor del registro r3. Algo similar sucederÃa para bnez r8,Fin respecto a sge r8,r2,r5.

Se obtuvieron los siguientes resultados:

Ejecucion sin forwarding:

Se ejecutan 108 ciclos y 64 instrucciones , con 2 instrucciones en el pipeline al finalizar.

$$CPI = 108 ciclos / 64 instrucciones = 1{,}69$$

Se contaron un total de 43 stalls, divididos en las siguientes categorÃas:

- 10 stalls de control (9.26
- 2 stalls correspondientes a la instruccion trap (1.85
- 31 stalls RAW (Read After Write) (28.70

SpeedUp respecto al codigo $sinreordenamiento = 1{,}2$

Ejecución con forwarding:

9 Se ejecutan 84 ciclos y 64 instrucciones , con 2 instrucciones en el pipeline al finalizar.

$$CPI = 84 ciclos / 64 instrucciones = 1{,}31$$

SpeedUp respecto al codigos$inreordenamiento = 1{,}17$

Se contaron un total de 19 stalls, divididos en las siguientes categor'ias:

- 10 stalls de control (11.09
- stalls correspondientes a la instruccion trap (2.38
- stalls RAW (Read After Write) (8.33

6.

3. Conclusión