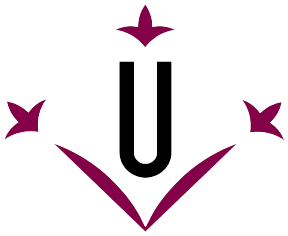

Universitat de Lleida
Escola Politècnica Superior

Arquitectura de computadors

Pràctica processament segmentat

Alejandro Clavera Poza
Jordi Rafael Lazo Florensa

20 de maig de 2020

Índex

1 Introducció

En aquesta pràctica hem ralitzat un programa anomenat, *comparar_major.s*, el qual donat un vector de dades i un número, calcula quants nombres del vector son majors que el número donat, a més a més, que emmagatzema en un altre vector de dades quins d'ells ho són. Per a això es col·loca un 0 en els números menors i un 1 en els majors.

2 Taula de dades

	Sense opcions de millora	**Forwarding**	**Delay slot**	**Forwarding + delay slot**
Nombre instruccions	66	66	67	67
Nombre cicles	140	99	132	91
CPI	2.121	1.500	1.970	1.358
RAW stalls	61	20	61	20
WAW stalls	0	0	0	0
WAR stalls	0	0	0	0
Structural stalls	0	0	0	0

Taula 1: Resultats execució codi

3 Comparativa d'execucions

Podem observar que quan executem el codi amb l'opció *forwarding* i *forwarding + delay slot* hi ha una millora substancial. El nombre de cicles per instrucció es redueix en un 35 % si es decideix activar la opció de *forwarding + delay slot* respecte a l'execució sense i del 30 % si es decideix activar només la opció de *forwarding*. D'altra banda si es compara a una execució seqüencial es podria observar que el rendiment d'aquesta empitjoraria pel fet que cada instrucció es produeix una darrere d'una altra per tant el que augmenta el nombre de cicles.

3.1 Tipus de depèndencia

Com podem observar en la taula 1, només es produeix la dependencia *RAW stalls*, aquesta dependencia *RAW (Read After Write)* es produeix quan una instrucció j llegeix un operand que abans escriu la instrucció i.
Uns exemples de dependencies RAW que es pot observar en el nostre codi es:

- *slt r5, r2, r4*
  dadd r3, r3, r5
- *daddi r1, r1, 8*
  bne r1, r6, 16

Si tenim en compte l'execució seqüencial podem observar que aquesta no té cap dependència de dades pel fet que una instrucció s'executa a continuació de l'anterior.

3.2 *Speedup* obtingut

El *speedup* es la multiplicació del número de instruccions per el número de etapes de segmentació, que son 5 *(IF,ID,EX,MEM,WB)*, dividt per el número de cicles d'xecució. Es a dir:

$$\frac{\text{número instruccions}}{\text{cicles execució}} \times 5$$

- *Sense opció de millora*:
$$\frac{66}{140} \times 5 \ = 1.88$$
- *Forwarding*:
$$\frac{66}{99} \times 5 \ = 3.33$$
- *Delay slot*:
$$\frac{67}{132} \times 5 \ = 2.53$$
- *Forwarding + delay slot*:
$$\frac{67}{91} \times 5 \ = 3.68$$

Tenint en compte que el *speedup* màxim és 5, es pot observar com es produeix una millora en les simulacions realitzades amb el *forwarding* i *forwarding + delay slot* activats.

3.3 Idoneïtat

Per a calcular la idoneïtat de cada opció, es divideix el *speedup*, trobat, entre el *speedup* ideal:

- *Sense opció de millora*:
$$\frac{1.88}{5} \ = 0.37$$
- *Forwarding*:
$$\frac{3.33}{5} \ = 0.66$$
- *Delay slot*:
$$\frac{2.53}{5} \ = 0.50$$
- *Forwarding + delay slot*:
$$\frac{3.68}{5} \ = 0.73$$

Finalment, com podem observar, *forwarding* i *forwarding + delay slot* són les opcions més idònies, ja que són aquelles el valor de les quals s'aproxima més a 1.