

## Parte 3: Atascos por dependencias de control

### 1)Atascos por salto (BTS)

Ejecutar el programa del ejercicio 3 de la Segunda Parte, reordenado y con Forwarding, de modo que no haya atascos RAW. El programa sigue teniendo algunos atascos, llamados Branch Taken Stall (BTS), también conocidos como atascos por salto.

1. Cambiar el valor de B a 1000 y ejecutar. ¿Cómo cambió la cantidad de BTS? ¿y el CPI?
2. Completar la siguiente frase: Al ejecutar un loop simple con N iteraciones, se producen \_\_\_\_\_ atascos BTS.
3. ¿Por qué se producen los BTS? ¿Qué sucede con la instrucción siguiente al salto?
4. ¿Pueden evitarse los BTS reordenando instrucciones?
5. Al ocurrir un BTS, una instrucción se empieza a ejecutar y luego se descarta, pero ¿debe contarse como una instrucción para el cálculo del CPI?

1. Los atascos cambiaron de 2 a 999 y el CPI de 1.46 a 1.334
2. “Al ejecutar un loop simple con N iteraciones, se producen N-1 atascos BTS.”
3. Se producen cuando hay un salto condicional. La cpu sabe que una instrucción es de salto cuando se decodifica, por lo tanto hay que descartar la instrucción que se capto en la etapa IF que no se va a ejecutar (porque se va a ejecutar la instrucción correspondiente al salto). ”
4. No, no se pueden evitar ya que no es por dependencia de datos, sino porque empieza a procesar una instrucción que se no se va a ejecutar.
5. Las instrucciones descartadas no cuentan para las estadísticas. Si la instrucción se ejecuta en otro momento si se la contara, pero solo en el momento que se ejecute efectivamente.

### 2)Reducción de BTS con Branch Target Buffer

Habilitar la opción Branch Target Buffer (BTB) y volver a ejecutar el programa anterior con B=3.

1. ¿Cómo cambió la cantidad de BTS? ¿y el CPI?
  2. Notar que ahora aparece un nuevo tipo de atasco, llamado Branch Misprediction Stall (BMS). ¿Por qué sucede? ¿Cuántos ocurren?
  3. Cambiar el valor de B a 1000, habilitar BTB y ejecutar. ¿Cómo cambió la cantidad de BTS y BMS? ¿y el CPI?
  4. Completar la siguiente frase: Al ejecutar un loop simple con N iteraciones, si se habilita BTB, se producen\_\_\_\_\_ atascos de tipo BTS y\_\_\_\_\_ atascos de tipo BMS.
  5. En base a los resultados anteriores ¿es mejor utilizar BTB cuando se realizan pocas o cuando se realizan muchas iteraciones?
- 
1. Ocurrieron 2 BMS. El CPI pasó a 1.615
  2. Ocurren cuando hay una mala predicción en el salto. Si se predijo que el salto iba a ocurrir y no ocurre o viceversa, hay un atasco de ese tipo.
  3. BTS y BMS se mantiene igual ya que ocurren los mismos atascos que antes. El CPI baja a 1.003 porque suben los ciclos pero no los atascos.
  4. “Al ejecutar un loop simple con N iteraciones, si se habilita BTB, se producen 2 atascos de tipo BTB y 2 atascos de tipo BMS”.
  5. Lo óptimo es utilizarlo cuando hay muchas iteraciones.

### 3)Utilidad del BTB en distintos casos

El BTB puede aumentar el desempeño significativamente en la mayoría de los lazos. No obstante, en algunos puede tener un comportamiento patológico y de hecho reducir la eficiencia del programa. El siguiente programa calcula el máximo de un vector.

```

.data
A: .word 2,1,3,1,4,1
MAX: .word -1
.code
ld $t1, MAX($0) daddi $t2,$0,0 daddi
$t3,$0,6
loop: ld $t4, A($t2) slt $t5,$t1,$t4
beqz $t5, chico daddi $t1,$t4,0
chico:daddi $t2,$t2,8 daddi $t3, $t3,
-1 bnez $t3, loop sd $t1, MAX($0)
halt

```

- a) Antes de ejecutar en el simulador, encontrar las instrucciones de salto, y pensar cómo se comportará

el BTB en cada caso. Recordar que la predicción del BTB guarda un bit de historia distinto por cada instrucción de salto.

- Ejecutar el programa en el simulador con y sin BTB. ¿Qué programa es más eficiente?
- Es más eficiente sin BTB

#### 4) Reducción de BTS con Delay Slot (DS)

El delay slot (DS) cambia el funcionamiento de los saltos. Cuando está activado, el salto se realiza con un retardo de un ciclo. Esto significa que la instrucción **siguiente** al salto también se ejecuta. De esta forma, los BTS desaparecen completamente, aunque esto no significa necesariamente que la ejecución sea más eficiente.

- Anotar en la tabla de abajo las estadísticas del programa del ejercicio 3 de la Segunda Parte al ejecutarse sin forwarding, y con BTB y **delay slot** desactivados
- Ejecutar el programa del ejercicio 3 de la Segunda Parte, pero ahora con BTB activado y forwarding desactivado.
- Ejecutar el programa del ejercicio 3 de la Segunda Parte, pero ahora con **delay slot** activado, BTB desactivado y forwarding activado. Anotar los CPI y la cantidad de instrucciones en la tabla de abajo. ¿Qué sucede con la instrucción **sd \$t1, C(\$0)**? ¿Cuántas veces se ejecuta? ¿Son necesarias todas las ejecuciones?
- Modificar el programa del ejercicio 3 de la Segunda Parte agregando un NOP antes de la instrucción **sd \$t1, C(\$0)**. Ejecutar el programa en el simulador y anotar los CPI. Los CPI serán menores que en el caso anterior
- Modificar el programa del ejercicio 3 de la Segunda Parte, pero ahora reordenando las instrucciones de modo de ejecutar una instrucción útil debajo del salto, sin aumentar el número total de instrucciones a ejecutar.

Ejercicio 3	Sin mejoras	BTB	DS	DS + NOP	DS + Reordenamiento
BTS	2	2	0	0	0
BMS	0	2	0	0	0
CPI	$\frac{23}{13} = 1.769$	$\frac{25}{13} = 1.923$	$\frac{19}{15} = 1.267$	$\frac{20}{16} = 1.250$	1.615
#Instrucciones	13	13	15	16	13

#### 5) Corrección de programas con Delay Slot

En el programa del ejercicio 3 de la Segunda Parte, habilitar DS hace que una instrucción se ejecute varias veces. Si bien el programa es más ineficiente, el resultado final es el mismo. No obstante, en otros casos habilitar DS puede hacer que el programa no funcione correctamente.

- Correr el programa del ejercicio 3 de esta parte pero ahora con DS habilitado y anotar el CPI. Notar el resultado almacenado en la variable MAX ¿Por qué es incorrecto?
- Reordenar las instrucciones del programa para que funcione correctamente, y además la instrucción **sd \$t1, MAX(\$0)** solo se ejecute una vez, sin agregar instrucciones a ejecutar. Anotar el CPI, y comparar con el caso anterior ¿qué mejora se obtuvo?

a.-  $\frac{74}{52} = 1.423 \text{ CPI}$ . Es incorrecto porque siempre se guarda el valor que se quiere analizar el loop (o sea, siempre se guarda el ultimo)

b.-  $\frac{63}{44} = 1.432 \text{ CPI}$ . Se redujo la cantidad de ciclos de 74 a 63. La cpu ejecuta 8 instrucciones menos.