Examen Final de Arquitectura de Computadores: CS221

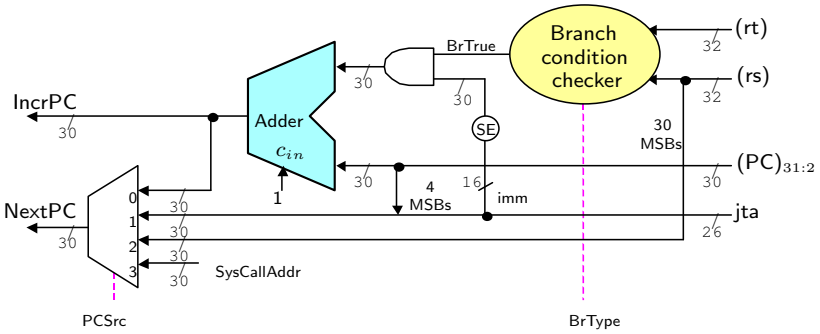
Escuela Profesional de Ciencia de la Computación: CCOMP3-1, CCOMP3-2

30 de Noviembre de 2017

EJECUCIÓN MIPS EN CICLO ÚNICO / ALU

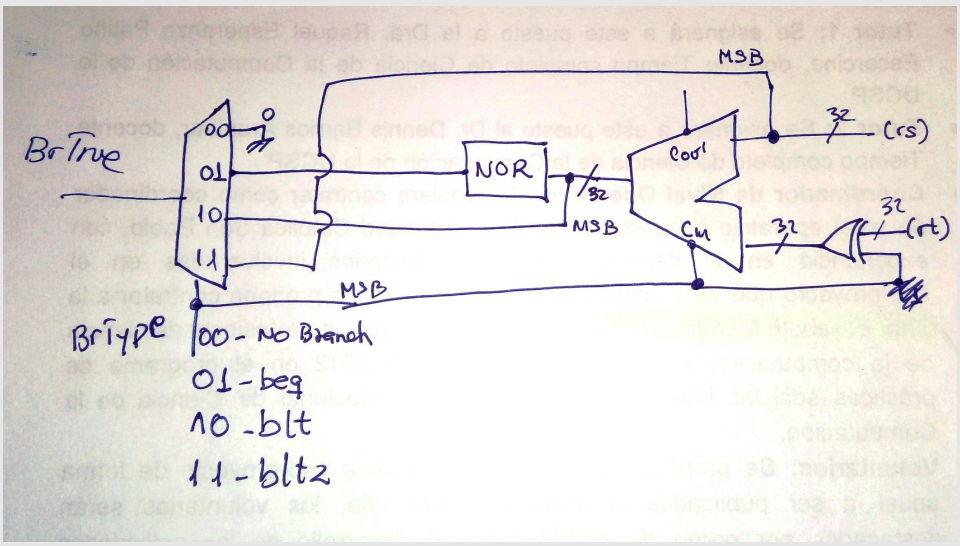
Pregunta 1: **[4pt, Competencias: b2, i3]**

Sea la Lógica de "próxima dirección" para ejecución en ciclo único de la siguiente figura:



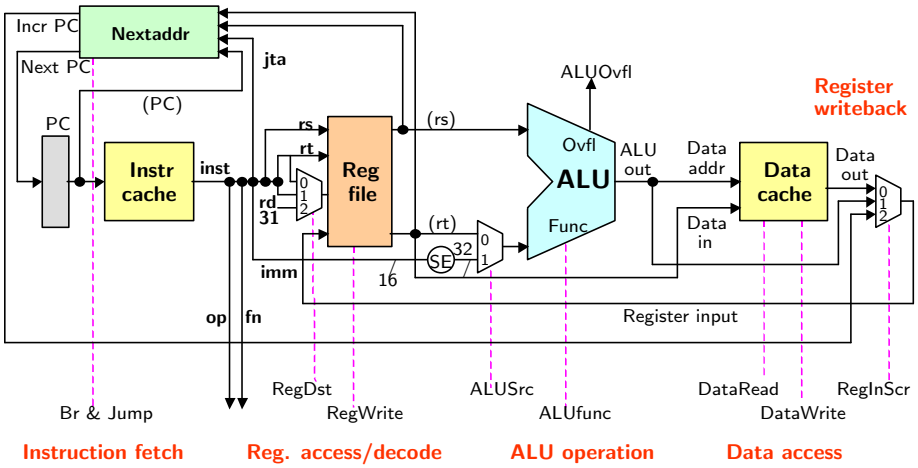
Se desea cambiar la instrucción **`bne`** por una nueva instrucción **`blt`** que verifica si el valor en `(rs)` es menor que el valor en `(rt)`. Haga un diseño de la lógica de verificación de condición de bifurcación (*branch condition checker*) que soporte las instrucciones **`beq`**, **`blt`**, **`bltz`**, (puede usar multiplexadores, sumadores binarios y otras puertas lógicas)

En este caso la instrucción **`blt`** que compara si `(rs)` es menor que `rt`, exige hacer una substracción por lo que se debe incorporar un sumador binario con el artificio de convertirlo en restador en complemento-2. Las bifurcación **`beq`** puede aprovechar este sumador que al restar también sirve de comparador de igualdad (verificar resultado cero). La bifurcación `blqz` sólo pide observar el MSB de `(rs)` verificando si es negativo.



Pregunta 2: **[4pt, Competencias: b2, i3]**

Sea la arquitectura de ejecución de instrucciones en ciclo único:



¿Es posible que la pseudo instrucción `li` (cargar un valor inmediato directamente en un registro, por ejemplo `li $v0, 10` sea ejecutada como instrucción real en esta arquitectura? Caso sea posible, indique la configuración de las líneas de control requerida para esta ejeución.

Felizmente, la pseudoinstrucción `li` puede ejecutarse mediante una sola instrucción `addi $v0, $zero, 10` (adecuada al ejemplo), la cual tendría la siguiente configuración de líneas de control:

op = 00100 (8)	
RegWrite = 1 (se graba registro)	RegDst = 00 (rt es destino instr I)
RegInSrc = 01 (se grabará ALUOut)	ALUSrc = 1 (imm 2do operando ALU)
Add'Sub = 0 (add)	FnClass = 10 (Add/sub)
DataRead = 0 (no lee datos)	DataWrite = 0 (no graba datos)
BrType = 00 (No es bifurcacion)	PCSrc = 00 (PC aumenta normalmente)

EJECUCIÓN MIPS EN MULTI CICLO

Pregunta 3: **[4pt, Competencias: i2]**

Sea el siguiente trecho de código mips:

```
      addi $t0,$zero,0      # $t0 = 0
      addi $t1,$zero,1      # $t0 = 1
      lw   $t8,0($t9)       # $t9 apunta a value
      addi $t4,$zero,2      # $t4 = 2
      beq  $t8,$zero,fin    # si $t8 es cero
      bltz $t8,fin          # si $t8 es negativo
loop: add  $t2,$t0,$t1      # bucle
      beq  $t4,$t8,fin
      addi $t4,$t4,1
      addi $t0,$t1,0
      addi $t1,$t2,0
      j loop
fin:  ...                   # salida de bucle
```

Calcule la cantidad de ciclos de reloj requeridos para la ejecución de este código suponiendo que la variable value en memoria se define como

```
value: .word 4
```

Se desdobla el loop para ver cómo se ejecuta este trecho de codigo para luego contar los ciclos de reloj totales requeridos para la ejecución de todo el desdoblado. Tomar en cuenta: instrucciones ALU = 4 ciclos de reloj, lw = 5 ciclos, sw = 4 ciclos, bucles y saltos 3 ciclos.

```
4         addi $t0,$zero,0
4         addi $t1,$zero,1
5         lw   $t8,0($t9)       # $t8 = 4
4         addi $t4,$zero,2      # $t4 = 2
3         beq  $t8,$zero,fin    # no se cumple
3         bltz $t8,fin          # no se cumple
4   loop: add  $t2,$t0,$t1      # bucle
3         beq  $t4,$t8,fin      # 2 != 4 no se cumple
4         addi $t4,$t4,1        # $t4 = 3
4         addi $t0,$t1,0
4         addi $t1,$t2,0
3         j loop
4   loop: add  $t2,$t0,$t1
3         beq  $t4,$t8,fin      # 3 != 4 no se cumple
4         addi $t4,$t4,1        # $t4 = 4
4         addi $t0,$t1,0
4         addi $t1,$t2,0
3         j loop
4   loop: add  $t2,$t0,$t1
3         beq  $t4,$t8,fin      # 4 = 4 si se cumple
    fin:  ...                   # salida de bucle
```

Luego sumando los ciclos obtenemos:
23 ciclos antes del loop
22 ciclos en el loop (que se repite 2 veces)
7 ciclos en el ultimo paso del loop donde se cumple la bifurcación de salida
total 23 + 44 + 7 = 74 ciclos

ARQUITECTURA EN PIPELINE

Pregunta 4: **[4pt, Competencias: i2]**
Para el mismo código de la pregunta anterior, encuentre la cantidad mínima de ciclos de reloj gastados en la ejecución cuando se considera un *Pipeline* con 5 etapas (Fetch, Acceso a registros, ALU, Acceso a datos, acceso a registros) que soporta ejecución reordenada, *forwarding*, lectura/escritura simultánea en registros y predicción simple de bifurcación (pensar que la bifurcación nunca se cumplirá y cuando se equivoque, mande vaciar las instrucciones falladas del *pipeline*) perdiendo un ciclo de reloj por el error y la reposición de instrucciones correctas hasta llenar el resto del pipeline.

En Pipeline cada instrucción inicia por cada ciclo de reloj, pero no se debe olvidar que al final de la ejecución hay la necesidad de terminar los pasos remanentes de las ultimas instrucciones entradas al pipeline (*draining stage*).

Si se revisa el código desdoblado de la pregunta anterior, se podrá notar que no hay dependencias de datos tipo I ni tipo II. Habiéndolas, la opción *forwarding* las resolvería.

Existen dependencias de control para todas las bifurcaciones, en este caso aplicando la predicción simple de bifurcación, se asume que no se cumplirá nunca la condición de bifurcación, por lo que el único caso en que esta predicción se equivocará es en la última instrucción del código desdoblado.

Hasta el inicio de la bifurcación que se cumple han ocurrido 20 ciclos de reloj.

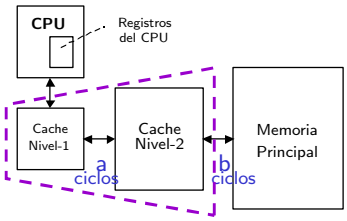
La verificación de la condición de bifurcación será notada en el ciclo 21, ya en el ciclo 22 se debe corregir el pipeline (vaciar las instrucciones que fueron puestas asumiendo que continua el bucle, y reemplazarlas por lo que sigue a la etiqueta fin:) En el ciclo 22 también está finalizando la ejecución de instrucción de bifurcación que toma 3 ciclos de reloj, aunque ese ejecutor se disponibilizará finalizando el ciclo 24.

El trecho de código se ejecuta usando 24 ciclos de reloj (podría pensarse en 22 ciclos de ejecución efetiva)

MEMORIA CACHÉ

Pregunta 5: **[4pt, Competencias: b2]**

Suponiendo el sistema de memoria caché de dos niveles L1 y L2 de la figura



considerando que la arquitectura tiene un CPI = 1.1 sin pérdidas por caché y un promedio de 1.1 accesos a memoria por instrucción.
Encontrar el CPI considerando pérdidas de caché si el acierto de L1 es 98% con penalidad de 6 ciclos de reloj por cada pérdida y el acierto de L2 es 85% con penalidad de 80 ciclos de reloj por cada pérdida.

El CPI por las pérdidas de caché es el costo promedio de acuerdo a las probabilidades de pérdida en ambos cachés multiplicado por el promedio de accesos a memoria, que es calculado por la siguiente ecuación:

$$C_{eff} = C_{fast} + Prom(1-h_1)[C_{med} + (1-h_2)C_{slow}] \quad (1)$$

La CPI = 1.1 ya está incluída en C_{fast}, a la cual deberá adicionarsele la CPI de las pérdidas de la Ecuación 1, así:

$$\begin{aligned} CPI &= 1.1 + 1.1 \times (1-0.98)[6+(1-0.85)80] \\ CPI &= 1.1 + 1.1 \times 0.02 \times [6+12] \\ CPI &= 1.1 + 1.1 \times 0.02 \times 18 \\ CPI &= 1.1 + 0.396 \\ CPI &= 1.496 \end{aligned}$$