Arquitectura del Procesador II

Práctico U2.2 Predicción de Saltos

Ejercicio 1: Para el siguiente fragmento de código:

```
        addi   $1, $0, 10
loop:   sw     $1, 0($2)
        addi   $2, $2, #4
        addi   $1, $1, -1
        bne    $1, $0, loop
        addi   $1, $0, 10
```

Muestre el diagrama de ciclos de reloj para el MIPS segmentado con unidad de adelantamiento para saltos, donde la dirección efectiva del salto se resuelve en la etapa ID, suponiendo:

1. Que se utiliza la política: Detener/Atascar la segmentación.
2. Que se utiliza la política: Salto no tomado.
3. Que se utiliza la política: salto demorado (planifique el ***slot*** de forma adecuada).

Luego, Compare estas mismas alternativas contra un procesador MIPS segmentado que no posea adelantamiento para saltos.

Ejercicio 2: Supongamos la arquitectura básica del MIPS segmentado de 5 etapas, sobre la cual se implementa una política de saltos demorado (delayed branch). Los saltos se resuelven en la etapa ID. Indique si las siguientes afirmaciones son ciertas o falsas:

□ a) Cuando el **delay slot** se rellena con una instrucción del destino del salto y el salto no se toma, entonces se produce una penalización de un ciclo de reloj.

□ b) La siguiente transformación es legal:

```
        sub $1, $2, $3                          sub $1, $2, $3
        beq $1, $0 LAB                          beq $1, $0, LAB1
        nop                                     xor $4, $2, $2
        add $4,$1,$1          >>>>>             add $4,$1,$1
        . . .                                   . . .
LAB:    xor $4, $2, $2                  LAB:    xor $4, $2, $2
        ld $1, 0(r4)                    LAB1:   ld $1, 0(r4)
```

□ c) La siguiente transformación es legal :

```
LAB:  add $6, $6, $2                          . . .
      subi $1, $1, 0x1                  LAB:  subi $1, $1, 0x1
      bne $1, $0, LAB                         bne $1, $0, LAB
      nop                   >>>>>             add $6, $6, $2
      add $5, $4, $6                          add $5, $4, $6
      . . .                                   . . .
```

□ d) La siguiente transformación es legal :

```
      sub $1, $2, $3                          sub $1, $2, $3
      beq $1, $0, LAB                         beq $1, $0, LAB
      nop                                     add $6, $1 ,$1
      add $6, $1, $1        >>>>>             xor $8, $6, $5
      xor $8, $6, $5                          . . .
      . . .                             LAB:  sub $5, $6, $7
LAB:  sub $5, $6, $7
```

□ e) Si se cambia a una política de predicción de saltos como no tomado (predict-not-taken), entonces la instrucción que está a continuación del salto se lee igualmente de la memoria de instrucciones, pero su ejecución se cancela en caso de que el salto no se tome.

<u>Ejercicio 3:</u> Para el siguiente fragmento de código MIPS:

```
        addi   $1, $0, 10
loop:   sw     $1, 0($2)
        addi   $2, $2,  4
        addi   $1, $1, -1
        bne    $1, $0, loop
        nop
        add    $29,$30,$31
```

1. Muestre el diagrama de ciclos de reloj de esta secuencia de instrucciones para el MIPS segmentado donde la dirección efectiva del salto se resuelve en la etapa ID y se utiliza un predictor de saltos dinámico de 2 bits y unidades de adelantamiento para instrucciones de salto.

2. Luego, suponiendo que se utiliza la política *salto demorado*, planifique la franja de instrucciones (slot) indicado por “nop” y muestre el diagrama de ciclos de reloj donde la dirección efectiva del salto se resuelve en la etapa ID.