PRÁCTICA 2

Estudio del cauce segmentado

Conceptos desarrollados:

Riesgos RAW.
Adelantamiento (forwading).
Aceleración.
Salto retardado.

El programa de prueba.

Para la realización de esta práctica utilizaremos un programa de ordenación mediante el algoritmo de la burbuja y observaremos su comportamiento bajo determinadas condiciones. Se trata de un método sencillo que nos permitirá ordenar un conjunto de enteros almacenados en un vector. El código correspondiente sería algo así:

```
int   n = 10;   // Longitud del vector.
int   A [10] = {1,5,3,7,2,8,5,9,0,7};   // Valores del vector.
int   i, j, k;

for  ( i = 0; i < n-1; i++ )
    for  ( j = n-1; j > i; j-- )
        if  (A [ j ] < A [ j-1 ]) {
            k = A [ j ];
            A [ j ] = A [ j-1 ];
            A [ j-1 ] = k;
        }
```

El programa ensamblador correspondiente podría ser el que se muestra a continuación y se suministra en el fichero Burbuja.s.

```
.data

N:      .byte   10
A:      .byte   1,5,3,7,2,8,5,9,0,7

.text

        lb      r1,N(r0)
        daddi   r1,r1,-1            ; N - 1
        dadd    r2,r0,r0            ; i = 0   (i es r2)
for1:   beq     r2,r1,sigue1        ; for ( i = 0; i < n-1; i++ )
          dadd    r3,r0,r1            ; j = N - 1   (j es r3)
for2:     beq     r3,r2,sigue2        ; for ( j = n-1; j > i; j-- )
            lb      r4,A(r3)            ; r4 = A[j]
            lb      r5,A-1(r3)          ; r5 = A[j-1]
            slt     r6,r4,r5
            beqz    r6,salta            ; if (A [j] < A [j-1]) {
              dadd    r7,r0,r4            ; k = A [j];
              dadd    r4,r0,r5
              sb      r4,A(r3)            ; A [j] = A [j-1];
              sb      r7,A-1(r3)          ; A [j-1] = k;
salta:    daddi   r3,r3,-1
          j       for2
sigue2: daddi   r2,r2,1
        j       for1
sigue1: halt
```

Primera parte.

1) Asegurarse de que no están seleccionadas ninguna de las opciones de la pestaña de configuración.

2) Cargar y ejecutar el programa comprobando que funciona correctamente.

3) Anotar los datos siguientes:

Ciclos:	Paradas por RAW:
Instrucciones:	Paradas por saltos (Branch taken):
CPI:	

4) Comprobar que el valor CPI coincide con el resultado del cociente entre el número de ciclos y las instrucciones ejecutadas.

$$\frac{\text{nº de ciclos}}{\text{nº de instrucciones}} =$$

5) Ahora vamos a fijarnos en la **primera vez que se ejecuta la instrucción "slt"** y contestaremos a las preguntas formuladas a continuación. Pero antes de nada hay que aclarar dos cuestiones.

La primera no representa gran dificultad y es simplemente que el simulador numera los ciclos de manera que el primero es el ciclo cero y no el uno como solemos hacer en clase.

La segunda es que en la ventana "Cycles" no aparecen todas las instrucciones que se han ejecutado ya que tiene un buffer limitado y guarda solamente las últimas. Por lo tanto, es posible que lo que ocurrió al comienzo de la ejecución ya no esté disponible y será necesario volver a ejecutar el programa desde el principio, parando en el lugar en que la instrucción aparece por primera vez.

¿En qué ciclo comienza la extracción (IF) de "slt"?

Cuantos ciclos de espera se producen:

¿En qué etapa se produce la espera?

¿A qué es debida dicha espera?

6) Observaremos ahora el comportamiento de la instrucción que va a continuación: la instrucción "beqz".

¿Por qué permanece tantos ciclos en la etapa IF?

¿Cuantos ciclos de espera se introducen en la etapa ID?

¿A qué se debe la espera?

7) Observa como coinciden la lectura y escritura de un mismo registro (coincidencia de las etapas ID y WB) en las dependencias.

¿Qué permite que ambas etapas coincidan en el mismo ciclo?

¿En qué ciclo se produce por primera vez esta coincidencia?

¿Cuál es el registro causante de dicha dependencia?

Segunda parte.

8) Entrar en la pestaña de configuración y activar la opción de adelantamiento (forwarding).

9) Volver a cargar y ejecutar el programa comprobando que sigue funcionando.

10) Anotar los nuevos datos:

Ciclos:	Paradas por RAW:
Instrucciones:	Paradas por saltos (Branch taken):
CPI:	

11) Calcular la aceleración respecto a la ejecución sin adelantamiento.

12) Comprobar que se obtiene la misma aceleración tanto si se comparan los tiempos de ejecución como si se comparan los valores de CPI.

13) Buscar un caso en el que se produzca un adelantamiento de la etapa **MEM a la EX** y escribir las dos líneas de la ventana "Cicles" correspondientes a las instrucciones involucradas.

14) En este caso el simulador presenta un comportamiento diferente al visto en clase. ¿Cuál es esa diferencia?

15) Buscar ahora un caso en el que se produzca un adelantamiento desde la etapa **EX a la ID** escribiendo las líneas afectadas.

Tercera parte.

16) Entrar nuevamente en la pestaña de configuración y activar la opción de hueco de retardado (delay slot) además de la de adelantamiento (forwarding).

17) Volver a cargar el programa y observar el resultado. ¿A qué se debe este comportamiento?

18) Solucionar el problema mediante el uso de instrucciones nop, comprobando de forma experimental cuál será el **mínimo** de dichas instrucciones y **dónde** habrá que colocarlas.

19) Una vez solucionado el problema, ejecutar la nueva versión y anotar los siguientes datos:

Ciclos:	Paradas por RAW:
Instrucciones:	Paradas por saltos (Branch taken):
CPI:	

20) Ahora, intentar solucionar el problema con reordenamiento de código en vez de instrucciones nop. Indicar los cambios introducidos:

21) Volver a ejecutar con la última modificación y anotar los datos siguientes:

Ciclos:	Paradas por RAW:
Instrucciones:	Paradas por saltos (Branch taken):
CPI:	

22) Compara estos resultados con los obtenidos con la versión anterior (la de las instrucciones nop).

¿Qué ocurre?

¿Cuál es la explicación?