UNIVERSIDAD CARLOS III DE MADRID

Ejercicios de evaluación. Scoreboard y Tomasulo

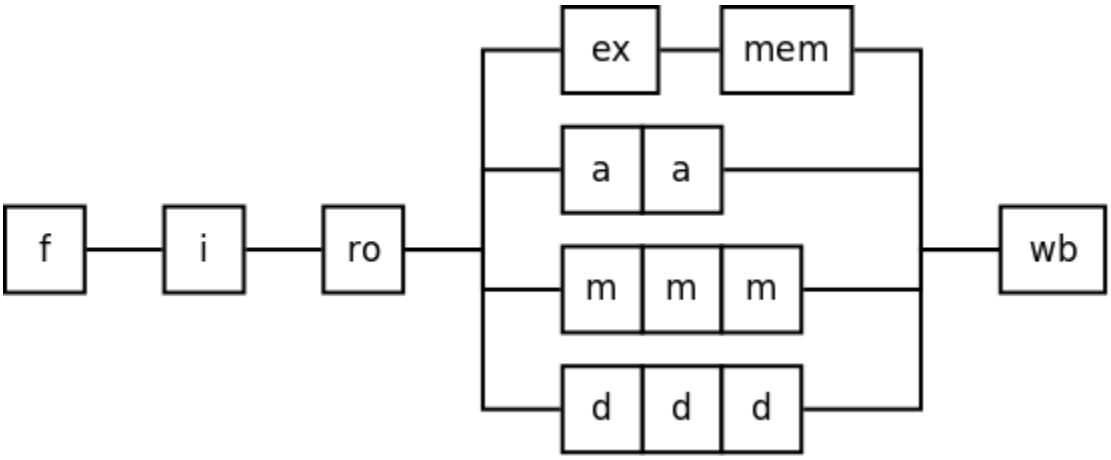
Departamento de Ingeniería de Sistemas y Automática

RAÚL PÉRULA MARTÍNEZ
LUIS ENRIQUE MORENO LORENTE
ALBERTO BRUNETE GONZALEZ
CESAR AUGUSTO ARISMENDI GUTIERREZ
DOMINGO MIGUEL GUINEA GARCIA ALEGRE
JOSÉ CARLOS CASTILLO MONTOYA

Universidad Carlos III de Madrid

Ejercicio 1

Se dispone de un procesador con la siguiente arquitectura:



Dado el siguiente programa en ensamblador:

```
1.  bucle:  ld f2, 0(r1)
2.          addd f4,f0,f2
3.          sto 0(r1), f4
4.          ld f6, -4(r1)
5.          addd f8, f6, f2
6.          sto -4(r1), f8
7.          sub r1,r1, #8
8.          bnez r1, bucle
9.
```

Se pide:

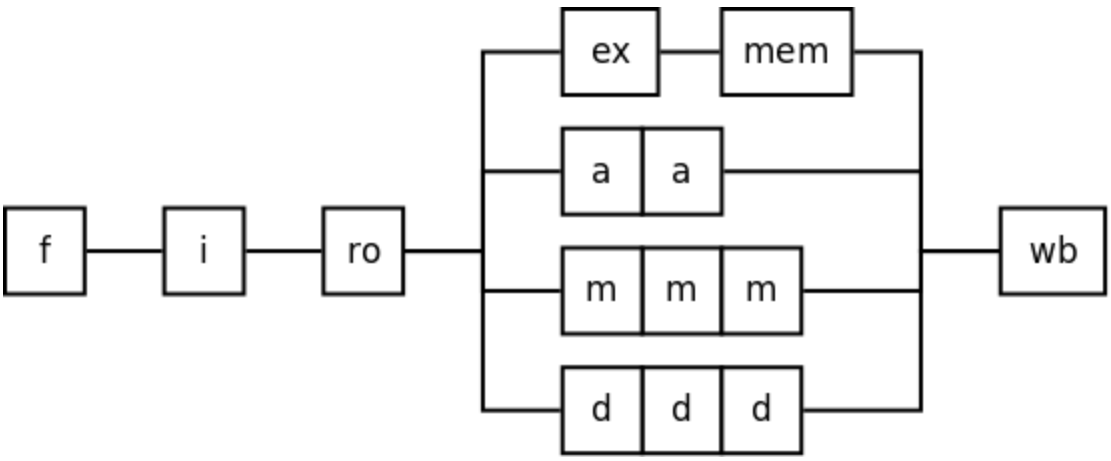
1) Cronograma de ejecución con Scoreboard (con el cauce entero NO segmentado).
2) Cronograma de ejecución con Tomasulo si cada unidad funcional tiene dos estaciones de reserva.
3) Cronograma de ejecución con Tomasulo si cada unidad funcional tiene dos estaciones de reserva para dos iteraciones del bucle, teniendo en cuenta que se usa la estrategia de predicción estática BT-FNT.

Ejercicio 2

Suponiendo que se tiene el siguiente código en lenguaje ensamblador

```
      L0:   ld r3, # 100
            ld r2, DIR_X
            ld r2, DIR_Y
 1    L1:   ld f2, 0(r1)
 2          blt f2, 0 , L3
 3    L2:   ld f4, 0(r2)
 4          ld f6, 4(r2)
 5          add f8, f2, f4
 6          add f8, f8, f6
 7          store 0(r2), f8
 8          br L4
 9    L3:   store 0(r2), f2
10    L4:   add r1, r1, #4
11          add r2, r2, #4
12          sub r3,r3, #4
13          blt r3, 0 L1
```

donde se lista en primer lugar el operando de destino y a continuación los operandos fuentes, y en el caso de las instrucciones de salto se listan primero los operandos a ser comparados y después la dirección del salto. La estrategia de resolución de saltos es Predict Not Taken. La arquitectura del procesador es la siguiente:



Suponiendo que 0(r1)=10. Se pide:

1) Obtener el cronograma mediante el algoritmo de Scoreboard (con el cauce entero segmentado).
2) Obtener el cronograma mediante el algoritmo de Tomasulo suponiendo dos unidades de reserva para cada unidad funcional

Ejercicio 3

El siguiente fragmento de código aplica un filtro sobre un vector A y almacena el resultado en un vector B:

```
for (int i = 1 ; i < 100 ; i++)
{
        b[i] = d * a[i-1] + a[i] + d * a[i+1];
}
```

El compilador lo traduce al siguiente código DLX

```
                addi r3, r1, 800        ; condición de final
                addi r1, r1, 8          ; inicialización de los índices
                addi r2, r2, 8
                ld f0, d                ; Carga el coeficiente d
1.      loop    ld f2, -8(r1)           ; carga de a[i-1]
2.              ld f4, 0(r1)            ; carga de a[i]
3.              ld f6, 8(r1)            ; carga de a[i+1]
4.              multd f8, f2, f2        ; d * a[i-1]
5.              multd f10, f0, f6       ; d * a[i+1]
6.              addd f4, f4, f8         ; d * a[i-1] + a[i]
7.              addd f4, f4, f10        ; d * a[i-1] + a[i] + d * a[i+1]
8.              sd 0(r2), f4            ; Guarda en b[i]
9.              addi r2, r2, 8          ; incremento de índices
10.             addi r1, r1, 8
11.             addi r1, r1, 8
12.             slt r4, r1, r3
13.             bnez r4, loop
```

Las operaciones de división/multiplicación en punto flotante necesitan 4 ciclos para ejecutarse y las operaciones de suma en punto flotante necesitan 3.

1) Obtener el cronograma mediante el algoritmo de Scoreboard (con cauce entero segmentado).
2) Obtener el cronograma mediante el algoritmo de Tomasulo suponiendo dos unidades de reserva para cada unidad funcional.

Ejercicio 4

Dado el código ensamblador que implementa la operación vectorial

$$Y = (a * X + Y) / X:$$

```
                ld r1, dir_x
                ld r2, dir_y
                ld r4, dir_cont
                ld f0, dir_a
1.      loop:   ld f2, 0(r1)          ; carga X[i]
2.              multd f4, f2, f0      ; multiplica a * X[i]
3.              ld f6, 0(r2)          ; carga Y[i]
4.              addd f6, f4, f6       ; suma a * X[i] + Y[i]
5.              divd f6, f6, f2       ; Divide (a * X[i] + Y[i]) / X[i]
6.              sd 0(r2), f6          ; Almacena Y[i]
7.              addi r1, r1, 8        ; incrementa el índice de x
8.              addi r2, r2, 8        ; incrementa el índice de Y
9.              sgt r3, r1, r4        ; comprueba si fin
10.             bequz r3, loop
```

Considerando que el código se ejecuta en una máquina DLX segmentada con unidades funcionales con las siguientes características:

Unidad funcional	Ejecución (ciclos)
ALU entera	1
Suma PF	2
Multiplicación PF	3
División PF	4

1) Obtener el cronograma mediante el algoritmo de Scoreboard.
2) Obtener el cronograma mediante el algoritmo de Tomasulo suponiendo tres unidades de reserva para cada unidad funcional.

Ejercicio 5

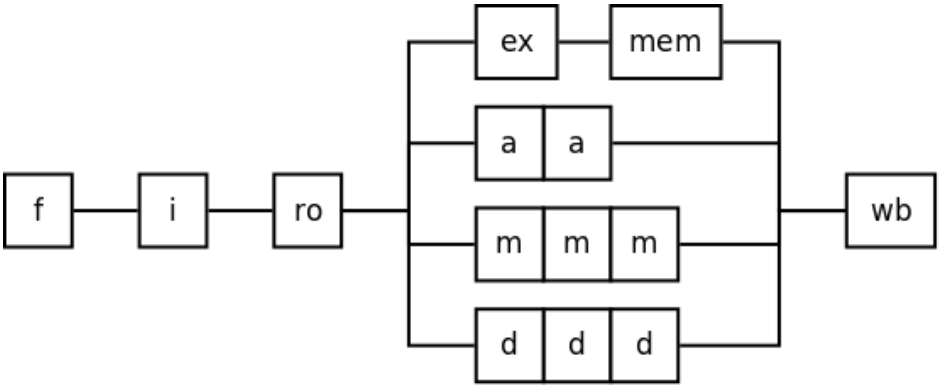
Suponiendo que el siguiente código en C

```
for ( i = 0; i<100; i++){
        if (X[i ]> 0){
                Y[i]= X[i ]*X[i] –Y[i];
        }
        else{
                Y[i]= X[i]*X[i]-Y[i+1]+10;
        }
}
```

tiene la siguiente traducción a un cierto lenguaje ensamblador

```
        L0:     ld      r3, #100
                ld      r1,DIR_X
                ld      r2,DIR_Y
1       L1:     ld      f2, 0(r1)
2               mul     f4, f2, f2
3               blt     f2, 0, L3
4       L2:     ld      f6, 0(r2)
5               sub     f6, f4, f6
6               store   0(r2), f6
7               br      L4
8       L3:     ld      f8, 4(r2)
9               sub     f6, f4, f8
10              addi    f4,f6, #10
11              store   0(r2), f4
12      L4:     add     r1, r1, #4
13              add     r2, r2, #4
14              sub     r3, r3, #-1
15              blt     r3, 0, L1
```

Los saltos se resuelven en la primera mitad de la etapa Mem y las instrucciones pueden leerse en la segunda mitad de la etapa If. La arquitectura del procesador es la siguiente (DLX, predict-not-taken):



1) Obtener el cronograma mediante el algoritmo de Scoreboard (cauce entero segmentado).
2) Obtener el cronograma mediante el algoritmo de Tomasulo suponiendo dos unidades de reserva para cada unidad funcional.