PROBLEMAS DE ESTRUCTURA DE COMPUTADORES

SEGMENTACIÓN

1. Sea un procesador segmentado con cinco etapas como el visto en clase que tiene las siguientes características:

- Se puede escribir y leer en el banco de registros en el mismo ciclo de reloj.
- No tiene técnicas para reducir o eliminar paradas en caso de riesgos de datos (NO tiene implementado cortocircuito).
- Los saltos se resuelven en la etapa ID. Después del salto siempre se busca la instrucción siguiente y en caso de que se produzca el salto se anula la instrucción buscada.

Se ejecuta en dicho procesador una aplicación con las siguientes características:

- El 20 % de las instrucciones son saltos condicionales y el 40 % de los saltos se realizan. No hay dependencias de datos en las instrucciones de salto.
- El 18% de las veces, las instrucciones I_{i+1} tienen una dependencia de LDE con las instrucciones I_i (el 30 % de éstas corresponden a instrucciones de load).
- El 6% de las veces las instrucciones I_{i+2} tienen dependencias de LDE con la instrucción I_i, (el 30 % de éstas corresponden a instrucciones de load), y en esos casos nunca hay dependencias entre I_{i+1} e I_{i+2}.

Calcular:

a) Los ciclos por instrucción de dicho procesador.

b) Los ciclos por instrucción si al procesador se le añade cortocircuito.

c) El speep-up del segundo caso frente al primero.

2. El siguiente fragmento de código se ejecuta en un ARM segmentado de cinco etapas:

```
sub   R1,R2,R3
add   R4,R5,R6
sub   R5,R4,R8
add   R7,R2,R3
add   R9,R7,R3
ldr   R1,10(R6)
add   R3,R1,R4
sub   R6,R7,R8
```

Suponiendo que se puede escribir un dato en el banco de registros y leer su nuevo valor en el mismo ciclo, calcular el número de ciclos necesarios para ejecutar este código en los siguientes casos:

a) No existe la posibilidad de anticipar operandos (NO tiene implementado cortocircuito) ni de reordenar el código.

b) Existe anticipación de operandos (tiene implementado cortocircuito), pero NO existe la posibilidad de reordenación de código.

c) No hay anticipación de operandos (NO tiene implementado cortocircuito) pero existe la posibilidad de reordenación de código. Reordenar el código para conseguir que el número de ciclos sea mínimo.

3. Sobre la estructura del computador ARM segmentado, se ejecuta la siguiente secuencia de instrucciones:

```
ADD R1, R2, R3
SUB R4, R2, R3
AND R5, R2, R3
OR  R6, R2, R3
```

Si la instrucción ADD está colocada en la dirección de memoria 00002000 (Hex), y el contenido de los registros es:

R1=00000005
R2=00000004
R3=00000002
PC=00002000

a) Señalar el contenido de los siguientes registros al cabo de 4 ciclos de reloj: PCSrcW, PCSrcM, PCSrcE, PCSrcD, RD1, RD2, ALUResulE, ALUOutM, ALUOutW, WA3E, WA3M, WA3W, BranchD, BranchE

4. Supongamos un computador como el ARM segmentado en cinco etapas con las siguientes características:

- Posee anticipación de operandos (tiene implementado cortocircuito).
- Posee una sola memoria cache para el almacenamiento de instrucciones y datos. Por esta razón no resulta posible leer una instrucción y realizar la lectura o escritura de un dato en el mismo ciclo de reloj (no puede coincidir etapa MEM de un load o store con otra instrucción en etapa IF, en ese caso la instrucción que entraría en IF se queda en parada en esa etapa hasta que se libere la memoria).
- Las escrituras en el banco de registros se hacen en la primera mitad de la fase WB, mientras que las lecturas se hacen en la segunda mitad de la fase WB.

Supongamos que este computador ejecuta el siguiente programa:

```
BUCLE:  LDR   R2,4(R6)
        LDR   R3,8(R6)
        SUB   R2,R2,R3
        ADD   R2,R2,R1
        SUB   R6,R6,#4
        STR   R2, 10(R6)
        CMP   R6, #0
        BNE   BUCLE
        ADD   R1,R1,#1
        SUB   R3,R3,R7
```

Se supone que el valor inicial de R6 es 2000.

a) Construye el diagrama de tiempo correspondiente a la primera iteración del bucle, indicando sobre el diagrama los cortocircuitos que se activan.
b) Calcula el valor del CPI
c) Si el computador trabaja con una frecuencia de 1 GHz, determina el rendimiento en ARM.

5. Sea un ARM segmentado de cinco etapas con las siguientes características:

- Un dato se puede leer y escribir en el banco de registros en el mismo ciclo de reloj.
- Existe anticipación de operandos (tiene implementado cortocircuito).
- La detección de riesgos LDE y generación de paradas se realiza en la etapa de decodificación

- Los riesgos estructurales referidos a memoria se detectan y se resuelven mediante espera en la última etapa de cada unidad funcional
- Los riesgos de EDE entre dos instrucciones A y B tal que A precede a B se resuelven mediante inhibición de escritura de la instrucción A.
- Las unidades funcionales del procesador son:

UF	Cantidad	Latencia	Segmentación
FP ADD	1	2	sí
FP MUL	1	5	sí
Int ALU	1	1	No

Sabiendo que el siguiente fragmento de código se ejecuta sobre dicho procesador

```
LD      F10, 0(R1)
MULD    F4, F0, F10
LD      F12, 0(R2)
ADDD    F2, F12, F4
LD      F4, 8(R1)
MULD    F12, F4, F12
LD      F12, 16(R1)
```

a) Representar el diagrama instrucción-tiempo para la primera iteración e indicar los cortocircuitos realizados. Indicar claramente las paradas y sus causas.
b) Determinar el CPI

6. Sea un procesador segmentado de cinco etapas con las siguientes características:
- Un dato se puede leer y escribir en el banco de registros en el mismo ciclo de reloj.
- Existe anticipación de operandos.
- La detección de riesgos LDE y generación de paradas se realiza en la etapa de decodificación.
- Los riesgos estructurales referidos a memoria se detectan y se resuelven mediante espera en la última etapa de ejecución.
- Los riesgos de EDE entre dos instrucciones A y B tal que A precede a B se resuelven mediante inhibición de escritura de la instrucción A.
- Las unidades funcionales de las que dispone el procesador son las siguientes:

UF	Cantidad	Latencia	Segmentación
FP ADD	1	2	No
FP MUL	1	3	No
INT ALU	1	1	No

a. Completar el diagrama instrucción-tiempo para el siguiente fragmento de código, indicando claramente los cortocircuitos realizados, las paradas y sus causas.

```
LD      F1, 0(R1)
LD      F2, 0(R2)
MULD    F2, F2, F1
MULD    F3, F3, F1
ADDD    F4, F2, F1
```

b. Completar el diagrama instrucción-tiempo para el siguiente fragmento de código, indicando claramente los cortocircuitos realizados, las paradas y sus causas.

```
MULD  F1, F2, F3
ADDD  F1, F1, F2
SD    F1, 0(R1)
```

c. Completar el diagrama instrucción-tiempo para el siguiente fragmento de código, indicando claramente los cortocircuitos realizados, las paradas y sus causas.

```
MULD  F1, F2, F1
MULD  F1, F3, F1
MULD  F5, F4, F2
```

d. Calcular el CPI de la ejecución completa del siguiente código, siendo R0=0 y el R2=32 al comienzo de la ejecución.

```
loop: ADDD  F7, F7, F6
      AND   R6, R2, #1
      BEQ   R6, R0, if
      ADD   R3, R3, #2
      ADD   R5, R3, R5
 if:  ADDD  F7, F7, F1
      SUB   R2, R2, #1
      ADD   R4, R3, R5
      CMP   R0, R2
      BNE   loop
      SD    F7, 256(R1)
```

7. En un procesador segmentado es necesario ejecutar el siguiente código:

```
for (n = 0; n < 100; n++) {
    for (k = 0; k < 4; k++) {
       out[n] += filter[k] * input[n+k];
    }
}
```

Queriendo conseguir el mayor rendimiento, se decide realizar la siguiente implementación en ensamblador:

```
loop_n :  ADD   R2, R0, #0
loop_k :  LD    F3, 0(R3)    ; lee de memoria 4 bytes (load de un float)
          LD    F4, 0(R4)    ; lee de memoria 4 bytes (load de un float)
          MULD  F6, F3, F4   ; mult. en punto flotante (de dos float)
          LD    F5, 0(R5)    ; lee de memoria 4 bytes (load de un float)
          ADDD  F5, F6, F5   ; suma en punto flotante (de dos float)
          SD    F5, O(R5)    ; escribe en memoria 4 bytes (store de un float)
          ADD   R3, R3, #4   ; actualizo puntero a filter
          ADD   R4, R4, #4   ; actualizo puntero a input
          ADD   R2, R2, #1
          CMP   R2, #4
          BLT   loop_k   ; salta a loop_k si R2 < 4
          ADD   R5, R5, #4     ; avanzamos el puntero sobre out
```

```
SUB     R3, R3, #16        ; R3 vuelve a apuntar a filter[0]
SUB     R4, R4, #12    ; R4 apunta al siguiente input[n]
ADD     R1, R1, #1
CMP     R1, #100
BLT     loop_n  ; salta a loop_n si R1 < 100
```

Se supone que:

- Inicialmente R1=0, R3 apunta a filter[0], R4 apunta a input[0] y R5 apunta a out[0]
- Un dato se puede escribir en un registro y leer su valor en el mismo ciclo.
- Se dispone de lógica de cortocircuito (forwarding).
- La detección de todo tipo de riesgos (estructurales y LDE) y generación de paradas se realiza en la etapa de decodificación.
- Los riesgos EDE se resuelven mediante paradas hasta que la instrucción ya lanzada entre en la etapa de memoria.
- Dos instrucciones no pueden acceder simultáneamente a la etapa de acceso a memoria ni tampoco a la de escritura en el banco de registros.
- Los saltos se resuelven en la etapa de ejecución y usan cortocircuito.
- Se dispone de las siguientes unidades funcionales:

UF	Cantidad	Latencia	Segmentación
FP ADDD/ SUBD	1	2	Sí
FP MULD	1	3	Sí
INT ALU	1	1	No

a) Rellena 25 ciclos del diagrama instrucción-tiempo correspondiente al comienzo del bucle interno con k=3 y n = 0.

b) A partir del apartado anterior, estima el número de ciclos que tardará en ejecutarse el código completo y determina el CPI. Indica claramente los cálculos que realices para llegar a los resultados.

8. Supongamos un procesador segmentado de cinco etapas en el que se ejecutan 2 aplicaciones diferentes. La aplicación A tiene un 17% de instrucciones con una penalización de 3 ciclos de reloj y un 12% de instrucciones con una penalización de 2 ciclos de reloj. La aplicación B tiene un 25% de instrucciones con 1 ciclo de penalización.

a) ¿Cuál es el CPI de cada una de las aplicaciones?
b) ¿Cuál es el CPI promedio del procesador?
c) ¿Cuál es el speedup frente al procesador sin segmentar?
d) ¿Cuál es la eficacia frente al procesador segmentado ideal?

9. Sea un procesador ARM segmentado con las siguientes características:

- Se puede escribir un dato en el banco de registros y leer su nuevo valor en el mismo ciclo
- Los saltos se resuelven en la etapa de ejecución y se cancelan las siguientes instrucciones si el salto es tomado.
- La detección de todo tipo de riesgos y generación de paradas se realiza en la etapa de decodificación.
- En el caso de los riesgos EDE se produce una parada hasta que la instrucción ya lanzada entre en la etapa de memoria (en ese momento la segunda instrucción saldría de su etapa ID).
- Los registros r1 y r2 contienen inicialmente el valor 4.
- Dos instrucciones no pueden acceder simultáneamente a la etapa de acceso a memoria ni tampoco a la de escritura en el banco de registros.
- Se dispone de las siguientes unidades funcionales:

UF	Cantidad	Latencia	Segmentación
FP ADD	1	2	No
FP MUL	1	3	No
FP DIV	1	5	No
Int ALU	1	1	No

En este procesador se ejecuta el siguiente fragmento de código:

```
L0:  ADDD  F2, F4, F0
     SD    F2, 0(r1)
     DIVD  F4, F4, F0
     ADDD  F4, F0, F2
     MULD  F2, F2, F4
     LD    F4, 0(r1)
     ADDD  F0, F2, F4
     ADDD  F8, F6, F8
     SUB   r2, r2, #1
     ADD   r1, r1, #1
     CMP   R2, #0
     BNE   L0
     ADDD  F0, F8, F2
     ADDD  F2, F0, F8
     end
```

a) Representar el diagrama instrucción-tiempo e indicar los cortocircuitos realizados. Indicar claramente las paradas y sus causas.
b) Calcular los CPI (ciclos por instrucción) resultantes de la ejecución del código.

10. Sea un ARM segmentado con las siguientes características:

- Un dato se puede escribir en el banco de registros y leer su valor en el mismo ciclo.
- Se dispone de lógica de cortocircuito (*forwarding*).
- Los saltos se resuelven en la etapa de ejecución y se cancelan las siguientes instrucciones si el salto es tomado.
- La detección de todo tipo de riesgos (estructurales y LDE) y generación de paradas se realiza en la etapa de decodificación.
- Los riesgos EDE se resuelven mediante inhibición de escritura.
- Dos instrucciones no pueden acceder simultáneamente a la etapa de acceso a memoria ni tampoco a la de escritura en el banco de registros.
- Se dispone de las siguientes unidades funcionales:

UF	Cantidad	Latencia	Segmentación
FP ADDD	1	3	Sí
FP SUBD	1	3	Sí
FP MULD	1	4	Sí
FP DIVD	1	5	No
INT ALU	1	1	No

En este procesador se ejecuta el siguiente fragmento de código

```
     ADD   r3, r0, #3
L1 : SUBD  F2, F6, F8
     SUBD  F4, F8, F6
     SD    F4, 0(r3)
```

```
DIVD    F2, F4, F8
ADDD    F2, F8, F8
SUB     r3, r3, #1
DIVD    F6, F4, F8
MULD    F4, F2, F6
SUBD    F10, F2, F6
LD      F4, 0(r3)
ADDD    F0, F4, F2
CMP     r3, #0
BNE     L1
MULD    F4, F2, F2
end
```

a) Representar el diagrama instrucción-tiempo para la primera iteración e indicar los cortocircuitos realizados. Indicar claramente las paradas y sus causas.
b) A la vista del diagrama obtenido, y sin necesidad de desarrollar todas las iteraciones, calcular el número de ciclos que toma la ejecución completa del código.