Unidad III: Paralelismo en monoprocesadores

Trabajo Práctico Nº 3: Planificación en ILP

1.- Planifique el siguiente fragmento de programa usando el algoritmo de planificación de bloque básico Arriba-Abajo:

```
…
X = A + B – (C * A) / X
Y = X + D * F
If (Y > 1000) then
…
```

Considere los siguientes retardos
Multiplicación-División: 3 ciclos
Suma-Resta: 1 ciclo
Lectura: 2 ciclos
Escritura: 1 ciclo

2.- Ejecute el fragmento resultante del ejercicio 1 en un procesador escalar (pipeline paralelo de grado **s**=1) de 2 unidades funcionales multifunción (ALU y acceso a memoria cada una). Repita el procedimiento para un superescalar de grado 2 con 2 unidades funcionales multifunción (ALU y acceso a memoria cada una). ¿Cuánto mas rápido se ejecuta el código en el superescalar?

3.- Repita el ejercicio 2 pero considerando en ambos casos 2 unidades funcionales dedicadas: una ALU y una para acceso a memoria. ¿Cómo son los rendimientos en comparación a los del ejercicio 2?

4.- Dado el siguiente fragmento de programa en pascal tradúzcalo a assembler de DLX.

```
var   a : array [1 .. 1000] of real;
      b : array [1 .. 1000] of integer;
      i : integer;

for i:=1 to 1000 do
begin
      a[i]:= a[i] * 5;
      b[i]:= b[i] * 5;
end;
```

5.- Suponga que se dispone de un procesador superescalar de grado 2 con 3 vías de ejecución: una ALU de punto flotante, una ALU de punto fijo y una unidad de acceso a memoria (para carga y almacenamiento de enteros y punto flotante indistintamente).

a) Realice una optimización (planificación de bloque básico), del bloque de sentencias internas del bucle del ejercicio anterior (en DLX) a fin de minimizar el número de ciclos de reloj necesario para su ejecución.
b) ¿Cuál es la aceleración lograda respecto de la versión sin optimizar ejecutada en el mismo procesador?

Considere los siguientes retardos	
Operaciones P. Flotante: 3 ciclos	Operaciones Enteras: 1 ciclo
Carga datos P. Flotante: 2 ciclos	Carga datos Enteros: 1 ciclo
Instrucciones de salto: 2 ciclos	Almacenamientos: 1 ciclo

6.- Aplique la técnica de **loop unrolling** al programa del ejercicio 4.

a) Suponer un procesador escalar (grado 1) con cuatro unidades funcionales multifunción (entero, flotante, memoria c/u).
b) ¿Cuál es la aceleración obtenida respecto a la versión secuencial sin el desenrollado de lazo?
c) ¿Cuál sería la aceleración si el procesador fuese un superescalar de grado 2 con 2 unidades funcionales multifunción (entero, flotante, memoria c/u)?
d) ¿Cuál sería la aceleración si el procesador fuese un superescalar de grado 4 con 5 unidades multifunción especializadas (2 de enteros, 2 de punto flotante, 1 de memoria)?

7.- Aplique la técnica de **software pipelining** al fragmento de programa siguiente. Suponga que se dispone de un procesador superescalar con 2 unidades de multiplicación de 3 ciclos segmentada y 1 de suma de 1 ciclo.
¿Cuál es la aceleración obtenida respecto de la versión secuencial sin optimizaciones?

```
var   a : array [1 .. 500] of integer;
      b : array [1 .. 500] of integer;
      i : integer;
for i := 3 to 502 do
begin
      A[i]:= A[I-2] * B[i-1];
      B[i]:= A[I-1] + 5;
end;
```

8.- Suponga un sistema que dispone de:

- una unidad LD/SD (2 ciclos)
- una unidad FP (3 ciclos)
- una ALU para operaciones con enteros (2 ciclos)
- una unidad de tratamiento de saltos (1 ciclo)

El siguiente código corresponde a la suma de un escalar "**s**" a los elementos de un vector "**x**".

Código	Compilación básica	Emisión *
`for (i=1,i < 100, i++)` `  x[i]=x[i]+s;`	I_1: **Loop**: *LD* F0, 0(R1) I_2: *ADDF* F4, F0,F2 I_3: *SD* F4, 0(R1) I_4: *ADDI* R1, R1,#8 I_5: *BNE* R1, R2, **Loop** I_6: ...	n+1 n+3 n+6 n+8 n+10

* Relativa al ciclo n

Valores iniciales

- El escalar "**s**" está contenido en **F2**
- **R1** apunta al primer componente del vector **x**.
- 8(**R2**) apunta al último componente del vector **x**.

a) Indique los ciclos que tarda la ejecución de 4 iteraciones del bucle.

b) ¿Qué mejora se obtiene en los ciclos de ejecución si un compilador realiza el desenrollamiento de las 4 iteraciones? Suponga que el compilador evita las dependencias entre las instrucciones de cada iteración utilizando renombramiento de registros y para ello dispone de 16 registros FP (F0, F1, F2,… F15) y además recalcula el desplazamiento adecuado para cada referencia de memoria en base al contenido de R1.

c) Dibuje el grafo de dependencias y realice una planificación estática del código del inciso b), para mejorar el rendimiento.

d) Indique los ciclos de la ejecución del código correspondiente a las cuatro iteraciones en este último caso.

9.- Considere el siguiente fragmento de código.

```
S1:   A = B * C
S2:   D = E + F * P
S3:   G = H + A / D
S4:   O = D – P << 2
S5:   A = X + Y and Z
S6:   D = O * A
```

a) Generar la secuencia de operaciones básicas (*load*, *operation*, *store*)

b) Dada una máquina VLIW con las unidades funcionales definidas a continuación, genere (empíricamente) una secuencia de instrucciones VLIW que distribuya las operaciones del inciso a) de forma que no viole ninguna dependencia.

MUL/DIV	ADD/SUB	ADD/SUB	Logic Op	Shift/Rot	R/W1	R/W2

Considere los siguientes retardos

Multiplicación-División: 3 ciclos
Suma-Resta: 2 ciclos
Operaciones lógicas: 2 ciclos
Shift-Rotación: 1 ciclo
Lectura-Escritura: 1 ciclo

c) Calcular el **speedup** de la arquitectura considerando que en su versión secuencial las instrucciones se ejecutan en un procesador con una única unidad funcional (multifunción).

10.- Un procesador que emite una instrucción VLIW en cada ciclo dispone de los siguientes formatos de instrucción:

Tipo	Op1	Op2	Op3
0	Operación Mem	Operación FP	Operación FP
1	Operación Mem	Operación FX	Operación FX/Salto

Considere los siguientes retardos
Operación Memoria: 2 ciclos
Operación FP: 2 ciclos
Operación FX/Salto: 1 ciclo

Suponiendo que se ejecuta el siguiente código:

```
I1:  Lazo: LOAD   F4, M (R1)
I2:        LOAD   F2, M (R1+100)
I3:        MULTF  F2, F2, F0      //Op FP
I4:        SUBF   F4, F2, F4      //Op FP
I5:        STORE  M(R1), F4
I6:        SUB    R1, R1 #8       //Op FX
I7:        BNEZ   R1, Lazo
```

a) Indique, mediante un grafo, la dependencia entre las instrucciones y los ciclos de espera que existan entre ellas.

b) Escriba la codificación VLIW que resulta de compilar, sin ninguna optimización, el programa que corresponde al código anterior.

Tipo	Instrucción VLIW			Ciclo
(0 o 1)	Op1	Op2	Op3	Ciclo

c) ¿Cuántas unidades funcionales debe tener el procesador para poder ejecutar sin problemas estructurales los dos tipos de instrucciones que se han definido?

d) Optimice el código VLIW para lograr una mayor eficiencia en el uso de las unidades funcionales e indique el IPC alcanzado durante la primera iteración sin desenrollar el bucle.

e) Repita el apartado anterior desenrollando el bucle y calcule el IPC cuando han transcurrido 10 ciclos.