Apellidos _____ Nombre _____ Grupo _____

Arquitectura de Computadores. Examen final 27/01/2021. Teoría.

Puntuación: Cada pregunta vale 1,25 puntos. **Tiempo:** 70 minutos

1. Para mejorar la velocidad de una aplicación, se ejecuta un procedimiento, P, que consumía el 90% del tiempo, sobre un cierto número de procesadores, NProc, en paralelo. De esta forma se consigue que P se ejecute Nproc veces más rápido. El 10% restante no admite la ejecución en paralelo. Supón que puedes seleccionar el valor de Nproc y explica razonadamente si la eficiencia aumenta o disminuye al incrementar Nproc.

Usamos la definición de Eficiencia: $$\mathbf{E} = \frac{Speedup}{x} = \frac{\frac{1}{(1-F)+\frac{F}{x}}}{x} = \frac{1}{x(1-F)+F}$$

En este caso, x = NProc y F=0,9, lo que da lugar a:

$$\mathbf{E} = \frac{1}{Nproc(0{,}1)+0{,}9}$$

Luego, al aumentar Nproc disminuye la Eficiencia.

2. Explica brevemente en qué consiste el concepto de "software pipeline", e indica cuáles son sus principales ventajas e inconvenientes en comparación con el desarrollo de bucles.

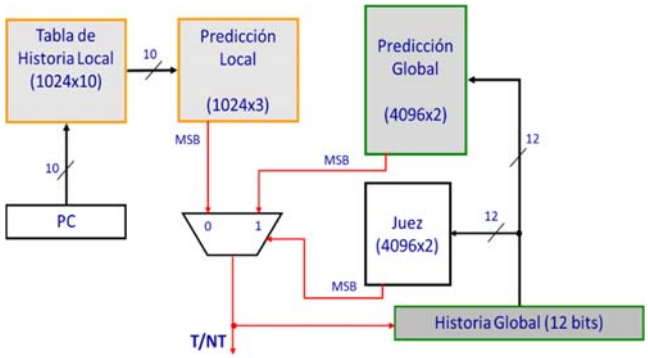
Dado un bucle en el que las sucesivas iteraciones son independientes, un bucle sw pipeline se crea a base de tomar instrucciones que pertenecen a diferentes iteraciones del bucle inicial, al objeto de aumentar la distancia entre instrucciones dependientes.

En comparación con el desarrollo del bucle original, el sw pipeline presenta las ventajas de que no incrementa el tamaño del código, no requiere de registros adicionales para reescribir el código y evita dependencias en el cuerpo del bucle. Por el contrario, el sw pipeline no reduce el nº de saltos y necesita de un código especial para el inicio (cabecera) y la finalización (cola).

3. Supongamos un predictor de saltos como el implementado en el Alpha 21264 (tal como muestra la figura de la derecha). Supongamos, además, que el siguiente bucle se itera 1000 veces:

Label: Instr. 1
Instr. 2
Instr. 3
BEQ R1, R2, Label

Suponiendo que todas las tablas y el registro de historia global están a cero antes de comenzar la ejecución del bucle, explica razonadamente a partir de qué iteración el predictor comienza a acertar.



Iter	Valor THL=Dir. TPL accedida	Valor en TPL(Dir)	Pred TPL	Nuevo valor en TPL(Dir)	Valor RHG = Dir acceso a TPG	Valor en TPG(Dir)	Pred TPG	Nuevo valor TPG(Dir)
1	00 00 00 00 00	000	N	001	00 00 00 00 00 00	00	N	01
2	00 00 00 00 01	000	N	001	00 00 00 00 00 01	00	N	01
3	00 00 00 00 11	000	N	001	00 00 00 00 00 11	00	N	01
…								
9	00 11 11 11 11	000	N	001	00 00 11 11 11 11	00	N	01
10	01 11 11 11 11	000	N	001	00 01 11 11 11 11	00	N	01
11	11 11 11 11 11	000	N	001	00 11 11 11 11 11	00	N	01
12	11 11 11 11 11	001	N	010	01 11 11 11 11 11	00	N	01
13	11 11 11 11 11	010	N	011	11 11 11 11 11 11	00	N	01
14	11 11 11 11 11	011	N	**100**	11 11 11 11 11 11	01	N	**10**
15	11 11 11 11 11	100	**T**	**101**	11 11 11 11 11 11	10	**T**	**11**
16	Etc.							

Ambos predictores comienzan a acertar a partir de la iteración 15.

4. (0,25 puntos cada apartado) Consideremos un procesador superescalar en desorden, que hace el renombramiento de registros mediante un Fichero de Registro Unificado. Indica qué cambios se producen en las "Register Alias Tables" (si es que hay algún cambio) en los siguientes casos.
a) Se hace la fase de "rename" de una instrucción de la forma: LD F2, 4(R1)
b) Se hace la fase de "rename" de una instrucción de la forma: BEQ R2, R3, Label
c) Se hace la fase de "rename" de una instrucción de la forma: MUL.D F4, F2, F3
d) Se hace la fase "commit" de una instrucción de la forma: SD 0(R1), F4
e) Se hace la fase "commit" de la instrucción descrita en el apartado b), asumiendo que la predicción del salto fue incorrecta.

a) Se busca un RF libre, digamos RFx, y se asigna Frontend RAT(F2) ← RFx

b) Las RATs no se modifican, dado que no hay ningún registro destino en esta instrucción

c) Se busca un RF libre, digamos RFx, y se asigna Frontend RAT(F4) ← RFx

d) No ocurre nada en las RATs, dado que el destino de SD no es ningún registro. *(Nota adicional: lo único que se hace es Mem(0+R1) ← Valor almacenado en cabecera de la Store Queue. Esta explicación adicional no se pide como parte de la respuesta, dado que no hay modificación de las RATs)*

e) Los valores de la Frontend RAT deben eliminarse debido a la predicción incorrecta. Para reconstruir la FE RAT a un estado válido, se pueden copiar todas las entradas de la Ret RAT sobre la FE RAT.

5. Explica brevemente en qué consiste el concepto de "wake-up anticipado" de instrucciones y cuál es el problema que esta técnica pretende resolver. En la respuesta puedes utilizar un diagrama de tiempo, si lo estimas conveniente.

En principio, cuando una instrucción finaliza su ejecución y hace la fase de escritura (write-back), las instrucciones que están en la(s) Issue Queue(s) comparan sus marcas internas con la marca (nº de ROB, nº de registro físico) de la instrucción que está haciendo la fase de escritura. Si hay coincidencia en una o varias filas de la(s) Issue Queue(s), las instrucciones correspondientes pueden ser despertadas. Desde ese momento, hasta que comienza su ejecución pueden transcurrir varios ciclos de reloj (select, drive), provocando la aparición de burbujas al ejecutar instrucciones que están muy próximas en el programa y que presentan dependencias LDE

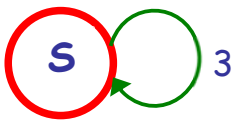
Con el wake-up anticipado, cuando se selecciona una instrucción, I, que genera un resultado, se manda la señal de wake-up en el mismo ciclo (si la latencia de la instrucción I es de 1 ciclo) o N-1 ciclos de reloj más tarde (si la latencia de I es de N ciclos). De este modo, una instrucción despertada, J, con dependencia LDE respecto de I, puede llegar a la entrada la UF donde se tiene que ejecutar en el mismo ciclo en que la instrucción I está produciendo el resultado, con lo que J puede recibir su operando mediante by-pass. Con ello se evitan burbujas en el pipeline y se gana tiempo, a costa de incrementar la complejidad de las interconexiones entre UFs. (Alternativa: hacer la explicación con un gráfico).

6. Explica razonadamente si el bucle

```
for (i=0; i<64; i=i+1)
      A[i+3] = A[i] - B[i];
```

es vectorizable.

Se puede construir el grafo de dependencias:



Con ello se ve que hay una dependencia de la única sentencia con ella misma con una distancia de tres iteraciones. Es decir, los resultados calculados en una iteración se usan como operandos tres iteraciones más tarde. Esto implica la necesidad de ejecutar los cálculos secuencialmente, y por lo tanto el bucle no es vectorizable.

Alternativa. Se pueden desarrollar algunas iteraciones del bucle y comparar el resultado de la ejecución secuencial con el código vectorial correspondiente.

Escalar:

i=0 A[3] = A[0] - B[0];

i=1 A[4] = A[1] - B[1];

i=2 A[5] = A[2] - B[2];

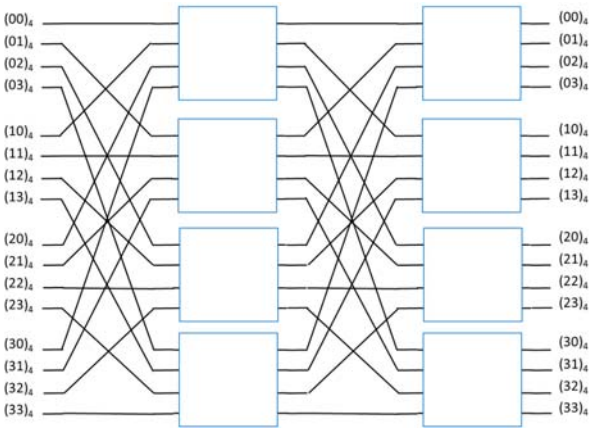
i=3 A[6] = A[3] - B[3];

Etc.

7. Supongamos la red de interconexión mostrada en la figura.

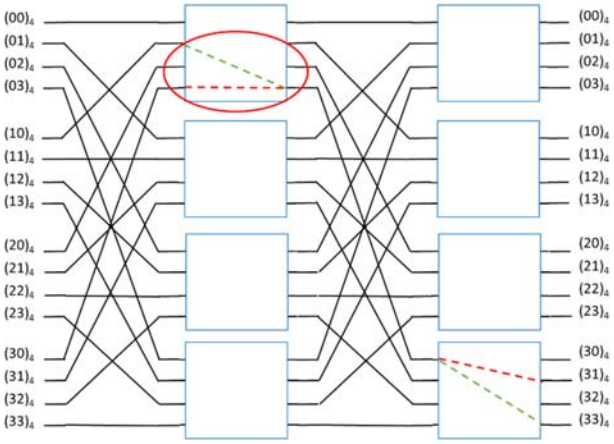
a) Explica qué tipo de red es, de acuerdo con la clasificación estudiada en clase ¿Qué tipo de permutaciones entre etapas utiliza?

b) Explica razonadamente si las comunicaciones $(30)_4 \rightarrow (31)_4$ y $(10)_4 \rightarrow (33)_4$ son compatibles.

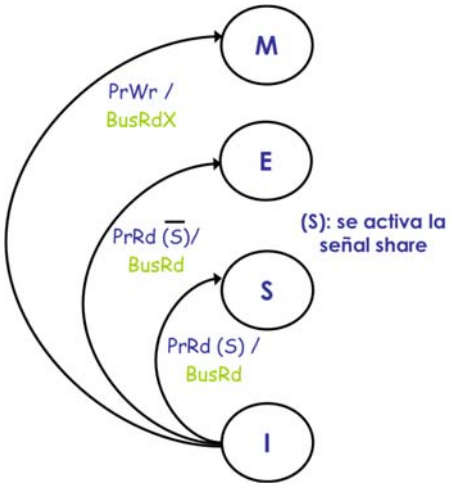


a) Es una Red MIN Omega en base 4 (k=4, n=2; $N=4^2=16$ nodos), dado que todas las permutaciones son de baraje perfecto en base 4 (σ), excepto la permutación final, que es una identidad.

b) Sobre la figura proporcionada se pueden trazar los dos caminos. Se comprueba que, en el conmutador superior de la primera etapa, ambos caminos necesitan usar la misma salida. Por tanto, no son compatibles.



8. En un multiprocesador con coherencia de cache tipo Snoopy que utiliza el protocolo MESI, muestra con un diagrama las transacciones que son posibles desde el estado I. Comenta el significado de cada una de ellas.



Si un procesador ejecuta una operación de lectura de memoria (PrRd) que produce un fallo de cache, la dirección a leer se manda a través del bus mediante una orden BusRd.

Si alguno de los controladores de cache conectados al bus detecta que la dirección solicitada se encuentra en su cache, activará la señal compartida, S, para que adopte el valor 1. El procesador que generó el fallo de cache detecta el valor de S, y pone el bloque que recibe a través del bus en su cache en estado Compartido (transición I ➔ S). En caso contrario (ninguna cache tiene una copia del bloque), la señal S permanecerá en cero, y el procesador pondrá el bloque en estado Exclusivo (transición I➔E).

Finalmente, si un procesador ejecuta una operación de escritura de memoria (PrWr) que produce un fallo de cache, la dirección a escribir se manda a través del bus mediante una orden BusRdX (lectura con exclusividad). El procesador que generó el fallo pondrá el nuevo bloque obtenido a través del bus en estado Modificado (transición I➔M).

Apellidos _____ Nombre _____Grupo_____

Arquitectura de Computadores. Examen final 27/01/2021. Problemas.

Tiempo: 1 hora y 50 minutos

1. (1,5 puntos) En la ejecución de un cierto programa, P, un procesador, que trabaja a una frecuencia de 1 GHz, ejecuta 1200 millones de instrucciones en 3 segundos. Sabemos que la distribución de instrucciones de P es la siguiente: enteras (50%), FP (25%), memoria (20%) y saltos (5%). Además, se sabe que CPI_{SALTOS}=1,1, CPI_{MEM} = 1,5 y que CPI_{FP} es el doble de CPI_{INT}. Calcula el valor de CPI_{FP}.

Solución:

T = Nº instr x CPI x t. En este caso: $3 = 1.2 \times 10^9 \times CPI \times 1 \times 10^{-9} \rightarrow CPI = 3/1.2 = 2.5$

$2.5 = 0.5 \times CPI_{INT} + 0.25 \times CPI_{FP} + 0.2 \times CPI_{MEM} + 0.05 \times CPI_{BRANCH} =$

$= 0.5 \times CPI_{INT} + 0.25 \times 2 \times CPI_{INT} + 0.2 \times 1.5 + 0.05 \times 1.1.$

$2.5 = CPI_{INT} + 0.355$; $CPI_{INT} = 2.145$; **$CPI_{FP} = 2 \times 2.145 = 4.29$**

2. (3 puntos) Disponemos de un procesador con planificación dinámica de instrucciones mediante el algoritmo de Tomasulo, con especulación, que utiliza un ROB de 5 entradas. Se dispone de las unidades funcionales que se muestran en la tabla siguiente:

Tipo UF	Nº de UFs	Latencia (ciclos)	Segmentada	Nº de Buffers o ER
FP *	1	7	Si	1
FP /	1	12	Si	2
FP + y -	1	4	Si	2
Load	1	2	SI	6
Store	1	2	SI	
Enteros	1	1		6

Además, tiene las siguientes características.

- Los datos que se escriben en la etapa de escritura NO se pueden usar para la ejecución de una instrucción hasta el ciclo siguiente
- Hay un solo bus común de datos (CDB)
- Las estaciones de reserva se liberan al final de la fase de escritura
- Las entradas del ROB se liberan al final de la fase commit

Indica en qué ciclo, o ciclos, de reloj se realiza cada una de las etapas de ejecución para las instrucciones siguientes. Utiliza llamadas en la tabla para explicar separadamente las dependencias que motivan cualquier retraso que se produzca en la temporización. Nota: SUBI y BNEZ son instrucciones enteras.

#	
1	L.D F2, 0(R1)
2	MUL.D F0, F2, F4
3	MUL.D F4 F0, F4
4	ADD.D F2, F2, F0
5	L.D F0, 0(R2)
6	L.D F6, 0(R3)
7	L.D F8, 0(R4)
8	DIV.D F10, F2, F8
9	DIV.D F4, F6, F4
10	SUB.D F2, F6, F0
11	ADD.D F8, F2, F8
12	S.D 0(R1), F2
13	SUBI R1, R1, #8

14	BNEZ R1, Label

Solución:

Como el nº de instrucciones L.D es menor que el nº de Load Buffers, no es preciso llevar la cuenta de su ocupación. Lo mismo ocurre con las ER correspondientes a la UF de operaciones enteras.

#		Issue	Exec	Write	Com	ROB#	* 1ER	/ 2ER	+ - 2 ER
1	L.D F2, 0(R1)	1	2-3	4	5	1			
2	MUL.D F0, F2, F4	2	LDE 5-11	12	13	2	1		
3	MUL.D F4 F0, F4	ER 13	14-20	21	22	3	1		
4	ADD.D F2, F2, F0	14	15-18	19	23	4	1		1
5	L.D F0, 0(R2)	15	16-17	18	24	5	1		1
6	L.D F6, 0(R3)	16	17-18	CDB 20	25	1	1		1
7	L.D F8, 0(R4)	17	18-19	CDB 22	26	2			
8	DIV.D F10, F2, F8	ROB 23	24-35	36	37	3	0	1	0
9	DIV.D F4, F6, F4	24	25-36	37	38	4		2	
10	SUB.D F2, F6, F0	25	26-29	30	39	5		2	1
11	ADD.D F8, F2, F8	26	LDE 31-34	35	40	1		2	2
12	S.D 0(R1), F2	27		LDE 30	41	2		2	
13	SUBI R1, R1, #8	ROB 38	39	40	42	3		0	0
14	BNEZ R1, Label	39	LDE 41	42	43	4			

3. (3,5 puntos) Se desea realizar el cálculo:

```
for (i=0;i<107;i++)
    D[i] = A[3,i]*B[i,3] + (A[i,i]*C[i])/3.5;
```

donde A y B son matrices de 107x107 componentes, que están almacenadas por filas en la memoria de un computador vectorial, CompV, a partir de las direcciones indicadas por los registros Ra y Rb, respectivamente. Los vectores C y D, de 107 componentes, están almacenados en la memoria de CompV, a partir de las direcciones indicadas por los registros Rc y Rd, respectivamente.

CompV tiene las mismas características del VMIPS, excepto:

- La memoria está organizada en 64 bancos
- Tiene 2 pipes de load /store
- El ciclo de reloj dura 0.5 ns
- La latencia de los pipes de load/store es de 7 ns

a) (0,75 puntos) Realiza las comprobaciones necesarias para averiguar si se pueden realizar todos los accesos a memoria sin conflictos.

b) (1,25 puntos) Escribe un programa en el lenguaje máquina simbólico del VMIPS que ejecute el cálculo, utilizando strip- mining y tratando de minimizar el tiempo de ejecución. Notas: i) dentro de un mismo convoy no pueden coexistir instrucciones que presenten un riesgo estructural; ii) las instrucciones vectoriales deben acompañarse de comentarios; iii) se pide solamente el código vectorial, organizado en convoyes, y con comentarios que expliquen su significado (no hace falta el código escalar); iv) se puede asumir que los registros enteros y de PF utilizados contienen los valores adecuados, pero se debe especificar claramente cuáles son esos valores.

c) (1,5 puntos) Construye el diagrama de tiempo correspondiente al código del apartado anterior, y calcula el rendimiento obtenido, en MFLOPS.

Solución:

a) El acceso a B[i,3] implica el acceso a un vector columna de B. Como cada fila de B tiene 107 componentes, el stride para este acceso es de 107 palabras.

107 es un nº primo ➔ Acceso sin conflictos si Nº bancos >= Tacceso a memoria.

Nº bancos=64; Tacceso=7ns / 0,5ns/ciclo = 14 ciclos. Luego, el acceso a B[i,3] está libre de conflictos.

El acceso a A[i,i] (diagonal principal de A) tiene un stride = (nº elementos por fila)+1 = 108

El acceso sin conflictos requiere que mcm(stride, Nº bancos) / stride >= Tacceso a memoria.

stride = 108 = 2^2x3^3 ; Nº bancos = 2^6 ; Luego, mcm = 2^6x3^3

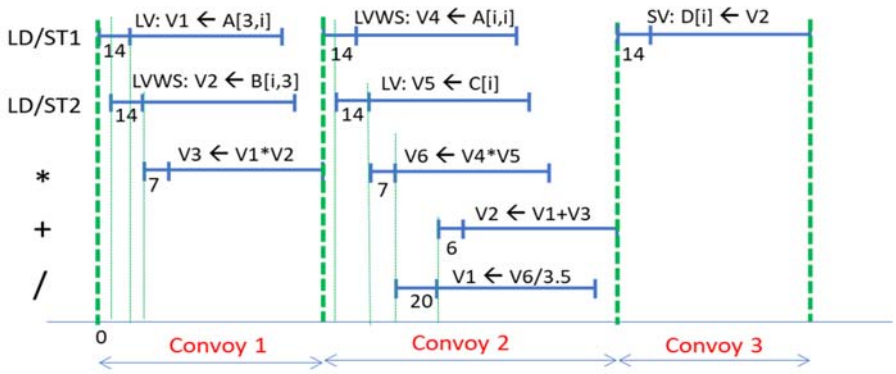
mcm/stride = 2^6x3^3 / 2^2x3^3 = 2^4= 16 > 14. Luego el acceso está libre de conflictos.

b)

Ra3: contiene la dirección inicial de la fila 3 de A. Su valor se obtiene mediante Ra+(8*3*107)
Rb3: contiene la dirección inicial de la columna 3 de B. Rb3 = Rb+(8*3)
Rs1: contiene el valor 107*8 (stride para el acceso a columna de B)
Rs2: contiene el valor 108*8 (stride para acceso a diagonal principal de A)
Fcte: contiene la constante 3.5

```
LV          V1, Ra3        ; load A[3,i]
LVWS        V2, (Rb3,Rs1)  ; load B[i,3]
MULVV       V3,V1,V2       ; A[3,i]*B[i,3]
-------------------------------------------------------------------
LVWS        V4, (Ra,Rs2)   ; load A[i,i]
LV          V5, Rc         ; load C
MULVV       V6,V4, V5      ; A[i,i]*C[i]
DIVVS       V1,V7,Fcte     ; (A[i,i]*C[i])/3.5
ADDVV       V2,V1,V3       ; Calcula valores de D[i]
-------------------------------------------------------------------
SV          Rd,V2          ; store D[i]
```

c)



T_{chime} = 3

La latencia de los pipes de LD y ST es 7 ns, lo cual son 14 ciclos (tiempo de ciclo = 0.5 ns).

T_{start} = 1+14+7+1+14+7+20+6+14 = 84 ciclos

$$T_n = \left\lceil \frac{n}{MVL} \right\rceil \times (T_{loop} + T_{start}) + n \times T_{chime} = \left\lceil \frac{107}{64} \right\rceil \times (15 + 84) + 107 \times 3 =$$

$$= 198 + 321 = 519 \text{ ciclos}$$

R = (4*107 FLOP) / 519 ciclos = 4*107 / 519 FLOP/ciclo = 0.825 FLOP/ciclo ;

(0.825 FLOP/ciclo) / (0.5×10^{-9} s/ciclo) = 1.65×10^{9} FLOP/s = **1650 MFLOPS**

4. (2 puntos) Supongamos una red MIN Butterfly con 64 entradas y 64 salidas (numeradas del 0 al 63), construida con conmutadores de grado 8. Internamente los conmutadores de grado 8 han sido implementados por medio de una red tipo *crossbar*. Al realizar la comunicación entre la entrada $(10)_{10}$ y la salida $(60)_{10}$ de la red, explica razonadamente qué comunicaciones se bloquean debido a la contención en los canales de comunicación entre los conmutadores de grado 8.

N= 64; Base k =8; $N = k^n$ ➔ Nº etapas = n = 2

Por definición:

Red Butterfly ➔ $C_0 = C_2 = I$; $C_1 = \beta_1$;

$\beta_1(a,b) = (b,a)$ siendo $a,b \in \{0..7\}$, dado que β_1 intercambia el dígito de la posición 1 con el dígito de la posición 0.

Entrada: $(10)_{10} = (12)_8$ Salida: $(60)_{10} = (74)_8$

Manejamos todos los números en octal. Como el enlace rojo grueso de la figura queda reservado para la conexión de la entrada $(12)_8$ con la salida $(74)_8$, cualquier otra conexión que utilice ese enlace no será posible. En particular, tal como muestra la figura, quedan bloqueadas las conexiones de todas las demás entradas que están conectadas al Xbar A y tienen que usar la salida $(17)_8$ para llegar al Xbar B. Es decir, quedan boqueadas las conexiones entre las entradas:

10, 11, 13, 14, 15, 16 y 17 (todas en base 8)

con las demás salidas que se alcanza desde el enlace rojo grueso a través de Xbar B:

70, 71, 72, 73, 75, 76 y 77 (todas en base 8)