Arquitectura e Ingeniería de Computadores. Examen Parcial (Problemas). 7/02/2012

Nombre----------Grupo----------

1) Sea un procesador segmentado con planificación dinámica mediante el algoritmo de Tomasulo sin especulación. El procesador tiene las siguientes características:

- Los datos que se escriben en la etapa de escritura no se pueden usar en la etapa de ejecución de una instrucción dependiente hasta el ciclo siguiente.
- Las instrucciones de load y store tienen ambas una latencia de 2 ciclos y utilizan una unidad funcional común para su ejecución.
- Existe un único bus común de datos (CDB).
- Se dispone de las siguientes unidades funcionales, estaciones de reserva, buffers de load (LB) y buffers de store (SB):

UF	Cantidad	Latencia	Segmentación
FP ADDD	1	3	Sí
FP MULD	1	6	Sí
FP DIVD	1	8	No
LOAD/STORE	1	2	Sí
INT ALU	1	1	No

Estaciones de reserva /LB /SB	Cantidad
FP ADDD	2
FP MULD	1
FP DIVD	2
LOAD	1
STORE	1

a) Para el siguiente código mostrar en qué ciclo o ciclos se llevan a cabo cada una de las tres fases del algoritmo de Tomasulo para cada instrucción, indicando también en cada caso el tipo de parada que se produce. **(1.5 ptos)**

	Instrucción	**ISSUE**	**EJECUCIÓN**	**ESCRITURA**
1	**MULD F2,F8,F6**			
2	**ADDD F8,F0,F6**			
3	**ADDD F6,F2,F4**			
4	**LD F2, 0(R3)**			
5	**DIVD F0, F4, F8**			
6	**ADDD F2,F2,F8**			
7	**MULD F6,F6,F8**			
8	**SD 0(R3),F2**			
9	**DIVD F4,F8,F2**			
10	**ADDD F8,F2,F2**			

b) Indicar el estado de las estaciones de reserva, buffers de loads, buffers de stores y banco de registros en punto flotante al final del ciclo 10. **(0.5 ptos)**

2) Supongamos un procesador con predicción dinámica de saltos que usa un predictor competitivo (Tournament Predictor) como el del Alpha 21264**.** Supongamos un programa que posee dos únicas instrucciones de salto con la siguiente estructura:

```
        Inst1
loop:   Inst2
        Inst3
        Inst4
        BEQ R1, loop
        Inst6
        Inst7
        BNE R2, loop
```

donde Insti representa una instrucción que no es de salto. Supongamos que la instrucción BEQ se ha ejecutado 1000 veces con el siguiente comportamiento: T-T-NT-NT-T-T-NT-NT-……El comportamiento de la instrucción BNE es que se toma siempre.

a) Determinar qué entradas de la Tabla de Predicción Local han sido accedidas como consecuencia de las 100 últimas ejecuciones de la instrucción BEQ y determinar cuál es el contenido de dichas entradas al final de las 1000 ejecuciones. **(1 pto)**

b) Si la instrucción BEQ se ejecuta 1000 veces más, siguiendo el mismo patrón de comportamiento, ¿cuántos fallos de predicción se producen en estas 1000 ejecuciones? **(0.5 ptos)**

3) Sea un procesador de 16 bits con una memoria direccionable en bytes que dispone de una memoria cache de 4 KB, con organización directa, líneas de 512 bytes y que realiza precarga del bloque siguiente en caso de fallo.
Suponiendo la siguiente secuencia de accesos a memoria principal: 25AFh, 3601h, 4E11h, 38AAh, 5100h, 427Ah, mostrar el contenido del directorio cache. Indicar con el símbolo "-" un fallo, con un "+" un acierto y con "*" una precarga. **(1.5 ptos)**

Solución

1)

a)

	Instrucción	ISSUE	EJECUCIÓN	ESCRITURA
1	**MULD F2,F8,F6**	1	2-7	8
2	**ADDD F8,F0,F6**	2	3-5	6
3	**ADDD F6,F2,F4**	3	9-11 LDE	12
4	**LD F2, 0(R3)**	4	5-6	7
5	**DIVD F0, F4, F8**	5	7-14 LDE	15
6	**ADDD F2,F2,F8**	7 EST	8-10	11
7	**MULD F6,F6,F8**	9 EST	13-18 LDE	19
8	**SD 0(R3),F2**	10	12-13 LDE	-
9	**DIVD F4,F8,F2**	11	15-22 EST	23
10	**ADDD F8,F2,F2**	12	13-15	16

b)

	Op	Busy	Vj	Vk	Qj	Qk
Suma1	Suma	Sí	[F2]	[F8]	0	0
Suma2	Suma	Sí	[F2]	[F4]	0	0
Mul1	Mul	Sí		[F8]	Suma2	0
Div1	División	Sí	[F4]	[F8]	0	0
Div2		No				

	Busy	Dir.	Qi
Store1	Sí	0+R3	Suma1

	Busy	Dir.
Load1	No	

	F0	F2	F4	F6	F8
UF	Div1	~~Mul1~~ ~~Load1~~ Suma1		~~Suma2~~ Mul1	~~Suma1~~

2)

a) Las posiciones modificadas en la Tabla de predicción local como consecuencia de las últimas 100 ejecuciones (de la 901 a la 1000) son las siguientes (se indexan de acuerdo a los 10 bits que marcan el comportamiento del salto en sus 10 ejecuciones inmediatamente anteriores:

Posición 204 (0011001100)
Posición 409 (0110011001)
Posición 819 (1100110011)
Posición 614 (1001100110)

El contenido de las posiciones 204 y 409 es 111, mientras que el de las posiciones 819 y 614 es 000

Cuando se accede a las posiciones 819 y 614 el salto es siempre no tomado, por lo que después de 1000 ejecuciones, e independientemente de cuál era el contenido inicial de la tabla, los bits de predicción en ambas entradas saturarán a cero (000).
Cuando se accede a las posiciones 204 y 409 el salto es siempre tomado, por lo que después de 1000 ejecuciones, e independientemente de cuál era el contenido inicial de la tabla, los bits de predicción en ambas entradas saturarán a siete (111).

b) El predictor acertará siempre en cada una de las iteraciones, por lo que el número de fallos será 0.

3)

El número de bloques que caben el cache es: (Tamaño de la cache)/(Tamaño de línea) = (2^{12} / 2^{9}) = 8, por lo que se necesitan 3 bits de la dirección para expresar el índice de cache.

Como el tamaño de línea es de 512 B, necesitamos 9 bits de la dirección como selector del byte. Los 16 – (3+9) bits restantes de la dirección (4) se corresponden con el tag. Por tanto:

Bits 0-8 → selector de byte

Bits 9-11 → índice de cache

Bits 12-15 → tag

	25AF	3601	4E11	38AA	5100	427A
0			5*	5	5+	5
1						4-
2	2-	2	2	2	2	4*
3	2*	3-	3	3	3	3
4		3*	3	3+	3	3
5						
6						
7			4-	4	4	4