Apellidos _____ Nombre _____ **Grupo:___**

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

Instrucciones.- Cada pregunta consta de cinco afirmaciones, y cada una de las afirmaciones puede ser cierta o falsa. Si considera que la afirmación es cierta marque con un aspa la casilla de la columna "**C**"; por el contario, si considera que es falsa marque con un aspa la casilla de la columna "**F**". Si considera que alguna respuesta es ambigua y, por tanto, podría considerarse cierta o falsa en función de la interpretación, ponga una llamada y explique sus argumentos al dorso de la hoja. No se permite la utilización de calculadora.

Puntuación.- Pregunta con 5 aciertos: 1 punto. 4 aciertos: 0,6 puntos. 3 aciertos: 0,2 puntos. Menos de 3 aciertos: 0 puntos.

1. Supongamos una Mc de datos directa de 4 Kbytes, con bloques de 64 bytes. En este sistema ejecutamos los programas que se describen a continuación, teniendo en cuenta que en todos los casos los elementos de los vectores son palabras de 64 bits y los arrays se almacenan en memoria por filas. Determine si los siguientes enunciados sobre son ciertos o falsos.

C F

☒ ☐ a) Al ejecutar el siguiente código:

```
double A[1024];
double B[1024];
for (i = 0; i < 1024; i = i + 1)
  C = C + (A[i] + B[i]);
```

Se producen 2048 fallos en Mc de datos.

☐ ☒ b) Al ejecutar el siguiente código:

```
struct fusion{
  double A;
  double B;
} array[1024];
for (i = 0; i < 1024; i = i + 1)
  C = C + (array[i].A + array[i].B);
```

Se producen 512 fallos en Mc de datos.

☒ ☐ c) Al ejecutar el siguiente código:

```
double A[1032];
double B[1024];
for (i=0; i < 1024; i=i+1)
  C = C + (A[i] + B[i]);
```

Se producen 256 fallos iniciales en Mc de datos.

☐ ☒ d) Al ejecutar el siguiente código:

```
double A[32][32];
for (i=0; i < 32;i=i+1)
  for (j=0; j < 32;j=j+1)
  {
   C = C * A[i][j];
   D = D + A[i][j];
  }
```

Se producen un total de 256 fallos en Mc de datos.

☒ ☐ e) Si duplicamos el tamaño de la Mc de datos, el número de fallos iniciales en la Mc de datos al ejecutar el código del apartado d) se mantiene constante.

2. Supongamos un DLX con planificación estática de instrucciones, segmentado en 7 etapas (IF1, IF2, DE, EX, ME1, ME2, WB) que implementa una política de saltos retardados. Las instrucciones de salto se resuelven en la etapa DE. La carga de trabajo está constituida por los programas A, B y C. El % de instrucciones de salto que cada programa ejecuta es A=10%, B=20%, C=5%. Se supone que las dependencias LDE no provocan paradas. La capacidad del compilador para rellenar los "delay slots" viene dada por la siguiente tabla:

Programa	% de instrucciones ejecutadas en "delay slots" que realizan trabajo útil	% de instrucciones ejecutadas en "delay slots" que son NOPs
A	50	40
B	20	10
C	60	25

Marque si las siguientes afirmaciones son ciertas o falsas.

C F

☒ ☐ a) En el programa A el 60% de las instrucciones ejecutadas en "delay slots" son instrucciones del programa que el compilador ha movido para rellenar los "delay slots".

☒ ☐ b) La penalización media por cada salto en el programa A es 1 ciclo.

☐ ☒ c) La penalización media por instrucción en el programa B es 0,16 ciclos.

☐ ☒ d) El CPI del programa C es 1,02

☒ ☐ e) El % de instrucciones de salto que ejecuta el programa C es más típico de un programa de Punto Flotante que de un programa entero.

3. Supongamos una Unidad de Ejecución que utiliza el algoritmo de Tomasulo con especulación para la planificación dinámica de instrucciones, tal como se explicó en clase. La máquina tiene 32 registros de PF, 16 Load Buffers, 8 estaciones de reserva (ER) para suma/resta, 4 ER para multiplicación/división y un buffer de reordenamiento (ROB) con 36 entradas. Marque si las siguientes afirmaciones son ciertas o falsas:

C F

☐ ☒ a) El tamaño de la ventana de instrucciones es 28.

☐ ☒ b) La anchura mínima del campo "Destino" de las ER es 5 bits.

☐ ☒ c) Cada entrada del ROB está compuesta por los campos (TAG, VALOR, TIPO_DE_INSTRUCCION, LISTO).

☒ ☐ d) En esta arquitectura no pueden aparecer dependencias EDE con memoria debido a que las instrucciones tipo STORE hacen la fase de finalización (commit) en orden.

☒ ☐ e) Siempre que una instrucción de la forma

ADDD F2, F4, F6

ejecuta la fase commit, el resultado de la suma se guarda en F2.

4. En un DLX se ejecuta un programa P que consta de 10^{10} instrucciones. El 40% de las instrucciones son de punto flotante y el resto son enteras. Las instrucciones en punto flotante tienen una penalización media por instrucción de 2,5 ciclos, mientras que las enteras tienen un CPI igual a 1. Suponiendo que la frecuencia de reloj es de 500 MHz, marque si las siguientes afirmaciones son ciertas o falsas.

C F

☐ ☒ a) Si duplicamos la frecuencia de reloj, el rendimiento en MIPS se multiplica por 1,2

☒ ☐ b) El CPI medio del programa es 2

☒ ☐ c) El tiempo de ejecución del programa es 40 segundos

☐ ☒ d) El rendimiento es 200 MFLOPS

☐ ☒ e) Si introducimos una cierta mejora que permite ejecutar las instrucciones en punto flotante el doble de rápido, entonces de alcanza un Speedup de 1,25.

5. Indique si las siguientes afirmaciones sobre procesadores con multithreading (MT) son ciertas o falsas:

C F

☒ ☐ a) Existen procesadores con MT en los que las instrucciones de cada thread se ejecutan en orden.

☐ ☒ b) Sea un procesador con MT simultáneo de 2 threads. Entonces se puede afirmar que el procesador lanza a ejecución una instrucción de cada thread en cada ciclo de reloj.

☐ ☒ c) En un procesador con MT de grano grueso, cuando se produce un cambio de thread el sistema operativo se encarga de salvar el contexto del thread suspendido.

☒ ☐ d) Los procesadores MT de grano fino cambian de thread en cada ciclo de reloj.

☐ ☒ e) En los procesadores con MT, normalmente cada thread tiene su propia memoria cache de instrucciones.

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