Arquitectura e Ingeniería de Computadores. Examen Parcial (Problemas). 12/09/2011

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 una latencia de 2 ciclos y cada una de ellas utiliza su propia unidad funcional 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 DIVD	1	7	No
LOAD	1	2	No
STORE	1	2	No
INT ALU	1	1	No

Estaciones de reserva /LB /SB	Cantidad
FP ADDD	2
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. **(2.5 ptos)**

	Instrucción	**ISSUE**	**EJECUCIÓN**	**ESCRITURA**
1	**LD F0,0(R2)**			
2	**ADDD F2,F2,F0**			
3	**DIVD F0,F4,F6**			
4	**SD 0(R3),F0**			
5	**ADDD F6,F4,F6**			
6	**DIVD F0,F2,F4**			
7	**LD F4,0(R1)**			
8	**ADDD F4,F8,F2**			
9	**ADDD F0,F6,F4**			
10	**ADDD F2,F6,F2**			
11	**SD 0(R5),F0**			

b) Entre las instrucciones 6 y 7 existe una dependencia EDL sobre F4. De acuerdo con el algoritmo de Tomasulo, ¿cómo afecta esta dependencia a los ciclos de ejecución de dichas instrucciones? ¿Por qué? **(0.5 ptos)**

c) El valor de F0 que la instrucción 11 debe escribir en memoria se corresponde con el resultado obtenido en la fase de ejecución ¿de la instrucción 6 o de la instrucción 9? ¿Por qué? **(0.5 ptos)**

d) Indicar el estado de las estaciones de reserva, buffers de loads, buffers de stores y banco de registros en punto flotante en el ciclo 7. **(1.5 ptos)**

2) Sea un sistema con las siguientes características:

Un procesador con un
- CPI ideal de 1,5
- 27 % de las instrucciones de acceso a memoria

Una Cache unificada accedida mediante direcciones virtuales con las siguientes características
- 128 KB
- Líneas de 16 bytes
- Emplazamiento directo, postescritura y asignación en escritura, sin bit sucio
- 35% de las líneas modificadas
- Tasa de fallos =0,018

Memoria principal
- Latencia de 50 ciclos
- Tasa de transferencia de bloques 8 bytes/ciclo

TLB
- Tasas de fallos 0,025
- Penalización 8 ciclos

a) Calcular el CPI real. **(3 ptos)**

b) Suponiendo que al sistema anterior se le realizan las siguientes modificaciones:
- las caches utilizan direccionamiento físico
- se utiliza bit sucio
- los valores de la tasa de fallos y penalización del TLB son los mismos

Indicar el speedup frente a la primera organización **(2 ptos)**

Solución

1)

a)

	Instrucción	ISSUE	EJECUCIÓN	ESCRITURA
1	**LD F0,0(R2)**	1	2-3	4
2	**ADDD F2,F2,F0**	2	5-7 LDE	8
3	**DIVD F0,F4,F6**	3	4-10	11
4	**SD 0(R3),F0**	4	12-13 LDE	-
5	**ADDD F6,F4,F6**	5	6-8	9
6	**DIVD F0,F2,F4**	6	11-17 EST	18
7	**LD F4,0(R1)**	7	8-9	10
8	**ADDD F4,F8,F2**	9 EST	10-12	13
9	**ADDD F0,F6,F4**	10	14-16 LDE	17
10	**ADDD F2,F6,F2**	14 EST	15-17	19 CDB
11	**SD 0(R5),F0**	15	18-19 LDE	-

b) No afecta, dado que en el ciclo 6, al hacer issue la instrucción 6, ya queda marcado quién debe proporcionar el valor de F4 que debe utilizar la instrucción 6 en su fase de ejecución, por lo que la instrucción 7 puede escribir en cualquier ciclo sin preocuparse de la instrucción anterior.

c) Lo toma de la instrucción 9. Aunque la instrucción 6 realiza la fase de escritura con posterioridad, ésta no modifica el contenido del registro dado que el tag del mismo indica que no debe escribir. Además, por seguir la secuencialidad del programa, se debe tomar este valor de la instrucción 9.

d)

	Op	Busy	Vj	Vk	Qj	Qk
Suma1	Suma	Sí	[F2]	[F0]	0	0
Suma2	Suma	Sí	[F4]	[F6]	0	0
Div1	División	Sí	[F4]	[F6]	0	0
Div2	División	Sí		[F4]	Suma1	

	Busy	Dir.	Qi
Store1	Sí	0+R3	Div1

	Busy	Dir.
Load1	Sí	0+R1

	F0	F2	F4	F6
UF	Div2	Suma1	Load1	Suma2

2)

A) $CPI_{REAL}=CPI_{IDEAL}+P_{MEMORIA}+P_{TLB}$
Donde:
$P_{MEMORIA}$ es la penalización por acceso a la MP cuando se producen fallos en la cache
P_{TLB} es la penalización que se produce cuando falla la tlb
A continuación se estudian estas dos penalizaciones

1. $P_{MEMORIA} = AMPI * T_{FALLOS} * P_{FALLO}$

Donde
AMPI= Nº de accesos medio por instrucción
T_{FALLOS} es la tasa de fallos de la cache
P_{FALLO} es la penalización que se produce cada vez que hay un fallo

1.1. AMPI
AMPI= Nº de accesos medio por instrucción
Las caches unificadas contiene tanto instrucciones como datos, luego:
$AMPI = AMPI_{INSTRUCCIONES} + AMPI_{DATOS}$
A la memoria cache se accede cada vez que se lee una instrucción por lo tanto
$AMPI_{INSTRUCCIONES} = 1$
Por otro lado el 27% de instrucciones son de acceso a memoria luego
$AMPI_{DATOS} = 0,27$

AMPI= 1+ 0,27 =1,27

1.2. P_{FALLO} = Nºaccesos por fallo *ciclos de acceso a MP
Donde nº de accesos por fallo es el número de veces que hay que acceder a la mp cuando se produce un fallo tanto de lectura como de escritura
Ciclos de acceso a mp, el número de ciclos que se tarda en un acceso a la mp

1.2.1. Nºaccesos por fallo:

La memoria no tiene bit sucio por lo tanto no se sabe si el bloque que se tiene que sobreescribir guarda la coherencia con la memoria principal.

Cuando se produce un fallo de lectura, tenemos que traer un bloque de la memoria principal, pero antes tenemos que salvar el bloque que va a ser sobreescrito en la memoria principal, por lo tanto se producen 2 accesos a memoria principal.

Cuando se produce un fallo de escritura, debido a que se utiliza la técnica de asignación en escritura, tenemos que traer el nuevo bloque de la memoria principal a la memoria cache. Igual que sucedía en el caso anterior, como no sabemos si el bloque a sustituir está modificado o no, tenemos que salvarlo en la mp principal antes de traer el nuevo bloque . luego en este caso también se producen 2 accesos a mp

Nºaccesos por fallo= 2

1.2.2. Ciclos de acceso a MP

Ciclos de acceso a MP = latencia +(tamaño línea/tasa de transferencia)=
50+ (16/8)=52

P_{FALLO} =2x52=104

$P_{MEMORIA}$ =AMPI*T_{FALLOS}*P_{FALLO} = 1,27x0,018x104= 2,3774

2. P_{TLB}

La memoria cache usa direcciones virtuales, luego sólo se accede a la tlb cuando se produce un fallo en la cache.

P_{TLB} =AMPI* T_{FALLOS_MC} *T_{FALLOS_TLB}*P_{FALLO_TLB} =1,27x0,018x0,025x8=0,004572

Luego:

CPI_{REAL}=CPI_{IDEAL}+$P_{MEMORIA}$+P_{TLB}=1,5+2,3774+0,004572=3.88197

b) $CPI_{REAL}=CPI_{IDEAL}+P_{MEMORIA}+P_{TLB}$

1. $P_{MEMORIA} =AMPI*T_{FALLOS}*P_{FALLO}$

1.1. AMPI= $ACCESOS_{INSTRSUCCIONES}$+$ACCESOS_{DATOS}$

1.1.1. $ACCESOS_{INSTRSUCCIONES}$ =1

1.1.2. $ACCESOS_{DATOS}$= 0,27

AMPI=1,27

1.2. P_{FALLO} = Nºaccesos por fallo *ciclos de acceso a MP

1.2.1. Nºaccesos por fallo:

En este caso existe bit sucio por lo tanto:

- Si el bit sucio =0 tanto en fallos de lectura como de escritura se puede traer el bloque de mp directamente puesto que el bloque almacenado en cache no se ha modificado y por lo tanto existe coherencia.
- Si bit sucio =1 tanto para lectura como para escritura se escribe y se lee un bloque en Mp.

Luego Nºaccesos por fallo= 0,35x2+0,65X1=1,4

1.2.2. Ciclos de acceso a MP = latencia +(tamaño línea/tasa de transferencia)= 50+ (16/8)=52

P_{FALLO} =1,4x52=72,8

$P_{MEMORIA} =AMPI*T_{FALLOS}*P_{FALLO}$ = 1,27x0,018x72,8= 1,6642

2. P_{TLB}

La memoria cache usa direcciones físicas, luego se accede a la tlb siempre que se accede a la memoria cache :

P_{TLB} =AMPI*T_{FALLOS_TLB}*P_{FALLO_TLB} =1,27x0,025x8=0,254

Luego:

$CPI_{REAL}=CPI_{IDEAL}+P_{MEMORIA}+P_{TLB}$=1,5+1,6652+0,254=3,4192

Speedup =**3.8819/3,4192 = 1,1353**