Comenzado el	Friday, 29 de October de 2021, 09:00
Estado	Finalizado
Finalizado en	Friday, 29 de October de 2021, 11:55
Tiempo empleado	2 horas 54 minutos
Calificación	**5,77** de 10,00 (**58**%)

Pregunta **1**

Correcta

Puntúa 0,25 sobre 0,25

Suponga que un programa posee 1250 instrucciones de acceso a memoria (loads y stores) y que 750 de esos datos de memoria solicitados se encuentran en la caché. Los otros 500 datos se entregan al procesador desde la memoria principal.

a) ¿Cuál es la tasa de fallos (miss rate) de la caché?

Tasa de fallos =

40

✔ %

b) ¿Cuál es la tasa de aciertos (hit rate) de la caché?

Tasa de aciertos =

60

✔ %

Nota: para los decimales usar punto o coma, según corresponda a su configuración de Moodle.

Pregunta **2**

Correcta

Puntúa 0,75 sobre 0,75

Suponga un sistema computacional que trabaja a 1.4GHz, con hit time = 1 ciclo de clock, CPI = 1.5 (sin stalls por fallo de caché) y que posee una organización de memoria de dos niveles de jerarquía, una caché y la memoria principal. Los tiempos de acceso y las tasas de fallo (miss rate) de ambas memorias se muestran en la siguiente tabla:

Nivel de memoria	Tiempo de acceso	Tasa de fallo
Caché	1 ciclo de clock	15%
Memoria principal	80 ciclos de clock	0%

a) ¿Cuál es el tiempo promedio de acceso a memoria, dados los tiempos de acceso y tasas de fallo (miss rate) de la tabla?

Tiempo promedio de acceso a memoria =

9,2846

✔ ns

b) Calcule el CPI promedio si el sistema de memoria no es ideal, considerando sólo los fallos de la caché de datos. Asuma la distribución de instrucciones:

R-Type	CBZ/CBNZ	B	LDUR	STUR
55%	12%	4%	18%	11%

CPIpromedio =

4,98

✔

c) ¿Qué tasa de fallo se necesitaría para reducir el tiempo promedio de acceso a memoria a 4ns?

Tasa de fallo =

5,8

✔ %

Pregunta **3**

Parcialmente correcta

Puntúa 0,67 sobre 1,00

Considere una CACHE con los siguientes parámetros:

- Criterio de correspondencia: Asociativa por conjuntos
- Asociatividad (N-vias): 2
- Tamaño de bloque: 2 words
- Tamaño de palabra (word): 32 bits
- Tamaño de la cache: 32K words
- Tamaño de dirección: 32 bits
- Cada palabra es directamente direccionable en memoria

Responder:

A- Completar cada casillero con el numero de bits de cada campo del formato de dirección de memoria principal:

Memory Address			
Tag	**Index/Set**	**Word**	**Offset ***
18	13	1	0
✔	✔	✔	✔

* Llenar con 0 en el caso que no corresponda

B- Cual es el tamaño de toda el area de Tag de la cache, expresada en bits?

151552

✖ bits.

C- Suponga que cada LINEA de la cache contiene además un bit de validación (V) y un tiempo de vida de 8 bits. Cual es el tamaño completo (expresado en bits) de un CONJUNTO de la cache, considerando datos, tags y los bits de status antes mencionados?

182

✔ bits.

Pregunta **4**

Parcialmente correcta

Puntúa 0,83 sobre 2,00

Considere que un sistema tiene una CACHE para DATOS de correspondencia ASOCIATIVA POR CONJUNTOS de 4 VÍAS de 256Kbyte y 4 palabras por línea, sobre un procesador de 32bits, CPI = 1, que resuelve todos los *data* y *control hazard* sin necesidad de stalls, tiene una memoria principal de 4G palabras de 1 byte cada una.

1. Muestre el formato de memoria principal:

Memory Address			
Tag	Index	Word	Offset (0 si no corresponde)
14 ✖	16 ✖	2 ✔	0 ✖

Considere ahora la ejecución en este sistema de los siguientes segmentos de código LEGv8 equivalentes para N > 0, donde N→ X3, start = 0x00520FC y X6 contiene la dirección base del arreglo A[] del tipo uint32_t. Considerar que el resto de los registros están inicializados en 0 y la CACHÉ vacía al inicio de la ejecución de cada segmento.

Segmento #1:

```
 start: ADD X9,X3,XZR
  loop: CBZ X9, end
        LDURH X10, [X6,#0]
        LDURH X11, [X6,#8]
        ADD X12, X10, X11
        LSR X12, X12, #1
        ADDI X6, X6, #4
        SUBI X9, X9, #1
        B loop
   end: …
```

Segmento #2:

```
 start: LDURH X10, [X6,#0]
        LDURH X11, [X6,#8]
        ADD X12, X10, X11
        LSR X12, X12, #1
        ADDI X6, X6, #4
        ADDI X9, X9, #1
        SUBS XZR, X9, X3
        B.NE start
   end: ...
```

2. Cuantos MISS de CACHE se produjeron en 4 iteraciones del lazo para el segmento #1 si la dirección base del arreglo A es 0x00051F0C?

Rta:

3

✔ MISS de caché.

3. Ahora considere que se incorpora una CACHE exclusiva para INSTRUCCIONES de correspondencia FULL ASOCIATIVA de una sola línea de 8 palabras de 32 bits c/u. Considerando que cada MISS de caché supone una penalidad de 5 ciclos de clk. Determine qué segmento de código se ejecuta en menor tiempo para N = 3. Argumente su respuesta indicando los ciclos totales en cada caso. Suponer que el pipeline ya se encuentra en régimen y que la CACHE de DATOS solo produce aciertos en los accesos, por lo que no se tiene penalidades por acceso a datos.

Rta:

32

✖ ciclos la ejecución del segmento 1.

28

✖ ciclos la ejecución del segmento 2

Pregunta **5**

Parcialmente correcta

Puntúa 0,96 sobre 2,25

Dado un procesador de arquitectura LEGv8 2-issue, que predice los saltos perfectamente, de modo que los hazard de control son manejados por hardware. Para el siguiente fragmento de código LEGv8:

```
        1> ADD X10, XZR, XZR
        2> ADDI X11, XZR, #64
L1:     3> SUBIS X2, X10, #64
        4> B.EQ end
        5> LDUR X0, [X10, #0]
        6> ADDI X11, X11, #8
        7> LDUR X1, [X11, #0]
        8> SUBS X3, X0, X1
        9> STUR X0, [X11, #0]
        10> B.LT L2
        11> SUB X1, XZR, X1
L2:     12> STUR X1, [X10, #0]
        13> ADDI X10, X10, #8
        14> B L1
end:    ...
```

a) Sin alterar el orden de las instrucciones, mostrar en la siguiente tabla cómo organizaría los issue packets para ejecutar el programa en la menor cantidad posible de ciclos de clock (cada instrucción sólo puede agruparse con la inmediata anterior, la inmediata posterior o una nop). El compilador asume toda la responsabilidad insertar instrucciones nop para que el código se ejecute sin necesidad de generación de stalls. Mostrar sólo una iteración del loop L1.

Formato:
- Escribir en los cuadros el número de instrucción o la letra “N” en caso de insertar un nop.
- Puede haber filas de más en la tabla. Completar los espacios no utilizados con guion medio o dejar los espacios vacíos. En el segundo caso, puede aparecer un aviso de Moodle diciendo que faltan completar espacios al terminar.

ALU or branch instruction	Data transfer instruction
1 ✔	N ✔
2 ✔	N ✔
3 ✔	N ✔
4 ✔	5 ✖
6 ✔	N ✖
7 ✖	N ✖
8 ✖	N ✔

10 ✖	9 ✔
11 ✖	N ✔
12 ✖	N ✔
13 ✔	N ✖
14 ✔	N ✔
- ✔	- ✔
- ✔	- ✔
- ✔	- ✔
- ✔	- ✔

b) Calcular el tiempo de ejecución del código dado en un procesador LEGv8 de 1-issue con forwarding-stall y en el procesador LEGv8 2-issue, si ambos trabajan a una frecuencia de 1.5GHz. Suponer que en ambos procesadores los saltos son perfectamente predichos y los hazard de control son manejados por hardware (no hay flush de instrucciones). Suponer que siempre X0>X1. Considerar la totalidad de las iteraciones realizadas por el código.

Tiempo de ejecución en 1-issue =

71,28

✖ ns

Tiempo de ejecución en 2-issue =

61,33

✖ ns

c) Aplicar la técnica de loop unrolling para que cada iteración del nuevo bucle se corresponda con dos iteraciones del bucle “L1” original. Luego responder si las siguientes afirmaciones son verdaderas (V) o falsas (F):

- El nuevo loop posee 4 instrucciones que acceden a memoria. F ✔
- Para aplicar la técnica es necesario cambiar de lugar las instrucciones 3 y 4 (del código original). F ✔
- Ya sea que aplique o no la técnica de register renaming, el nuevo loop posee 4 instrucciones de salto. F ✖
- Para direccionar correctamente los accesos a memoria, la instrucción 6 (del código original) debe necesariamente ejecutarse dos veces en el nuevo loop. F ✔

Pregunta **6**

Parcialmente correcta

Puntúa 0,45 sobre 1,25

Considerando que el procesador que ejecuta el siguiente código, cuenta con un predictor de saltos global de dos niveles, siendo m y n igual a 8.

```
0x00: for (i = 0; i <= 100; i++) {

0x14:  for (j = 0; j <= 2; j++) {

      ...

0x24: if (j < 1)

 ...

       ...
   }

}
```

Mostrar como queda la tabla de historial de patrones (PHT), considerar que ya se ejecutaron varias iteraciones del ciclo externo y el predictor ya paso la fase de warmup:

PC	GR	Predicción
0x	0x	0b
00	E5	11
✔	✖	✔
0x	0x	0b
14	DB	11
✔	✖	✖
0x	0x	0b
24	E7	11
✖	✖	✔
0x	0x	0b
14	AD	11
✔	✖	✔
0x	0x	0b
24	5C	00
✖	✖	✖
0x	0x	0b
14	BA	11
✖	✖	✖
0x	0x	0b
24	75	00
✔	✖	✔
0x	0x	0b
14	DA	00
✖	✖	✖

0x	0x	0b
✖	✖	✖

Importante:

- Se considera taken cuando el código dentro del loop o if, se ejecuta.
- En el código, el valor en hexadecimal al costado de cada loop o if indica la posición en memoria donde esta el salto
- Los valores PC y GR debe completarse en hexadecimal y la predicción en binario.
- La tabla puede contener mas filas de las necesarias.
- Ordenar las direcciones de la siguiente forma: el tag (PC,GR) mas bajo en la parte superior de la tabla y luego en forma creciente.

Pregunta **7**

Parcialmente correcta

Puntúa 1,85 sobre 2,50

Considerando un microprocesador out-of-order implementado mediante el algoritmo de Tomasulo (**sin** especulación), muestre el contenido de las tablas de *Status register*, *Reservation stations* y *stage* luego de ejecución de la siguiente secuencia de código para el **4to ciclo de clk** (incluido):

```
   1>> addi x3,xzr,#80
L: 2>> subi x1, x1, #8
   3>> ldurd D2, [x1]
   4>> ldurd D6, [x1, #100]
   5>> fmuld D4, D2, D0
   6>> faddd D6,D4,D6
   7>> sturd D6, [x1, #100]
   8>> cbnz x1, L
   9>> add x1, xzr, #160
```

Asumir que el *salto fue predicho* ***correctamente*** *como taken.* Considerar el siguiente Hardware:

Hardware
Los saltos son predichos como *taken*
Issue = 4 instrucciones
Load = 4 RS / 2 clk
Store = 4 RS / 2 clk
Suma entero = 4 RS, 2 FU / 1 clk
Suma punto flotante = 4 RS, 2 FU / 2 clk
Multiplicación punto flotante = 4 RS, 2 FU / 3 clk
Branch = 1 RS, 1 FU / 1clk

Reservation station

Nombre	Busy	Op	Vj	Vk	Qj	Qk	A
Load 1	1 ✗	load ✓	[x1] ✓	- ✓	0 ✓	0 ✓	[x1 + #0] ✗
Load 2	1 ✗	load ✓	[x1] ✓	- ✓	0 ✓	0 ✓	✗
Load 3	0 ✓	✗	✗	✗	✗	✗	✗
Load 4	0 ✓	✗	✗	✗	✗	✗	✗
Store 1	1 ✗	store ✓	[x1] ✓	- ✓	0 ✓	load 2 ✗	#100 ✓
Store 2	0 ✓	✗	✗	✗	✗	✗	✗
Int alu 1	0 ✓	✗	✗	✗	✗	✗	✗

Int alu 2	0							
Int alu 3	0							
Int alu 4	0							
FP alu 1	0	add	-	-	FP mult 1	load2	-	
FP alu 2	0							
FP alu 3	0							
FP alu 4	0							
FP mult 1	0	mult	--	[D0]	load 1	0	-	
FP mult 2	0							
FP mult 3	0							
FP mult 4	0							
Branch	1	cbnz	[x1]	-	0	0	-	

Nota 1: En la columna Busy, se debe indicar con uno si la FU esta ocupada, o en caso contrario con cero

Nota 2: Al indicarse una reservation station, utilizar el nombre dado (en la primer columna) sin modificarlo ni agregar espacios extras.

Register Status

	D0	D1	D2	D3	D4	D5	D6
Qi			load 1		FP mult 1		load 2

	X0	X1	X2	X3	X4	X5	X6
Qi							

Stage

clk	IF	ID/IS	Ex	WB

0	1,2,3,4 ✔	✖	✖	✖
1	5,6,7,8 ✔	1,2,3,4 ✔	✖	✖
2	2,3,4,5 ✔	3,4,5,6,7,8 ✔	1,2 ✔	✖
3	2,3,4,5,6,7,8 ✖	3,4,5,6,7,8 ✔	✖	1,2 ✔
4	2,3,4,5,6,7,2,3,4,5 ✖	5,6 ✖	3,4,7,8 ✖	✖

Nota: Separar los números de las instrucciones con comas pero sin espacios.

◄ Parcial 1

Ir a...

ej 6 - parcial 2 - resuelto ►