Pregunta 1

Sin responder aún

Puntúa como 1,00

Marcar pregunta

Para un procesador de 16 bits que es capaz de direccionar 4G byte de memoria principal (32 bits de dirección), se requiere implementar una memoria caché con capacidad total de 8Mbits para datos. La memoria caché utiliza correspondencia (mapeo) directo, con un tamaño de bloque de 16 palabras.

A) ¿Cuántos **bits** hay en los diferentes campos del formato de dirección de memoria principal?

Memory Address			
Tag	Index	Word	Offset (si aplica)

B) ¿Cuántas **líneas** contiene la memoria caché?

C) ¿Cuál es la **capacidad total** de memoria (**expresada en bits**) necesaria para implementar dicha caché? Considerar que cada línea está compuesta, además, del campo Tag y 1 bit extra de validación.

NOTA: No utilizar el campo de respuesta mostrado a continuación. Las respuestas de los ejercicios deben ser enviadas todas juntas en un ÚNICO archivo comprimido (tar, zip, rar, etc) con un archivo pdf por ejercicio, el cual debe contener los desarrollos y cálculos realizados.

Respuesta:

aún

Puntúa como 1,00

Marcar pregunta

```
1>              ADDI X9, XZR, #2
2>              MOVZ X0, 0xFFFF, LSL 16
3>              SUBI X10, X0, #8
4>              EOR X11, X11, X11
5>      loop:   SUBI X9, X9, #1
6>              CBZ X9, end
7>              ADDI X12, XZR, #8
8>              LDUR X3, [X0, #0]
9>              ADDI X0, X0, #16
10>             SUB X3, XZR, X3
11>             STUR X3, [X10, #0]
12>             SUB X10, X10, X12
13>             LDUR X4, [X0, #-8]
14>             SUB X5, X3, X4
15>             STUR X5, [X11, #0]
16>             ADD X11, X11, X12
17>             B loop
18>     end:    ...
```

1. Analice en el código las dependencias de datos y determine cuales generan *data hazards*. En cada caso indique: el tipo de *hazard*, el operando en conflicto y los números de las instrucciones involucradas.
2. Dibuje un diagrama de pipeline que muestre cómo se ejecuta el código LEGv8 dado anteriormente en un procesador de 2-issue. Sin modificar el orden de ejecución, organice el código para evitar la mayor cantidad posible de stalls. Deje indicados los caminos de forwarding utilizados. *Considere el procesador LEGv8 two-issue descripto en el libro de Patterson y Hennessy, con forwarding-stall, donde en cada "issue packet" una de las instrucciones puede ser una operación de la ALU o un salto y la otra puede ser un load o store. Suponga que los saltos están perfectamente predichos, de modo que los control hazard son manejados por hardware.*
3. Indique la cantidad de ciclos de clock que toma la ejecución del código dado en el procesador de 1-issue descripto en el libro de Patterson y Hennessy (con forwarding-stall y los saltos perfectamente predichos) y en el procesador de 2-issue. Luego calcule el aumento de velocidad en la ejecución del código.
4. Aplique la técnica de *loop unrolling* al código LEGv8 para que cada iteración del nuevo bucle se corresponda con dos iteraciones del bucle original. Luego, reorganice/reescriba su nuevo código para lograr un mejor rendimiento en el procesador de 2-issue. Está permitido utilizar otros registros si se considera necesario. (Suponga que el registro X9 se inicializa con un número impar mayor que 1.)

NOTA: No utilizar el campo de respuesta mostrado a continuación. La respuesta a esta pregunta debe enviarse en un único archivo, en formato pdf, con el nombre ej3.pdf. Pueden ser fotos del ejercicio resuelto en papel (fotos de buena calidad y resuelto en forma prolija) o puede ser resuelto en computadora.

Pregunta **4**

Sin responder aún

Puntúa como 1,00

Marcar pregunta

Considerando un microprocesador out-of-order implementado mediante el algoritmo de Tomasulo (**sin** especulación), muestre el contenido de las tablas de *Status*, las *Reservation stations* y el flujo de ejecución para la siguiente secuencia de código para el 7to ciclo de clk (incluido):

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

Asumir que X1 inicialmente contiene el valor 80. Considerar el siguiente Hardware:

Hardware
Issue = 4 instrucciones
Load = 4 RS / 1 clk
Store = 4 RS / 1 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

Se recomienda utilizar el template subido a moodle para la resolución del ejercicio y la entrega del mismo

No utilizar el campo de respuesta mostrado a continuación. Las respuestas de los ejercicios deben ser enviadas todas juntas en un ÚNICO archivo comprimido (tar, zip, rar, etc) con un archivo pdf por ejercicio, el cual debe contener los desarrollos y cálculos realizados.