Universidad Carlos III de Madrid

Departamento de Informática
Grado en Ingeniería Informática
Arquitectura de Computadores

ARCOS

Examen Extraordinario
23 de junio de 2015

ATENCIÓN:

- Lea atentamente todo el enunciado antes de comenzar a contestar.
- No se podrán utilizar libros ni apuntes, ni calculadoras de ningún tipo.
- Los teléfonos móviles deberán permanecer desconectados durante la prueba (apagados, no silenciados).
- Solamente se corregirán los ejercicios contestados con bolígrafo. Por favor no utilice lápiz.
- **La duración del examen es de 150 minutos.**
- **Presente cada ejercicio en una hoja distinta. Si no hace un ejercicio entregue una hoja en blanco con el número del ejercicio.**

NOMBRE: Lucas
APELLIDOS: Menéndez Sánchez
NIA: 100316670

Ejercicio 1 [3.5 puntos]:

Sea el siguiente fragmento de código:

```
bucle:
        lw $t4, 0($t0)
        lw $t5, 0($t1)
        sub $t6, $t5, $t4
        mul $t6, $t6, $t6
        sw $t6, 0($t2)
        addi $t0, $t0, 4
        addi $t1, $t1, 4
        addi $t2, $t2, 4
        beq $t0, $t3, bucle
```

```
lw  $t7  0x4($t0)
lw  $t8  4($t1)
sub $t9  $t7  $t8
sw  $t6  4($t2)
```

Se pide:

a) Elabore una lista con todas las dependencias RAW de datos que hay en el código.

b) Elabore un diagrama de tiempos para una arquitectura MIPS de 5 etapas con las siguientes consideraciones:

- Si hay hardware de forwarding.
- La arquitectura permite que una instrucción escriba en un registro y otra instrucción lea de un registro en el mismo ciclo de reloj sin problemas.
- Las bifurcaciones se tratan asumiendo el salto como tomado.
- Las referencias a memoria requieren un ciclo de reloj.
- La dirección efectiva de salto se calcula en la etapa de ejecución.

c) Determine cuantos ciclos de reloj se requiere para ejecutar todas las iteraciones del bucle si los valores iniciales de $t0, $t1, $t2 y $t3 son inicialmente 0x00A00000, 0x00B00000, 0x00C00000, y 0x00A0100.

0x00A00100

d) Escriba el bucle más desenrollado posible asumiendo que puede usar solamente los registros que van desde $t0 hasta $t9.

e) Determine cuantos ciclos de reloj se requiere ahora para ejecutar todas las iteraciones del bucle.

Universidad Carlos III de Madrid

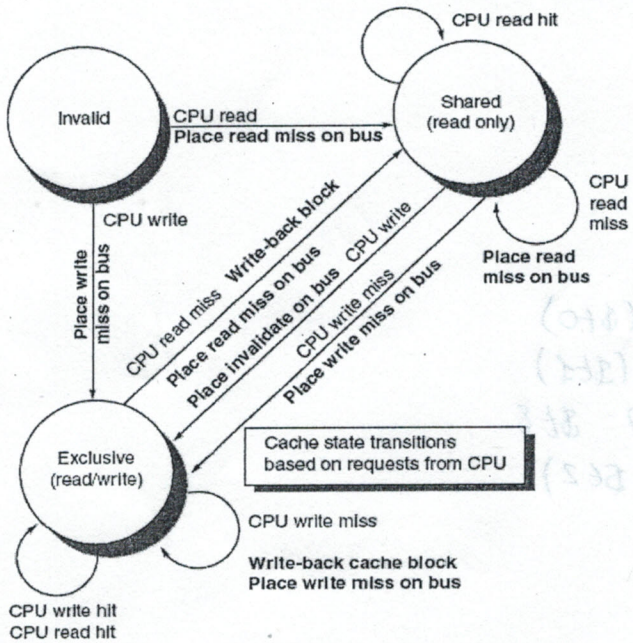
Departamento de Informática
Grado en Ingeniería Informática
Arquitectura de Computadores

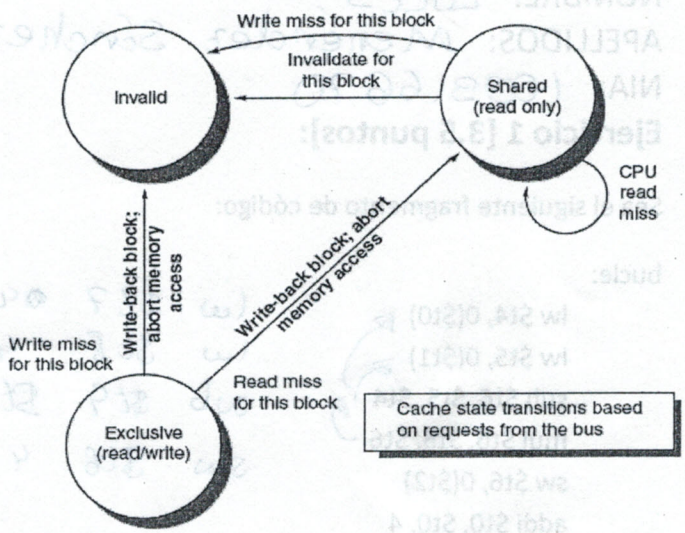
ARCOS

Examen Extraordinario
23 de junio de 2015

Ejercicio 2 [2.5 puntos]:

Sea un procesador con dos núcleos con arquitectura de memoria compartida simétrica basado en bus con protocolo de espionaje o *snooping*. Cada núcleo tiene un único nivel de memoria caché que es privada para el núcleo y cuya coherencia se mantiene usando el protocolo MSI. Las memorias cachés utilizan correspondencia directa y cada línea caché puede almacenar bloques de ocho palabras.





Se tiene en memoria principal dos variables X e Y, cuyos valores iniciales son para ambas 0. La variable X se almacena en la dirección de memoria 0x0000A000 y la variable Y en la dirección de memoria 0x0000A004.

En dicho procesador se ejecuta código de dos hilos (T1 y T2) según la siguiente tabla:

Orden	Hilo	Instrucción
1	T1	lw $t1, X
2	T1	li $t1, 99
3	T1	sw $t1, X
4	T2	lw $t5,Y
5	T2	li $t6, 88
6	T2	sw $t6, Y
7	T1	lw $t2, X
8	T2	sw $t5, Y

Si todas las cachés están inicialmente vacías, se pide indicar tras la ejecución de cada instrucción:

- El valor de cada registro que se haya modificado.
- El valor de las posiciones de memoria X, e Y, tanto en memoria principal como en las cachés locales.
- El estado del protocolo de coherencia de caché MSI para cada línea de caché afectada.

Ejercicio 3 [2 puntos]: Dado el siguiente código.

```
#ifndef SEQ_BUFFER_H
#define SEQ_BUFFER_H

#include <memory>

struct full_buffer {};
struct empty_buffer {};

template <typename T>
class seq_buffer {
public:
  seq_buffer(int n) : size_{n}, buf_{new item[size_]} {}
  ~seq_buffer() = default;

  int size() const noexcept { return size_; }
  bool empty() const noexcept { return next_read_ == next_write_; }
  bool full() const noexcept {
    const int next = next_position(next_write_);
    return next == next_read_;
  }

  void put(const T & x, bool last);
  std::pair<bool,T> get();

private:
  int next_position(int p) const noexcept { return p + ((p+1>=size_)?(1-size_):1); }

private:
  const int size_;

  struct item {
    bool last;
    T value;
  };

  std::unique_ptr<item[]> buf_;
  int next_read_ = 0;
  int next_write_ = 0;
};

template <typename T>
void seq_buffer<T>::put(const T & x, bool last)
{
  const int next = next_position(next_write_);
  if (next == next_read_) throw full_buffer{};
  buf_[next_write_] = item{last, x};
  next_write_ = next;
}
```

```
template <typename T>
std::pair<bool,T> seq_buffer<T>::get()
{
  if (empty()) throw empty_buffer{};
  auto res = buf_[next_read_];
  next_read_ = next_position(next_read_);
  return std::make_pair(res.last, res.value);
}

#endif
```

Modifique el código fuente de esta clase para que soporte su uso en aplicaciones multi-hilo bajo las siguientes restricciones:

- La solución debe ser libre de cerrojos (no podrá hacer uso de mutex ni variables condición).
- Debe soportar un productor y un consumidor.
- No es necesario dar soporte a múltiples productores ni múltiples consumidores.