Universidad Carlos III de Madrid

Computer Architecture

Group 89

Final exam 2011-2012

Name:_____

NIA: _____

I. (10 points) Answer the following questions. ***A correct answer scores 1 point, while an incorrect answer subtracts 0.25 points.***

Question	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Answer	**A**	**A**	**C**	**C**	**A**	**B**	**E**	**B**	**B**	**A**

1. Write propagation means a) writes by a CPU become visible to other CPUs b) writes by a CPU are updated in the local cache c) Writes to a cache are updated in the main memory

2. Intuitively if a memory location is more intensively read than written by several CPUs the most adequate cache coherence protocol is a) update b) invalidate

3. Which orders are relaxed in the processor consistency model? a) read after read b) write after read c) read after write d) write after write e) true dependencies f) none of the above

4. The orders relaxed based on a consistency models can be used for reordering by a) Only static methods (compilers) b) Only dynamic methods (hardware) c) Both static and dynamic methods d) Neither static or dynamic methods

5. Test-and-test-and-set compares twice the same value of a register with the same value of a memory location. a) True b) False

6. When store conditional fails generates bus traffic. a) True b) False

7. Which of the following lock implementation requires most storage? a) test-and-set b) test-and-test-and-set, c) test-andset with backoff d) load-locked/store conditional e) array-based lock

8. The bisection width of a hypercube of $n=2^d$ nodes is a) d b) n/2 c) d/2

9. The routing strategy with the best performance in absense of contention is a) Store-and-forward b) Cut-though

10. RAID6 allows recovery from more than 1 error a) True b) False

II. (30 points) Shortly answer the following questions:

a) Flynn classification

b) What coherence protocols scale better: snoopy protocols or directory protocols? Justify your answer.

c) What is the difference between coherence and consistency

d) Explain RAID5 based on a drawing for a RAID5 with 4 disks and a file with 6 blocks.

e) Explain shortly the cache optimizations critical word first and early restart

f) 90% of a sequential application can be perfectly parallelized. What is the maximum speedup that can be obtained assuming that you have an infinite number of processors available?

Solutions:

a) slides

b) Directory scale better. Snoopy require either a bus or a broadcasting mechanisms, while the directory operations do not involve all caches at the same time.

c) Slides

d) Slides

e) Slides

f) Speedup = 1 / non-parallizable percentage = 1/.1 = 10

III. (20 points) Given the following code:

```
1:    DADDUI R3, R1, #40
2: loop:
3:    L.D F0, 0(R1)
4:    L.D F2, 0(R2)
5:    ADD.D F4, F0, F2
6:    S.D F4, 0(R1)
7:    DADDUI R1, R1, #8
8:    DADDUI R2, R2, #8
9:    BLE R1, R3, loop
```

and the following latencies between instuctions:

Instruction producing result (previous)	Instruction using result (next)	Latency (clock cycles)
FP ALU op	FP ALU op	5
FP ALU op	Store/load double	4
Load double	FP ALU op	2
Load double	Load double	1
Store double	FP ALU op	2
FP ALU op	Jump	4

and assuming that a jump has a latency of 1 cycle and does not have any delay slot, answer the following questions:

a) Identify data dependencies.
b) Assume that: you have one execution pipeline, each executing one instruction per cycle, you have enough fetch/decode bandwidth and no instruction reordering is performed. How many cycles does the loop require?

c) Reorder the instructions to improve performance. How many cycles does the loop require? What is the speedup over the code without reordering?

d) Modify the code by unrolling two loop iterations. What is the speedup over the code without reordering?

Solution:

a) RAW: 3-5, 4-5, 5-6, 7-9, 1-9

WAR: 6-7,4-8

b)

```
        DADDUI R3, R1, #40    : Only first time
loop:
        L.D F0, 0(R1)
        L.D F2, 0(R2)
        Stall
        Stall
        ADD.D F4, F0, F2
        Stall
        Stall
        Stall
        Stall
        S.D F4, 0(R1)
        DADDUI R1, R1, #8
        DADDUI R2, R2, #8
        Stall
        Stall
        Stall
        BLE R1, R3, loop

        1+16*5 = 81 cycles
```

c)

```
        DADDUI R3, R1, #40    : Only first time
loop:
        L.D F0, 0(R1)
        L.D F2, 0(R2)
        Stall
        DADDUI R1, R1, #8
        ADD.D F4, F0, F2
        DADDUI R2, R2, #8
        Stall
        Stall
        Stall
        S.D F4, -8(R1)
        BLE R1, R3, loop

        1+11*5 = 56 cycles

        Speedup = 81/56
```

d)

Unrolling:

```
        DADDUI R3, R1, #40    : Only first time
loop:
        // First iteration
        L.D F0, 0(R1)
        L.D F2, 0(R2)
        Stall
        DADDUI R1, R1, #8
        ADD.D F4, F0, F2
        DADDUI R2, R2, #8
```

Stall
Stall
Stall
S.D F4, **-8**(R1)

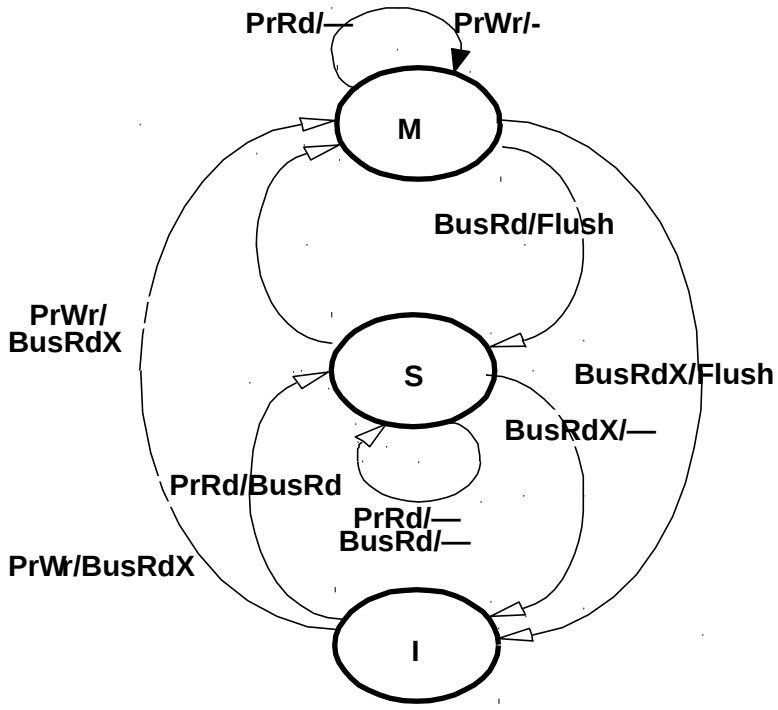
//Second iteration
L.D F6, 0(R1)
L.D F8, 0(R2)
Stall
DADDUI R1, R1, #8
ADD.D F10, F6, F8
DADDUI R2, R2, #8
Stall
Stall
Stall
S.D F10, **-8**(R1)
BLE R1, R3, loop

// 5th iteration outside loop
L.D F0, 0(R1)
L.D F2, 0(R2)
Stall
DADDUI R1, R1, #8
ADD.D F4, F0, F2
DADDUI R2, R2, #8
Stall
Stall
Stall
S.D F4, **-8**(R1)

1+2*21+10 = 53 cycles
Speedup = 66/53

IV. (25 points) You are given the parallel program from the following table which runs on a *cache-coherent bus-connected shared memory* machine. You may assume that each instruction of the parallel program executes atomically. The cache coherence protocol is MSI, whose state transition diagram is depicted lower. Assume that each invalidation generates a bus traffic of 2 bytes and each cache miss requires a traffic of 16 bytes for requesting the cache line and 64 bytes for transferring the cache line.

Processor: instruction	P1 transition	P2 transition	P3 transition	Bus action	Bus traffic
P1: read x	S -> S	S	S	-	-
P2: write x	S -> I	S -> M	S -> I	BusRdX	2
P2: write x	I	M	I	-	-
P3: write x	I	M -> I	I -> M	BusRdX Flush	16+64+2
P1: write x	I -> M	I	M -> I	BusRdX Flush	16 + 64+2
P2: write x	M-> I	I -> M	I	BusRdX Flush	16 + 64+2



Answer the following questions:

a) What type of write policy does MSI employ: write-through or write-back? Justify your answer.

Solution: write back, the cache is flushed for the transitions: M-S and M-I

b) **Assume that initially all the processors have the cache line containing x in their cache.** For the program given in the first column of the table, fill the other table columns with the following information:

- MSI transitions
- MSI bus actions
- The traffic in bytes

V. (15 points) Given the following class which implements a spin-lock:

```
class spinlock_mutex {

            private:
              std::atomic_flag f;

            public:

              spinlock_mutex() : f(ATOMIC_FLAG_INIT) {}


              void lock() {
                while (f.test_and_set(std::memory_order_acquire)) {}
              }

              void unlock() {
                flag.clear(std::memory_order_release);
              };
}
```

Assume that you want to employ this class in two versions of a parallel program for synchronizing the access to shared queue:
a) A version launching 8 threads on a 8-core processor
b) A version launching 256 threads on a 8-core processor

Based on your experience in the lab assignment with atomic operations, explain what is the effect of using this class for synchronization in each of these cases.

Solution:

The test and set implementation produces busy waiting, i.e. actively checking if the flag changes and generating bus traffic. N a) case each thread runs on its own core, but in b) case there are 32 threads on each core doing active polling. Therefore, in b case there is both CPU and bus contention and expectedly it will perform worse.

A blocking implementation or a t&s with back-off may reduce both the bus traffic and the CPU contention.