CSE 661

Parallel and Vector Architectures

Midterm Exam – Fall 2007

Tuesday, April 4, 2007

7:00 – 9:30 pm

Computer Engineering Department

College of Computer Sciences & Engineering

King Fahd University of Petroleum & Minerals

Student Name: _____

Student ID: _____

Q1	/ 20	Q2	/ 35
Q3	/ 25	Q4	/ 25
Total	/ 105		

Q1. (20 pts) In the Illinois MESI cache coherence protocol, when a block is modified in one cache and there is a Bus Read request from a second cache on a read miss, the cache block is flushed on the bus by the first cache. The data is picked up by the requesting cache and is written as well to memory. The cache block is now shared in both caches and memory is updated.

The problem here is that main memory is much slower than the requesting cache. Cache-to-cache transfer is very desirable, but we should avoid updating memory. Therefore, we need to modify the MESI protocol to become MOESI, where a fifth state (O = Owned) is introduced. The Owned state is a shared modified state indicating that memory is NOT modified, but will be updated on a writeback transaction when the block in the Owned state is replaced inside the cache.

Draw the MOESI state diagram, showing all transitions on processor read (PrRd), processor write (PrWr), on a block replacement (Replace), as well as observing a bus read (BusRd), bus read exclusive (BusRdX), and bus upgrade (BusUpgr). Cache-to-Cache transfer (C2C) should be supported only, but Not cache flushing. Bus writeback (BusWB) should happen only when a modified or owned block is replaced.

Q2. (35 pts) Consider the execution of a parallel loop on a bus-based multiprocessor with 2 processors. Processor P0 executes the even iterations and Processor P1 executes the odd one. For each iteration, each processor reads elements **A[i]** and **B[i]**, does the addition, and then writes element **A[i]**, generating two memory reads and one memory write.

```
Original Loop:
for (i=0; i<N; i++) A[i] = A[i] + B[i];


Processor P0 executes even iterations:
for (i=0; i<N; i=i+2) A[i] = A[i] + B[i];


Processor P1 executes odd iterations:
for (i=1; i<N; i=i+2) A[i] = A[i] + B[i];
```

Each processor has a private data cache with **32-byte blocks**. Each cache block can fit **8 array elements**, where each element is 4 bytes. Assume a cold start. The caches blocks are initially invalid forcing initial cache misses to read the blocks from memory.

a) (10 pts) Assuming that the caches implement the **Illinois MESI coherence protocol**, consider the **worst-case** scenario that results in the **worst number of bus transactions**. Complete the table (next page) showing the execution of 5 iterations in P0 and in P1. Compute the bus transactions issued by P0 and P1 for all **N** iterations, where **N** is even.

b) (10 pts) Repeat part **(a)**, assuming that the caches now implement the **Dragon update coherence protocol**. Complete the table showing the execution of 5 iterations in P0 and in P1. Compute the number of bus transactions issued by P0 and P1 for all **N** iterations.

c) (15 pts) Modify the code executed by P0 and P1 to minimize the cache misses and bus transactions. Complete the table showing the execution of 2 iterations only in P0 and in P1 for the MESI and Dragon coherence protocols. Re-compute the total bus transactions issued by P0 and P1 for all **N** iterations for the MESI and Dragon protocols.

Answer the three parts of this question on the next three pages.

Part (a) Solution: MESI Cache Coherence Protocol

Read/Write Operation	Bus Transaction	P0 Cache State	P1 Cache State
Initial Cold Start		I	I
P0 reads A[0]	Bus Read	E	I
P1 reads A[1]			
P0 reads B[0]			
P1 reads B[1]			
P0 writes A[0]			
P1 writes A[1]			
P0 reads A[2]			
P1 reads A[3]			
P0 reads B[2]			
P1 reads B[3]			
P0 writes A[2]			
P1 writes A[3]			
P0 reads A[4]			
P1 reads A[5]			
P0 reads B[4]			
P1 reads B[5]			
P0 writes A[4]			
P1 writes A[5]			
P0 reads A[6]			
P1 reads A[7]			
P0 reads B[6]			
P1 reads B[7]			
P0 writes A[6]			
P1 writes A[7]			
P0 reads A[8]			
P1 reads A[9]			
P0 reads B[8]			
P1 reads B[9]			
P0 writes A[8]			
P1 writes A[9]			

Total number of bus transactions (all iterations) =

Part (b) Solution: Dragon Cache Coherence Protocol

Read/Write Operation	Bus Transaction	P0 Cache State	P1 Cache State
Initial Cold Start		X	X
P0 reads A[0]	Bus Read	E	X
P1 reads A[1]			
P0 reads B[0]			
P1 reads B[1]			
P0 writes A[0]			
P1 writes A[1]			
P0 reads A[2]			
P1 reads A[3]			
P0 reads B[2]			
P1 reads B[3]			
P0 writes A[2]			
P1 writes A[3]			
P0 reads A[4]			
P1 reads A[5]			
P0 reads B[4]			
P1 reads B[5]			
P0 writes A[4]			
P1 writes A[5]			
P0 reads A[6]			
P1 reads A[7]			
P0 reads B[6]			
P1 reads B[7]			
P0 writes A[6]			
P1 writes A[7]			
P0 reads A[8]			
P1 reads A[9]			
P0 reads B[8]			
P1 reads B[9]			
P0 writes A[8]			
P1 writes A[9]			

Total number of bus transactions (all iterations) =

Part (c) Solution:

Modified Code for P0 and P1

Executing 2 iterations in P0 and in P1 using the MESI and Dragon Coherence Protocols

	MESI Coherence Protocol			Dragon Coherence Protocol		
Read/Write Operation	**Bus Transaction**	**P0 Cache State**	**P1 Cache State**	**Bus Transaction**	**P0 Cache State**	**P1 Cache State**
Cold Start		I	I		X	X

Number of bus transactions for MESI (all iterations) =

Number of bus transactions for Dragon (all iterations) =

Q3. (25 pts) Consider the following nested loops, where **A** and **B** are 2-dimensional arrays with 64×64 elements.

```
for (i=1; i<64; i++)
  for (j=0; j<64; j++)
    A[i][j] = B[i][j] + A[i-1][j] * B[i-1][j];
```

(a) (5 pts) Draw matrix A showing the data dependences among its elements and indicate whether the outer or inner loop can be vectorized.

(b) (10 pts) Translate the above nested loops to VMIPS assembly code (see Figure G.3 for the VMIPS vector instructions). The VMIPS vector registers have 64 elements, which match the number of elements along each dimension of matrices **A** and **B**. Here is a sample of VMIPS instructions that you might find useful:

ADDV.D	**V3, V1, V2**	**# V3 = V1+V2 (64-element vector registers)**
MULV.D	**V3, V1, V2**	**# V3 = V1*V2 (64-element vector registers)**
LV	**V1, R1**	**# Load V1 from memory starting at address R1**
SV	**R1, V1**	**# Store V1 into memory starting at address R1**

(c) (10 pts) Estimate the execution time of the above nested loops on a VMIPS vector processor. The VMIPS is a single-issue processor that can issue one instruction per clock cycle and has only one lane. It has one FP add/subtract unit, one FP multiply unit, and one vector load/store unit. The functional units are deeply pipelined and have a startup latency overhead equal to 6 cycles for the FP add/subtract unit, 5 cycles for the FP multiply unit, and 20 cycles for the vector load/store unit. There is also an integer ALU for executing scalar instructions at the rate of 1 cycle per scalar instruction.

If **no chaining** is provided, how many cycles does it take to compute all elements of matrix **A**?

If **chaining** is now supported, how many cycles does it take to compute all elements of matrix **A**?

Q4. (25 pts) Consider the following pseudocode describing sequential Gaussian elimination:

```
procedure Eliminate(A)
begin
  for k = 0 to n – 1 do
  begin
     for j = k+1 to n – 1 do
        A[k][j] = A[k][j] / A[k][k];
     end for
     A[k][k] = 1;
     for i = k+1 to n – 1 do
        for j = k+1 to n – 1 do
           A[i][j] = A[i][j] – A[i][k] * A[k][j];
        end for
        A[i][k] = 0;
     end for
  end for
end procedure
```

Assuming a decomposition into rows and an assignment into blocks of adjacent rows, **write a shared address space parallel version** using *LOCK* and *BARRIER* primitives for synchronization. Assume the existence of *P* processes executing *Eliminate*(*A*) in parallel. Indicate which variables are shared and which ones are private to each process.