School of Electrical and Computer Engineering

Georgia Institute of Technology

Advanced Computer Architecture

Moinuddin K. Qureshi, Instructor

**Homework 2, Submit Only Homework** (You will be receive points as long as you make reasonable attempt at solving the problem. We will release the solutions shortly after the due date so that you can grade it yourself).

**Name : Advait Koparkar**



**GT Account:**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Problem 1 (40 points): \_\_\_\_\_\_\_



Problem 2 (20 points): \_\_\_\_\_\_\_



Problem 3 (20 points): \_\_\_\_\_\_\_



Problem 4 (20 points): \_\_\_\_\_\_\_



Problem 5 (20 points): \_\_\_\_\_\_\_



Problem 6 (20 points): \_\_\_\_\_\_\_



**Total (140 points)** : \_\_\_\_\_\_



This exam is given under the Georgia Tech Honor Code System. Anyone found to have submitted copied work instead of original work will be dealt with in full accordance with Institute policies.

Georgia Tech Honor Pledge: “**I have neither given nor received aid on this homework.”**

[**MUST sign**:] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Note: Where needed, show all your intermediate results to receive full credit. Do all your work in this examination handout.

Use the back of the exam sheets if necessary.

Note: Please be sure your name is recorded on each sheet of the exam.

**GOOD LUCK!**

Name:\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem 1 – “Potpourri” (40 points, there are four parts, each worth 10 points)

(I) For a DRAM system, the time to open a row (tACT) is 15ns, to read a line from an open row (tCAS) is 15ns, and to close the row (tPRE) is also 15ns.

The total time to access a line with row buffer hit is 15 ns, whereas for a row buffer conflict is 45 ns.

If we find an invalid row buffer on memory access, the time to access a line would be 30 ns.

For a workload with row buffer hit rate of 40%, would you recommend using open-page policy or closed-page policy? closed

(II) n/a

(III) Multiprocessor performance can suffer because of false sharing between different threads. Array of locks can be a common source of false sharing, and a typical way to avoid this is to use padded locks. Shown below is the code for Padded Locks.

#define PAD\_SIZE ???

struct PaddedLock{

int Lock;

int pad[PAD\_SIZE];

};

PaddedLock MyLocks[NUM\_LOCKS];

If each int variable uses four bytes, then to avoid false sharing for a system with cache line-size of 64 bytes (without causing more than necessary wasted space), we must set PAD\_SIZE to 15

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem 2 “Power of Amdahl’s Law” (20 points)

A given program spends 20% of the instructions in portions that cannot be parallelized. The remaining 80% of the program is perfectly parallelizable, in that it gives linear speedup with the number of cores.

1. Using Amdahl’s law, the maximum possible speedup for this program using multiple processors is 5

Now, let us assume that this program takes 1 second on a single-core 1GHz processor. We have a choice between a dual-core or a quad-core chip that must run within the same power budget as the single-core processor. Assume that we use frequency scaling to limit power consumption, and that core performance is directly proportional to frequency. Also assume that the serial portion is quite small that we cannot turn-off unused cores.

1. For the dual core processor,

* What should be the revised frequency to maintain power budget? 0.5\*\*1/3
* `What is the execution time for this program? 7/10
* What is the speedup compared to uni-processor? 10/7

1. For the quad-core processor,

* What should be the revised frequency to maintain power budget? 0.25\*\*13
* What is the execution time for this program? 0.5
* What is the speedup compared to uni-processor? 2

IV. If the serial portion was 50%, then the execution time with the dual-core processor would be 0.81 seconds and with the quad-core processor it would be 0.69 seconds.

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem 3 “A MESI question” (20 points)

Consider two locations A and B that happen to be in different cache blocks. We will compare the state and activity of MSI and MESI protocols for a dual-core system where each core has a private 2-way set associative cache. Assume that at the start, cache blocks for both A and B are not present in the cache of any cores.

1. For the following sequence of accesses with MSI protocol, show the state of the block in each core (P1 and P2), whether that access causes a coherence-related writeback to memory, and if it induces a coherence message to other cores for that access. The first row is filled for your benefit.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Sequence (CoreID: Inst) | P1-state | P2-state | Writeback  Necessary? | Coherence Message on Bus? |
| P1: Load A | S | I | No | Yes |
| P1: Store A | M | I | No | Yes |
| P2: Load A | M | I | No | No |
| P2: Load B | I | S | No | Yes |
| P1: Store B | M | I | Yes | Yes |
| P2: Store B | I | M | Yes | Yes |

(II) Now, for the same sequence of accesses with MESI protocol, show the state of the block in each core (P1 and P2), whether that access causes a coherence-related writeback to memory, and if it induces a coherence message to other cores for that access. The first row is filled for your benefit.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Sequence (CoreID: Inst) | P1-state | P2-state | Writeback  Necessary? | Coherence Message on Bus? |
| P1: Load A | E | I | No | Yes |
| P1: Store A | M | I | No | No |
| P2: Load A | S | S | Yes | Yes |
| P2: Load B | I | E | No | Yes |
| P1: Store B | M | I | Yes | Yes |
| P2: Store B | I | M | Yes | Yes |

School of Electrical and Computer Engineering

Georgia Institute of Technology

ECE4100/ECE6100 (Section A, Q): Advanced Computer Architecture

Moinuddin K. Qureshi, Instructor

Midterm 2, April 7, 2015

**Name :\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**GT Account:**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Problem 1 (40 points): \_\_\_\_\_\_\_



Problem 2 (15 points): \_\_\_\_\_\_\_



Problem 3 (15 points): \_\_\_\_\_\_\_



Problem 4 (15 points): \_\_\_\_\_\_\_



Problem 5 (15 points): \_\_\_\_\_\_\_



Problem 6 (15 points): \_\_\_\_\_\_\_



**Total (100 points)** : \_\_\_\_\_\_\_\_



**Note: Problems 1 is compulsory.** You may choose to solve **ANY FOUR of the remaining five** problems (Problems 2-6) and mark X for the problem you choose not to do. If you solve all the problems, then we will discard the problem with the lowest score.

This exam is given under the Georgia Tech Honor Code System. Anyone found to have submitted copied work instead of original work will be dealt with in full accordance with Institute policies.

Georgia Tech Honor Pledge: “**I have neither given nor received aid on this exam.”**

[**MUST sign**:] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Note: Where needed, show all your intermediate results to receive full credit. Do all your work in this examination handout.

Use the back of the exam sheets if necessary.

Note: Please be sure your name is recorded on each sheet of the exam

**GOOD LUCK!**

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem 1 – “Potpourri Start” (40 points, there are five parts, each worth 8 points)

A) A processor has an L1 data cache that is 64KB and uses a linesize of 64 bytes. The processor supports a pagesize of 16KB. The design can access the L1 cache without having the TLB access in the critical path while incurring very little hardware and software complexity. To do so, the L1 cache must have been designed as a “Virtually Indexed Physically Tagged” cache, and it must be architected as at least a 2 way set-associative structure.

We have a machine with 32-bit virtual address space. We wish to design this machine for a system that uses a pagesize of 16KB and can address a maximum of 28-bit physical address space. Each Page Table Entry (PTE) contains the valid bit and modified bit in addition to the PFN.

The size of each PTE for this system is 16 bits

The total size of the Page Table is 2\*2\*\*18 bytes.

(B) DRAM banks and DRAM channels both allow independent request to be serviced concurrently. However, the fundamental difference between the two is (less than ten words):

Channels have their own buses, banks don’t  
  
Which option, bank or channel, is likely to be more costly and why ? channel – more hardware

(C) We are interested in beating the world grandmaster on the game *Go*. The game Go is played on a grid of 13x13, where each cell can have a black marble, or a white marble, or nothing. The state of the game can be captured in a cache line of 64 bytes using efficient encoding. We want to design a machine called DeepThinker which is a 10 million node machine for playing Go. Would you recommend that we design DeepThinker as a “Message Passing” machine or a “Shared Memory” machine? Message Passing

Why (in ten words or less)? State size is small good for parallel message passing interface

(D). There are two processors P1 and P2, each with an 8-way private cache with a line size of 64 bytes. Consider the following sequence:

Cycle 1: P1 reads the word “L” and gets a cache hit

Cycle 2: P2 writes to word “M” and gets a cache miss

Cycle 3: No cache activity from P1 and P2

Cycle 4: No cache activity from P1 and P2

Cycle 5: P1 reads the word “L” again but now get a cache miss.

Is this possible (yes/no)? Yes Why/how? False sharing. M and L lie in the same block

If it happens, how can you avoid it? Add padding

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem 1 – Continued

(F) Consider two locations A and B that happen to be in different cache blocks. We will figure out the cache state and activity for MESI protocol for a dual-core system where each core has a private 2-way set associative cache. Assume that at the start, cache blocks for both A and B are not present in the cache of any cores.

For the following sequence of accesses with MESI protocol, show the state of the block in each core (P1 and P2), whether that access causes a coherence-related writeback to memory, and if it induces a coherence message to other cores for that access. Use a “\*” to denote the absence of the line in the cache (for example at the start, the state for both lines A and B in all caches will be \*). Use “I” to denote that the line is present in the cache but in an invalid state. The first row is filled for your benefit.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Sequence (CoreID: Inst) | P1-state  A : B | P2-state A : B | Writeback  Necessary? | Coherence Message on Bus? |
| P1: Load B | \* : E | \* : \* | No | Yes |
| P1: Store B | \*:M | \*:\* | No | No |
| P2: Load B | \*:S | \*:S | Yes | Yes |
| P2: Load A | \*:S | E:S | No | Yes |
| P1: Store A | M:S | I:S | No | Yes |
| P2: Store A | I:S | M:S | Yes | Yes |
| P1: Store B | I:M | M:I | No | Yes |

Problem 2 “Prefetching”

You are analyzing a business processing application for a large-scale company. The data for their employees is stored as an array of structure, where each structure spans 256 bytes. You are analyzing a kernel that goes through the array of employee records but touches only the first 128 bytes of that record sequentially before accessing the next record. Therefore the access stream looks like below (assuming each address shown e.g. A, A+1, A+2, etc. is a cache line address, and cache line is 64 bytes):



We will analyze prefetching for this access pattern. Assume that a prefetch request is sent to memory only when there is a demand access for some other line (which may or may not be present in the cache).

1. What would be the accuracy of a next line prefetcher for this stream? 0.5
2. What would be the accuracy of a stride prefetcher (single entry) for this stream? 0.0
3. Given that this is a regular access pattern, we should be able to prefetch it with high accuracy. How?

Markov prefetcher will be better at finding access patterns

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem 4 “With great power comes great responsibility” (15 points)

A given program spends 40% of the instructions in portions that cannot be parallelized. The remaining 60% of the program is perfectly parallelizable, in that it gives linear speedup with the number of cores.

1. The maximum possible speedup for this program using infinite processors is 2.5

Now, let us assume that this program takes 1 second on a single-core 1GHz processor. We have a choice between a dual-core, a tri-core, or a quad-core chip that must run within the same power budget as the single-core processor. Assume that we use frequency scaling to limit power consumption, and that core performance is directly proportional to frequency. Also, assume that leakage power is negligible and that we do not have any design resources to implement power gating or clock gating.

[**NOTE: Report all your numbers till 3 decimal places**]

1. For the dual-core processor,

* What should be the revised frequency to maintain power budget? \_\_\_\_\_
* What is the execution time for this program? \_\_\_\_\_
* What is the speedup compared to uni-processor? \_\_\_\_

1. For the tri-core processor,

* What should be the revised frequency to maintain power budget? \_\_\_\_\_
* What is the execution time for this program? \_\_\_\_\_
* What is the speedup compared to uni-processor? \_\_\_\_

1. For the quad-core processor,

* What should be the revised frequency to maintain power budget? \_\_\_\_\_
* What is the execution time for this program? \_\_\_\_\_
* What is the speedup compared to uni-processor? \_\_\_\_

1. Which machine has the highest performance (dual-core, tri-core, or quad-core)? \_\_\_\_\_\_

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem 6 “Deep dive into Lab 4: Cache and Memory” (15 points)

Consider a system with two levels of cache, similar to Lab4. L1 is 32KB, 8-way, LRU-managed with 1-cycle latency, and L2 is 1MB 8-way LRU-managed with 10-cycle latency. The DRAM in the system consists of 1 bank with a row buffer of 256 byte. DRAM returns data in 50 cycles on Row Buffer Hit and 100 cycles otherwise. The system uses a linesize of 64B.

We are interested in doing a memory system analysis for the following kernel, which calculates the number of elements in an array that are above the average value (to do so, it first needs to calculate the average value). For the problem you are considering **N=64K, and each array element is four bytes.** Assume that the values for sum, average, and retval are always stored in registers, and we will ignore the activity incurred on the side of the instruction cache. Assume cold start for all structures.

**Num\_Values\_Greater\_than\_Average(N){**

**sum=0;**

**for(ii=0; ii<N; ii++)**

**sum += array[ii];**

**average = sum/N;**

**retval=0;**

**for(ii=0; ii<N; ii++)**

**if (array[ii] > average) retval++;**

**return retval;}**

A. What is the total size of the “array” data structure given each element is four bytes? 256 KB

1. How many L1 data accesses are generated by this kernel? 2\*64K
2. What is the hit rate of the L1 data cache (ignore conflict misses) ? 15/16
3. How many L2 accesses are generated by this kernel (hint: use B and C)? (1/16) \* 128K
4. What is the hit rate of the L2 cache (ignore conflict misses)? 0.5

1. How many DRAM accesses are generated by this kernel (hint: use D and E)? (1/32)\*128K
2. What is the row buffer hit rate of DRAM, assuming consecutive lines in same rowbuf? 0.75
3. What is the average latency to access DRAM? 62.5

**I: The average memory access time of the system is \_\_\_\_\_\_ cycles.**

**(Hint: Use C, E, and H) 3.203**

School of Electrical and Computer Engineering

Georgia Institute of Technology

ECE6100Q: Advanced Computer Architecture

Moinuddin K. Qureshi, Instructor

Midterm 2, (Due: November 28, 2017, 2pm EST)

**Name :\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**GT Account:**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Problem 1 (30 points): \_\_\_\_\_\_\_



Problem 2 (10 points): \_\_\_\_\_\_\_



Problem 3 (10 points): \_\_\_\_\_\_\_



Problem 4 (15 points): \_\_\_\_\_\_\_



Problem 5 (15 points): \_\_\_\_\_\_\_



Problem 6 (15 points): \_\_\_\_\_\_\_



Problem 7 (15 points): \_\_\_\_\_\_\_



**Total (110 points)** : \_\_\_\_\_\_\_\_



This exam is given under the Georgia Tech Honor Code System. Anyone found to have submitted copied work instead of original work will be dealt with in full accordance with Institute policies.

Georgia Tech Honor Pledge: “**I have neither given nor received aid on this exam.”**

[**MUST sign**:] \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Note: Where needed, show all your intermediate results to receive full credit. Do all your work in this examination handout.

Use the back of the exam sheets if necessary.

Note: Please be sure your name is recorded on each sheet of the exam

**GOOD LUCK!**

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem 1 – “Potpourri Small” (30 points, there are six parts, each worth 5 points)

**(B)** Shown below are examples of different processors, targeting different market segments. Arrange them from the lowest number of hardware threads on the chip to the highest.

W. A state-of-the-art GPU processor  
X. An extremely low power “nano” core targeted for wearable market  
Y. A “Quad-Core” processor with 4-way SMT  
Z. A 12-core server processor with no SMT.

(Fewest Threads) X < Z < Y < W (Most Threads)

**(E)** SMT is an area-efficient means of providing multiple hardware contexts in the same core. While many processor resources are shared between threads, some of the hardware structures are kept private to the thread. For the following structures, specify if they are typically “Shared Between Threads (SBT)” or kept “Private Per Thread (PPT)” in SMT.

* Instruction Cache: PPT
* Global History Register (GHR): SBT
* Return Address Stack: PPT
* Floating point unit: SBT
* Architectural Register File: PPT

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem 2 – “Cache Coherence” (10 points)

Consider two locations X and Y that happen to be in different cache blocks. We will figure out the cache state and activity for MESI protocol for a dual-core system where each core has a private 2-way set associative cache. Assume that at the start, cache blocks for both X and Y are not present in the cache of any cores.

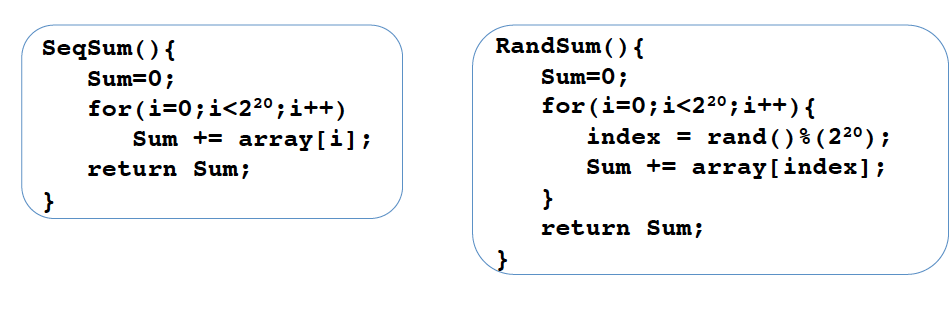
For the following sequence of accesses with MESI protocol, show the state of the block in each core (P1 and P2), whether that access causes a coherence-related writeback to memory, and if it induces a coherence message to other cores for that access. Use a “\*” to denote the absence of the line in the cache (for example at the start, the state for both lines X and Y in all caches will be \*). Use “I” to denote that the line is present in the cache but in an invalid state. The first row is filled for your benefit.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Sequence (CoreID: Inst) | P1-state  X : Y | P2-state X : Y | Writeback  Necessary? | Coherence Message on Bus? |
| P1: Load X | E: \* | \* : \* | No | Yes |
| P2: Store Y | E:\* | \*:M | No | Yes |
| P1: Load Y | E:S | \*:S | Yes | Yes |
| P1: Store Y | E:M | \*:I | No | Yes |
| P1: Store X | M:M | \*:I | No | Yes |
| P1: Load Y | M:M | \*:I | No | No |
| P2: Load Y | M:S | \*:S | Yes | Yes |

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem 4 “Refresher on DRAM Memories” (15 points)

We will analyze two kernels (SeqSum and RandSum) that perform an accumulation operation on a one million element (2^20) array. Note: both kernels are likely to return different results by design.



Assume that each element of the array is an integer, which occupies four bytes. Also that the local variables such as Sum, index, and the temporary registers for the **rand()** function are all resident in the register file (and hence do not cause any memory operations).

We are interested in analyzing the memory performance of these kernels for a system with a 4KB L1 cache with a linesize of 64 bytes. The 8GB single-bank DRAM memory has a row buffer of 512 bytes. The time to open a row (tACT) is equal to 16 cycles. The time to close a row (tPRE) is also equal to 16 cycles. The time to provide a column access strobe (tCAS) is 12 cycles. The time to transfer a line on the bus (tBUS) is equal to 4 cycles. Assume that the memory **employs an open-page policy**.

(**Hint:** The total size of the array data structure is equal to 2\*\*20 \* 4 Bytes)

**A.** How many cache misses are incurred by the SeqSum kernel? 1/16 \* 2\*\*20

**B.** How many cache misses are incurred by the RandSum kernel? (1-½\*\*10) \* 2\*\*20 (Approximately)

**C.** What is the Row Buffer hit rate in DRAM for SeqSum kernel? 7/8

**D.** What is the Row Buffer hit rate in DRAM for RandSum kernel? ½\*\*13(Approximately)

**E.** The average number of cycles for servicing a cache miss for SeqSum is 20 cycles

**F.** The average number of cycles for servicing a cache miss for RandSum is approximately 48 cycles

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem 6 “With great power comes great responsibility” (15 points)

We have a uniprocessor machine (M1) that consumes a power of 100 watts and operates at a frequency of 1 GHz. For this machine, we have a workload that spends one third of the time in serial portion (X), one third of the time in limited parallel portion which only has a maximum parallelism of two (Y), and one third of the time in in embarrassingly parallel portion (Z) Each portion takes one second on M1, as shown below.

(X=1 sec) (Y=1 sec) (Z=1 sec)

We are considering a Tri-Core machine (M2) that must be implemented within the same power budget as M1. Assume that the leakage power of all systems is negligible.

**A.** If all the three cores are in operation, then what should be the operating frequency of M2? 0.70 GHz

**B.** Assume that once the frequency of M2 is set to the one determined in the previous question, and that M2 could not dynamically change the frequency even if the available power increases at runtime, then how long will it take M2 to execute the following portions:

X = 1/0.7 secs

Y = 0.5/0.7 secs

Z = 0.33/0.7 secs

**C.** What is the speedup of M2 with respect to M1 for this workload? 3/1.83

**D.** Assume we now implement clock gating on M2 and enable Turbo mode (power saved can be dynamically used to improve the performance of the other active cores). How long will it take for M2 to execute the following portions?

X = 1 secs

Y = 0.5 / 0. 8 secs

Z = 0.33/0.7 secs

**E.** For the machine in part **D**, what is the speedup with respect to M1 for this workload? \_\_\_\_\_

Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Problem 7 “Lets Play Fetch” (15 points)

This problem deals with the efficiency of the Fetch unit of different architectures. Lets say we are concerned with a kernel that divides all the data elements of a million-element array by 2 (which can be easily done by a right shift operation). Each element of the array is one byte. Shown below is the sample code for a single core processor.

**uint8\_t array[220];**

**...**

**for(ii=0; ii< 220; ii++)**

**array[ii] = (array[ii]>>1); // right shift, for divide by 2**

For all problems, assume we have perfect branch prediction.

**A.** The baseline system is a uniprocessor machine with a 64-bit architecture, and does not support SIMD. How many total instructions are fetched for this kernel? 6 \* 2\*\*20

(Hint: each iteration contains LD, SHFT, ST, IndexIncr, CounterDecr, BRNZ)

**B.** For the machine in part **A,** we now implement SIMD instructions in the ISA (including a SIMD LD, ST, and SHIFT). Assuming we rewrite the code to exploit the SIMD capability of this machine, then the total number of instructions fetched for this kernel is 2\*\*20 / 8 \* 6

**C.** We decide to instead use a Vector processor that has a maximum VLEN of 64, and supports Vector Loads, Stores, and Shift instructions. Assuming the code is modified to work well on this Vector processor, the total number of instructions fetched for this kernel is \_\_\_\_\_\_

**D.** Now, lets change the machine to a GPU processor that can support 16 warps, each of which contains 32 threads. The total number of instructions fetched for this kernel now is \_\_\_\_\_\_