Last (family) name: _____

First (given) name: _____

Student I.D. #: _____

Department of Electrical and Computer Engineering
University of Wisconsin - Madison

ECE/CS 757 Advanced Computer Architecture II

Midterm Exam 2

Take-home, distributed 5/6/09, due Tues 5/12/09 12:00 noon.

Instructions:

1. This exam is open books, open notes, and open all handouts (including previous homeworks and exams, project descriptions, textbook, and readings).
2. All work is to be completed individually.
3. You may separate exam pages for convenient reference; please re-staple when you turn it in.
4. Feel free to use a word processor to type up your answers.
5. **You must show your complete work**. Points will be awarded only based on what appears in your answers.
6. Please submit the completed exam in person to Prof. Lipasti in his office at EH4613.
7. Failure to follow instructions may result in forfeiture of your exam and will be handled according to UWS 14 Academic misconduct procedures.

Section	*Type*	*Points*	*Score*
1	Discussion questions	10	
2	Cray T3E E-Registers	8	
3	Router Microarchitecture	6	
4	Dimension-sliced Crossbar Analysis	8	
5	Data Parallel Algorithms	5	
6	Transactional Memory	5	
7	SIMD Extensions for CUDA	8	
Total		50	

SECTION 1: DISCUSSION QUESTIONS

Answer the following questions with well-constructed, brief essays.

1) Dataflow machines such as MIT's Monsoon were an attempt to expose fine-grain parallelism by eliminating the Von Neumann fetch bottleneck. Compare and contrast the fundamental differences between *static* and *dynamic* dataflow machines. (2 points)

2) Comment on the implementation challenges for *dynamic* dataflow machines, and describe some of the proposed solutions to these challenges. (2 points)

3) What aspects, if any, of dataflow machines do you think are most likely to be adopted in future machines? Justify your answer. (1 point)

4) The AMD Opteron system and the Intel 870 chipset architecture provide two very different approaches for building small-scale symmetric multiprocessor machines. Based on the assigned readings, identify the key architectural differences and describe the relative strengths and weaknesses of these two designs. (2 points)

5) Performance scaling for MPPs can be problem-constrained, memory-constrained, or time-constrained. In your own words, describe what these terms mean, and comment on the relative parallel scaling of applications in each category. (2 points)

6) The Cray T3D and T3E machines provide virtual, logical, and physical processing element numbering. Describe what each of these are and why they are needed. (1 point)

SECTION 2: CRAY T3E E-REGISTERS

For this problem, assume a T3E-like address centrifuge that uses a mask in the E register to construct a virtual address and a virtual processing element (PE) number, which is then used to perform a get from remote memory.

Assume that three two-dimensional, row-major arrays A, B, and C (e.g. double B[1024][1024]) are physically distributed across 1024 processing elements, with one row in each PE (row 0 in PE 0, etc.). Further assume that you are computing the cross-product C = A x B in a parallel fashion by having each PE compute the dot products for its row of C by multiply-accumulating the local row of A against the remote column of B. Hence, references to A and C will be local, while references to the column of B will consist of a gather operation against remote nodes, using an E register. Further, elements of B are gathered using an 8-element vector get to fully populate a 64B cache line.

Here is the partial pseudo-code executing on each PE:

```
row = __virtualprocelemnum__;
for(col=0;col<1024;++col) {
  C[row][col] = 0.0;
  for(i=0;i<1024/8;++i) {
    double Bvec[8];
    __ereg_vget(/* dest= */ Bvec, /* ereg= */ 0, /* index=* / _______);
    for(j=0;j<8;++j)
      C[row][col] += (A[row][i*8 + j] * Bvec[j]);
}
```

The __ereg_vget will fetch 8 words starting from a global address based on *index* and place them into local memory at Bvec, using base, mask and stride values from *ereg* (in this case, ereg 0). Assume that stride increments are applied before the address centrifuge.

1) Given this information, specify the following. Show your work and state any assumptions. (5 points)

   Expression for *index* (in terms of row, col, i, j, …) : _____

   Mask for ereg 0: _____

   Stride for ereg 0: _____

2) Assuming that the latency of __ereg_vget() is approximately twice as long as the computation kernel (dot product of eight elements), rewrite the pseudo-code above to fully pipeline this latency. (3 points)

SECTION 3: ROUTER MICROARCHITECTURE

As single-chip multiprocessors evolved, much of on-chip router microarchitecture was adapted from off-chip routers. On-chip networks offer 1) much wider links (increased inter-node bandwidth) and 2) much faster links (lower latency). Briefly discuss how the move from off-chip to on-chip affects the performance impact of each of the following.

1) Adaptive routing (2 points)

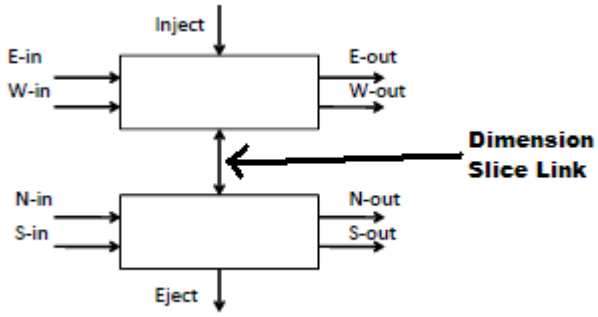
2) Lookahead routing (1 point)

3) Speculation (Virtual Channel Allocation and Switch Allocation) (1 point)

4) Area (specifically address control logic, buffering, and datapath) (2 points)

SECTION 4: DIMENSION-SLICED CROSSBAR ANALYSIS

Assume we have a torus network which uses dimension-sliced crossbars to reduce the total crossbar cost, as shown in the figure below.



The dimension slice link bandwidth at a single node is a critical network parameter (labeled in the diagram).

1) For 2D (NxN) and 3D (NxNxN) torus networks which use dimension-ordered routing, derive equations which specify the required inter-crossbar link bandwidth (expected average load) relative to the injection rate (p) and size N of the network. For the 3D torus assume a third sliced crossbar for the third dimension. (4 points)

2) Now, assume minimally adaptive routing is performed for a 4x4 2D torus. Calculate the required worst-case inter-crossbar bandwidth relative to injection rate p (C*p). For worst-case bandwidth, assume every packet must make the maximum possible number of turns. (4 points)

Note: We recommend coding a short program to find the answer to (2). Please provide pseudo-code, or actual code (it shouldn't be that long).

SECTION 5: DATA PARALLEL ALGORITHMS

Consider the data parallel code to perform a radix sort. Shown below the code is an unsorted list of 16 values. Draw the communication steps that are used for forming the sorted list, similar to Figure 1 in the Hillis & Steele paper. (5 points)

```
for j:=1 to log2(maxint) do
 for all k in parallel do
   if (x[k] mod 2^J< 2 ^J-1) then
     y[k] := enumerate
     c := count
   fi
   if (x[k] mod 2^J) >= 2 ^J-1
     y[k] := enumerate + c
     fi
   x[y[k]] := x[k]
 od
od
```

Note: Assume that all processors(n) are active, that sort keys are unsigned integers and that maxint is the value of the largest representative key value.

25	43	3	19	57	4	43	33	23	21	19	78	15	11	24	65

SECTION 6: TRANSACTIONAL MEMORY

The following code segments were taken from a parallel application.

```
bool flag1=false, flag2=false;
pthread_mutex_t lockA, lockB;
```

P1	P2

`pthread_mutex_lock(&lockA);`
`// omitted code`
`while(!flag1) { }`
`flag2 = true;`
`// omitted code`
`pthread_mutex_unlock(&lockA)`
	`pthread_mutex_lock(&lockB);`
	`// omitted code`
	`flag1 = true;`
	`while(!flag2) { }`
	`// omitted code`
	`pthread_mutex_unlock(&lockB)`

1) Perform a straight forward transformation of these code sections to be transaction-based by directly converting lock sections to atomic{} blocks. (1 point)

2) Discuss the performance impacts and potential problems that each the lock-based versions and transaction-based versions face. (4 points)

SECTION 7: SIMD INSTRUCTION EXTENSION FOR CUDA

For this problem, we will consider matrix multiplication as outlined in the assigned paper about CUDA. Assume that the CUDA hardware stays the same but we add SIMD instructions that use 4 single-precision floating point values (like SSE in the x86) for both ALU ops and vector load/stores. Initial assumptions about the program include:

- 10 registers per thread in the scalar (not vector) version
- Scheduling: 256 threads/block
- ¼ of instructions are loads
- 32 bit registers only (no vector registers; vector ops use 4 adjacent registers)
- 16-element tiled code (each thread computes 16 elements; Fig. 3b)

1) After converting the thread to use SIMD instructions, what happens to the theoretical peak performance of the program? (2 points)

2) Will the theoretical required memory bandwidth change, and if so, what is it? (2 points)

3) Comment on what you expect would happen to the register count per thread if SIMD instructions are used in the code? (1 point)

4) Considering your answers above, do you expect the number of blocks/SM to change? Why/why not? (1 point)

5) Finally, comment on how the above changes will affect the actual performance of the program, identifying any potential or actual performance bottlenecks. (2 points)