Exam

ECE 742 – Parallel Computer Architecture, Spring 2010

Instructor: Onur Mutlu

TA: Chris Fallin

April 9, 2009

NAME: _____SOLUTION_____

Problem	Points	Score
Name	2	
1	98	
2	15	
3	30	
4	35	
5	35	
6	20+20	
7	15	
Total	250 + 20	

- **READ THE QUESTIONS VERY CAREFULLY.**
- **Please write your name on *every* sheet (2 points).**
- This is a closed book exam. You are not allowed any cheat sheets.
- No electronic devices may be used.
- This exam lasts 1 hour 59 minutes.
- If you make a mess, clearly indicate your final answer. Clearly indicate what should be graded.
- For questions requiring brief answers, provide brief answers. You will be penalized for verbosity and irrelevance.
- Please show your work and reasoning. We cannot give you partial credit if you do not clearly show how you arrive at an answer.

Name: _____

Problem 1 (Short answers) [98 Points]

Please be concise, clear, and to-the-point in all your answers.

i) Amdahl's Law [6 points]

Amdahl's abstraction of a parallel program was two portions, one in which the program is completely serial (the serial portion) and the other in which the program is running completely in parallel (the parallel portion). In practice, the parallel portion doesn't achieve perfect speedup. Why is this so? Give three reasons, no more than 10 words each:

1. ___synchronization/communication overhead_____
2. ___load imbalance (between threads/tasks) _____
3. ___shared resource contention (between threads/tasks) _____

ii) Coupling Multiple Processors [5 points]

What key characteristic differentiates loosely coupled and tightly coupled multiprocessors? Explain (in less than 10 words).

Tightly coupled multiprocessors have a shared global address space.

iii) ROCK Runahead [6 points]

What is the purpose of runahead execution (or hardware scouting) employed in the Sun ROCK processor? In less than 10 words:

To tolerate memory latency

What does runahead execution achieve that multiple threads run on separate hardware contexts cannot achieve? In less than 10 words:

Improves single thread performance

Name: _____

iv) Combining Operations [8 points]

What is the idea of "combining operations" in the network? Why could it be useful? Explain in no more than 40 words.

The idea is to have the routers/switches combine multiple different operations on a shared memory location. This reduces synchronization overhead because multiple processors can update the same location in the network without incurring latency to transfer it first to their local execution units.

What kind of network topology is particularly suitable for combing operations? Give an example. Explain in no more than 20 words.

A multi-stage interconnection network, like a tree or omega network.

v) Network Contention [10 points]

Assume we have a 2-dimensional mesh network. When multiple packets arriving from different input ports need to go to the same output port, we say the output port is being contended for. There are three general ways to handle this contention. First is to _____buffer_____ some of the packets, second is to ____misroute_____ some of the packets, third is to ____drop_____ some of the packets.

Assuming contention is extremely rare, which option would you choose? Explain why, clearly, in less than 20 words.

Misroute. Buffering has unnecessary complexity/energy overhead of buffer hardware. Dropping adds complexity to enable retransmission. Misrouting avoids both problems without increasing latency.

vi) Locks (6 points)

Explain two reasons why a lock may not be necessary during dynamic execution even though it is required statically.

1. __The lock may be protecting a conditional write to shared data that never occurs dynamically_____
2. __Threads may update disjoint pieces of a data structure which is protected by a lock in its entirety__
3. __Threads may not contend for the critical section at any point in time during dynamic execution___

vii) Dataflow (6 points)

What does a dataflow implementation provide that is difficult to provide any other way? Explain in less than 20 words.

Irregular parallelism (Exploitation of irregular instruction level parallelism)

Name: _____

viii) Dependencies in Speculative Parallelization [6 points]

A speculative parallelization mechanism breaks a long sequential program into multiple tasks, e.g. as we saw in TLS, Multiscalar, Dynamic Multithreading. Tasks can be dependent on each other via registers or memory locations. What is the **key difference** between inter-task register and memory dependences in terms of how they are handled?

Register dependences: Known at compile time. Compiler inserts special instructions to communicate them.
Memory dependences: Known at run time. Hardware detects dependences dynamically and ensures execution obeys them.

ix) Multiscalar [7 points]

A multiscalar processor can reduce the branch misprediction penalty compared to a superscalar processor with a large sequential instruction window. TRUE or FALSE? Circle one.

Explain your reasoning in less than 30 words.

TRUE. A misprediction that happens within a task (i.e. not affecting task order) can be handled within the task, without requiring the flush of subsequent tasks. Any misprediction requires flush of all subsequent instructions in a larger sequential window.

x) Cache Coherence [6 points]

What is the advantage of the Illinois protocol (i.e., the addition of the Exclusive state) over the MSI cache coherence protocol? Be brief (20 words at most).

This state allows a cache/processor to write to a location (known to be the only clean copy) without notifying any other processor/cache.

xi) Parallel Performance Analysis [10 points]

We measure the performance of a parallel application with 4 threads on 4 cores, finding its execution time is 125 seconds. Then, we measure the performance of the same application with 32 threads on 32 cores, finding its execution time is 155 seconds. Assuming there were no software changes involved, describe how this is possible. (Assume each core can execute only one thread)

Performance decreases as the number of threads increases due to, e.g. communication overhead (e.g., ping-ponging) between threads.

We then measure the execution time of the application with a single thread on a single core machine. Can the execution time we find be 25 seconds? How? Describe your reasoning.

Yes. This is possible if the performance loss due to communication overhead is higher than the performance gain due to parallel execution of 4 threads.

Later, we measure the execution time of the same application with 4 threads on a 4-way multithreaded processor. We find the execution time of the application to be 300 seconds. Is there anything interesting about the result? Explain why this could be the case.

Yes. The execution time on the 4-way multithreaded processor is much higher than that on the 4-core processor.
This is very likely due to the additional resource contention threads cause each other in the processor pipeline and first-level caches in the MT processor.

xii) Routing (6 points)

a. What is the key difference between an oblivious routing algorithm and an adaptive routing algorithm? Explain in 10 words or less.

Oblivious → Unaware of network state, Adaptive → Aware of network state

b. Remember that Valiant's algorithm performs routing in two steps. First, it routes the packet to a randomly chosen intermediate node via dimension order routing. Next, the packet is routed to the destination node from the intermediate node, again via dimension order routing.

Is this algorithm **oblivious** or adaptive? Circle one. Oblivious.

Is this algorithm minimal or **non-minimal**? Circle one. Non-minimal.

c. Give an example of a non-minimal adaptive algorithm, in less than 10 words.

Deflection routing.

xiii) Dataflow Strikes Back (8 points)

Two "traditional control flow" constructs that are not possible to handle in a static dataflow architecture are ____loops_____ and __function calls____. Explain why, in less than 20 words:

A static architecture does not distinguish between tokens from different invocations of the loop or function call.

Dynamic dataflow architectures solve this problem by using ____tags_____. Explain how, in less than 20 words.

Each token is tagged with the call site or loop iteration it belongs to. A node fires only if its input tokens have the same tag.

Name: _____

xiv) Cache Coherence Again (8 points)

A designer has built a directory-based cache coherent system using the MESI invalidation protocol. Under certain workloads, however, the system performs very poorly, and after a simulation run and closer examination, the designer finds that there is a constant stream of invalidation requests between four of the nodes for one particular cache block. What is this phenomenon called? (no more than 5 words)

Ping ponging

Where and how is the problem best solved?

If due to true sharing, rewrite code to reduce sharing or use synchronization primitives that reduce communication. If not possible, consider using an update-based coherence protocol. If due to false sharing, change data layout via compiler or code rewrite to eliminate false sharing.

Name: _____

Problem 2 (Asymmetry) [15 Points]

An enthusiastic software engineer wrote a parallel program which spends a large fraction of its execution time in Amdahl's serial bottleneck (serial portion) when executed on a symmetric multi-core processor with 16 small, in-order-execution cores, each running at 1GHz. To improve the performance of the program by speeding up the serial portion, the engineer uses two different asymmetric multi-core processors, both of which are area equivalent to the 16-core symmetric machine.

First, she runs the program on a dynamically asymmetric multi-core processor that executes serial sections on one of the cores at 2GHz. She finds that the program speeds up by 1%.

Next, she runs the program on a statically asymmetric multi-core processor that has 12 small cores and one large, out-of-order-execution core. Each core executes at 1GHz. She finds the program speeds up by 10%.

What could be the reason for the higher speedup on the statically asymmetric multi-core processor? Explain in enough detail. (Assume there are no faults and the program runs correctly)

Extracting ILP and MLP by using out-of-order execution improves performance more than simply speeding up execution by increasing frequency in serial sections. This could be because the serial section's performance is memory bound, which cannot be improved by simply changing processor clock frequency.

Name: _____

Problem 3 (Simultaneous Multithreading) [30 Points]

Two designers for a microprocessor company found out that long latency load instructions were causing performance problems in the new 2-way SMT processor they architected. Explain why this could be a problem.

Long-latency load that blocks one thread's (Thread M) execution can cause the other thread (Thread C) to stall since threads share pipeline and instruction window resources, and the first thread's instructions can fill up those resources.

To reduce the performance problem, the designers came up with a creative solution, after days of experimenting and sleepless nights (and mounds of coffee, of course). Their idea was to flush the subsequently fetched instructions of a thread immediately after a load instruction is found to be a data cache miss. Fueled by the "Eureka moment" excitement of their new idea, they spent a few more sleepless nights and trillions of simulation cycles (not to mention the more mounds of coffee, of course) to test this idea on a suite of 1000 2-way multiprogrammed workloads. They found the idea improved performance (weighted speedup, harmonic speedup) in 700 workloads, but it degraded performance in 300, compared to the baseline case of SMT where no flushing is performed after a load miss.

Explain why and when the engineers' idea could be a good idea. (for performance; give as many reasons as makes sense)

Frees up pipeline/window resources and allows Thread C (thread without the long-latency stall) to make progress without being restricted by the stalling thread.

Explain why and when the engineers' idea could be a bad idea. (for performance; give as many reasons as makes sense)

Reduces the performance of Thread M (stalling thread) because (1) it cannot overlap other independent misses under the shadow of the miss, (2) it needs to refetch all instructions after the load miss and fill its window.

Name: _____

What could be the characteristics of the workloads that lost performance with the new idea? (assume each workload consists of two single-threaded applications)

The thread that incurs long-latency stalls has high memory-level parallelism (or a lot of independent operations) under a miss. Flushing that thread reduces the possibility of overlapping the latency of the miss, degrading that thread's performance (and possibly overall system performance).

You are a colleague of the engineers, who has listened to the discussions about alternative ways of solving the problem. Suggest one way of improving the proposed idea (feel free to use the ideas we have discussed and read about in class). Describe your technique.

Idea 1: Wait until Thread M stalls due to a full instruction window before flushing its instructions. This allows any independent misses to be generated and sent to memory before the flush.

Idea 2: Predict if there will be any independent misses happening in the shadow of the miss. If so, execute up to the last miss-causing instruction and flush after that. Otherwise, flush immediately after the first miss is seen.

Idea 3: Predict if there is enough MLP under the miss. If so, put Thread M into runahead mode to exploit MLP. Otherwise, flush Thread M right after the miss.

What are the advantages and disadvantages of your technique? Is your technique strictly better than the engineers' proposal?

Advantages (Ideas 2 and 3):

- Higher performance when prediction is correct

Disadvantages (Ideas 2 and 3):

- More hardware complexity
- Wasted prediction energy and execution (in case of runahead) on incorrect prediction
- Possibly lower performance on incorrect prediction (due to not releasing resources early for Thread C)

No, due to the above disadvantages, these techniques are not strictly better.

Name: _____

Problem 4 (Higher Performance Locking) [35 Points]

In lecture, we have discussed at least two approaches to improving the performance of lock-based synchronization: Accelerated Critical Sections (ACS) and Speculative Lock Elision (SLE). Compare and contrast these two following the questions below:

i. What is the basic idea of ACS? (less than 20 words)

Execute the critical section on a more powerful core that can execute it faster and preserve shared data locality.

ii. What is the basic idea of SLE? (less than 20 words)

Speculatively execute a critical section without acquiring the lock, check if there is any data conflict; if not, commit critical section; if conflict, roll back execution.

iii. Which approach, if any, requires the programmer to modify their programs?

Neither.

iv. Give at least two reasons why ACS can provide better performance than SLE. Be concise, but specific, describing any conditions that lead to this result.

1. ACS speeds up the execution of a critical section; SLE does not.
2. ACS keeps shared data/lock in a single cache, reducing ping-ponging; SLE does not.
3. ACS does not incur re-execution overhead since it does not speculatively parallelize critical sections; SLE does on a data conflict.

When critical sections are contended, long, and update the same shared data location, ACS can therefore provide significantly higher performance as all of the above three conditions are exercised.

Name: _____

v. Give at least two reasons why SLE can provide better performance than ACS. Be concise, but specific, describe any conditions that lead to this result.

1. SLE does not incur overhead to transfer private data to a large core or overhead to transfer control to a large core; ACS does.
2. SLE executes non-conflicting instances of a critical sections in parallel; ACS executes them serially.
3. SLE uses the area of a large core (dedicated to critical section execution in ACS) to smaller cores, which can execute parallel threads in the parallel section. This improves parallel throughput assuming the large core cannot execute parallel threads in ACS.

When critical sections are contended, long, and without any data conflict (i.e. do not update the same data locations), SLE can therefore provide significantly higher performance as all of the above three conditions are exercised.

vi. Assume each instance of a heavily contended, long critical section updates the same element in a shared data structure. Which approach, if any, would you expect to perform the best? Why?

ACS, because it accelerates each instance of the critical section, reducing serialization.

On the other hand, SLE will speculatively execute yet roll back each instance, causing additional overhead due to ping-ponging and rollback, which likely reduces performance compared to conventional locking.

vii. Assume each instance of a heavily contended, long critical section updates disjoint elements in a shared data structure. Which approach, if any, would you expect to perform the best? Why?

Likely SLE, because it executes instances of the critical section completely in parallel, eliminating the need for and overhead of serialization due to locking.

On the other hand, ACS serially executes those critical sections… although it executes them faster and does not suffer from false sharing if it exists… therefore, the best answer actually is:

IT DEPENDS (if parallelizing critical section instances provides better performance than executing them serially yet faster, then SLE; otherwise ACS).

Name: _____

Problem 5 (Memory Scheduling) [35 Points]

In class, we examined three memory scheduling techniques that can be employed when DRAM is shared between multiple threads: the commonly used FR-FCFS (or row-hit first) scheduler which aims to optimize for DRAM throughput, the stall-time fair memory scheduler (STFM) and the parallelism aware batch scheduler (PAR-BS). In this question, you will compare and contrast the three approaches.

i. How does the FR-FCFS (row-hit first) scheduling algorithm work? No more than 20 words.

Prioritization order:

1. Row hits before everything else
2. Older requests (all else being equal)

ii. What are the disadvantages of FR-FCFS in a multi-core system?

Thread-unaware → threads with high row hit rates (e.g. streaming threads) cause temporary starvation to other threads, leading to unfairness and low system utilization.

iii. Explain the key idea of STFM algorithm. No more than 20 words.

Estimate slowdown of threads due to sharing the main memory system. If unfairness is intolerable, prioritize the most slowed-down thread's requests.

iv. Explain the key idea of the PAR-BS algorithm. No more than 20 words.

Preserve each thread's bank level parallelism and improve system throughput by forming a ranking of threads and servicing threads in rank order. Batch requests and prioritize older batches to prevent starvation due to ranking.

Name: _____

v. What are the advantages of STFM over PAR-BS?

1. Can provide better fairness and allows the system to trade-off fairness and throughput
2. Better suited to allow the system software to enforce slowdown guarantees

vi. What are the advantages of PAR-BS over STFM?

1. Preserves intra-thread bank parallelism
2. Improves system throughput by using a system-performance aware thread ranking mechanism (shortest job first)
3. Provides strict starvation freedom using batching; STFM cannot if it estimates slowdowns incorrectly
4. Simpler: No need to explicitly estimate slowdowns

vii. You find that the FR-FCFS algorithm provides higher DRAM throughput (data bytes transferred across DRAM bus per second) than the STFM algorithm, but STFM provides higher system throughput (total number of instructions retired in the entire system over a period of time) than FR-FCFS. Explain how this is possible.

Optimizing for DRAM throughput unfairly prioritizes threads that have better DRAM bandwidth utilization. These are the threads that cause starvation to other threads.

STFM allows the otherwise-stalling cores to make progress by reducing starvation they encounter in the memory system. This improves system throughput, but reduces DRAM throughput because cores with not-as-good DRAM bandwidth utilization are allowed to make progress.

Name: _____

Problem 6 (Design Question - Smart resources, dumb resources) [20+20 Points]

In class, we have discussed solutions to reducing inter-thread interference in shared multi-core resources by designing control mechanisms in resources (e.g. caches, on-chip network, and memory controllers) that aim to reduce inter-thread interference. The goal of these mechanisms was to improve fairness and performance – let's call this approach the "smart resources" approach. An alternative, which we briefly discussed, is to keep resources simple and "dumb" (as they are in existing system) without such control mechanisms, but control the rate at which each core can inject in the resources. Let's call this latter approach "source throttling" as we did in the lecture.

What are the advantages of "smart resources" over "source throttling"?

1. Each resource is designed to be as efficient as possible → more efficient design using custom control techniques for each resource
2. No need for estimating interference across the entire system (to feed a throttling algorithm). Interference estimation can be local to each resource.

What are the advantages of "source throttling" over "smart resources"?

1. Prevents overloading of any or all resources (if employed well)
2. Can keep each resource simple; no need to redesign each resource
3. Provides prioritization of threads in the entire memory system; instead f per resource
4. Eliminates conflicting decision making between resources

BONUS (20 points):

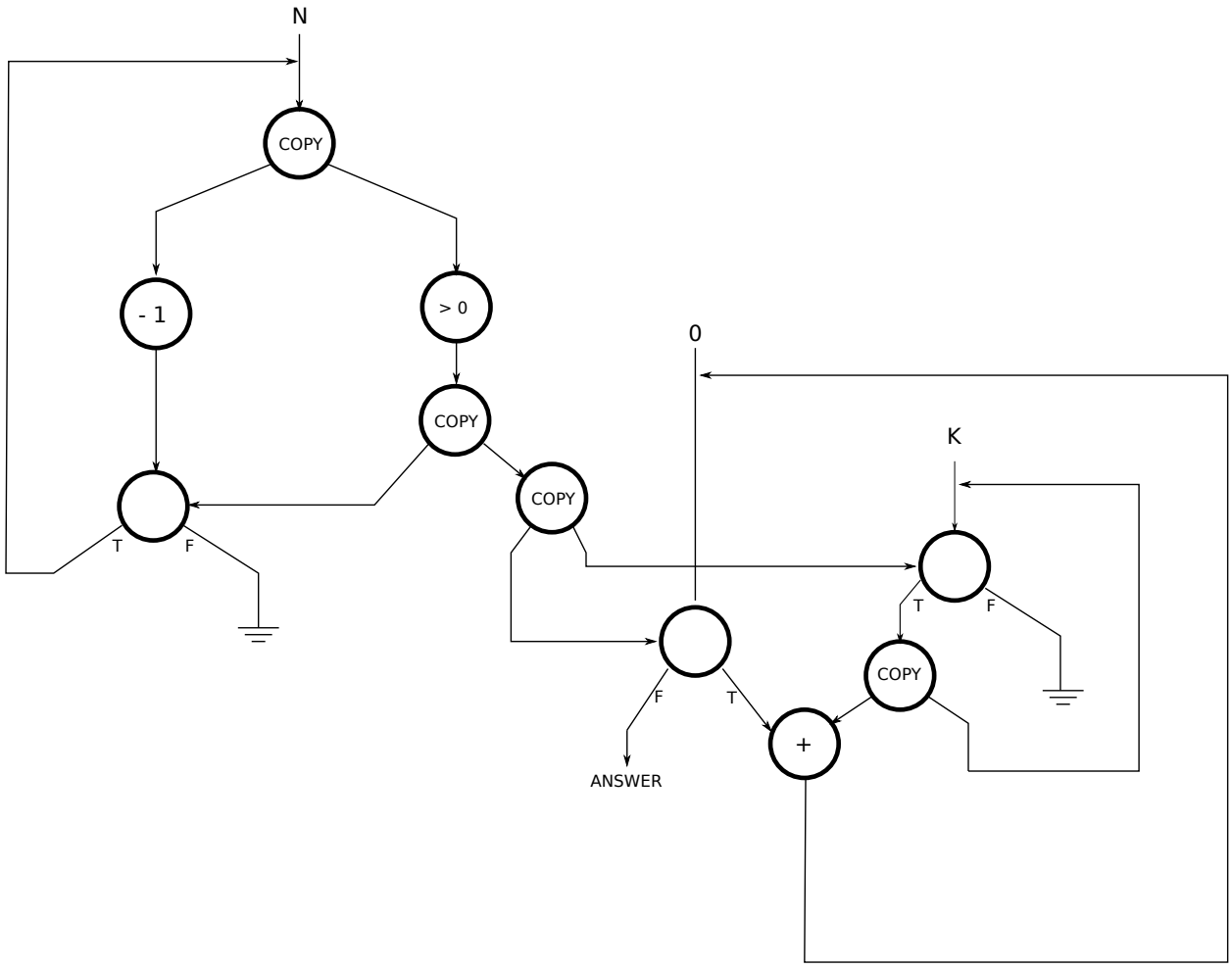
Can both approaches be combined to provide higher performance than either approach alone? If so, describe how you would go about combining them. Explain your ideas clearly.

Yes! Talk to us if you are interested.

Name: _____

Problem 7 (Dataflow) [15 Points]

The following data flow graph requires three non-negative integer input values 0 (zero), N, k as shown. It produces a value "Answer", also shown.



What function does this data flow graph perform (in less than ten words, please)?

N * k