Name: Jared Raiola GT Username: jraiola3

CS4290: Advanced Computer Organization

Summer 2020
Final Exam
07/23/2020

150-Minute Exam (Start: 2:40, End: 5:10)

- This is a closed-book exam.
- Write your answers in the space provided.
- Please write your name and GTID on every page.
- For fractions that are difficult to compute (ex. ⅓), you may leave them as is.
- Write any assumptions you may make, if they are important to your solution.

Question	Points	Points Earned
1	15	
2	10	
3	5	
4	15	
5	20	
6	20	
7	15	
Total	**100**	

Name: Jared Raiola GT Username: jraiola3

Problem 1. Architecture Basics 15 points

A. Say that we have developed an ***in-order***, 4-wide superscalar processor with the traditional 5-stage pipeline. For this processor, we use a register file that allows 8 reads and 4 writes in a given cycle. However, while reads to the same register in a cycle are supported, the writes must be to different registers. For the instruction scheduler we are adding to the processor, what types of data dependencies does the scheduler need to handle? (**2 points**)

- The scheduler must handle the Read-After-Write or RAW dependency. This is the case because an instruction might attempt to read a register that is currently being written to.
- Depending on the structure, the scheduler also may have to handle the Write-After-Write or WAW dependency, as data may be lost by being overwritten.

B. What is the difference between data dependencies and hazards (write two)? (**2 points**)

- Data dependencies are issues happen with the scheduler and are typically handled by stalling.
- Hazards happen in the pipeline and require the pipeline to be flushed, or some other method like checkpointing to deal with the occurrences.

C. In terms of registers, what is edge-triggered logic? And Why it helps? (**2 points**)

- Edge triggered logic is only allowing the updating the registers once every cycle. This helps because it forces only one register write per cycle, meaning less write after write and read after write dependencies.

Name: Jared Raiola GT Username: jraiola3

D. During exception handling, the system will want to flush the ROB before jumping to the exception handler code. Someone suggests that this can be done by moving the tail pointer up to the point after the exception causing instruction. Why is this not sufficient for correct re-execution? Specifically, ***clearly*** state what other steps that must be taken to ensure correct execution of the exception handling code. (**2 points**)

- This is not sufficient because the return target may change depending on the exception handler code, along with data in registers. The return target, along with the registers must be pushed onto the Return Address Stack or RAS in order to properly deal with the exception handling. When the system returns from the exception handler, the stack will be torn down, restoring all old values.

E. Given the following breakdown for Program A, which contains 10 billion instructions:

Instruction Type	**% of Program**	**CPI**
Integer	30	3
Floating-Point	20	7
Branch	10	5
Memory	40	8

Calculate the Average CPI of the program? (**2 points**)

- AvgCPI = (.3 * 3 + .2 * 7 + .1 * 5 + .4 * 8) = (.9+1.4+.5+3.2) = 6

Name: Jared Raiola GT Username: jraiola3

F. Given a Program B with the following characteristics:
   - 20% of time is spent on graphics processing
   - 30% of time is spent on physics computation
   - 50% of time is spent on machine learning computation

   And a new machine with the following specifications:
   - A new GPU that improves performance by 2x
   - A specialized vector unit that improves physics computations by 3x
   - A machine learning accelerator that increases ML computations by 5x

   What is the speedup of running Program B on the new machine? (**2 points**)

- Speedup is AvgCPIA / AvgCPIB
- 1 / (0.2 / 2 + 0.3 / 3 + 0.5 / 5) = 3.33
- 3.33 is the speedup of running program B on the new machine

G. For the given hardware structures, fill-in the label **A** if the component is considered part of the Architecture, or fill-in in the label **M** if it is considered part of the Microarchitecture. (**3 points**)
   Note: Architecture states are visible/managed by programmer/OS.

a. A - Register File	g. M - TLB
b. M - Functional Units	h. M - VIVT Cache
c. M - PC	i. M - PIPT Cache
d. M - Branch Predictor	j. A - Ld/St Queue
e. A - ROB	k. M - Write Buffer
f. M - Pipeline Latches	l. IGNORE - DVFS Logic

Name: Jared Raiola GT Username: jraiola3

Problem 2. Branch Prediction 10 points

Say there is a 32-bit machine with the following specifications:

- Machine implements an ISA with **fixed-length, word-sized** instructions
- Machine contains **byte-addressable** memory.
- Machine contains a **g-share** PHT with **64-entries**, with the GHR initialized to all not-taken.
- Each PHT entry uses a **2-bit** Smith counter, with each counter initialized to **weakly-taken**.

A. Given the following branch stream, then what are the PHT entries that below four branches will map to? Ignore the repetition factor, aka calculate the indices of the entries the branches will map to during the first iteration of execution. The stream is given in the form of "PC - Direction", where T means the branch is Taken and NT means the branch is Not-Taken. (**2 points**)

```
0x000C - NT, 0x0014 - T, 0x000C - T, 0x0014 - T, Repeat
```

- G-share we xor the bits.
- 0x000C goes to index 12, updates the PHT to weakly not taken and updates the Global History with a not taken.
- 0x0014 goes to index 20, updates the PHT to strongly taken and updates the Global History with a taken.
- 0x000C goes to index 13, updates the PHT to strongly taken and updates the Global History with a taken.
- 0x000C goes to index 16, updates the PHT to strongly taken and updates the Global History with a taken.

B. Assume that now, the predictor is changed to use the Two-Level Adaptive scheme. What is the minimum storage size such that the branch stream from part A does not result in a conflict in the PHT entry? Make sure to show work of all size values used to calculate total minimum storage size. (**3 points**)

- The minimum total storage size would require a history for every branch and a PHT for every HRT entry. This means that minimum size is equal to HRT * the number of bits per HRT entry + HRT entry number * PHT size

Name: Jared Raiola GT Username: jraiola3

C. Aside from branch identification and misprediction rate, detail the issues and possible solutions that a global history predictor may have with ***very deep*** pipelines. (**3 points**)

- The issue that a global history predictor may have with deep pipelines is that it will make multiple predictions before receiving the results from previous predictions. This means that if the first prediction is wrong, the history is incorrect, causing the incorrect PHT indices to be accessed when predicting other branches.

D. What are the pros/cons with predication (one for each)? (**2 points**)

- Pro: The ability to schedule other instructions sooner, creating for an overall faster pipeline as all stages will be used and instructions will not have to wait.
- Con: The introduction of hazards as data may be lost during incorrect predictions, as the pipeline will need to be flushed and some systems may not be built to handle that situation.

Name: Jared Raiola GT Username: jraiola3

Problem 3. Tomasulo's Algorithm 5 points

A **single-issue** processor uses Tomasulo's algorithm in its floating-point unit, which has one adder and one multiplier, each with its own set of reservation stations. There is only one CDB, and broadcast on this CDB takes an entire cycle. The processor is executing the following sequence of instructions and, for each instruction, we show the cycle in which the instruction is fetched, decoded, issued, begins to execute, and writes the result.

	Instruction	Fetch	Decode	Issue	Execute	Write
I1	`mul t1, t2, t2`	1	2	3	4	8
I2	`add t1, t1, t2`	2	3	4	9	10
I3	`mul t2, t2, t3`	3	4	5	8	13
I4	`add t3, t1, t1`	4	5	6	11	12
I5	`mul t1, t1, t1`	5	6	7	12	16
I6	`add t2, t3, t4`	6	7	11	**??**	**??**
I7	`add t1, t5, ??`	7	8	13	17	18

A. Assuming a physical register file, describe the actions that occur when instruction **`I5`** is in the write stage. (**3 points**).

- When I5 is in the write stage the processor looks up t1 in the TLB, gets the address, puts t1 on the CDB and then writes it to t1s address.

B. Someone suggests we can allow more instructions in-flight by removing an instruction from its reservation station once it finishes execution and is ready to enter the write stage. Why is this a bad idea? (**2 points**)

- This is a bad idea because in doing so we develop a hazard where we could be reading a register before the proper value is written to it, making our instruction perform in incorrect operation.

Name: Jared Raiola GT Username: jraiola3

Problem 4. Advanced Architecture 15 points

A. Someone suggests that instead of checkpointing at every branch, we should only checkpoint at branches with high misprediction rates. What are the pros and cons of using such a scheme? (**2 points**)

- The pros are that we would use less memory space by checkpointing only at branches with high misprediction rates,
- The cons are that this scheme would require more complex logic to implement and would completely fail for a system that does not branch the same way often. This would cause no checkpoints to be created.

B. Again, based on using the scheme where a checkpoint is only created for branches with high misprediction rates, what should be done when a misprediction occurs on a branch where a checkpoint was not created? Give two possible mechanisms. (**2 points**)

- Mechanism one would be flushing the entire pipeline, losing all the progress on the current instructions.
- Mechanism two would be to stall the pipeline with NOPs and retry that same branch instruction.

C. The Load/Store Queue is one of the more complicated structures in a modern microarchitecture. Describe its purpose and what mechanisms make this structure so complicated (at least two). (**3 points**)

- The load and store queue is used for the primary purpose of regulating when data is loaded and stored to and from memory. It is complicated because it takes into account all data dependencies and sends specific information depending on what data is dependent.

Name: Jared Raiola GT Username: jraiola3

D. What is the biggest roadblock to a compiler exploiting ILP in a program? Give a microarchitecture optimization technique for instruction cache that tries to address this issue. (**2 points**)

- The biggest roadblock to a compiler exploiting ILP in a program is small blocks, since compilers schedule on block granularity. A microarchitecture optimization for this is to loop unroll these blocks, so the compiler can branch less times.

E. What type of information is stored in the TLB?
Hint: Don't forget about security. (**2 points**)

- The TLB stores the addresses of locations in memory for Physically Indexed Physically tagged caches. Specifically, it stores the bits that correspond to those locations, and is protected through

F. How does SECDED provide Memory Reliability and why do errors happen? (**2 points**)

- SECDED provides memory reliability by providing the detection of two-bit errors or the correction of one-bit errors, this prevents bit flip errors that could be due to alpha particles or attacks.

G. For each data structure, specify a type of prefetcher that would be useful if a program traversed said data structure. (**2 points**)

a. Arrays write

b. Linked List no-write

Name: Jared Raiola GT Username: jraiola3

Problem 5. Caches and Memory 20 points

A. What are the **four** types of cache misses and give an example of a caching mechanism or strategy that can help reduce such misses? (**4 points**)

- Compulsory-miss
- Cold miss
- Cache miss
- Conflict miss
- A caching strategy that can help reduce misses is keeping the recent data available in available memory in case data is needed again immediately after.

B. What is the aliasing problem and for a Physically Tagged, Virtually Indexed Cache, what must be true about the index bits for aliasing not be an issue? (**3 points**)

- The aliasing problem for a VIPT cache is that since there is no TLB, some indexes might mask to the same memory address. In order for it not to be the case, the amount of index bits must be able to access all locations in memory.

C. The Tag and Data portions of a cache can be accessed in serial or parallel. Specify the different cache levels in a multi-level cache system and detail whether you expect the tag/data accesses to occur in serial or parallel and why? (**3 points**)

Name: Jared Raiola GT Username: jraiola3

D. What is the purpose of the MSHR? (**3 points**)

- The MSHR is used to hold recently used data so it is quickly available for access again, instead of having to recall the data back from its memory location.

E. Describe the destructive read property of DRAM and how DRAM deals with the problem. (**2 points**)

- The destructive read property of DRAM is when the charge of DRAM clears the data from a cell after reading from it. DRAM deals with this issue by storing the data in a row buffer and then writing the data back to the cell.

F. What are the different DRAM hardware structures that a memory controller can take advantage of to increase memory-level parallelism? (**2 points**)

- The DRAM hardware structure can use open page and closed page policies to increase memory level parallelism.
- Open page is when the contents of a row are not given back until the next row is available to access
- Closed page is when the contents of a row are given back immediately after being read.

Name: Jared Raiola GT Username: jraiola3

G. What is FR-FCFS scheduling and why would a memory controller use it? (**3 points**)

- FR-FCFS scheduling is schedules the first instruction available, even if it is not the first instruction in the queue. A memory controller would want to use this type of scheduling because it would consistently allow available instructions to be scheduled immediately, instead of stalling and waiting to do the instructions in order.

Name: Jared Raiola GT Username: jraiola3

Problem 6. Coherence and Consistency 20 points

A. Given a machine with the following details:
   - 32-bit machine with a **byte-addressable** 2GB physical memory space that uses 4KB pages
   - An 8KB 4-way set-associative write-back, write-allocate cache
     - Cache block is 64B
     - Cache uses true LRU replacement policy
     - Cache is virtually indexed and physically tagged
     - Cache implements MOESI protocol.

   How big (in bits) is the tag store (Do not include valid/dirty bits)? (**3 points**)

- A tag store is the number of lines * the tag size + LRU size.

B. Given a system with a basic MSI protocol, what is the benefit of extending the protocol with the O state? (**2 points**)

C. Given a system with a basic MSI protocol, what is the benefit of extending the protocol with the E state, and what additional hardware should be added to support the E state? (**2 points**)

D. Given a system with a basic MSI protocol, what is the benefit of adding a bus response arbitration scheme? (**2 points**)

- The benefit of adding a bus response arbitration scheme is that we can see when information leaves the bus so we know when to use it sooner.

Name: Jared Raiola GT Username: jraiola3

E. Directory based systems can suffer poor scalability due to metadata overhead and a centralized location. Why do those factors lead to poor scalability and what are possible solutions (write two) to these problems? (**3 points**)

- A centralized location and metadata overhead leaves to poor scalability because many processes can attempt to access the same memory at the same time. Solutions include localizing memory and

F. To implement synchronization operations such as mutex locks or barriers, what considerations should be made in terms of cache coherence to gain the best performance. (**2 points**)

- When implementing synchronization operations such as mutex locks and barriers, to gain the best performance, one must consider how many processes will be trying to access a certain memory location as once, how to deal with different data dependencies (specifically read after write), and how each separate process should be able to access the memory locations.

G. Describe sequential consistency and explain why processors typically implement weaker consistency models. (**2 points**)

- Sequential consistency is when all writes must happen before any read access is given at all. Processors will typically implement weaker consistency models because these reads and writes can be sped up. If a memory location is not being written to, a process does not have to necessarily wait for read access in order to get that data value, allowing for less latency and quicker execution.

H. High-performance computing applications share data in various ways. One well-known sharing pattern is *migratory data sharing (MDS*), in which core A writes to a (cache) block, core B reads it, then later B writes to the block, and then core C reads it, and then later writes to it, etc. This occurs commonly for pipelined parallel patterns.

Name: Jared Raiola GT Username: jraiola3

Cache coherence can be maintained, but at low performance, by only implementing the “I” and “M” states. Consider that implementing any state beyond M and I cost a significant amount of manufacturing money to verify the designs are correct, to test new chips for correctness as they are being manufactured, etc. For MDS sharing pattern, propose adding ***one or more*** states to M, I to enhance performance **and justify your answer**. (**4 points**)

Name: Jared Raiola GT Username: jraiola3

Problem 7. Many-Core/GPU 15 points

A. Detail the pros and cons of Simultaneous Multi-Threading? (**2.5 points**)

- Simultaneous Multi-Threading allows multiple processes to complete at the same time.
- Pros:
  - When a process is stalled, during its wait time another process can take over, allowing for less downtime.
- Cons:
  - Can cause hazards and stalls when multiple threads of the same process are attempting to access the same memory location.

B. Describe the cases where one-thread may stall the execution of another thread when performing SMT execution. (**2.5 points**)

- One thread may stall the execution of another thread when they are both attempting to access the same data through writes and reads.
- A thread can also stall the execution of another thread when all threads are waiting for the final thread to finish in order for the process to finish, stalling all execution.

C. Describe 3 different NoC configurations and for each configuration, give an example of a machine that would use such a configuration. (**2.5 points**)

Name: Jared Raiola GT Username: jraiola3

D. Out-of-order machines try to hide periods of long latency instructions through the overlapping execution of different, independent instructions. How do GPUs typically try to deal with periods of long latency instructions? (**2.5 points**)

- GPUs typically try to deal with periods of long latency instructions by multi-threading their processes.

E. Why does control-flow divergence cause performance inefficiencies in GPUs? (**2.5 points**)

- Control-flow divergence causes performance inefficiencies in GPUs because it causes a division of resources, where some threads will stall other threads, causing for a long waiting period.

F. What is requirement for memory coalescing in GPUs, and why does it provide a performance benefit (**2.5 points**)