CHALMERS

2007-12-20

Exam in DAT 105 Computer Architecture

Time: December 20, 2007 after lunch in the M building

Person in charge of the exam: Minh Quang Do, 031-772 5216

Supporting material/tools: Chalmers approved calculator.

Exam Review: January 8 2008 between 10-12. Location will be announced on the web.

Grading intervals:

- **Fail**: Result < 24
- **Grade 3**: 24 <= Result < 36
- **Grade 4:** 36 <= Result < 48
- **Grade 5:** 48 <= Result

Important note: Answers should be given in English

GOOD LUCK!
Per Stenström

[General disclaimer: If you feel that sufficient facts are not provided to solve a problem, either 1) ask the teacher when he visits the exam, or 2) make your own additional assumptions. Additional assumptions will be accepted if they are reasonable and required to solve the problem. Always make sure to motivate your answers.]

ASSIGNMENT 1

A) You are comparing three computers to see which offers the best performance for your applications. To compare them you have measured the execution time of three equally important representative benchmark programs. The results of these measurements are provided in the table below. Summarize the performance of the computers so that the one with the best performance can be identified. Which computer is best? (4 points)

Benchmark	Computer A	Computer B	Computer C
B1	2000	1800	2200
B2	200	160	180
B3	600	660	660

B) You have developed a new sensor interface card that can be used by a computer system to read the input from up to 64 different sensors. To read a sensor value, a drive signal is first sent to the selected sensor using a special drive circuit, and then the value of the sensor can be read from a shared sensor bus. This procedure means that only one sensor can be measured at a time. This takes on average 50 ms including both the drive time and read time. To efficiently handle multiple sensor read requests from the computer system, the interface card allows multiple requests to be queued and then processed in order. Assuming that on average 10 sensor read requests per second are made, and that the interval between the requests is exponentially distributed, what is the average time it takes for the computer system to get a response on a sensor read request? (4 points)

C) In a program you are running, 45% of the instructions access data in memory. The first level cache configuration of your computer system is such that the hit time is one clock cycle, and the miss penalty is 20 clock cycles. For your program the miss rate for data accesses becomes 10%. Simulations show that the CPI assuming a perfect cache (0% miss rate) is 1.5. What is the best speedup of your program that you can hope for by improving the miss rate for data accesses? (4 points)

ASSIGNMENT 2

Assume a MIPS processor with a simple five stage pipeline of the type presented in appendix A of the course book (Instruction fetch, Instruction decode, Execute, Memory access, Write back). Furthermore, assume that all memory accesses complete in a single cycle (cache hit time is one cycle), that pipeline forwarding is used whenever possible to resolve RAW hazards, and that there is one branch delay slot (all branch computations are completed in the ID stage). Integer multiply operations are performed in two steps; the first in the Execute stage, and the second in the Memory Access stage, so multiply operations are fully pipelined like all other arithmetic instructions, but the result is not available until the end of

the Memory access stage. You need to write a piece of MIPS assembly code to perform the following computation in as few clock cycles as possible.

```
for(int i=0; i<n; i=i+1) {
           x[i] = x[i]*x[i];
}
```

The x array is an array of long integers (8 bytes). You may assume that register R1 already is loaded with the start address of the x array, i.e. the address of x[0], and that you are allowed to change the value of R1. The address of x[n] is likewise available in register R2. You may assume that n always is greater than zero, and use that fact in your code.

A) The following is a first version of the code:

```
LOOP:   LD      R4, 0(R1)
        DMUL    R5, R4, R4
        SD      R5, 0(R1)
        DADDI   R1, R1, 8
        BNE     R1, R2,
                LOOP
        NOP
```

How many cycles does this code take to execute per loop of the original program? Also, specify all types of dependencies and unresolved hazards in this code. (4 points)

B) Modify the code above to require as few clock cycles as possible. How many clock cycles does it take now? Also specify any remaining hazards. (4 points)

C) During an effort to improve the performance of the program, you find out that n always is an even number. Try to modify the code further using this knowledge. How many clock cycles does it take now? Also specify any remaining hazards. (4 points)

ASSIGNMENT 3

A) In a processor with dynamic scheduling according to Tomasulo's algorithm, hardware based speculation using a reorder buffer (ROB), and dynamic branch prediction, execution of each instruction follows a sequence of steps that is somewhat different depending on the type of instruction. A description of the execution of each of four main types of instructions (arithmetic/logic, branch, load, and store) is provided below. However, there are some gaps marked by ['gap number'] in these descriptions. Fill in each of these gaps. Write your answers on a separate page as "gap number: missing text". (8 points)

Arithmetic/Logic Operation Processing

1. Issue when reservation station and ROB entry is available
   - Read already available operands from registers and instruction
   - Send instruction to reservation station
   - Tag unavailable operands with ROB entry
   - Tag destination register with ROB entry
   - Write destination register to ROB entry
   - Mark ROB entry as busy
2. Execute after issue
   - Wait for operand values on CDB (if not already available)
   - Compute result
3. Write result when CDB and ROB available
   - [1]
   - Update ROB entry with result, and mark as ready
   - Free reservation station
4. Commit when at head of ROB and ready
   - Update destination register with result from ROB entry
   - Untag destination register
   - Free ROB entry

Branch Processing

1. Issue when reservation station and ROB entry is available
   - Read already available operands from registers and instruction
   - Tag unavailable operands with ROB entry
   - Write destination address and outcome prediction to ROB entry
   - Mark ROB entry as busy
2. Execute after issue
   - Wait for operand values on CDB (if not already available)
   - Compute result (branch condition)
3. Write result when ROB available
   - Update ROB entry with result, and mark as ready
   - Free reservation station
4. Commit when at head of ROB and ready
   - [2]
   - If result did not agree with prediction
     - [3]
   - Else, free ROB entry

Load Processing

1. Issue when reservation station and ROB entry is available
   - Read already available operands from registers and instruction
   - Tag unavailable operands with ROB entry
   - Tag destination register with ROB entry
   - Write destination register to ROB entry
   - Mark ROB entry as busy
2. Execute step 1 after issue
   - Wait for base address register value on CDB (if not already available)
   - Compute address
3. Execute step 2 when [4]
   - [5]
4. Write result when CDB and ROB available
   - Send result on CDB to reservation stations
   - Update ROB entry with result, and mark as ready
   - Free reservation station
5. Commit when at head of ROB and ready
   - Update destination register with result from ROB entry
   - Untag destination register
   - Free ROB entry

Store Processing

1. Issue when reservation station and ROB entry is available
   - Read already available operands from registers and instruction
   - Tag unavailable operands with ROB entry
   - Mark ROB entry as busy
2. Execute after issue
   - Wait for operand values on CDB (if not already available)
   - Compute address and [6]
3. Write result when CDB and ROB available
   - Update ROB entry with [7], and mark as ready
   - Free reservation station
4. Commit when at head of ROB and ready
   - [8]
   - Free ROB entry

B) What changes and/or additions to the algorithms in the previous question are required to provide support for handling of precise exceptions? (4 points)

ASSIGNMENT 4

A) Describe two cache memory optimization techniques that may improve hit performance (latency and throughput). For each technique, specify how it affects hit time and fetch bandwidth. (4 points)

B) Describe two cache memory optimization techniques that may reduce miss rate, and define the miss type (compulsory, capacity, conflict) that is primarily affected by each technique. (4 points)

C) Describe two cache memory optimization techniques that may reduce miss penalty. (4 points)

ASSIGNMENT 5

Nearly all computer manufacturers offer today multi-core microprocessors. This assignment focuses on concepts central to how thread-level parallelism can be exploited to offer a higher computational performance.

A) The performance of a superscalar processor is limited by the amount of instruction-level parallelism in the program. In particular, when a load instruction must fetch data from memory, it can be difficult to find a sufficient number of independent instructions to execute while the data is being fetched from memory.

Multithreading is a technique to do useful work while waiting for the data to be returned from memory. Explain how the following concepts can keep the processor busy doing useful work:

(i) Fine-grain multithreading
(ii) Coarse-grain multi-threading
(iii) Simultaneous multithreading

(3 points)

B) What structures in a superscalar processor must be replicated to realize a simultaneous multithreaded processor? (2 points)

C) Flynn classifies computer architectures that leverage thread-level parallelism into four categories. Which ones? (4 points)

D) Shared-memory multiprocessors is an important class of architectures that form the basis for multi-core microprocessors. The memory model is such that all processors access the same memory.

(i) What is cache coherence?
(ii) How does an invalidation-based cache coherence protocol work?
(iii) How is the lock primitive in a critical section implemented using test-and-set instructions?

(3 points)

****** GOOD LUCK! ******