Computer Architecture and Engineering

CS152 Quiz #5

April 23rd, 2009

Professor Krste Asanovic

Name: Answer Key

This is a closed book, closed notes exam.

80 Minutes

8 Pages

Notes:

- **Not all questions are of equal difficulty, so look over the entire exam and budget your time carefully.**
- **Please carefully state any assumptions you make.**
- **Please write your name on every page in the quiz.**
- **You must not discuss a quiz's contents with students who have not yet taken the quiz. If you have inadvertently been exposed to the quiz prior to taking it, you must tell the instructor or TA.**
- **You will get no credit for selecting multiple-choice answers without giving explanations.**

Writing name on each sheet	**_____**	**1 Point**
Question 1	**_____**	**20 Points**
Question 2	**_____**	**18 Points**
Question 3	**_____**	**17 Points**
Question 4	**_____**	**24 Points**
TOTAL	**_____**	**80 Points**

NAME:_____

Problem Q5.1: VLIW 20 points

In this problem, you will port code to a simple VLIW machine, and modify it to improve performance. Details about the VLIW machine:

- Three fully pipelined functional units (Integer ALU, Memory, and Floating Point)
- Integer ALU has a 1 cycle latency
- Memory Unit has a 2 cycle latency
- FPU has a 3 cycle latency and can complete one add or one multiply (but not both) per clock cycle
- No interlocks

C Code:

```
for(int i=0; i<N; i++)
  C[i] = A[i]*A[i] + B[i];
```

Assembly Code:

```
loop:  ld   f1, 0(r1)
       ld   f2, 0(r2)
       fmul f1, f1, f1
       fadd f1, f1, f2
       st   f1, 0(r3)
       addi r1, r1, 4
       addi r2, r2, 4
       addi r3, r3, 4
       bne  r3, r4, loop
```

Problem Q5.1.A 7 points

Schedule operations into the VLIW instructions in the following table. Show only one iteration of the loop. Make the code efficient, but do not use software pipelining or loop unrolling. You do not need to write in NOPs (can leave blank).

ALU	Memory Unit	FPU
addi r1, r1, 4	ld f1, 0(r1)	
addi r2, r2, 4	ld f2, 0(r2)	
		fmul f1, f1, f1
		fadd f1, f1, f2
addi r3, r3, 4		
bne r3, r4, loop	st f1, -4(r3)	

What performance did you achieve? (in FLOPS per cycle): 2/9_____

NAME:_____

Problem Q5.1.B **7 points**

Unroll the loop by one iteration (so two iterations of the original loop are performed for every branch in the new assembly code). You only need to worry about the steady-state code in the core of the loop (no epilogue or prologue). Make the code efficient, but do not use software pipelining. You do not need to write in NOPs (can leave blank).

ALU	Memory Unit	FPU
	ld f1, 0(r1)	
addi r1, r1, 8	ld f3, 4(r1)	
	ld f2, 0(r2)	fmul f1, f1, f1
addi r2, r2, 8	ld f4, 4(r2)	fmul f3, f3, f3
		fadd f1, f1, f2
		fadd f3, f3, f4
addi r3, r3, 8	st f1, 0(r3)	
bne r3, r4, loop	st f3, -4(r3)	

What performance is achieved now? (in FLOPS per cycle): **4/10**_____

Problem Q5.1.C **6 points**

With unlimited registers, if the loop was fully optimized (loop unrolling and software pipelining), how many FLOPS per cycle could it achieve? What is the bottleneck?
(Hint: You should not have to write out the assembly code.)

It could achieve 2/3 flops per cycle. It will be bottlenecked by memory accesses, since each iteration has 3 memory ops (2 loads and 1 store) and only 2 floating point ops, and there is only one functional unit for each.

Many people picked the FPU because it has the longest latency. In steady state, it is a matter of throughput rather than latency.

NAME:_____

Problem Q5.2: Vector 18 points

In this problem we will examine how vector architecture implementations could affect the performance of various codes. As a baseline implementation, assume:

- 64 elements per vector register
- 8 lanes
- One ALU per lane; 2 cycle latency
- One load/store unit per lane; 8 cycle latency
- No dead time
- No support for chaining
- Scalar instructions execute on a separate five-stage pipeline

Between two given alternatives, pick the modification that will yield the greatest performance improvement and explain why (assuming everything else is held constant). Be sure to explain why the other choice will not help as much.

Problem Q5.2.A 6 points

```
# Vector Assembly

   LV   V0, R1
   ADDV V1, V1, V2
   MULV V2, V2, V2
   ADDV V3, V3, V4
   ADDV V1, V1, V0
```

Circle one:

- Double number of lanes
- Add support for chaining

There will be no gain from chaining because there aren't any stalls caused by dependencies. The instructions aren't all entirely independent since the last two instructions use previously computed vectors. If you work out the latencies, you will see that V3 and V1 will be ready before they are needed so there will be no stalls.

Doubling the number of lanes will improve performance because 16 lanes is still less than the vector length.

NAME:_____

Problem Q5.2.B **6 points**

```
# C Code
#  Vector reduction from lecture
#  VL is vector length (power of 2)

   do {
      VL = VL/2;
      sum[0:VL-1] += sum[VL:2*VL-1];
   } while (VL>1);
```

Circle one:

- Double number of lanes
- Double vector unit clock frequency

In this vector reduction code, the vector length keeps getting halved. This means for a significant portion of its execution it will be using short vectors. Additional lanes can't help with short vectors.

Doubling the clock rate will still offer the theoretical doubling of throughput, but it will be able to achieve that even with short vectors.

Problem Q5.2.C **6 points**

```
# Vector Assembly

   LV   V0, R1
   MULV V0, V0, V0
   LV   V1, R2
   ADDV V1, V1, V0
   SV   V1, R2
```

Circle one:

- Double number of lanes
- Add support for chaining

Many of these instructions are dependent, so even with more lanes, the system will need to stall for dependencies. Chaining will allow for the biggest performance improvement.

NAME:_____

Problem Q5.3: Multithreading 17 points

Consider the following code on a multithreaded architecture. You can assume each thread is running the same program. You can assume:

- Single-issue and in-order machine pipeline
- ALU is fully pipelined with a latency of 2
- Branches (conditional and unconditional) take 2 cycles to complete (if branch is started on cycle 1, the following instruction can't start until cycle 3)
- Memory Unit is fully pipelined with a latency of 16

Code:

```
loop:   beq r1, r0, end      1     36
        lw  r2, 4(r1)        3
        add r3, r3, r2       19
        lw  r1, 0(r1)        20
        j   loop             21
end:
```

Problem Q5.3.A 5 points

How many cycles does it take for one iteration of the loop to complete if the system is single threaded?

36 – 1 = 35 cycles
The jump executes while the last lw is in progress, but the next iteration can't start until the load is done.

Problem Q5.3.B 6 points

If the system is multithreaded with fixed round-robin scheduling, how many threads are needed to fully utilize the pipeline?

It needs to cover 15 cycles of latency (between lw and add), so 15 + 1 = 16 threads will be needed.

Problem Q5.3.C 6 points

What is the minimum number of threads needed for peak performance if the system is changed to data-dependent thread scheduling (i.e., can pick next instruction from any thread that is ready to execute) and you are allowed to re-schedule instructions in the assembly code? Explain.

If the second load is moved up to below the first, latency of iteration becomes 21 cycles (5 issues + 16 stall cycles). The ceiling of (21/5) = 5 so at least five threads are needed. Four threads is almost just enough. If you assume no thread switches on branches, then 3 threads should be sufficient.

NAME:_____

Problem Q5.4: Memory Latency 24 points

In this problem, we will compare how the three architectures studied (VLIW, vector, and multithreaded) tolerate memory latency, both in their hardware design and in how they are programmed. To make the comparison fair, each system has only one floating-point unit (FPU). For the rest of the problem, please consider the following program (vector add):

```
for (int i=0; i<n; i++)
  C[i] = A[i] + B[i];
```

Problem Q5.4.A 15 points

Assume the latency to memory is always 100 cycles (no cache). How do each of the architectures tolerate the memory latency? Be sure to describe what programming techniques and what hardware mechanisms help for each machine type.

i) VLIW

The impact of memory latency is reduced by getting more iterations of the loop in flight at the same time. With a VLIW system, this must be explicitly managed by the programmer (or compiler). Software pipelining allows iterations to be overlapped, while loop unrolling allows iterations to be combined. A rotating register file can be used to simplify and shorten the code.

ii) Vector

A vector machine hides the latency of instructions by overlapping their execution. With sufficiently long vectors, the memory latency can be amortized over many elements. With chaining, the multiple vector instructions can be overlapped, further hiding the latency. Multiple lanes doesn't help to hide the latency, and for this problem it was stated there was only one FPU. To use a vector machine, the programmer must stripmine the code.

iii) Multithreaded

This system hides the latency by running other threads while waiting. The programmer must partition the program into multiple threads.

NAME:_____

Problem Q5.4.B **9 points**

What if the memory latency increased from 100 (Problem Q5.4.A) to 1000 cycles? For each architecture, what changes are needed to hardware and software to continue to hide the latency?

i) VLIW

To hide a longer latency, more iterations will need to be in flight at the same time which means the code will need to have its loops unrolled more times and be software pipelined to a greater degree. This will require explicit and significant changes to the code by the programmer. It will also require many more physical registers to support this.

ii) Vector

To best hide the latency, longer vectors will be needed. To utilize longer vectors will require negligible changes to the software.

iii) Multithreaded

To hide more latency will require more threads, and these will need to be created by the programmer. Each additional thread will require more registers to holds its architectural state.