EE 565 Computer Architecture

Midterm-Grad version

Tuesday, October 27, 1998

Name: ____

Limit your answers to the space provided. Unnecessarily long answers will be penalized. If you use more space than is provided, you are probably doing something wrong. Use the back of the page for any scratch work. Write your last name on each page.

Some problems may take longer than others. Plan your time accordingly.

Please make sure that you have all the 11 pages of the exam.

Problem 1 ____________________ (out of 26 points)

Problem 2 ____________________ (out of 12 points)

Problem 3 ____________________ (out of 15 points)

Problem 4 ____________________ (out of 24 points)

Problem 5 ____________________ (out of 15 points)

Problem 6 ____________________ (out of 8 points)

Total ____________________ (out of 100 points)

1 Pipelining (26 points)

1.1 (5 points)

Identify all the data hazards in the following code by drawing arrows and naming the arrows with the hazard:

r1 := r2 + r3

r2 := r1 - r4

r1 := r3 + r4

1.2 (5 points)

Assume that in the 5-stage DLX pipeline, register file writes are too slow to fit a write in the first half of the cycle and a read in the next half. What problem will this cause? How can we fix it?

1.3 (5 points)

Assume that in the 5-stage DLX pipeline, there is hardware hazard detection to insert a 1-cycle stall for an instruction that immediately follows a load and uses the load value. Schedule the following instructions to avoid any such stalls as much as possible.

```
# the compiler knows that r28 = 0xffff0000

# the compiler knows that r29 = 0x10000000

# the compiler does not know the value of r30

lw r2, 4[r28]

lw r3, 0[r28]

add r1, r2, r3

sw 4[r29], r1

lw r5, 8[r28]

lw r6, 4[r30]

add r4, r5, r6

sw 8[r29], r4
```

1.4 (5 points)

Assume that in a modified DLX pipeline, not-taken branches do not cause any pipeline bubbles. The delay-slot (single cycle) of the branch is filled by instructions from only the taken path and will not be executed if the branch is not taken. 20% of all instructions are branches, 85% of which are taken.What is the overall CPI, if 60% of the delay slots are filled and the CPI, assuming perfect branches, is 1?

1.5 (6 points)

Answer yes/no: Can the 5-stage DLX pipeline machine guarantee precise interrupts if

(a) it allows load instructions before an excepting load instruction to write back?

(b) it allows an add instruction after an excepting load instruction to write back?

(c) it forces a divide-by-zero trap in the IF stage while the divide instruction is in the EX stage?

2 Performance calculations (12 points)

2.1 Hardware Alignment (6 points)

Modern microprocessors usually use a barrel shifter to perform alignment in hardware. If you use a barrel shifter, the critical path of the clock will worsen by 10%. If you don’t use a barrel shifter, then unaligned accesses will require an extra instruction, which will not cause any extra pipeline stalls as compared to the design with the shifter. Assuming that 10% of all loads and stores access unaligned locations and every fourth instruction is a load or a store, will you employ hardware alignment?

2.2 Directed optimizations (6 points)

Machine A is 20% faster than machine B for a specific SPEC95 benchmark. Machines A spends 40% of its execution time on FP multiplications in some benchmark. Machine B spends 60% of its execution time on FP multiplications. Through an optimization of the FP multiplier, the FP multiplications in the benchmark can be sped up on machines A and B by 15% and 25%, respectively. After the optimization, which machine is faster and by how much?

3 Short Questions (15 points)

3.1 (3 points)

What would you use to average the following:

Throughput (e.g., instructions/clock)

Speedups (i.e., old time/new time)

Execution times (e.g., in seconds)

3.2 (4 points)

Today’s RISC processors use separate explicit store and load instructions to save and restore registers on function calls and returns, instead of requiring the call and return instructions to implicitly perform saves and restores. Why?

3.3 (4 points)

Many ISA multimedia extensions (e.g., MMX) by various vendors allow adds of 8-bit and 32-bitd quantities but disallow adds of 16-bit quantities. What ISA principle do they violate?

3.4 (4 points)

Branch instructions in some ISAs have 16-bit offset field. Does this mean that programs that need to branch past 64Kbytes (2^{16}) cannot be compiled in these ISAs? Explain.

4 Dynamic Scheduling (24 points)

Consider the following code:

ADD r0, r2, r3

MULT r1, r2, r2

SUB r0, r0, r1

ADD r0, r3, r3

4.1 (12 points)

The above instruction sequence is issued to reservation stations with Tomasulo's algorithm. After all the instructions are issued, fill in the values for the reservation stations and the register result status table shown below. Assume that no instructions execute during the issue phase, and that all the reservations stations and registers are not busy at the beginning.

Reservation Stations							
FU	**Name**	**Busy**	**Op**	$\mathbf{V_j}$	$\mathbf{V_k}$	$\mathbf{Q_j}$	$\mathbf{Q_k}$
1	Add1						
2	Add2						
3	Add3						
4	Mult1						
5	Mult2						

Register Result Status			
Reg	**Busy**	$\mathbf{Q_i}$	**value**
r0			3
r1			2
r2			4
r3			5

4.2 (12 points)

Using the same code, repeat with register renaming using extra physical registers: the initial physical registers that are free are: P4, P5, P6, P7. Initial register map contents are shown. Fill in the final contents.

Reservation Stations						
FU	**Name**	**Busy**	**Op**	**P_i**	**P_j**	**P_k**
1	Add1					
2	Add2					
3	Add3					
4	Mult1					
5	Mult2					

Register Rename Table		
Reg	**Initial**	**Final**
r0	P0	
r1	P1	
r2	P2	
r3	P3	

5 Branch Prediction (15 points)

Consider the following code. The loop body contains no conditional branches.

```
while  (TRUE)  {                     /* infinite loop */

  for (i=0; i < 4; i++) {            /* loop four times */

    loop body

  }

}
```

we want to predict the conditional branch that terminates the inner loop. For the following branch schemes, what is the average prediction accuracy (#correct/#total)?

5.1 (5 points)

A prediction table with a single history bit accessed via a truncated program counter.

5.2 (5 points)

A prediction table with two-bit counters accessed via a truncated program counter.

5.3 (5 points)

A prediction table with a single history bit accessed via a 3-bit global branch history register.

Name: ______________________________________

6 Software Pipelining (8 points)

Software pipeline the following loop to maximize performance:

```
for (i = N; i > 0 ; i--)
   A[i] = A[i] + incr;
```

```
LOOP:
   LD        f0, A(r1)
   ADDD      f4, f0, f2          # f2 holds incr
   SD        A(r1), f4
   SUBI      r1, r1, 8           # r1 holds i
   BNEZ      r1, LOOP
```