Midterm Exam

October 13, 2014

Name: _____

Problem	1	2	3	4	**total**
Points					

Exam rules:

- Time: 90 minutes.
- Individual test: *No team work!*
- Open book, open notes.
- No electronic devices, except an *unprogrammed* calculator.
- *No phone, no internet!*

Good luck and have fun!

1. *Basics* (10 Pts.)

   Note: Answers should be no longer than *two sentences*.

   (a) Assume the 5 least significant bits of the branch address are used to form the branch prediction buffer, and the branch predictor has a 2-bit counter, what is the total size of the branch predictor?

   (b) Assuming infinitely many reservation stations, what is the best performance achievable when following Tomasulo's algorithm?

   (c) Assume a 4kB cache with a block size of 1kB. What values can N take for making that cache N-way set associative? How is the cache usually classified if N takes the maximum possible value?

   (d) Consider the case of a processor with an instruction length of 16 bits and with 64 general purpose registers, so the size of the address fields is 6 bits. How many two-address instructions can be encoded?

   (e) A two-level cache architecture features a hit time of 2 cycles for L1 cache, 8 cycles for L2 cache, and a miss penalty of 50 cycles for L2 cache misses. If on 1000 memory accesses there are 60 L1 misses and 30 L2 misses, what is the average *miss penalty* for L1 cache?

Additional space:

2. Multiple pipeline with hazards and forwarding

   Consider a single-issue, 5-stage pipeline design. For all instructions, fetching (F), decoding (D), memory (M), and write-back (W) take 1 cycle each. The execution time is 3 cycles for multiplication (E_MUL), 6 cycles for division (E_DIV), and 1 cycle for any other executions (E). The processor has an integer ALU, a multiplier and a divider. Furthermore, the memory (M) and writeback (W) stages (and only those) can accomodate multiple instructions without a structural hazard, as long as only one instructions needs memory access or access to registers, respectively. If more than one instruction requires access to memory or registers, preference is given to instructions in issuing order (while the other(s) stall(s)).

   (a) Show the forwarding and stalls for the following MIPS code for one loop. Indicate stalls by (S) and forwarding by clearly visible arrows between the appropriate states.

   (b) After heavy optimization, multiplication and division are merged into one unit that completes execution in 2 cycles. The processor now only has an integer ALU and one combined MUL/DIV engine.
   Show the forwarding and stalls for the following MIPS code for one loop. Compare the result to the previous engine. Assuming the second engine to be clocked 10% faster, express the performance gain for the given loop. Assume the execution time of the loop to end after the fetch of the last instruction.

```
Loop:  LD    F2, 0(R1)
       DIV   F8, F2, F0
       MULT  F2, F6, F2
       LD    F4, 0(R2)
       ADD   F4, F0, F4
       ADD   F10, F8, F2
       ADDI  R1, R1, 8
       ADDI  R2, R2, 8
       SD    F4, 0(R2)
       SUB   R20, R4, R1
       BNZ   R20, Loop
```

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
LD F2, 0(R1)																			
DIV F8, F2, F0																			
MULT F2, F6, F2																			
LD F4, 0(R2)																			
ADD F4, F0, F4																			
ADD F10, F8, F2																			
ADDI R1, R1, 8																			
ADDI R2, R2, 8																			
SD F4, 0(R2)																			
SUB R20, R4, R1																			
BNZ R20, Loop																			

Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
LD F2, 0(R1)																			
DIV F8, F2, F0																			
MULT F2, F6, F2																			
LD F4, 0(R2)																			
ADD F4, F0, F4																			
ADD F10, F8, F2																			
ADDI R1, R1, 8																			
ADDI R2, R2, 8																			
SD F4, 0(R2)																			
SUB R20, R4, R1																			
BNZ R20, Loop																			

3. Single Issue vs. Multiple Issue Processor

Consider the following code and latencies for this problem. We ignore instruction fetch and decode as well as write backs in this problem. All instructions execute in one cycle, plus the latency given in the below table.

```
Loop:  LD      F6, 0(R2)
       LD      F2, 8(R2)
       MULTD   F8, F2, F6
       LD      F4, 8(R2)
       ADD     F10, F0, F6
       ADD     F6, F8, F4
       ADDI    R2, R2, -16
       BNE     R2, R3, Loop
```

Latencies	
Memory LD	+3
Branch	+1
ADD,ADDI	+0
MULTD	+5

(a) What is the baseline performance of the code? Assume that one instruction per cycle can be issued and that the following instruction only executes after the previous has produced a result.

(b) Now we assume the instruction execution to be non-blocking. The execution is only delayed by true data dependencies. Rewrite the code (without reordering is) indicating when which instruction is starting execution and indicate stalls, i.e. cycles where no new instruction is starts execution. How many cycles does one execution of the code take now?

(c) Now consider a two-issue design. Suppose there are two execution pipelines, each capable of beginning execution of one instruction per cycle. Results are immediately forwarded between the pipelines. The pipelines only stall for true data dependences. Note that instructions may only be issued once all inputs are available and instructions must be issued in order. Write down the execution order per cycle as before. How many cycles does one execution of the loop take now?

Additional space:

	Problem A	Problem B	Problem C	
Cycle	Pipeline	Pipeline	Pipeline A	Pipeline B
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				

4. *Scoreboard Algorithm* (10 Pts.)

For this problem use the single-issue Scoreboard MIPS pipeline with the number of functional units and reservation stations listed in the table below

FU type	Execution Time	Number of FUs
Integer (LD, SD, ADDI)	1	3
FP adder	3	2
FP multiplier	5	2

When applying the Scoreboard algorithm, consider the following arbitration rules:

- *Only one instruction can be issued per clock cycle*.
- Functional units are not pipelined.
- The execution stage (EX) does both the effective address calculation and the memory access for loads and stores. Thus the pipeline is IS / RO /EX / WB.
- All stages (IS, RO, WB) except EX take 1 cycle to complete.
- There is no forwarding between functional units. Both integer and floating point results are communicated through the registers.
- Memory accesses use the integer functional unit to perform effective address calculation. All loads and stores access memory during the EX stage.
- If an instruction A has read-after-write dependency on instruction B, instruction A can read operands (RO) only AFTER the WB of instruction B is complete.
- Whenever there is a conflict for a functional unit, assume program order.
- When an instruction is done executing in its functional unit and is waiting for writing the result, it is still occupying the functional unit (meaning no other instruction may enter). However, in the clock cycle it writes the result, a new instruction can be issued to the FU.

(a) Using the Scoreboard approach, fill in the table below to indicate the cycle number at each stage for all instructions. Indicate the reason to stall by hazard type and the causing register or FU. Also indicate which FU the instruction is issued to.

(b) Indicate the state of the scoreboard and register file after instruction 7.

Instruction	Issue	Read Ops	Execute Complete	Write Result	Assigned FU	Reason to Stall
LD F0, 0(R0)						
MUL.D F1, F0, F1						
ADD.D F0, F0, F2						
SD F1, 0(R1)						
LD F2, 4(R0)						
MUL.D F3, F2, F1						
ADD.D F4, F1, F2						
SD F3, 4(R3)						
ADDI R3, R3, 1						
ADD.D F0, F4, F1						

The scoreboard and register state table should contain the state after cycle 7.

Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
Integer 1									
Integer 2									
Integer 3									
FP Add 1									
FP Add 2									
FP Mul 1									
FP Mul 2									

Name	FU	Name	FU
F0		R0	
F1		R1	
F2		R2	
F3		R3	
F4		R4	

Additional space:

Additional space: