Last (family) name: _____

First (given) name: _____

Student I.D. #: _____

Department of Electrical and Computer Engineering
University of Wisconsin - Madison

ECE/CS 552 Introduction to Computer Architecture

Midterm Exam

Friday, October 29, 2010, 2:25 – 3:15 PM

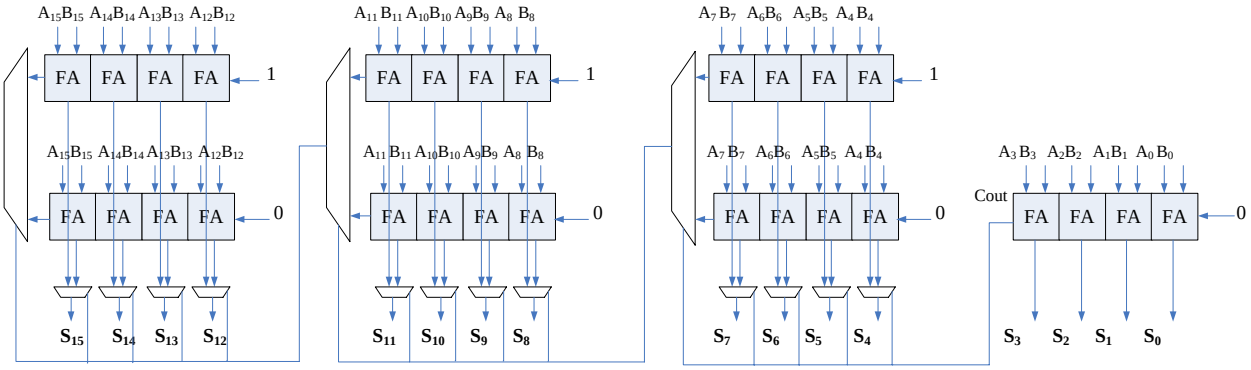
Instructions:

1. Close book/close notes examination (1 double-side hand-written "cheat sheet" is allowed).
2. Calculator is allowed. No hand-held computers or portable computers allowed.
3. **You must show your complete work**. Points will be awarded only based on what appears in your answer book.
4. No one should leave the room during the last 5 minutes of the examination.
5. Upon announcement of the end of the exam, stop writing on the exam paper immediately. Pass the exam to aisles to be picked up by the proctors. The instructor will announce when to leave the room.
6. Failure to follow instructions may result in forfeiture of your exam and will be handled according to UWS 14 Academic misconduct procedures.

Problem	*Type*	*Points*	*Score*
1	Arithmetic unit design	3	
2	Delay analysis	2	
3	Number formats	6	
4	Pipeline analysis	4	
5	Cycle time analysis	2	
6	CPI analysis	4	
7	Forwarding paths	2	
8	Delay slot scheduling	6	
9	Pipeline design	5	
10	Branch prediction	6	
Total		40	

Context: You are hired by 552 Inc., a world-wide computer hardware vendor, and you are the principle architecture engineer working on a recent project – MID-F10 Computer. MID-F10 has a 5-stage pipeline architecture: IF, ID, EX, MEM, WB, very similar to the MIPS architecture in the textbook. The ISA of MID-F10 is almost the same as the MIPS ISA, except that the data width of MID-F10 is 16 bits. Read each question carefully and show your work along with your answer, stating any assumptions you have made clearly.

1. (3 pts.) Your company has designed a 16-bit carry-select adder (CSA) in the ALU. The CSA consists of seven 4-bit Ripple Carry Adders. S[3:0] is calculated once using one ripple carry adder. S[15:4] is calculated twice, one time with the assumption of the carry being zero and the other assuming one. Then, the right answer will be selected via the actual carry-in.
Figure 1 shows the partial block diagram of CSA. FA means 1-bit full adder. All the muxes here are 2-to-1 muxes. Connect the select signal for each 2-to-1 muxes to complete the CSA diagram correctly.



2. (2 pts.) Assuming the delay for each Full Adder is 10 ns and the delay for each mux is 3 ns. What is the delay for this Carry-Select Adder? Show your work.

Four FA delays plus 3 2-to-1 mux delays.

4*10+3*3 = 49 ns

3. (6 pts.) MID-F10 uses half-precision floating-point format in its floating point registers. The IEEE 754 standard specifies a half-precision floating-point format as having:
a. Sign bit: 1 bit
b. Exponent width: 5 bits. The exponent bias is 15 (0x0f).
c. Significand precision: 11 (10 explicitly stored, unsigned, with implicit leading 1)

Given this format, fill in the missing values in the table (convert to/from a hexadecimal 16-bit value from/to half-precision floating point expressed in decimal). Show your work.

Hex. value	Half-prec Floating-Point
0F00	0.00042724609
C480	-4.5
5640	100_{10}

0x0F00 = 0b 0000 1111 0000 0000
sign: 0
exp: 3 – 15 = -12
fraction = (1.)11 0000 0000
Therefore the floating point value is
(-1)^0 * (1+0.5+0.25)*2^(-12) = 0.000427246094

0xC480 = 0b 1100 0100 1000 0000
sign: 1
exp: 17 – 15 = 2
fraction = (1.)00 1000 0000
Therefore the floating point value is
(-1)^1 * (1+0.125)*2^(2) = -4.5

100 = 64+32+4 = (-1)^0 * (1+0.5+0.0625)*2^(21-15)

Therefore the hex. value is:
0 10101 1001000000 = 0x5640

A careless engineer in your team lost some important information about the design: the cycle time and the information on forwarding paths . While being mad at him, you have to figure out this information by running some instruction sequences on the design and recording the WB completion times for these instructions.

You know that the register file is designed with latches, and is read in the second half of the ID stage and written in the first half of the WB stage (it is a write-before-read RF design).

Here are the results, assuming each memory reference is satisfied in one clock cycle in the MEM stage.

Program instruction sequence	WB timestamp	Elapsed execution time	Time per instruction	Cycles per instruction
<preceding inst> addi $2, $5, 10 sra $2, $3, 2	500 800	 300 ns	150 ns	1

<preceding inst>
lw $1, 0($6)
lw $2, 10($6)
add $8, $1, $2
sw $8, 0($7)
	1000
	
	1900
		
		900 ns
			225 ns	1.5

<preceding inst> and $2, $5, $6 sll $2, $2, 1 sw $2, 10($3)	2500 2950	 450 ns	150 ns	1
Average TPI and CPI			183.3 ns	1.22

4. (4 pts) Fill in the time per instruction column in the table, including the average TPI. Show your work.

5. (2 pts) What is the cycle time of MID-F10? Show your work.

150 ns. We know the CPI = TPI for the first instruction sequence.

6. (4 pts) Fill in the cycles per instruction (CPI) column in the table, including the average CPI. Show your work.

7. (2 pts) What type of the forwarding path does MID-F10 have (Full forwarding, ALU forwarding only, or no forwarding at all)? Justify your answer.

ALU forwarding only. From the CPI for second instruction sequence we know there is no MEM to EX forwarding; from the third sequence we know there is ALU forwarding.

8. (6 pts) Suppose you have the following instruction sequence to be executed:

```
lw      $1, 0($7)
addi    $1, $1, 1
sw      $10, 10($7)
lw      $2, 0($8)
addi    $2, $2, 1
sw      $20, 10($8)
```

Rearrange the instruction sequence so that it achieves the **same functionality** but **best performance (shortest execution time)** on MID-F10. You are only allowed to change the order of the six instructions. Do not modify or add new instructions. Calculate the execution time of the instruction sequence you rearranged.

Rearranged instruction sequence:

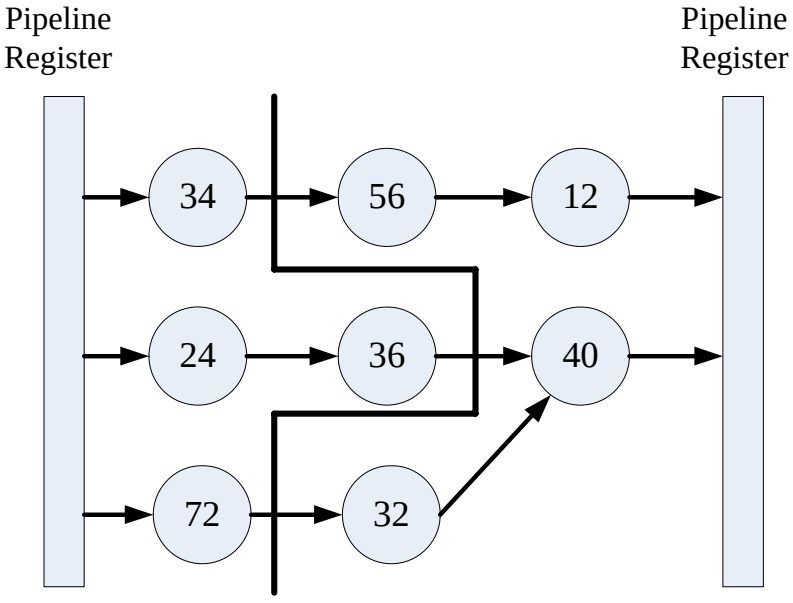
```
lw      $1, 0($7)
lw      $2, 0($8)
sw      $10, 10($7)
sw      $20, 10($8)
addi    $1, $1, 1
addi    $2, $2, 1
```

(as long as lw and the dependent addi instruction are separated by at least 2 other instructions)

Total execution time: _____6_____ cycles _____900_____ ns.

9. (5 pts.) Right now all the operations in their execution stage, including the half-precision floating point, finish in one clock cycle. As a result, the delay in execution stage becomes dominant in your pipeline (longer than any other stages). You plan to split the FP execution stage so that the execution takes more than one cycle, and the cycle time of the system can be smaller. In the following figure, each circle represents a combinational logic block that cannot be split into smaller units. The number in the circle is the latency of this combinational logic (in ns). The arrows show connections between logic blocks. Insert the smallest number of pipeline registers to maximize the throughput of the computer and the floating point unit. Assume that the delays of other stages are 70 ns or less.

Draw a vertical line between combinational logic blocks to indicate a pipeline register insertion.



Original critical path (ns): 72+32+40 = 144 ns

New pipelined critical path (ns): 72 ns

10. (6 pts.) MID-F10 computes branch target addresses in Instruction Decode stage (ID), and resolves branch conditions in the Execution stage (EX). This computer uses static branch condition prediction and does not predict branch target addresses. Given the following instruction sequence:

```
            addi    $1, $0, 100
LOOP:
            add     $5, $5, $6
            subu    $1, $1, 1           # $1 = $1 - 1
            bne     $1, $0, LOOP
END:
            sw      $5, 0($8)
```

Compute the total execution time for this instruction sequence when the branch is predicted as always taken, always not-taken, and backward-taken/forward-not-taken. Show your work.

a. Always-Taken:

1+99*(1+1+2) + (1+1+3) +1 = 403

b. Always-Not-Taken:

1+99*(1+1+3) + (1+1+1) +1 = 500

c. Backward-taken/forward-not-taken:

Same as Always-Taken