

Problem 1 (3 parts, 25 points)

Scoreboarding

Suppose we are using scoreboarding to dynamically schedule code. We have six functional units with the following execution stage latencies.

Floating point unit:	Latency (in cycles):
1: ALU integer	1
2: FP add/subtract	5
3: FP add/subtract	5
4: FP multiplication	10
5: FP multiplication	10
6: FP division	25

Consider the following program.

Instruction
I7: MULTF F2, F1, F3
I8: ADDF F4, F2, F5
I9: SUBF F1, F3, F6

Suppose the operands of the MULTF instruction (I7) have been read and the multiplication operation is currently in the process of being executed by functional unit 4. The processor has just finished executing the FP subtraction of the SUBF instruction (I9) and is ready to write its result to the register file.

1a) [10 points] **Show the current status of the scoreboard at this point in time.**

1. Instruction Status Queue

Instruction	Issued?	Operands Read?	Execution Complete?	Results Written?
I7: MULTF F2, F1, F3	<b>X</b>	<b>X</b>		
I8: ADDF F4, F2, F5	<b>X</b>			
I9: SUBF F1, F3, F6	<b>X</b>	<b>X</b>	<b>X</b>	

2. Functional Unit Status:

FU#	Busy?	Op	Dest	Source1	Source2	Unit1	Unit2	Rdy1?	Rdy2?
1(alu)	<b>N</b>								
2(+/-1)	<b>Y</b>	<b>+</b>	<b>F4</b>	<b>F2</b>	<b>F5</b>	<b>4</b>	<b>--</b>	<b>N</b>	<b>Y</b>
3(+/-2)	<b>Y</b>	<b>-</b>	<b>F1</b>	<b>F3</b>	<b>F6</b>	<b>--</b>	<b>--</b>	<b>*</b>	<b>*</b>
4(*1)	<b>Y</b>	<b>*</b>	<b>F2</b>	<b>F1</b>	<b>F3</b>	<b>--</b>	<b>--</b>	<b>*</b>	<b>*</b>
5(*2)	<b>N</b>								
6(/)	<b>N</b>								

3. Register Result Status:

Reg	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	...	F31
FU#		<b>3</b>	<b>4</b>		<b>2</b>								

1b) [8 points] Give an example of a DLX instruction that follows I9 in the program but that cannot issue in the current cycle. **Write the example DLX instruction in the I10 slot below.**

Instruction
I7: MULTF F2, F1, F3
I8: ADDF F4, F2, F5
I9: SUBF F1, F3, F6
<b>I10: ADDF F4, F7, F8</b>

**Explain why I10 cannot issue.**

**If I10 is any ADDF or SUBF instruction, it cannot issue because there is a structural hazard (no FP adders are available to execute the instruction).**

**If I10 has either F1, F2, or F4 as its destination register, it will have an output dependence with one of the previous instructions still pending, so it cannot issue.**

1c) [7 points] Can I9 (SUBF F1, F3, F6) proceed to write its results in the register file in the next clock cycle? **Yes**

**There is a potential anti-dependence hazard between I7 and I9, but I7 has already read its operands, so it is safe for I9 to write its results. So no stall occurs.**

Problem 2 (3 parts, 25 points)

Performance

Consider a basic 5-stage DLX pipelined machine (M1) which employs data forwarding and dynamic branch prediction. The dynamic instruction mix for this program is as follows.

Instruction Type	Dynamic Frequency
Loads	15%
Stores	10%
Integer ALU	51%
Branches	24%

2a) [10 points] Suppose the following:

- ? *The branch prediction accuracy is 95%.*
- ? *Branch mispredictions each cause a 3 cycle branch penalty.*
- ? 40% of all branches are taken.
- ? 60% of all load instructions are immediately preceded by an R-type instruction that computes the base register value used in calculating the effective address for the memory read (e.g., `ADD R1, R2, R3`  
`LW R4, 0(R1)`).
- ? *32% of all load instructions are followed immediately by an instruction that uses the value just loaded in.*
- ? 12% of all store instructions are immediately preceded by an R-type instruction that computes the value that is written to memory.

What is the throughput (in instructions per cycle) while executing this program? Assume all instructions go through all five stages and there is no additional pipelining overhead, except stalls.

**The only sources of stalls are from the italicized statements above (branch mispredictions and loads followed by use):**

$$s = (0.05)(0.24)(3) + (0.32)(0.15)(1)$$

$$IPC = 1/1+s$$

**Throughput of M1: 0.9225 instructions per cycle.**

2b) [5 points] If the clock rate of M1 is 750 MHz, what is the throughput (in instructions per second) of M1?

$$\text{IPS} = 750 \text{ MHz} * \text{IPC}_{\text{M1}}$$

$$\text{IPS} = 750\text{M cycles/sec} * 0.9225 \text{ I/Cycle} = 692 \text{ M I/sec}$$

**Throughput of M1: 692 instructions per second.**

2c) Suppose we run the same program on the newly released 2GHz (2 billion hertz) Pentium 4 (let's call it P4) and suppose we achieve a throughput of 0.59 IPC.

2c.1) [4 points] What is the throughput of P4 in IPS?

$$\text{IPS} = 2 \text{ G cycles/sec} * 0.59 \text{ I/cycle} = 1.18 \text{ G I/sec}$$

**Throughput of P4: 1.18 billion instructions per second.**

2c.2) [6 points] Compare the two machines in terms of execution time. Which is faster for this application and by what percent?

$$1.18 \text{ G}/692\text{M} = 1.70$$

**Machine P4 is 70 % faster than Machine M1 .**

## Problem 3 (3 parts, 25 points)

## Dynamic Branch Prediction

Consider a processor that is using bimodal dynamic branch prediction. It is executing the following DLX code fragment.

Instruction Address	Label	Instruction
1000		BEQZ R1, L1
1004		ADD R4, R1, R2
1008		BNEZ R2, L2
1012		SUB R4, R3, R2
1016	L2:	BEQZ R3, END
1020		ADD R4, R4, R1
1024		BEQZ R0, END
1028		SUB R4, R4, R2
1032	L1:	BEQZ R2, END
1036		SUB R4, R2, R2
1040	END:	ADD R4, R1, R3

Initially, the registers have these values:

R0: 0	R1: 1	R2: 2	R3: 3	R4: 0
-------	-------	-------	-------	-------

Assume a branch history table of 2-bit saturating counters is keeping track of branch history. This is the current status of the branch target address cache and the branch history table:

**BRANCH TARGET ADDRESS CACHE (BTAC):**

Branch Instruction Address	Branch Target Address
1000	1032
1024	1040
1032	1040

**BRANCH HISTORY TABLE (BHT) using 2-bit saturating counters:**

Branch Instruction Address	Prediction Bits
1000	WP
1008	SNP
1016	WNP
1024	SP
1032	WP

3a) [10 points] **Which branches (if any) are mispredicted when this code fragment is executed?** (If none are mispredicted, write “None.”)

**1000 and 1008. (1032 is not reached.)**

3b) [10 points] What is the status of the BTAC and BHT after this fragment executes?

**BRANCH TARGET ADDRESS CACHE (BTAC):**

Branch Instruction Address	Branch Target Address
<b>1024</b>	<b>1040</b>
<b>1032</b>	<b>1040</b>

**BRANCH HISTORY TABLE (BHT) using 2-bit saturating counters:**

Branch Instruction Address	Prediction Bits
<b>1000</b>	<b>WNP</b>
<b>1008</b>	<b>WNP</b>
<b>1016</b>	<b>SNP</b>
<b>1024</b>	<b>SP</b>
<b>1032</b>	<b>WP</b>

3c) [5 points] What is the branch prediction accuracy for this code fragment?

**4 branches are executed. 2 of them are predicted correctly.**

50 % branch prediction accuracy.

Problem 4 (3 parts, 25 points)

The Hazards of Multi-cycle Functional Units

Consider the following program fragment executing on a basic 5-stage DLX pipeline *with no data forwarding*. All stages take 1 cycle, except the Execute stage, which takes a variable number of cycles, depending on the functional unit used:

Functional unit	Number of EX cycles
Integer ALU	1
Floating Point Add	6
Floating Point Load/Store	2
Floating Point Multiply	3

Assume there are no instructions previous to instruction 1 and that there are 2 of each type of floating point functional unit. Assume registers are written in the first half of the clock cycle and read in the second half.

4a) [10 points] Suppose the instructions enter the pipeline in order, with a new instruction starting on each cycle. (That is, assume there is no hazard detection mechanism being used and no stalls are introduced to avoid hazards.) Determine which *data hazards* occur in executing this program fragment. Indicate the hazards as in the following example: if there is a WAR (anti-dependence) hazard between instructions 3 and 4, involving register F8, and 3 precedes 4, put “3(F8)” in the WAR column to the right of instruction 4. An instruction may cause more than one hazard. Show work on timing diagram below.

Instruction	RAW (true)	WAR (anti)	WAW (output)
1: MULTF F1, F2, F3	N/A	N/A	N/A
2: ADD R1, R1, R2	--	--	--
3: LF F5, 0(R1)	<b>2(R1)</b>	--	--
4: SUBF F5, F6, F1	<b>1(F1)</b>	--	--
5: SF 0(R1), F5	<b>4(F5)</b>	--	--
6: MULTF F5, F3, F4	--	--	<b>4(F5)</b>

Cycle \ Inst	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
(1)	F	D	E	E	E	M	W																							
(2)		F	D	E	M	W																								
(3)			F	D	E	E	M	W																						
(4)				F	D	E	E	E	E	E	E	M	W																	
(5)					F	D	E	E	M	W																				
(6)						F	D	E	E	E	M	W																		

4b) [10 points] Suppose the following instructions are executed in order, instruction (1) is fetched on clock cycle 1 and suppose a hazard detection mechanism inserts bubbles (stall cycles) into the pipeline to avoid data and control hazards. For each instruction, determine in which clock cycle the instruction is safely completed without hazards (i.e., when does its write back stage occur). Assume no data forwarding occurs. Show timing in the table below for maximum credit.

Instruction	Write Back Cycle
1: MULTF F1, F2, F3	<b>7</b>
2: ADD R1, R1, R2	<b>6</b>
3: LF F5, 0(R1)	<b>10</b>
4: SUBF F5, F6, F1	<b>15</b>
5: SF 0(R1), F5	<b>19</b>
6: MULTF F5, F3, F4	<b>21</b>

Cycle Inst	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
(1)	F	D	E	E	E	M	W																							
(2)		F	D	E	M	W																								
(3)			F	S	S	D	E	E	M	W																				
(4)						F	D	E	E	E	E	E	M	W																
(5)							F	S	S	S	S	S	S	S	D	E	E	M	W											
(6)															F	D	E	E	E	M	W									

4c) [5 points] What is the throughput (in instructions per cycle) of the DLX pipeline while executing these instructions? (show work)

**6 instructions are completed (written back) from cycle 6 to cycle 21 (16 cycles).**

---


$$\underline{\underline{6/16 = 0.375}} \quad \text{IPC}$$