Adam Ziel & Matt Bennett

ELEC 3275 - Computer Architecture
Assignment 11 - Putting It All Together

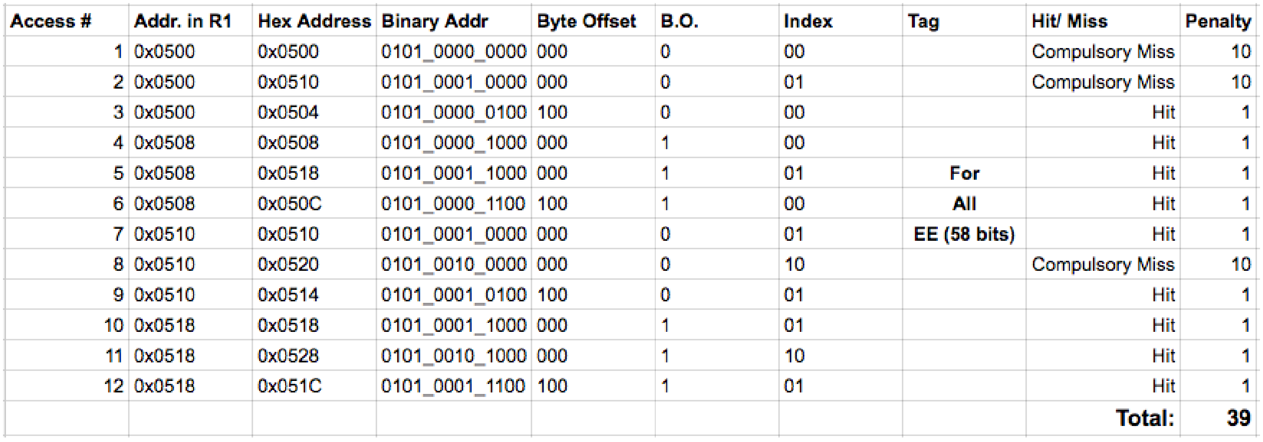
All screenshots in this document are available in this aggregated Google Sheet: Assignment11_BennettZiel Google Sheet.

1. *What is the ARM code?*

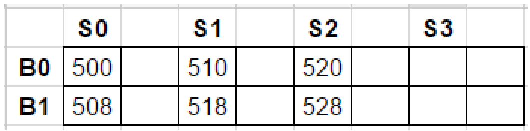
Addr.	Binary Inst.	Assembly Inst.	
0x0100	1001000100_000000000100_11111_00000	ADDI R0, R31, #4	//Loop iterations
0x0104	1001000100_010100000000_11111_00001	ADDI R1, R31, #1280	//Some value
0x0108	11111000010_000000000_00_00001_00010	LDUR R2, [R1,#0]	//Load R1[i]
0x010c	11111000010_000010000_00_00001_00011	LDUR R3, [R1,#16]	//Load R1[i+1]
0x0110	10001011000_00011_000000_00010_00100	ADD R4, R3, R2	//Sum R1[i] + R1[i+1]
0x0114	11111000000_000000000_00_00001_00100	STUR R0, [R1,#4]	//Store loop count in beginning of R1(?)
0x0118	1001000100_000000001000_00001_00001	ADDI R1, R1, #8	//Add 8 to R1
0x011c	1101000100_000000000001_00000_00000	SUBI R0, R0, #1	//Decrement loop count
0x0120	10110101_1111111111111111010_00000	CBNZ R0, #-6	//Go back to 1st load for 4 more iterations
0x0124	11001011000_001010_00000_00101_00101	SUB R5, R5, R10	

We compared each instruction's MSB's with the ARM/LEG opcode table until we found a match. Then, we interpolated the remaining immediate/ register values according to the specific instruction type.

2. *What are the cache hits/misses?*



Access #	Addr. in R1	Hex Address	Binary Addr	Byte Offset	B.O.	Index	Tag	Hit/ Miss	Penalty
1	0x0500	0x0500	0101_0000_0000	000	0	00		Compulsory Miss	10
2	0x0500	0x0510	0101_0001_0000	000	0	01		Compulsory Miss	10
3	0x0500	0x0504	0101_0000_0100	100	0	00		Hit	1
4	0x0508	0x0508	0101_0000_1000	000	1	00		Hit	1
5	0x0508	0x0518	0101_0001_1000	000	1	01	**For**	Hit	1
6	0x0508	0x050C	0101_0000_1100	100	1	00	**All**	Hit	1
7	0x0510	0x0510	0101_0001_0000	000	0	01	**EE (58 bits)**	Hit	1
8	0x0510	0x0520	0101_0010_0000	000	0	10		Compulsory Miss	10
9	0x0510	0x0514	0101_0001_0100	100	0	01		Hit	1
10	0x0518	0x0518	0101_0001_1000	000	1	01		Hit	1
11	0x0518	0x0528	0101_0010_1000	000	1	10		Hit	1
12	0x0518	0x051C	0101_0001_1100	100	1	01		Hit	1
								Total:	**39**

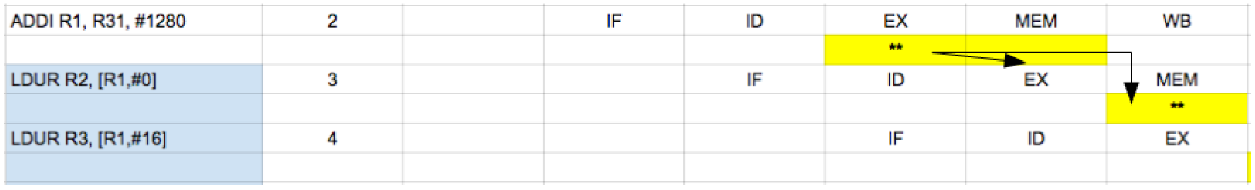


	S0		S1		S2		S3	
B0	500		510		520			
B1	508		518		528			

No replacements were needed.

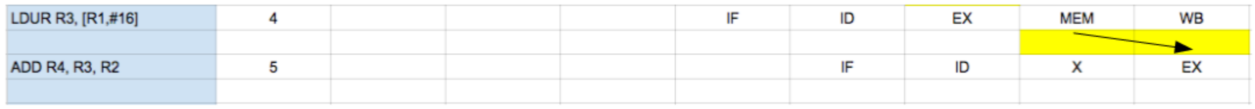
3. *Where are all the hazards in the code execution?*

[Yellow boxes indicate diagonal forwarding wherever arrows aren't explicitly drawn]
We first have a data hazard involving the first ADDI and the two LOAD']s that follow. To get around it, we have to simultaneously forward the result of the ADDI's EX stage to the EX stages of both LOAD's. This works because the address needed by the LOAD's becomes available after the ADDI's EX stage.



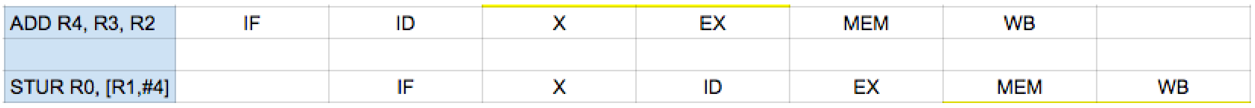
ADDI R1, R31, #1280	2		IF	ID	EX	MEM	WB	
					**			
LDUR R2, [R1,#0]	3			IF	ID	EX	MEM	
							**	
LDUR R3, [R1,#16]	4				IF	ID	EX	

Next, we have another data hazard between the second LOAD, and the ADD that follows it. Again, this is resolved through forwarding, since the register data is available after the LOAD's MEM stage.



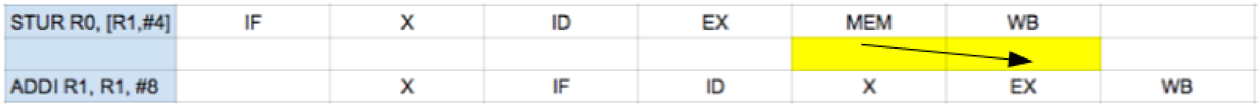
LDUR R3, [R1,#16]	4				IF	ID	EX	MEM	WB
ADD R4, R3, R2	5					IF	ID	X	EX

In the next line, we begin a trend of structural hazards. Because of the stall needed by the ADD, the STORE must stall until the ID stage is open. These hazards persist and compound throughout the entire program.



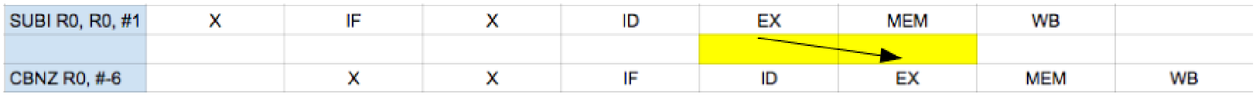
ADD R4, R3, R2	IF	ID	X	EX	MEM	WB	
STUR R0, [R1,#4]		IF	X	ID	EX	MEM	WB

Next, there's another data-conflict between the STORE and the x. It's dealt with using the same forwarding technique as the previous LOAD.



STUR R0, [R1,#4]	IF	X	ID	EX	MEM	WB	
ADDI R1, R1, #8		X	IF	ID	X	EX	WB

Last, there's a final data-hazard between the SUBI and the CBNZ. We can just forward between EX stages without stalling.



SUBI R0, R0, #1	X	IF	X	ID	EX	MEM	WB	
CBNZ R0, #-6		X	X	IF	ID	EX	MEM	WB

All hazards, except the first group mentioned, repeat since they fall within the loop.

Adam Ziel & Matt Bennett

4. *How many cycles does it take to run the code from beginning to end including all iterations?*

82 cycles. 43 come from the pipeline without accounting for cache accesses. 39 come from accessing cache.

Simplified pipeline:

	Cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...43	
Instruction	(no cache)																	
	Line																	
ADDI R0, R31, #4	1	IF	ID	EX	MEM	WB												
ADDI R1, R31, #1280	2		IF	ID	EX	MEM	WB											
					**	🟨												
LDUR R2, [R1,#0]	3			IF	ID	EX	MEM	WB										
							**											
LDUR R3, [R1,#16]	4				IF	ID	EX	MEM	WB									
								🟨	🟨									**Do this**
ADD R4, R3, R2	5					IF	ID	X	EX	MEM	WB							**loop 4 times**
																		total, but
STUR R0, [R1,#4]	6						IF	X	ID	EX	MEM	WB						**account for**
											🟨	🟨						**boundary**
ADDI R1, R1, #8	7							X	IF	ID	X	EX	WB					**stall cases**
SUBI R0, R0, #1	8								X	IF	X	ID	EX	MEM	WB			
													🟨	🟨				
CBNZ R0, #-6	9									X	X	IF	ID	EX	MEM	WB		
SUB R5, R5, R10	10										X	X	IF	ID	EX	MEM	WB	

Full pipeline:

Inst.	(no cache) Line																																																											
ADDI R0, R31, #4	1	IF	ID	EX	MEM	WB																																																							
ADDI R1, R31, #1280	2		IF	ID	EX	MEM	WB																																																						
							🟨	🟨																																																					
LDUR R2, [R1,#0]	3			IF	ID	EX	X	MEM	WB																																																				
LDUR R3, [R1,#16]	4				IF	ID	EX	X	MEM	WB																																																			
										🟨	🟨																																																		
ADD R4, R3, R2	5					IF	ID	X	X	X	EX	MEM	WB								1st Loop Iter.																																										
STUR R0, [R1,#4]	6						IF	X	X	X	ID	EX	MEM	WB																																																
														🟨	🟨																																													
ADDI R1, R1, #8	7							X	X	X	IF	ID	X	X	EX	WB																																													
SUBI R0, R0, #1	8								X	X	X	IF	X	X	ID	EX	MEM	WB																																											
																		🟨	🟨																																										
CBNZ R0, #-6	9									X	X	X	X	X	IF	ID	X	X	EX	MEM	WB																																								
LDUR R2, [R1,#0]	10										X	X	X	X	X	IF	X	X	ID	EX	MEM	WB																																							
LDUR R3, [R1,#16]	11											X	X	X	X	X	X	X	IF	ID	EX	MEM	WB																																						
																							🟨	🟨																																					
ADD R4, R3, R2	12												X	X	X	X	X	X	X	IF	ID	X	X	EX	MEM	WB								2nd Loop Iter.																											
STUR R0, [R1,#4]	13													X	X	X	X	X	X	X	IF	X	X	ID	EX	MEM	WB																																		
																											🟨	🟨																																
ADDI R1, R1, #8	14														X	X	X	X	X	X	X	X	X	IF	ID	X	X	EX	WB																																
SUBI R0, R0, #1	15															X	X	X	X	X	X	X	X	X	IF	X	X	ID	EX	MEM	WB																														
																															🟨	🟨																													
CBNZ R0, #-6	16																X	X	X	X	X	X	X	X	X	X	X	IF	ID	X	X	EX	MEM	WB																											
LDUR R2, [R1,#0]	17																	X	X	X	X	X	X	X	X	X	X	X	IF	X	X	ID	EX	MEM	WB																										
LDUR R3, [R1,#16]	18																		X	X	X	X	X	X	X	X	X	X	X	X	X	IF	ID	EX	MEM	WB																									
																																				🟨	🟨																								
ADD R4, R3, R2	19																			X	X	X	X	X	X	X	X	X	X	X	X	X	IF	ID	X	X	EX	MEM	WB									3rd Loop Iter.													
STUR R0, [R1,#4]	20																				X	X	X	X	X	X	X	X	X	X	X	X	X	IF	X	X	ID	EX	MEM	WB																					
																																								🟨	🟨																				
ADDI R1, R1, #8	21																					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	IF	ID	X	X	EX	WB																			
SUBI R0, R0, #1	22																						X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	IF	X	X	ID	EX	MEM	WB																	
																																												🟨	🟨																
CBNZ R0, #-6	23																							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	IF	ID	X	X	EX	MEM	WB														
LDUR R2, [R1,#0]	24																								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	IF	X	X	ID	EX	MEM	WB													
LDUR R3, [R1,#16]	25																									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	IF	ID	EX	MEM	WB												
																																																	🟨	🟨											
ADD R4, R3, R2	26																										X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	IF	ID	X	X	EX	MEM	WB								4th Loop Iter.	
STUR R0, [R1,#4]	27																											X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	IF	X	X	ID	EX	MEM	WB								
																																																						🟨	🟨							
ADDI R1, R1, #8	28																												X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	IF	ID	X	X	EX	WB						
SUBI R0, R0, #1	29																													X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	IF	X	X	ID	EX	MEM	WB				
																																																										🟨	🟨			
CBNZ R0, #-6	30																														X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	IF	ID	X	X	EX	MEM	WB	
SUB R5, R5, R10	31																															X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	IF	X	X	ID	EX	MEM	WB

Adam Ziel & Matt Bennett

5. *What percentage of the time is spent waiting on the memory?*

The instructions take 82 cycles. The processor is waiting on memory for 39 of them.
39 / 82 * 100 = **47.6%**

EXTRA CREDIT: You are shrunk to the height of a nickel and your mass is proportionally reduced so as to maintain your original density. You are then thrown into an empty glass blender. The blades will start moving in 60 seconds. What do you do?