

Problem 1

	Instruction	Fet	Dec	Iss	RS	LSQ	ROB	Exe	CDB	Com	
1	addi v0, a2, 0x3	1	2	3	0		0	4	5	6	
2	muli v0,v0,0x03	1	2	3	1		1	6	10	11	Wait for v0
3	sub sp, sp, v0	2	3	4	2		2	11	12	13	Wait for v0
4	addi a2, sp, 24	2	3	4	3		3	13	14	15	Wait for sp
5	sub sp, sp, v0	3	4	5	0		4	14	15	16	RS, Wait for sp, Adder
6	addi a3, sp, 24	3	4	7	1		5	16	17	18	RS, Wait for sp
7	sub sp, sp, v0	4	5	12	2		6	17	18	19	RS, Wait for sp
8	addi t0, sp, 24	4	7	14	3		7	19	20	21	RS, Wait for sp
9	sub sp, sp, v0	5	12	15	0		0	20	21	22	RS, Wait for sp, Adder
10	lw v1, -32740(sp)	7	14	17	1	0	1	22	24	25	RS, Wait for sp
11	addi a0, sp, 24	12	15	18	2		2	22	23	25	RS, Wait for sp
12	sub sp, sp, v0	14	17	20	3		3	23	24	26	RS, Wait for sp
13	lw v0, -32740(sp)	15	18	21	0	1	4	25	27	28	RS, Wait for sp
14	sw a3, 18732(v1)	17	20	23	1	2	5	26	28	29	RS
15	lw v1, -32740(sp)	18	21	26	2	0	6	29	31	32	RS, LSQ, Wait for prev st
16	sw a2, 18728(v0)	20	23	29	3	1	7	30	32	33	RS, LSQ
17	lw v0, -32740(sp)	21	26	30	0	2	0	33	35	36	RS, LSQ, Wait for prev st
18	sw a0, 18740(v1)	23	29	33	1	0	1	34	36	37	RS, LSQ
19	lw a0, -32740(sp)	26	30	34	2	1	2	37	39	40	RS, LSQ, Wait for prev st
20	sw t0, 18724(v0)	29	33	37	3	2	3	38	40	41	RS, LSQ
21	lw v0, -32740(sp)	30	34	38	0	0	4	41	43	44	RS, LSQ, Wait for prev st
22	lw t9, -28608(sp)	33	37	41	1	1	5	42	44	45	RS, LSQ, Wait for prev st
23	addi a1, sp, 24	34	38	41	2		6	42	43	45	RS
24	addi a0, a0, 18804	37	41	42	3		7	43	44	46	RS
25	addi a2, zero, 124	38	41	42	0		0	44	45	46	RS, Adder
26	sw a1, 18736(v0)	41	42	43	1	2	1	44	46	47	RS, LSQ, Wait for a1
27	move a1, zero	41	42	43	2		2	45	46	47	RS, LSQ, Wait for a1

Notes:

All instructions in given code are integer operations, and muli takes four cycles. Move is an integer operation.

All instructions should use RS and LSQ.

LSQ should be freed after commit.

RS can be freed after beginning of execution, or end of execution.

Stalls in issue stage propagate to fetch and decode pipeline stages.

A load instruction can be executed either (1) after commit of previous stores, (2) after previous stores execution (by forwarding), (3) without waiting for any previous stores (special recovering hardware needed).

Scoring:

Did not stall fetch and decode properly. (-3)

Incorrect fetch and decode stalls. (-2)

Freed RS at commit stage (this is too late). (-2)

Load and store did not use RS. (-2)

LSQ and ROB entries are used in out-of-order. (-2)

Each following mistakes are -1 for once, and -2 for more:

Did not fetch, decode, issue and commit 2 instructions per cycle when it is possible. (-1)

Waiting too long to be executed. (-1)

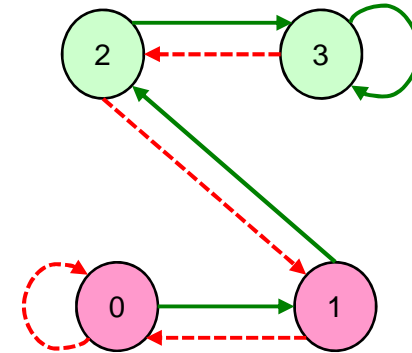
Getting RS too soon. (-1)

Out-of-order front-end (fetch, decode, and issue). (-1)

Errors in dependency, resources, and etc. (-1)

Problem 2.a

Address	T/NT	Entry	Prediction	Correct ?	Predictor State (0, 1, 2, 3)			
					0	1	2	3
0x4652a8	NT	0	NT	Yes	0	0	0	0
0x4652b1	NT	1	NT	Yes	0	0	0	0
0x4652b3	T	3	NT	No	0	0	0	1
0x4652c1	NT	1	NT	Yes	0	0	0	1
0x4652c2	NT	2	NT	Yes	0	0	0	1
0x465300	T	0	NT	No	1	0	0	1
0x4652a8	NT	0	NT	Yes	0	0	0	1
0x4652b1	NT	1	NT	Yes	0	0	0	1
0x4652b3	NT	3	NT	Yes	0	0	0	0
0x4652c1	T	1	NT	No	0	1	0	0
0x4652c2	NT	2	NT	Yes	0	1	0	0
0x465300	T	0	NT	No	1	1	0	0
0x4652a8	NT	0	NT	Yes	0	1	0	0
0x4652b1	T	1	NT	No	0	2	0	0
0x4652b3	T	3	NT	No	0	2	0	1
0x4652c1	T	1	T	Yes	0	3	0	1
0x4652c2	NT	2	NT	Yes	0	3	0	1
0x465300	T	0	NT	No	1	3	0	1
0x4652a8	T	0	NT	No	2	3	0	1
0x4652b3	NT	3	NT	Yes	2	3	0	0
0x4652a8	NT	0	T	No	1	3	0	0
0x4652b1	T	1	T	Yes	1	3	0	0
0x4652c1	T	1	T	Yes	1	3	0	0
0x4652c2	NT	2	NT	Yes	1	3	0	0
0x465300	T	0	NT	No	2	3	0	0



FSM for 2 bit counter

Problem 2.b

Address	T/NT	Pred?	Coor?	Predictor State (0, 1, 2, ..., 7)							
0x4652a8	NT	NT	Yes	<u>000</u>	000	000	000	000	000	000	000
0x4652b1	NT	NT	Yes		<u>000</u>						
0x4652b3	T	NT	No				<u>110</u>				
0x4652c1	NT	NT	Yes		<u>000</u>						
0x4652c2	NT	NT	Yes			<u>000</u>					
0x465300	T	NT	No	<u>110</u>							
0x4652a8	NT	NT	Yes	<u>010</u>							
0x4652b1	NT	NT	Yes		<u>000</u>						
0x4652b3	NT	NT	Yes				<u>010</u>				
0x4652c1	T	NT	No		<u>110</u>						
0x4652c2	NT	NT	Yes			<u>000</u>					
0x465300	T	T	Yes	<u>110</u>							
0x4652a8	NT	NT	Yes	<u>010</u>							
0x4652b1	T	NT	No		<u>111</u>						
0x4652b3	T	T	Yes				<u>110</u>				
0x4652c1	T	T	Yes		<u>111</u>						
0x4652c2	NT	NT	Yes			<u>000</u>					
0x465300	T	T	Yes	<u>110</u>							
0x4652a8	T	NT	No	<u>111</u>							
0x4652b3	NT	NT	Yes				<u>010</u>				
0x4652a8	NT	T	No	<u>010</u>							
0x4652b1	T	T	Yes		<u>111</u>						
0x4652c1	T	T	Yes		<u>111</u>						
0x4652c2	NT	NT	Yes			<u>000</u>					
0x465300	T	T	Yes	<u>110</u>							

If unclear, please ask in the newsgroup.