HW2 Answers:

1 The MemtoReg is “don’t care” when RegWrite signal is 0 (the register is not being written, thus the value of the data on the register data write port is not used). Therefore, MemtoReg can be “X” (don’t care) for STUR and CBZ (Regwrite should be 0).

Reg2Loc is “don’t care” for LDUR, since the second register is not used.

4.27

4.27.1
ADD X 5, X2, X1
NOP
NOP
LDUR X3, [X5, #4]
LDUR X2, [X2, #0]
NOP
ORR X3, X5, X3
NOP
NOP
STUR X3, [X5, #0]

4.27.2 It is not possible to reduce the number of NOPs.

4.27.3 The code executes correctly. We need hazard detection only to insert a stall when the instruction following a load uses the result of the load. That does not happen in this case.

4.27.4

Cycle	1	2	3	4	5	6	7	8	
ADD	IF	ID	EX	ME	WB				
LDUR		IF	ID	EX	ME	WB			
LDUR			IF	ID	EX	ME	WB		
ORR				IF	ID	EX	ME	WB	
STUR					IF	ID	EX	ME	WB

Because there are no stalls in this code, PCWrite and IF/IDWrite are always 1 and the mux before ID/EX is always set to pass the control values through.
(1) ForwardA = X; ForwardB = X (no instruction in EX stage yet)
(2) ForwardA = X; ForwardB = X (no instruction in EX stage yet)
(3) ForwardA = 0; ForwardB = 0 (no forwarding; values taken from registers)
(4) ForwardA = 2; ForwardB = 0 (base register taken from result of previous instruction)
(5) ForwardA = 1; ForwardB = 1 (base reguster taken from result of two instructions previous)
(6) ForwardA = 0; ForwardB = 2 (Rn = X5 taken from register; Rm = X3 taken from result of 1st LDUR—two instructions ago)
(7) ForwardA = 0; ForwardB = 2 (base register taken from register fi le. Data to be written taken from previous instruction)

4.27.5 The hazard detection unit additionally needs the values of Rd that comes out of the MEM/WB register. The instruction that is currently in the ID stage needs to be stalled if it depends on a value produced by (or forwarded from) the instruction in the EX or the instruction in the MEM stage. So we need to check the destination register of these two instructions if the instruction is an I-type, R-type, or load. (The hazard detection unit already has access to the op code.) The Hazard unit already has the value of Rd from the EX/MEM register as inputs, so we need only add the value from the MEM/WB register. No additional outputs are needed. We can stall the pipeline using the three output signals that we already have. The value of Rd from EX/MEM is needed to detect the data hazard between the ADD and the following LDUR. The value of Rd form MEM/WB is needed to detect the data hazard between the first LDUR instruction and the ORR instruction.

4.27.6

Cycle	1	2	3	4	5	6
ADD	IF	ID	EX	ME	WB	
LDUR		IF	ID	–	–	EX
LDUR			IF	–	–	ID

(1) PCWrite = 1; IF/IDWrite = 1; control mux = 0
(2) PCWrite = 1; IF/IDWrite = 1; control mux = 0
(3) PCWrite = 1; IF/IDWrite = 1; control mux = 0
(4) PCWrite = 0; IF/IDWrite = 0; control mux = 1
(5) PCWrite = 0; IF/IDWrite = 0; control mux = 1

4.28

4.28.1

Always taken: the CPI increases from 1 to 1.1375,

Here if branch outcomes are resolved at ID stage, we assume that one stall cycle is incurred. In this case, CPI=1+(1-0.45)*0.25*1=1.1375

4.28.2 Always-not-taken, CPI increases from 1 to 1.1125=1+(1-0.55)*0.25

4.28.3 2-bit predictor, CPI increases from 1 to 1.0375=1+0.25*(1-0.85)

4.28.4 The speedup is approximately 1.019.
Changing half of the branch instructions to an ALU instruction reduces the percentage of instructions that are branches from 25% to 12.5%. Because predicted and mispredicted branches are replaced equally, the misprediction rate remains 15%. Th us, the new CPU is 1 + 0.125*(1 − 0.85)= 1.01875. This represents a speedup of 1.0375 / 1.01875 = 1.0184

4.28.5 The "speedup" is .91.
Two ADD instruction as a branch with an "extra" cycle. Thus, half of the branches have 1 extra cycle; 15% of the other half have 1 extra cycles (the pipeline flush); and the remaining branches (those correctly predicted) have no extra cycles. This gives us a CPI of 1 + (.5)(.25)*1 + (.5)(.25)(.15)*1 = 1.14375 and a speedup of 1.0375 / 1.14375 = .91.

4.28.6 The predictor is 25% accurate on the remaining branches. We know that 80% of branches are always predicted correctly and the overall accuracy is 0.85. Th us, 0.8*1 + 0.2*x = 0.85. Solving for x shows that x = 0.25.

HW3 Answers

5 .9

5.9.1 AMAT for B = 8: 0.040 × (20 × 8) = 6.40
AMAT for B = 16: 0.030 × (20 × 16) = 9.60
AMAT for B = 32: 0.020 × (20 × 32) = 12.80
AMAT for B = 64: 0.015 × (20 × 64) = 19.20
AMAT for B = 128: 0.010 × (20 × 128) = 25.60
B = 8 is optimal.

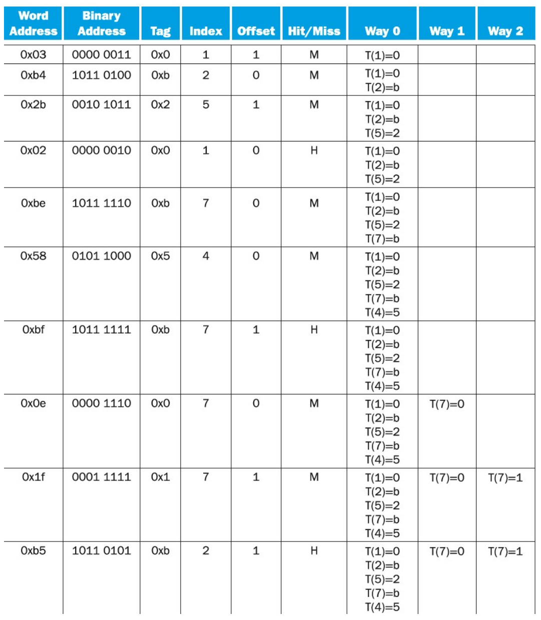
5.9.2 AMAT for B = 8: 0.040 × (24 + 8) = 1.28
AMAT for B = 16: 0.030 × (24 + 16) = 1.20
AMAT for B = 32: 0.020 × (24 + 32) = 1.12
AMAT for B = 64: 0.015 × (24 + 64) = 1.32
AMAT for B = 128: 0.010 × (24 + 128) = 1.52
B = 32 is optimal

5.9.3 B = 128 is optimal: Minimizing the miss rate minimizes the total miss latency.

5.11

5.11.1 Each line in the cache will have a total of six blocks (two in each of three ways). There will be a total of 48/6 = 8 lines.

5.11.2 T(x) is the tag at index x.



Word Address	Binary Address	Tag	Index	Offset	Hit/Miss	Way 0	Way 1	Way 2
0x03	0000 0011	0x0	1	1	M	T(1)=0		
0xb4	1011 0100	0xb	2	0	M	T(1)=0 T(2)=b		
0x2b	0010 1011	0x2	5	1	M	T(1)=0 T(2)=b T(5)=2		
0x02	0000 0010	0x0	1	0	H	T(1)=0 T(2)=b T(5)=2		
0xbe	1011 1110	0xb	7	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b		
0x58	0101 1000	0x5	4	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5		
0xbf	1011 1111	0xb	7	1	H	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5		
0x0e	0000 1110	0x0	7	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	
0x1f	0001 1111	0x1	7	1	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	T(7)=1
0xb5	1011 0101	0xb	2	1	H	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	T(7)=1

0xbf	1011 1111	0xb	7	1	H	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=0	T(7)=1
0xba	1011 1010	0xb	5	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=2 T(5)=b	T(7)=1
0x2e	0010 1110	0x2	7	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=2 T(5)=b	T(7)=1
0xce	1100 1110	0xc	7	0	M	T(1)=0 T(2)=b T(5)=2 T(7)=b T(4)=5	T(7)=2 T(5)=b	T(7)=c

5.11.3 8 sets 3-ways (each way consisting of two-words), You can slightly modify Fig.5.18, note the offset is 1 bit (word offset), indicating one of two-words (different from the byte-offset).

5.11.4 Because this cache is fully associative and has one-word blocks, there is no index and no off set. Consequently, the word address is equivalent to the tag.

Word Address	Binary Address	Tag	Hit/Miss	Contents
0x03	0000 0011	0x03	M	3
0xb4	1011 0100	0xb4	M	3, b4
0x2b	0010 1011	0x2b	M	3, b4, 2b
0x02	0000 0010	0x02	M	3, b4, 2b, 2
0xbe	1011 1110	0xbe	M	3, b4, 2b, 2, be
0x58	0101 1000	0x58	M	3, b4, 2b, 2, be, 58
0xbf	1011 1111	0xbf	M	3, b4, 2b, 2, be, 58, bf
0x0e	0000 1110	0x0e	M	3, b4, 2b, 2, be, 58, bf, e
0x1f	0001 1111	0x1f	M	b4, 2b, 2, be, 58, bf, e, 1f
0xb5	1011 0101	0xb5	M	2b, 2, be, 58, bf, e, 1f, b5
0xbf	1011 1111	0xbf	H	2b, 2, be, 58, e, 1f, b5, bf
0xba	1011 1010	0xba	M	2, be, 58, e, 1f, b5, bf, ba
0x2e	0010 1110	0x2e	M	be, 58, e, 1f, b5, bf, ba, 2e
0xce	1100 1110	0xce	M	58, e, 1f, b5, bf, ba, 2e, ce

5.11.5 Not given, modify Fig. 5.18.

5.11.6 Because this cache is fully associative, there is no index. (Contents shown in the order the data were accessed. Order does not imply physical location.)

Word Address	Binary Address	Tag	Offset	Hit/Miss	Contents
0x03	0000 0011	0x01	1	M	[2,3]
0xb4	1011 0100	0x5a	0	M	[2,3], [b4,b5]
0x2b	0010 1011	0x15	1	M	[2,3], [b4,b5], [2a,2b]
0x02	0000 0010	0x01	0	H	[b4,b5], [2a,2b], [2,3]
0xbe	1011 1110	0x5f	0	M	[b4,b5], [2a,2b], [2,3], [be, bf]
0x58	0101 1000	0x2c	0	M	[2a,2b], [2,3], [be, bf], [58, 59]
0xbf	1011 1111	0x5f	1	H	[2a,2b], [2,3], [58, 59], [be, bf]
0x0e	0000 1110	0x07	0	M	[2,3], [58, 59], [be, bf], [e,f]
0x1f	0001 1111	0x0f	1	M	[58, 59], [be, bf], [e,f], [1e,1f]
0xb5	1011 0101	0x5a	1	M	[be, bf], [e,f], [1e,1f], [b4, b5]
0xbf	1011 1111	0x5f	1	H	[e,f], [1e,1f], [b4, b5], [be, bf]
0xba	1011 1010	0x5d	0	M	[1e,1f], [b4, b5], [be, bf], [ba, bb]
0x2e	0010 1110	0x17	0	M	[b4, b5], [be, bf], [ba, bb], [2e, 2f]
0xce	1100 1110	0x67	0	M	[be, bf], [ba, bb], [2e, 2f], [ce,cf]

5.11.7 (Contents shown in the order the data were accessed. Order does not imply physical location.)

Word Address	Binary Address	Tag	Offset	Hit/Miss	Contents
0x03	0000 0011	0x01	1	M	[2,3]
0xb4	1011 0100	0x5a	0	M	[2,3], [b4,b5]
0x2b	0010 1011	0x15	1	M	[2,3], [b4,b5], [2a,2b]
0x02	0000 0010	0x01	0	H	[b4,b5], [2a,2b], [2,3]
0xbe	1011 1110	0x5f	0	M	[b4,b5], [2a,2b], [2,3], [be, bf]
0x58	0101 1000	0x2c	0	M	[b4,b5], [2a,2b], [2,3], [58, 59]
0xbf	1011 1111	0x5f	1	M	[b4,b5], [2a,2b], [2,3], [be, bf]
0x0e	0000 1110	0x07	0	M	[b4,b5], [2a,2b], [2,3], [e, f]
0x1f	0001 1111	0x0f	1	M	[b4,b5], [2a,2b], [2,3], [1e, 1f]
0xb5	1011 0101	0x5a	1	H	[2a,2b], [2,3], [1e, 1f], [b4,b5]
0xbf	1011 1111	0x5f	1	M	[2a,2b], [2,3], [1e, 1f], [be, bf]
0xba	1011 1010	0x5d	0	M	[2a,2b], [2,3], [1e, 1f], [ba, bb]
0x2e	0010 1110	0x17	0	M	[2a,2b], [2,3], [1e, 1f], [2e, 2f]
0xce	1100 1110	0x67	0	M	[2a,2b], [2,3], [1e, 1f],, [ce, cf]

5.11.8 Because this cache is fully associative, there is no index.

Word Address	Binary Address	Tag	Offset	Hit/Miss	Contents
0x03	0000 0011	0x01	1	M	[2,3]
0xb4	1011 0100	0x5a	0	M	[2,3], [b4,b5]
0x2b	0010 1011	0x15	1	M	[2,3], [b4,b5], [2a,2b]
0x02	0000 0010	0x01	0	H	[2,3], [b4,b5], [2a,2b]
0xbe	1011 1110	0x5f	0	M	[2,3], [b4,b5], [2a,2b], [be, bf]
0x58	0101 1000	0x2c	0	M	[58,59], [b4,b5], [2a,2b], [be, bf]
0xbf	1011 1111	0x5f	1	H	[58,59], [b4,b5], [2a,2b], [be, bf]
0x0e	0000 1110	0x07	0	M	[e,f], [b4,b5], [2a,2b], [be, bf]
0x1f	0001 1111	0x0f	1	M	[1e,1f], [b4,b5], [2a,2b], [be, bf]
0xb5	1011 0101	0x5a	1	H	[1e,1f], [b4,b5], [2a,2b], [be, bf]
0xbf	1011 1111	0x5f	1	H	[1e,1f], [b4,b5], [2a,2b], [be, bf]
0xba	1011 1010	0x5d	0	M	[1e,1f], [b4,b5], [ba,bb], [be, bf]
0x2e	0010 1110	0x17	0	M	[1e,1f], [b4,b5], [2e,2f], [be, bf]
0xce	1100 1110	0x67	0	M	[1e,1f], [b4,b5], [ce,cf], [be, bf]

5.12

5.12.1 Standard memory time: Each cycle on a 2Ghz machine takes 0.5 ps. Thus, a main memory access requires 100/0.5 = 200 cycles.

■ L1 only: 1.5 + 0.07*200 = 15.5

■ Direct mapped L2: 1.5 + .07 × (12 + 0.035 × 200) = 2.83

■ 8-way set associated L2: 1.5 + .07 × (28 + 0.015 × 200) = 3.67.

Doubled memory access time (thus, a main memory access requires 400 cycles)

■ L1 only: 1.5 + 0.07*400 = 29.5 (90% increase)

■ Direct mapped L2: 1.5 + .07 × (12 + 0.035 × 400) = 3.32 (17% increase)

■ 8-way set associated L2: 1.5 + .07 × (28 + 0.015 × 400) = 3.88 (5% increase).

5.12.2 1.5 = 0.07 × (12 + 0.035 × (50 + 0.013 × 100)) = 2.47

Adding the L3 cache does reduce the overall memory access time, which is the main advantage of having an L3 cache. Th e disadvantage is that the L3 cache takes real estate away from having other types of resources, such as functional units.

5.12.3 No size will achieve the performance goal.

We want the CPI of the CPU with an external L2 cache to be at most 2.83. Let x be the necessary miss rate.

1.5 + 0.07*(50 + x*200) < 2.83

Solving for x gives that x < − 0.155. Th is means that even if the miss rate of the L2 cache was 0, a 50- ns access time gives a CPI of 1.5 + 0.07*(50 + 0*200) = 5, which is greater than the 2.83 given by the on-chip L2 caches. As such, no size will achieve the performance goal.