

NATIONAL UNIVERSITY OF SINGAPORE

ANSWERS**CS2100 – COMPUTER ORGANISATION**

(Semester 2: AY2018/19)

ANSWER BOOKLET

Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This answer booklet consists of **SIX (6)** printed pages.
2. Fill in your Student Number **with a pen clearly** below. Do **NOT** write your name.
3. You may write your answers in pencil (2B or above).

STUDENT NUMBER
(fill in with a **pen**):

A	0							
----------	----------	--	--	--	--	--	--	--

For examiner's use only		
Question	Total	Marks
Q1	12	
Q2	4	
Q3	14	
Q4	16	
Q5	22	
Q6	18	
Q7	14	
Total	100	

Write your answers in the box/space provided.

1a.
[2]

\$t0 = 26 *-1 mark for wrong answer*
\$t1 = 31 *answer*

1b.
[2]

3
1 mark for 4

1c.
[2]

-1 or 0xFFFFFFFF or $2^{32}-1$ *0 mark otherwise*

1d.
[2]

0x00108042

-1 mark for each wrong digit

1e.
[2]

\$t0 is the number of leading zeros in \$s0 / number of 0s before first 1 / position of leftmost 1 / floor(log₂(\$s0))+1 if \$s0 != 0 else 32 / number of bits in minimum binary representation

[NOT ACCEPTED: log₂(\$s0) without the condition; use of digit instead of bits; no mention of minimum because all are 32-bits]

\$t1 is the (total) number of zeros in \$s0 / 32 – number of 1s

1f.
[2]

lw \$t0, 80(\$zero)
srl \$t0, \$t0, 16

1 mark for lw

1 mark for correct result at the end

-1 mark for each additional instructions after 2

Q1:

/12

2.
[4]

0xBE999999

-1 mark for each wrong digit

Q2:

/4

- 3a. [2] *The first 4 bits do not come from PC+4, OR
Use of sign-extend instead of zero-extend [this is an edge case where
address starts with bit 1]*

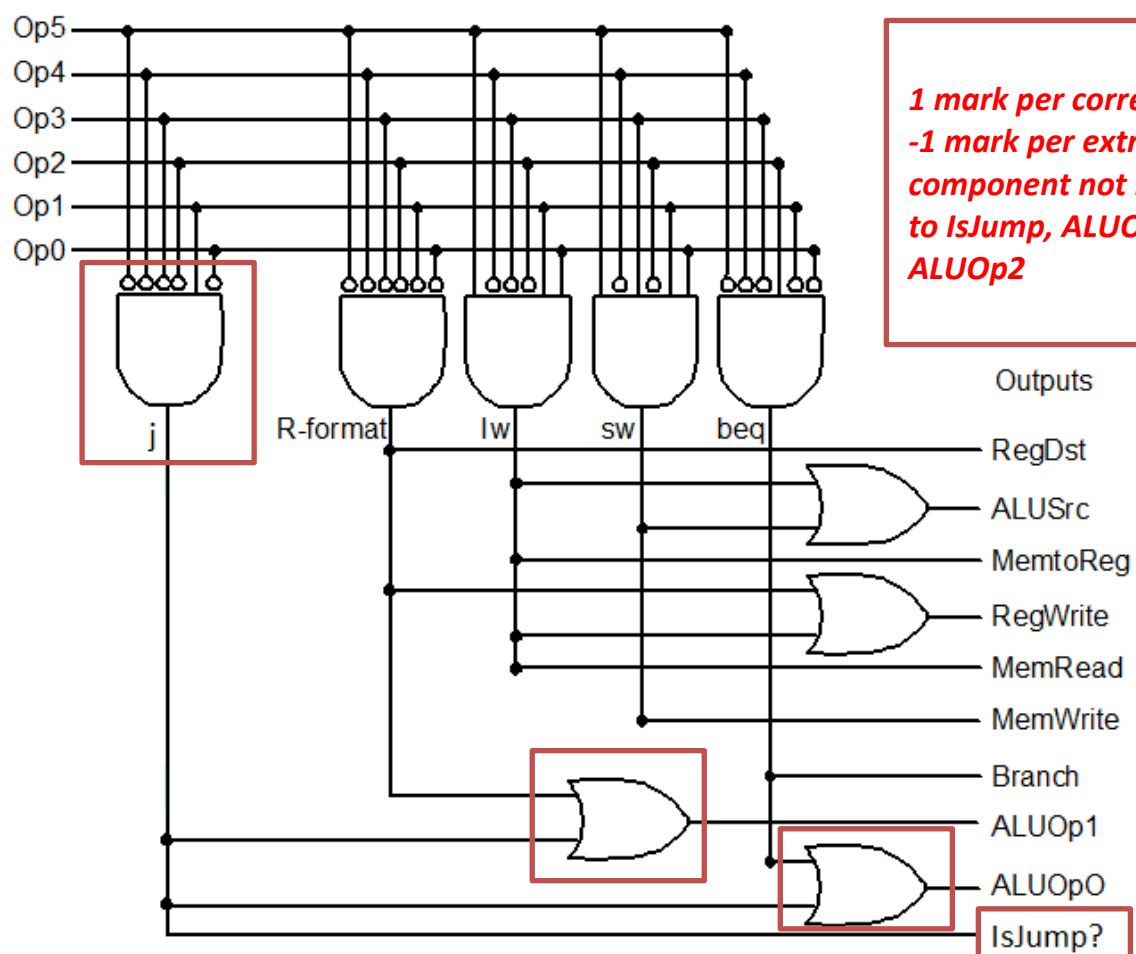
- 3b. [3] *0x00032100
-1 mark for each wrong digit*

- 3c. [3] *NOTE: X can be replaced with either 0 or 1; Mark per group all values in group must be correct*

	RegDst	ALUSrc	MtoR	Reg Write	Mem Read	Mem Write	Branch	IsJump?	ALUOp	
									Op1	Op0
R-type	1	0	0	1	0	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0	0
sw	X	1	X	0	0	1	0	0	0	0
beq	X	0	X	0	0	0	1	0	0	1
j	X	X	X	0	X	0	X	1	1	1

- 3d. [4]

Inputs



3e.
[2]**0111****-1 mark for each wrong digit**

Q3:

/14

4a.
[10]

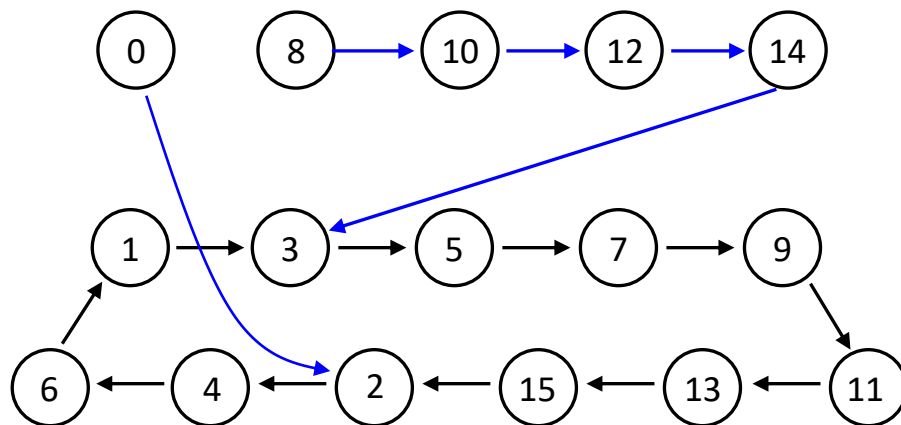
$$DA = A \cdot B' + A \cdot C' + A' \cdot B \cdot C \cdot D$$

$$TB = C$$

$$TC = A' + B' + C'$$

$$JD = B \cdot C$$

$$KD = A \cdot B \cdot C$$

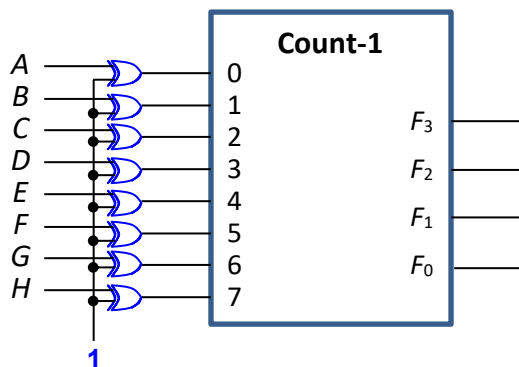
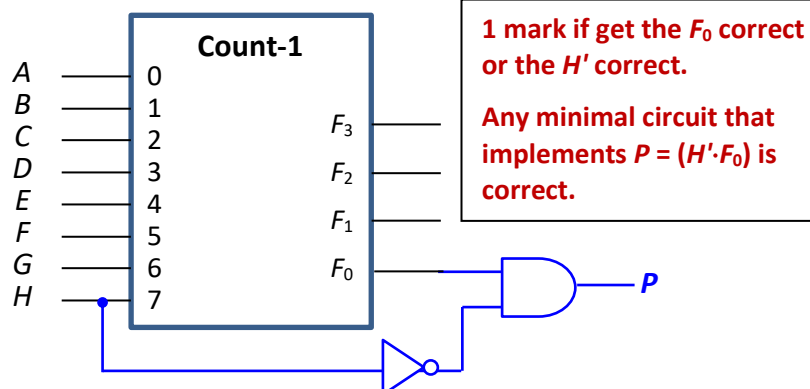
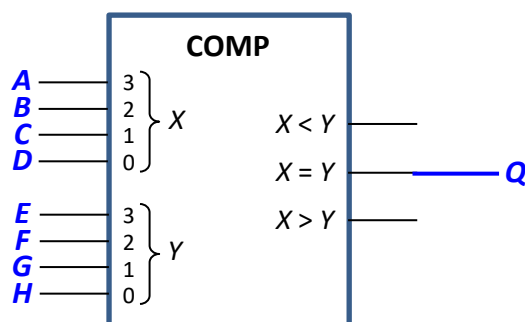
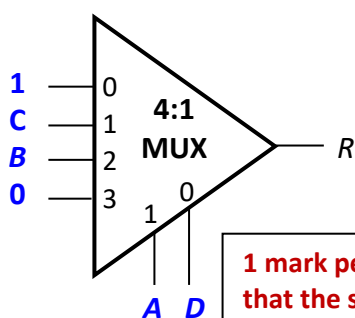
4b.
[5]4c.
[1]

Is the circuit self-correcting? Why?

Yes, it is self-correcting as any unused state can transit to a used state after a finite number of cycles.

Q4:

/16

5a.
[3]5b.
[3]5c.
[4]5d.
[6]5e.
[6]

$$X = A \cdot B$$

$$Y = A \oplus B$$

$$Z = A + B$$

1 mark per input, provided that the selector inputs are correct.

Q5:

/22

6a. [2] Set index: 3 bits; Offset: 4 bits

6b. [3] A[0] → Set 0; B[60] → Set 7; C[1032] → Set 6

6c. [6] Hit rate for array A: 7/8; array B: 3/4; array C: 3/4

6d. [2] Number of misses in first iteration: 6

6e. [2] Number of misses in second iteration: 2

6f. [3] Total number of misses: 2053 2 marks if answer is off by 1.
1 mark if answer is off by 2.

Q6: /18

7a. [2] **The jump (j) instruction incurs one stall cycle, if computation of the next PC value is done at ID stage (stage 2).**

For (b),(c): 2 marks if answer is off by 1.
1 mark if answer is off by 2.

For (d): Award mark if answer is one less than the answer of (c).

7b. [3] Without forwarding/branch decision at MEM stage <u>38</u> cycles	/c. [3] With forwarding/early branching/no branch prediction <u>27</u> cycles	/d. [3] With forwarding/early branching/branch predicted not taken <u>26</u> cycles
---	--	--

7e. [3] **Move instructions 10 and 11 to after instruction 13, reducing 2 stall cycles**

or

lw	\$t6, 0(\$t0)	# Inst8
lw	\$t7, 0(\$t1)	# Inst9
lw	\$t8, 4(\$t0)	# Inst12
lw	\$t9, 0(\$t2)	# Inst13
add	\$t6, \$t6, \$t7	# Inst10
sw	\$t6, 0(\$t0)	# Inst11
add	\$t8, \$t8, \$t9	# Inst14
sw	\$t8, 4(\$t0)	# Inst15

Other alternative answers possible. Too many to list here.

Q7: /14

=== END OF PAPER ===

Workings

Q1 (a) Via tracing and/or reasoning of the program (i.e., do Q1 (e) first).

(b) $43_{10} = 101011_2$. Based on Q1 (e), number of non-leading zeros = 2. Total = 3 (1 for last branch).

(c) -1. i.e., $0xFFFFFFFF$ = no leading zeros and no ones.

(d) R-format: **srl \$s0, \$s0, 1**

Encoding: **000000 00000 10000 10000 00001 000010**

Hexadecimal: **0x00108042**

(e) Program reasoning

(f) Minimum is 2: load and shift.

Q2 $0.3_{10} = 0.0 \ 1001 \ 1001 \ 1001 \ 1001 \dots_2$

Normalise: $1. \ 0011 \ 0011 \ 0011 \ 0011 \ 0011 \ 0011 \ 0011 \ 0011 \ 0011 \dots \times 2^{-2}$

Exponent: $-2 \rightarrow 127-2 = 125 = 0111 \ 1101$

Mantissa: $0011 \ 0011 \ 0011 \ 0011 \ 0011 \ 001$ (truncate to 23 bits)

Sign bit: 1

Binary: $1011 \ 1110 \ 1001 \ 1001 \ 1001 \ 1001 \ 1001 \ 1001$

Hexadecimal: **0xBE999999**

Q3 (a) The first 4 bits should come from PC+4 (OR sign extend is used)

(b) Since the first 4 bits do not come from PC+4 but instead, sign extend is used, the first 4 bits always follow the MSB. In this case MSB = 0, so first 4 bits = 0.

Binary: 0000 1000 0000 0000 1100 1000 0100 0000

Opcode: 000010

Address: 0000 (*incorrect, not from PC+4*) 00 0000 0000 1100 1000 0100 0000 00 (*last 2 bits always 00*)

Hexadecimal: **0x00032100**

(c) Mostly everything is don't care. However, we should not write into register and we should not write into memory. Mem Read actually don't really matter because we will not write into register anyway, so I accept both 0/X. Similarly with Branch as it is superseded by IsJump?.

(d) The AND gate in j should encode 000010. The OR gate should be used: $ALUOp1 = R\text{-type OR } j$; $ALUOp2 = \text{beq OR } j$.

(e) Binary: 0000 1000 0000 0000 0000 0000 0011 0001

Expand: 000010 00000 00000 00000 00000 110001

op rs rt rd shamt funct

$F0 = 1$; $F1 = 0$; $F2 = 0$; $F3 = 0$

$ALUControl3 = 0$

$ALUControl2 = ALUOp0 + X = 1 + X = 1$

$ALUControl1 = ALUOp1' + F2' = 0 + 1 = 1$

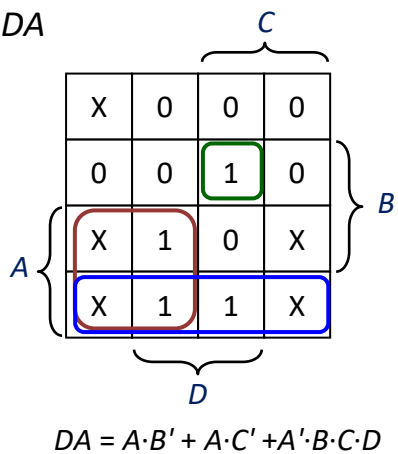
$ALUControl0 = (F0 + F3) \cdot ALUOp1 = (1 + 0) \cdot 1 = 1$

$ALUControl = \mathbf{0111}$

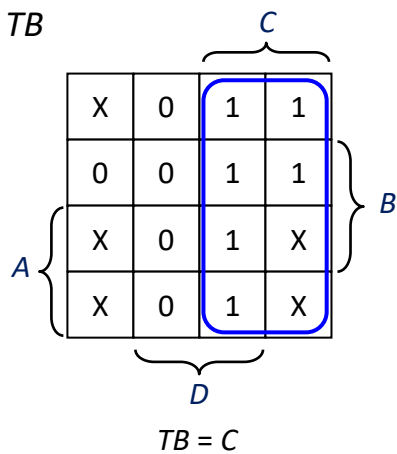
Q4.

Current state				Nex state							
A	B	C	D	DA=A ⁺	B ⁺	C ⁺	D ⁺	TB	TC	JD	KD
0	0	0	0	X(0)	X(0)	X(1)	X(0)	X(0)	X(1)	X(0)	X(0)
0	0	0	1	0	0	1	1	0	1	X	0
0	0	1	0	0	1	0	0	1	1	0	X
0	0	1	1	0	1	0	1	1	1	X	0
0	1	0	0	0	1	1	0	0	1	0	X
0	1	0	1	0	1	1	1	0	1	X	0
0	1	1	0	0	0	0	1	1	1	1	X
0	1	1	1	1	0	0	1	1	1	X	0
1	0	0	0	X(1)	X(0)	X(1)	X(0)	X(0)	X(1)	X(0)	X(0)
1	0	0	1	1	0	1	1	0	1	X	0
1	0	1	0	X(1)	X(1)	X(0)	X(0)	X(1)	X(1)	X(0)	X(0)
1	0	1	1	1	1	0	1	1	1	X	0
1	1	0	0	X(1)	X(1)	X(1)	X(0)	X(0)	X(1)	X(0)	X(0)
1	1	0	1	1	1	1	1	0	1	X	0
1	1	1	0	X(0)	X(0)	X(1)	X(1)	X(1)	X(0)	X(1)	X(1)
1	1	1	1	0	0	1	0	1	0	X	1

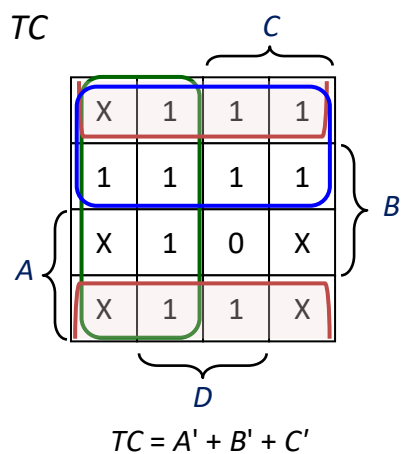
DA



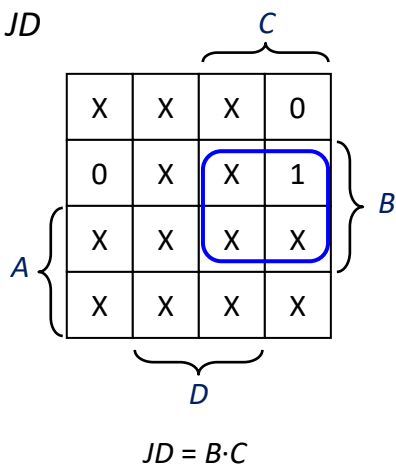
TB



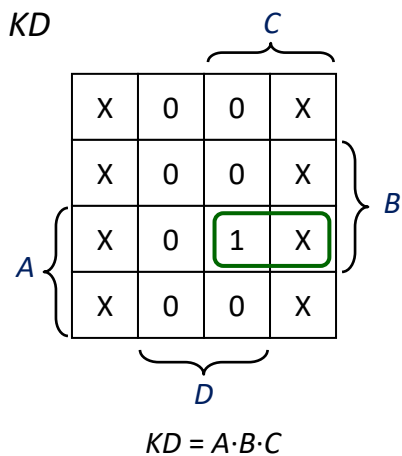
TC



JD



KD



Q5(d) $R(A,B,C,D) = \Sigma m(0, 2, 3, 4, 6, 7, 12, 14)$

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>R</i>
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

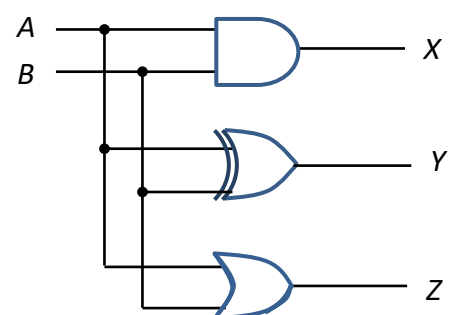
Q5(e)

<i>A</i>	<i>B</i>	<i>C</i>	<i>S</i>	<i>X</i>	<i>Y</i>	<i>Z</i>
0	0	0	0	0	0	0
0	1	0	1	0	1	1
1	0	0	1	0	1	1
1	1	1	0	1	0	1

$$X = A \cdot B$$

$$Y = A \oplus B$$

$$Z = A + B$$



Q6. Tested on QTSpm

```

# Q.asm
.data
A: .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
B: .word 101, 102, 103, 104, 105, 106, 107, 108
C: .word 201, 202, 203, 204, 205, 206, 207, 208
n: .word 8

.text
main: la    $s0, A          # $s0 is the base address of array A
      la    $s1, B          # $s1 is the base address of array B
      la    $s2, C          # $s2 is the base address of array C
      la    $t0, n          # $t0 is the address of n (size of array B)
      lw    $s3, 0($t0)     # $s3 is the content of n

#####

      add    $t0, $s0, $0    # Inst1
      add    $t1, $s1, $0    # Inst2
      add    $t2, $s2, $0    # Inst3
      add    $t3, $s3, $s3   # Inst4: $t3 = 2n
      add    $t4, $0, $0     # Inst5

Loop: slt    $t5, $t4, $t3   # Inst6: k < 2n
      beq    $t5, $0, End    # Inst7

      lw     $t6, 0($t0)     # Inst8
      lw     $t7, 0($t1)     # Inst9
      add    $t6, $t6, $t7   # Inst10
      sw     $t6, 0($t0)     # Inst11

      lw     $t8, 4($t0)     # Inst12
      lw     $t9, 0($t2)     # Inst13
      add    $t8, $t8, $t9   # Inst14
      sw     $t8, 4($t0)     # Inst15

      addi   $t0, $t0, 8     # Inst16
      addi   $t1, $t1, 4     # Inst17
      addi   $t2, $t2, 4     # Inst18
      addi   $t4, $t4, 2     # Inst19

      j      Loop           # Inst20
End:  li     $v0, 10         # system call code for exit
      syscall

```

Data:

User data segment [10000000]..[10040000]

[10000000]..[1000ffff] 00000000

[10010000]	0000000102	0000000203	0000000105	0000000206
[10010010]	0000000108	0000000209	0000000111	0000000212
[10010020]	0000000114	0000000215	0000000117	0000000218
[10010030]	0000000120	0000000221	0000000123	0000000224
[10010040]	0000000101	0000000102	0000000103	0000000104
[10010050]	0000000105	0000000106	0000000107	0000000108
[10010060]	0000000201	0000000202	0000000203	0000000204
[10010070]	0000000205	0000000206	0000000207	0000000208
[10010080]	0000000008	0000000000	0000000000	0000000000
[10010090]..[1003ffff]	00000000			

Array A

Array B

Array C

Q6. (a) 64 words; 1 block = 4 words = 16 bytes \rightarrow 16 blocks. 2 blocks per set \rightarrow 8 sets.
Set index: **3 bits**; Offset: **4 bits**.

(b) $A[0]$ at $0x00000080$

$00000080 \rightarrow 00 \dots 0000 \underline{1000} 0000 \rightarrow \text{Set 0}$

$B[0]$ at $0x00100000 \rightarrow B[60]$ at $0x001000F0$ ($60 \times 4 = 240 = 0xF0$)

$001000F0 \rightarrow 00 \dots 0000 \underline{1111} 0000 \rightarrow \text{Set 7}$

$C[0]$ at $0x00108040 \rightarrow C[1032]$ at $0x00109060$ ($1032 \times 4 = 4128 = 0x1020$)

$00109060 \rightarrow 00 \dots 0000 \underline{0110} 0000 \rightarrow \text{Set 6}$

(c) $A[0]$ at set 0; $B[0]$ at set 0; $C[0]$ at set 4

The cache content for the first 16 iterations is shown below (array element $A[k]$ is shown as A_k .)

	Block 0				Block 1			
Set 0	A_0	A_1	A_2	A_3 ✓	B_0	B_1	B_2	B_3 ✓
Set 1	A_4	A_5	A_6	A_7 ✓	B_4	B_5	B_6	B_7 ✓
Set 2	A_8	A_9	A_{10}	A_{11} ✓	B_8	B_9	B_{10}	B_{11} ✓
Set 3	A_{12}	A_{13}	A_{14}	A_{15} ✓	B_{12}	B_{13}	B_{14}	B_{15} ✓
Set 4	C_0	C_1	C_2	C_3 ✓	A_{16}	A_{17}	A_{18}	A_{19} ✓
Set 5	C_4	C_5	C_6	C_7 ✓	A_{20}	A_{21}	A_{22}	A_{23}
Set 6	C_8	C_9	C_{10}	C_{11} ✓	A_{24}	A_{25}	A_{26}	A_{27}
Set 7	C_{12}	C_{13}	C_{14}	C_{15} ✓	A_{28}	A_{29}	A_{30}	A_{31}

For parts (d), (e), (f):

Index = 2 bits; byte offset = 4 bits. Address $0x00FFFF18 = 00\dots 1111\ 1111\ 00\underline{01}\ 1000$.
Therefore, first instruction is at index 1 word 2.

The cache is shown below:

	Word0	Word1	Word2	Word3
Index 0	Inst11	Inst12	Inst13	Inst14
Index 1	Inst15	Inst16	Inst1 Inst17	Inst2 Inst18
Index 2	Inst3 Inst19	Inst4 Inst20	Inst5 Inst21	Inst6 Inst22
Index 3	Inst7	Inst8	Inst9	Inst10

- (d) First iteration, misses at instructions 1, 3, 7, 11, 15, 19 → **6 misses**.
- (e) Second iteration, misses at instructions 6, 19 → **2 misses**.
- (f) There are $2^{10} = 1024$ iterations. First iteration = 6 misses; second through 1024^{th} iterations = $1023 \times 2 = 2046$ misses; one more miss due to instruction 6.
Therefore, total = $6 + 2046 + 1 = \mathbf{2053}$ misses
Partial credit: 1 mark for 2052

Q7. (b) **38 cycles** (Partial credit: 1 mark for 37 or 39)

Cycle	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
I1 add	F	D	E	M	W																									
I2 add		F	D	E	M	W																								
I3 add			F	D	E	M	W																							
I4 add				F	D	E	M	W																						
I5 add					F	D	E	M	W																					
I6 slt					F	///	///	D	E	M	W	+2																		
I7 beq						F	///	///			D	E	M	W	+2															
I8 lw											///	///	///	F	D	E	M	W	+3											
I9 lw														F	D	E	M	W												
I10 add														F	///	///	D	E	M	W	+2									
I11 sw															F	///	///		D	E	M	W	+2							

F D E M W

Cycle	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38														
I12 lw	D	E	M	W	?. missing																								
I13 lw	F	D	E	M	W																								
I14 add		F			D	E	M	W	+2																				
I15 sw			F					D	E	M	W	+2																	
I16 addi				F					D	E	M	W																	
I17 addi					F					D	E	M	W																
I18 addi						F					D	E	M	W															
I19 addi							F					D	E	M	W														

Q7. (c) **27 cycles** (Partial credit: 1 mark for 26 or 28)

Cycle	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			
I1 add	F	D	E	M	W																									
I2 add		F	D	E	M	W																								
I3 add			F	D	E	M	W																							
I4 add				F	D	E	M	W																						
I5 add					F	D	E	M	W																					
I6 slt						F	D	E	M	W																				
I7 beq							F		D	E	M	W																		
I8 lw										F	D	E	M	W																
I9 lw											F	D	E	M	W															
I10 add												F	D		E	M	W													
I11 sw													F	D		E	M	W												
I12 lw														F	D		E	M	W											
I13 lw															F	D		E	M	W										
I14 add																F	D			E	M	W								
I15 sw																	F	D			E	M	W							
I16 addi																		F	D			E	M	W						
I17 addi																			F	D			E	M	W					
I18 addi																				F	D			E	M	W				
I19 addi																					F	D			E	M	W			

Q7. (d) This will save one stall cycle for instruction 8 (lw). Hence 26 cycles. Award mark if this answer is one less than answer for part (c).

Q7. (e) Move instructions 10-11 to after instruction 13. This would remove the data dependency arising from the add-after-lw instruction, reducing **two stall cycles**.