NATIONAL UNIVERSITY OF SINGAPORE

CS2100 – COMPUTER ORGANISATION

(Semester 2: AY2024/25)

Final Assessment Answer Sheets (ANSWERS)

Time Allowed: 2 Hours

PLEASE READ THE INSTRUCTIONS IN THE QUESTION PAPER CAREFULLY BEFORE PROCEEDING

Please write and shade your Student Number correctly on the box on the right with a pencil.

For Examiner's Use Only

Question	Marks
PART A: MCQs	/36
PART B: Q19	/12
PART B: Q20	/14
PART B: Q21	/13
PART B: Q22	/12
PART B: Q23	/13
TOTAL	**/100**

STUDENT NUMBER

A										
U	○	0	0	0	0	0	0	0	A	N
A	●	1	1	1	1	1	1	1	B	R
HT	○	2	2	2	2	2	2	2	E	U
NT	○	3	3	3	3	3	3	3	H	W
		4	4	4	4	4	4	4	J	X
		5	5	5	5	5	5	5	L	Y
		6	6	6	6	6	6	6	M	
		7	7	7	7	7	7	7		
		8	8	8	8	8	8	8		
		9	9	9	9	9	9	9		

Part A: Multiple Choice Questions (Total: 36 marks)

Please shade using **pencil** only ONE bubble for each question.

Q	(A)	(B)	(C)	(D)	(E)
1.	●	○	○	○	○
2.	○	○	○	●	○
3.	○	●	○	○	○
4.	○	●	○	○	○
5.	○	○	○	●	○
6.	○	●	○	○	○
	(A)	(B)	(C)	(D)	(E)

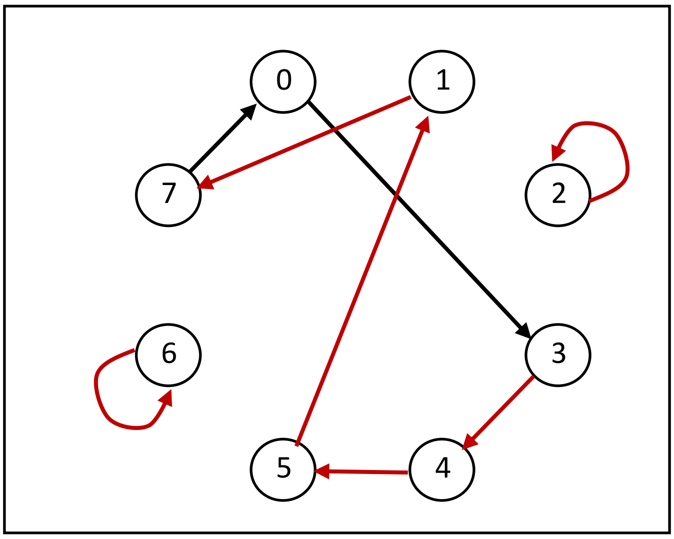
Q	(A)	(B)	(C)	(D)	(E)
7.	○	○	○	●	○
8.	○	○	●	○	○
9.	○	●	○	○	○
10.	○	●	○	○	○
11.	○	○	○	●	○
12.	○	○	●	○	○
	(A)	(B)	(C)	(D)	(E)

Q	(A)	(B)	(C)	(D)	(E)
13.	○	○	●	○	○
14.	○	○	●	○	○
15.	○	○	○	○	●
16.	○	○	●	○	○
17.	●	○	○	○	○
18.	●	○	○	○	○
	(A)	(B)	(C)	(D)	(E)

Part B (Total: 64 marks) Write your answers within the boxes provided.

19. Sequential circuit [12 marks]

(a)
[6]



(b) Sink states.
[1]

States 2 and 6

(c)
[4]

TB **=** $A' \cdot B' + B \cdot C$

TC **=** $B' \cdot C' + B \cdot C$

(d)
[1]

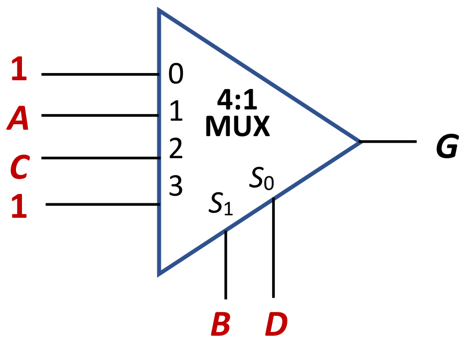
4 logic gates

20. Combinational circuits [14 marks]

(a)
[4]

F **=** $B' \cdot C'$

(b)
[4]



(c)
[6]

ALUcontrol2 = **ALUop0 + ALUop1·F3'**

ALUcontrol1 = **ALUop1' + F4**

ALUcontrol0 = **ALUop1·F5**

21. MIPS [13 marks]

(a) Contents of array *B* after the execution of the code:
[2]

B **= {11, 22, 32, 44, 54, 66, 77, 88, 98}**

(b) Write an equivalent C code using a for loop:
[3]

```
// $a0 = size; $a1 = arrayA; $a2 = arrayB
for (i = 0; i < size; i=i+2) {
   if ((A[i] % 4)) == 0) {
        B[i] = B[i]-1;
   }
}
```

Alternative: `if (!(A[i] % 4))`

(c) Total number of instructions executed:
[2]

61

(d) Encoding of `lw $s1, 0($t1)` in hexadecimal:
[2]

0x 8D31 0000

(e) Encoding of `bne $s3, $0, skip` in hexadecimal:
[2]

0x 1660 0002

(f) Encoding of `j loop` in hexadecimal:
[2]

0x 0807 FFF5

22. Pipelining [12 marks]

(a) [1] **17**

(b) [3] **11**

(c) [3] **6**

(d) [3] **3**

(e) [2] Better, Worse, or Same?

Same

23. Cache [13 marks]

Direct-mapped data cache:

(a) Byte-offset [1] **5**

(b) Index [1] **6**

(c) #hits [3] **256**

2-way set-associative data cache:

(d) Set-index [1] **5**

(e) #hits [3] **770**

Direct-mapped instruction cache:

(f) Index [1] **1**

(g) #hits [3] **402**

=== END OF PAPER ===

Working/explanation

Q1. A.

Q2. D.
a == -1 (true) so foo(&b) is not called (short-circuit evaluation). foo(&c) changes c to 0 (false). foo(&c) and !c returns true. result = a + b + c = -1 + 2 + 0 = 1.

Q3. B. $3333_4 = 11111111_2$; $11111111_2$2 = -1_{10}.

Q4. B.
0x3F838000 = 0b0 01111111 00000111000000000000000
Exponent = 127 – 127 = 0. So $1.00000111_2 \times 2^0 = 1.02734375_{10}$.

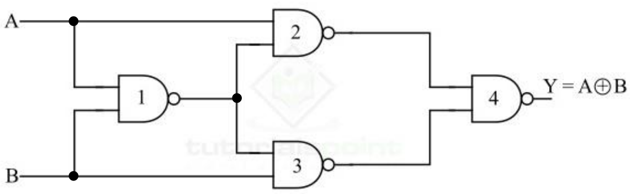
Q5. D.
0x212A000A = 0b001000 01001 01010 0000000000001010
Opcode = 0b001000 = addi; rs = \$9 = \$t1; rt = \$10 = \$t2; immed = $10_{10} - 32768_{10} = -32758_{10}$.

Q6. B.

Q7. D. Consensus theorem: $X{\cdot}Y + X'{\cdot}Z + Y{\cdot}Z = X{\cdot}Y + X'{\cdot}Z$

Q8. C. ← Give-away question!

Q9. B.
I actually expected many to give 5 NAND gates, a direct conversion of the 2-level AND-OR circuit in Q8 into a NAND-only circuits (by converting the AND gates and the OR gate into NAND gates). However, many students gave 3 NAND gates. How do you get 3 NAND gates?



Q10. B. $F(A,B,C,D,E) = \Sigma m(0,2,7,8,10,11,15,16,18,23,24,26,27,30,31)$.

Q11. D. See table below.

P	Q	R	S	X1	X0	Y1	Y0	X<Y
0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0
0	0	1	0	0	1	0	0	0
0	0	1	1	1	0	0	0	0
0	1	0	0	0	0	0	1	1
0	1	0	1	0	1	0	1	0
0	1	1	0	0	1	0	1	0
0	1	1	1	1	0	0	1	0

P	Q	R	S	X1	X0	Y1	Y0	X<Y
1	0	0	0	0	0	0	1	1
1	0	0	1	0	1	0	1	0
1	0	1	0	0	1	0	1	0
1	0	1	1	1	0	0	1	0
1	1	0	0	0	0	1	0	1
1	1	0	1	0	1	1	0	1
1	1	1	0	0	1	1	0	1
1	1	1	1	1	0	1	0	0

Q12. C. A maxterm is a sum term, not a product term.

Q13. C. Option (A): Inputs = *D*100*DD*11; (B) *C*1000111; (C) 01$\underline{A}$’11111 ; (D) *DD*011*D*01

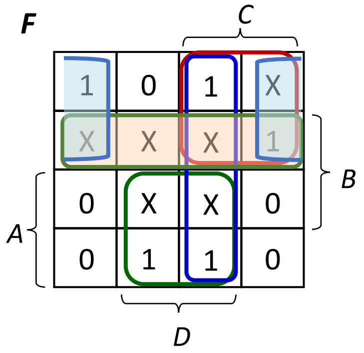
Q14. C.

5 PIs: A’·D’, A’·C, A’·B, C·D, A·D.

Note that B·D is not a PI.

Q15. E.

2 EPIs: A’·D’ and A·D.



Q16. C.

	0	1	2	3	4	5	6	7	8	9
84-2-1	0000	0111	0110	0101	0100	1011	1010	1001	1000	1111
5211	0000	0001	0011	0110	0111	1000	1001	1100	1110	1111

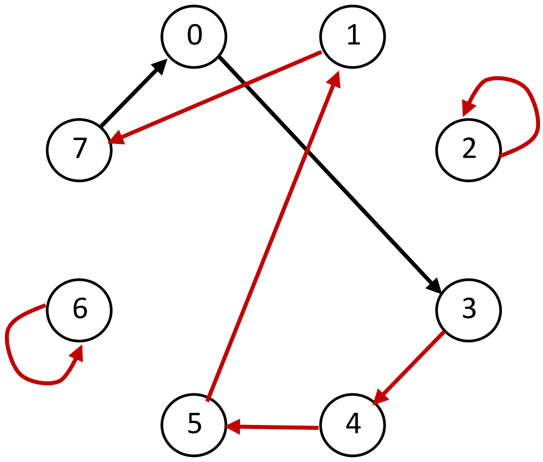
Q17. A.

Q18. A. State transition: 0101 → 1010 → 0101 → 1010 → … Hence, the 2 unique states are 0101 and 1010.

Part B: Answers and Workings

Q19. Sequential circuit (12 marks)

(a) [6 marks]



Present state			Next state			Flip-flop inputs			
A	*B*	*C*	A^+	B^+	C^+	*TA=C*	*JB=A'*	*KB= B·C*	*DC=B'*
0	0	0	0	1	1	0	1	0	1
0	0	1	1	1	1	1	1	0	1
0	1	0	0	1	0	0	1	0	0
0	1	1	1	0	0	1	1	1	0
1	0	0	1	0	1	0	0	0	1
1	0	1	0	0	1	1	0	0	1
1	1	0	1	1	0	0	0	0	0
1	1	1	0	0	0	1	0	1	0

(b) [1 mark]
States 2 and 6.

(c) [4 marks] *TB = A'·B' + B·C*; *TC = B'·C' + B·C*

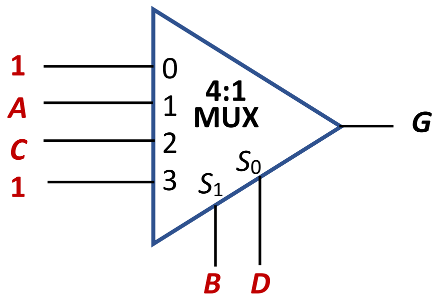
Present state			Next state			Flip-flop inputs		
A	*B*	*C*	A^+	B^+	C^+	*TA*	*TB*	*TC*
0	0	0	0	1	1	0	1	1
0	0	1	1	1	1	1	1	0
0	1	0	0	1	0	0	0	0
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	0	0	1	1	0	0
1	1	0	1	1	0	0	0	0
1	1	1	0	0	0	1	1	1

(d) [1 mark] A total of **4** logic gates. 3 gates for *TB* (2 AND gates and 1 OR gate, or 3 NAND gates) and 1 XNOR gate for *TC*.

Q20. Combinational circuits (14 marks)

(a) $F = B' \cdot A' \cdot C' + B' \cdot A \cdot C' = B' \cdot C'$ (4 marks)

(b) $G(A,B,C,D) = \Pi M(1,3,4,12)$. (4 marks)



A	*B*	*C*	*D*	*G*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

(c) ALUcontrol2 = ALUop0 + ALUop1·F3’ (6 marks)

ALUcontrol1 = ALUop1’ + F4

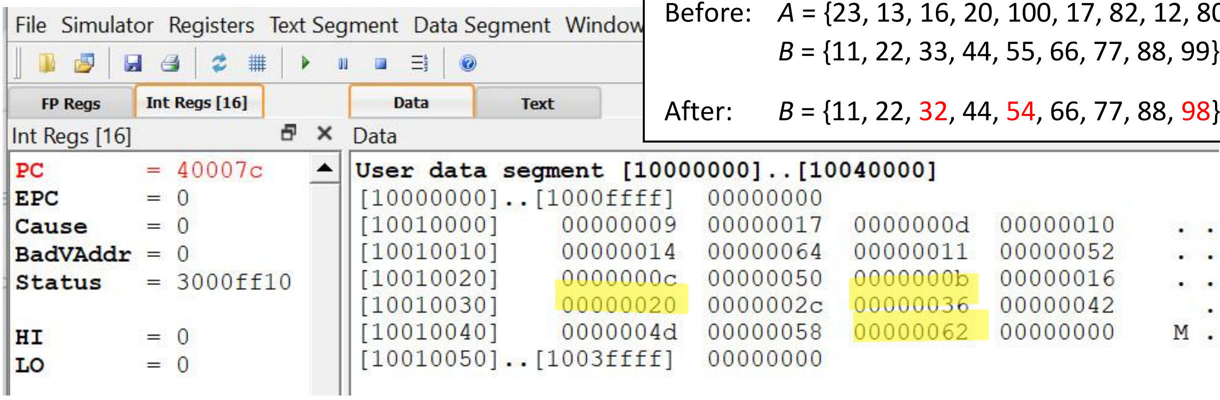
ALUcontrol0 = ALUop1·F5

27 = 011011_2; 19 = 010011_2; 11 = 001011_2; 43 = 101011_2; 51 = 110011_2.

	ALUop		funct						ALUcontrol			
	ALUop1	ALUop0	F5	F4	F3	F2	F1	F0	3	2	1	0
lw	0	0	X	X	X	X	X	X	0	0	1	0
sw	0	0	X	X	X	X	X	X	0	0	1	0
beq	0	1	X	X	X	X	X	X	0	1	1	0
add	1	0	0	1	1	0	1	1	0	0	1	0
sub	1	0	0	1	0	0	1	1	0	1	1	0
and	1	0	0	0	1	0	1	1	0	0	0	0
or	1	0	1	0	1	0	1	1	0	0	0	1
slt	1	0	1	1	0	0	1	1	0	1	1	1

Q21. MIPS (13 marks)

(a) (2 marks)



Before: *A* = {23, 13, 16, 20, 100, 17, 82, 12, 80}
B = {11, 22, 33, 44, 55, 66, 77, 88, 99}

After: *B* = {11, 22, 32, 44, 54, 66, 77, 88, 98}

(b) (3 marks)

```
// $a0 = size; $a1 = arrayA; $a2 = arrayB
for (i = 0; i < size; i=i+2) {
   if ((A[i] % 4)) == 0) {
         B[i] = B[i]-1;
   }
}
```

Alternative: `if (!(A[i] % 4))`

(c) (2 marks) Answer: 3 + 10 + 12 + 12 + 10 + 12 + 2 = **61**

If *A*[i] is divisible by 4 (2nd, 3rd and 5th iterations), 12 instructions in the loop, otherwise (1st and 4th iterations) 10 instructions. 3 instructions before the loops, and 2 instructions (Inst4 and Inst5) before exiting.

(d) (2 marks) Answer: **0x8D31 0000**

```
lw   $s1, 0($t1)
```

opcode = 0x23 = 0b100011

rs = $t1 = $9 = 0b01001; rt = $s1 = $17 = 0b10001

0b100011 01001 10001 0000 0000 0000 0000 = **0x8D31 0000**

Likely mistake (swapping rs with rt): 0b100011 10001 01001 0000… = 0x8E29 0000

(e) (2 marks) Answer: **0x1660 0002**

```
bne  $s3, $0, skip
```

opcode = 0x5 = 0b000101

rs = $s3 = $t19 = 0b10011; rt = $0 = 0b00000; skip = 0x10

0b000101 10011 00000 0000 0000 0000 0010 = **0x1660 0002**

Likely mistake (calculating number of instructions from PC instead of PC+4):

0b000101 10011 00000 0000 0000 0000 0011 = 0x1660 0003

(f) (2 marks) Answer: **0x0807 FFF5**

Address of Inst15 = 0x0020 0000. Therefore, address of Inst4 = 0x0020 0000 – 0x2C = 0x001F FFD4

```
j loop
```

opcode = 0x2 = 0b000010

Address at Inst4 = 0x001F FFD4

= 0b ~~0000~~ 0000 0001 1111 1111 1111 1101 01~~00~~

0b000010 0000 0001 1111 1111 1111 1101 01 = **0x0807 FFF5**

Q22. Pipeline (12 marks)

(a) 13 + 5 – 1 = **17** cycles. (1 mark)
(b) **11** (3 marks)
(c) **6** (3 marks)
(d) **3** (3 marks)

Delays are highlighted under the columns (b), (c), (d) for parts (b),(c),(d) below respectively.

Note that since *A*[0] = 23, Inst10 and Inst11 are skipped in the first iteration.

Code	(b)	(c)	(d)
`      add  $t0, $0, $0     # Inst1`			
`      addi $t1, $a1, 0     # Inst2`			
`      addi $t2, $a2, 0     # Inst3`			
`loop: slt  $t9, $t0, $a0  # Inst4`			
`      beq  $t9, $0, exit   # Inst5`	+2		+1
`      lw   $s1, 0($t1)     # Inst6`	+3	+3	
`      lw   $s2, 0($t2)     # Inst7`			
`      andi $s3, $s1, 3     # Inst8`	+1		
`      bne  $s3, $0, skip   # Inst9`	+2		+1
~~`      addi $s2, $s2, -1    # Inst10`~~			
~~`      sw   $s2, 0($t2)     # Inst11`~~			
`skip: addi $t0, $t0, 2     # Inst12`	+3	+3	+1
`      addi $t1, $t1, 8     # Inst13`			
`      addi $t2, $t2, 8     # Inst14`			
`      j    loop            # Inst15`			
`exit:`			
Total:	**+11**	**+6**	**+3**

(e) **Same** (2 marks)

Original:

6	`lw   $s1, 0($t1)`	F	D	E	M	W						
7	`lw   $s2, 0($t2)`		F	D	E	M	W					
8	`andi $s3, $s1, 3`			F		D	E	M	W			
9	`bne  $s3, $0, skip`				F				D	E	M	W

After swapping Inst7 and Inst8:

6	`lw   $s1, 0($t1)`	F	D	E	M	W						
8	`andi $s3, $s1, 3`		F			D	E	M	W			
7	`lw   $s2, 0($t2)`			F			D	E	M	W		
9	`bne  $s3, $0, skip`				F				D	E	M	W

Q23. Cache (13 marks)

(a) There are 8×4 = 32 = 2^5 bytes in each block. So there are **5** bits in the byte offset field. (1 mark)

(b) There are 512/8 = 64 = 2^6 blocks in the cache. So there are **6** bits in the index field. (1 mark)

(c) Address of *A*[0] = 0x10010004 = 0b……0000 0000 0000 0100. Index = 0, word 1.
1028 × 4 = 4112 = $2^{12} + 2^4$ = 0x1010.
Address of *B*[0] = 0x10011014 = 0b……0001 0000 0001 0100. Index = 0, word 5.
The following tables show the mapping of arrays *A* and *B* on the cache:

Index	Word0	Word1	Word2	Word3	Word4	Work5	Word6	Work7
0		A[0]M	A[1]	A[2]M	A[3]	A[4]M	A[5]	A[6]H
1	A[7]	A[8]M	A[9]	A[10]M	A[11]	A[12]M	A[13]	A[14]H

Index	Word0	Word1	Word2	Word3	Word4	Work5	Word6	Work7
0						B[0]M	B[1]	B[2]M
1	B[3]	B[4]M	B[5]	B[6]H	B[7]	B[8]M	B[9]	B[10]M
2	B[11]	B[12]M	B[13]	B[14]H	B[15]	B[16]M	B[17]	B[18]M

Therefore, the hits for array *A* are at *A*[6], *A*[14], *A*[22], …, *A*[1022], altogether 128 hits.
Similarly, 128 hits for array *B*. Hence, total = **256** hits. (3 marks)

(d) There are 512/8/2 = 32 = 2^5 sets in the cache. So there are **5** bits in the set index field. (1 mark)

(e) Address of *A*[0] = 0x10010004 = 0b……0000 0000 0000 0100. Set index = 0, word 1.
Address of *B*[0] = 0x10011014 = 0b……0001 0000 0001 0100. Set index = 0, word 5.
The misses for array *A* are at *A*[0], *A*[8], *A*[16], …, *A*[1024]. There are 129 misses, hence 514 – 129 = 385 hits. Similarly, 385 hits for array *B*. Hence, total = **770** hits. (3 marks)

(f) There are two blocks. So there is **1** bit in the index field. (1 mark)

(g) Address of Inst1 = 0x001F FFC4 = 0b… 1100 0100.
Note that Inst10 and Inst11 are not executed as all elements in array *A* contain the value 99.

The following table shows the mapping of the instructions on the cache and the misses and hits in the first iteration.

Index	Word0	Word1	Word2	Word3
0	Inst8(M)	Inst1(M)/Inst9(H)	Inst2(H)	Inst3(H)
1	Inst4(M)/Inst12(M)	Inst5(H)/Inst13(H)	Inst6(H)/Inst14(H)	Inst7(H)/Inst15(H)

The following table the misses and hits in the subsequent 49 iterations.

Index	Word0	Word1	Word2	Word3
0	Inst8(H)	Inst9(H)		
1	Inst4(M)/Inst12(M)	Inst5(H)/Inst13(H)	Inst6(H)/Inst14(H)	Inst7(H)/Inst15(H)

9 hits in the first iteration. 8 hits in each of the subsequent 49 iterations. Hit for Inst5 after the last iteration. Hence, total number of hits = 9 + (49 × 8) + 1 = 9 + 392 + 1 = **402** (3 marks)