NATIONAL UNIVERSITY OF SINGAPORE

ANSWERS

CS2100 – COMPUTER ORGANISATION

(Semester 2: AY2014/15)

Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper consists of ELEVEN (11) questions and comprises ELEVEN (11) printed pages.
2. This is a CLOSED BOOK examination. Two handwritten A4 reference sheets are allowed. Calculators are not allowed.
3. Answer all questions and write your answers in the ANSWER BOOKLET provided.
4. Fill in your Matriculation Number with a pen, clearly on odd-numbered pages of your ANSWER BOOKLET.
5. You may use pencil to write your answers.
6. The last three pages are for your rough work. They contain blank truth tables, K-maps, state table and timing charts for your use.
7. You are to submit only the ANSWER BOOKLET and no other document.

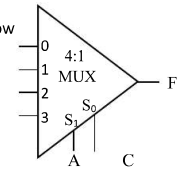
Questions 1 - 6: Each question has only one correct answer. Write your answers in the boxes provided in the Answer Booklet. One mark is awarded for a correct answer and no penalty for wrong answer.

1. To which of the following Boolean expressions is the Consensus Theorem applicable?

   i. (f + g)' h + h k + (f + g) k
   ii. W' X' Y' + W' X' Y + W X Y
   iii. a b c d + a b c' e + b d e
   iv. P Q R S + P R S T + Q' R S T

   A. Only (i) and (iv) Answer
   B. Only (iii) and (iv)
   C. Only (i), (ii) and (iii)
   D. Only (i), (iii) and (iv)
   E. Only (ii), (iii) and (iv)

2. Given a Boolean function F(A,B,C) = m(2, 4, 5, 7), how would you implement the function using a 4:1 multiplexer shown on the right, with the selector lines S_1 = A and S_0 = C?



   A. Input 0 = C; Input 1 = 0; Input 2 = C'; Input 3 = 1
   B. Input 0 = B; Input 1 = 0; Input 2 = B'; Input 3 = 1 Answer
   C. Input 0 = 0; Input 1 = B'; Input 2 = 1; Input 3 = B
   D. Input 0 = B; Input 1 = B; Input 2 = B'; Input 3 = B
   E. None of the above.

3. Which of the following Boolean expressions can be implemented using a single 2×4 decoder with 1-enable and active-high outputs, without any additional logic gates? Complemented literals are not available.

   i. A B' C
   ii. X' Y Z'
   iii. P Q'

   A. Only (i)
   B. Only (iii)
   C. Only (i) and (ii)
   D. Only (i) and (iii)
   E. All of (i), (ii) and (iii) Answer

4. If a program P executes on a machine M, which of the following sets of information allows us to calculate the execution time correctly?

   i. From P: number of instructions; From M: Clock Frequency
   ii. From P: number of instructions; From M: Clock Frequency and CPI
   iii. From P: number of instructions and average CPI; From M: Clock Period

   A. Only (i)
   B. Only (iii) Answer
   C. Only (i) and (ii)
   D. Only (i) and (iii)
   E. All of (i), (ii) and (iii)

5. Given a program P, if we know there are 30% of the instructions are of type A and 30% of the instructions are of type B, which of the following statements is/are true?

   i. Speeding up execution time for type-A instructions by 2x has the same impact on the overall execution time as speeding up type-B instructions by 2x.
   ii. There are same number of instructions of type A and type B in program P.
   iii. If we improve the execution time for both type-A and type-B instructions by 2x, the overall execution time will be (0.6/2 + 0.4) x original execution time.

   A. Only (i)
   B. Only (ii) Answer
   C. Only (ii) and (iii)
   D. All of (i), (ii) and (iii)
   E. None of the above.

6. Suppose register $s0 contains a random 32-bit value. Which of the following MIPS statements can copy the content of $s0 into $s1?

   i. xor $s1, $s0, $0
   ii. sll $s1, $s0, 1
       srl $s1, $s1, 1
   iii. andi $s1, $s0, 0xFFFF

   A. Only (i) Answer
   B. Only (i) and (ii)
   C. Only (ii) and (iii)
   D. All of (i), (ii) and (iii)
   E. None of the above.

7. [10 marks]
A sequential circuit cycles through the following states, whose state values are shown in decimal:

0 1 3 2 6 back to 0

a. The states are represented by 3-bit values ABC. Implement the sequential circuit using a JK flip-flop for A, a T flip-flop for B, and a D flip-flop for C. Write out the simplified SOP expressions for all the flip-flop inputs. Note that the simplified expression for KA has been done for you (KA = 1). You do not need to draw the logic diagram. [6 marks]

b. Complete the given state diagram on the answer booklet, by indicating the next state for each of the three unused states. [3 marks]

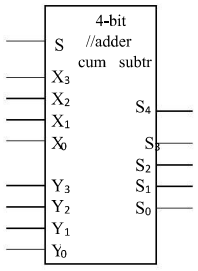
c. Which states are sinks and why? [1 mark]

8. [7 marks]
a. Given the following Boolean function:

$$F(A,B,C,D) = m(0, 5, 10, 15)$$

You are to implement F using the smallest magnitude comparator and only one of it, without any other devices or additional logic gates. Note that complemented literals are not available. Label your diagram completely. [4 marks]

b. You are given a 4-bit adder-cum-subtractor whose block diagram is shown on the right.



$X=X_3X_2X_1X_0$ and $Y=Y_3Y_2Y_1Y_0$ are unsigned binary values. The input S is a control signal. If S=0, it performs addition and the output is X + Y. If S=1, it performs subtraction and the output is X – Y.

You are to design a circuit to convert a 4-bit 2's complement value ABCD into its equivalent 4-bit 1's complement value PQRS. For example, if ABCD=1011 (representing -5 in decimal), then PQRS must be 1010 (representing -5 in decimal).

You may assume that the input ABCD=1000 will never happen since that represents -8 in decimal which is not possible to be represented in 4-bit 1's complement.

Implement the converter circuit by using the 4-bit adder-cum-subtractor and only one of it, without any other devices or additional logic gates. Note that complemented literals are not available. [3 marks]

9. [8 marks]
Study the following MIPS program:

```
    #register $s0 contains a 32-bit value
    #register $s1 contains a non-zero 8-bit value
    #       at the right most (least significant) byte
        add  $t0, $s0, $zero          #inst A
        add  $s2, $zero, $zero   #inst B
lp:  bne  $s2, $zero, done    #inst C
        beq  $t0, $zero, done    #inst D
        andi $t1, $t0, 0xFF      #inst E
        bne  $s1, $t1, nt        #inst F
        addi $s2, $s2, 1         #inst G
nt:  srl  $t0, $t0, 8         #inst H
        j    lp                 #inst J
done:
```

a. Translate instructions C and D into a single C-like high-level program statement. Remember to indicate whether this is an IF-THEN-ELSE or a WHILE statement. Use the register names as variable names. [2 marks]

b. If instruction J "j lp" is at memory address 0xBEBEC0C0, give the instruction encoding in hexadecimal. Opcode for jump is 0x02. [2 marks]

c. Give the final result (i.e. when the code terminates at "done:") of register $s2 if register $s0 contains 0xAFAFFAFA and $s1 contains 0xFF. [1 mark]

d. Give the final result (i.e. when the code terminates at "done:") of register $s2 if register $s0 contains 0xBABABABA and $s1 contains 0xBA. [1 mark]

e. Explain the purpose of the code in a single sentence. [2 marks]

10. [7 marks]
This question uses the same MIPS code in Question 9, duplicated below for your reference:

```
    #register $s0 contains a 32-bit value
    #register $s1 contains a non-zero 8-bit value
    #       at the right most (least significant) byte
        add  $t0, $s0, $zero          #inst A
        add  $s2, $zero, $zero   #inst B
lp:  bne  $s2, $zero, done    #inst C
        beq  $t0, $zero, done    #inst D
        andi $t1, $t0, 0xFF      #inst E
        bne  $s1, $t1, nt        #inst F
        addi $s2, $s2, 1         #inst G
nt:  srl  $t0, $t0, 8         #inst H
        j    lp                 #inst J
done:
```

We assume that the register $s0 contains 0xAFAFFAFA and $s1 contains 0xFF. (i.e. as in Q9, part (c)).

Given a 5-stage MIPS pipeline processor, for each of the parts below, give the total number of cycles needed for the 1st iteration of the execution from instructions A to H (i.e. excluding the "j lp"). Remember to include the cycles needed for instruction H to finish the WB stage. Note that the questions are independent from each other.

a. With only data forwarding mechanisms and no control hazard mechanism. [2 marks]

b. With data forwarding and "assume not taken" branch prediction. Note that there is no early branching. [3 marks]

c. By swapping two instructions (from Instructions A to H), we can improve the performance of early branching (with all additional forwarding paths). Give the two instructions that can be swapped. You only need to indicate the instruction letters in your answer. [2 marks]

5

CS2100 Mid Term + Finals Cheatsheet

Computer Organisation

100% (3)

2

CS2100 Final Cheat Sheet

Computer Organisation

100% (1)

2

Cheat Sheet Midterm

Computer Organisation

100% (1)

3

Lab01 Instruction - notes

Computer Organisation

100% (1)

11. [12 marks]
There are two major categories of memory accesses initiated by a processor:

a. Instruction memory access: Fetch an instruction by instruction address.

b. Data memory access: Read/Write a data value from/to memory.

We can analyze the cache performance by looking at the sequence of memory addresses generated during program execution. Below is the sequence of memory addresses generated by four iterations of the same loop.

1st Iteration		2nd Iteration		3rd Iteration		4th Iteration	
Inst.	Data	Inst.	Data	Inst.	Data	Inst.	Data
0xEBD8		0xEBD8		0xEBD8		0xEBD8	
0xEBDC		0xEBDC		0xEBDC		0xEBDC	
0xEBE0	0xAA18	0xEBE0	0xAA58	0xEBE0	0xAA28	0xEBE0	0xAA68
0xEBE4		0xEBE4		0xEBE4		0xEBE4	
0xFA10	0xBB94	0xFA10	0xBBA4	0xFA10	0xBBA4	0xFA10	0xBBB4
0xFA14		0xFA14		0xFA14		0xFA14	
0xEBE0	0xAA1C	0xEBE0	0xAA5C	0xEBE0	0xAA2C	0xEBE0	0xAA6C
0xEBE4		0xEBE4		0xEBE4		0xEBE4	

Note: The upper 16-bit for instruction addresses is 0xABCD. The upper 16-bit for data addresses is 0x0000. This table is also available on the last page if you need more space to work out the answers.

You can see that the instruction addresses are mostly sequential (first instruction of the loop at 0xABCD EBD8, then second instruction at 0xABCD EBDC, etc.) with a "jump" in the middle. Data addresses are generated only by memory instructions, e.g. the instruction 0xABCD EBE0 is likely to be memory load/store instruction which accesses memory location at 0x0000 AA18.

Given a 64 bytes direct mapping instruction cache with 16 bytes block, answer the following:

a) Give the index and offset for the first instruction access 0xABCD EBD8 in binary. [1 mark]

b) For the first iteration of loop only, what is the total number of cache misses for the instruction address sequence? [2 marks]

1st Iteration	
Inst.	Data
0xEBD8	
0xEBDC	
0xEBE0	0xAA18
0xEBE4	
0xFA10	0xBB94
0xFA14	
0xEBE0	0xAA1C
0xEBE4	

2nd Iteration	
Inst.	Data
0xEBD8	
0xEBDC	
0xEBE0	0xAA58
0xEBE4	
0xFA10	0xBBA4
0xFA14	
0xEBE0	0xAA5C
0xEBE4	

3rd Iteration	
Inst.	Data
0xEBD8	
0xEBDC	
0xEBE0	0xAA28
0xEBE4	
0xFA10	0xBBA4
0xFA14	
0xEBE0	0xAA2C
0xEBE4	

4th Iteration	
Data	Inst. Data
0xEBD8	
0xEBDC	
0xEBE0	0xAA68
0xEBE4	
0xFA10	0xBBB4
0xFA14	
0xEBE0	0xAA6C
0xEBE4	

Same table: duplicated for your convenience. Note: The upper 16-bit for instruction addresses is 0xABCD. The upper 16-bit for data addresses is 0x0000.

Given a 128 bytes 2-way set associative data cache with 16 bytes block, answer the following assuming Least Recently Used block replacement policy:

c) For the first iteration of loop only, what is the total number of cache misses for the given data address sequence? [2 marks]

d) For all four iterations, what is the total number of cache misses for the given data access sequence? You can assume that the data memory accesses are all generated by load word instructions. [3 marks]

Suppose we employ a combined direct mapping cache for instruction and data with 64 bytes total size and 16 bytes block size.

Note that if both instruction and data memory are accessed at the same "time step", the instruction memory access will be processed first.

e) For the first iteration of loop only, what is the total number of cache misses with the given instruction and data access sequences? [2 marks]

f) For all four iterations, what is the total number of cache misses with the given instruction and data access sequences? [2 marks]

~~ END OF PAPER ~~~

(Blank truth tables, K-maps, state table and timing charts are provided in the next three pages.)

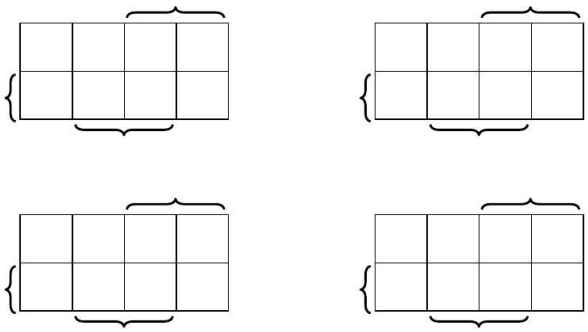
This page is for your rough work.

A	B		C
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

A	B		C	D
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

A	B		C	D			
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

This page is for your rough work.



A	B	C							
0	0	0							
0	0	1							
0	1	0							
0	1	1							
1	0	0							
1	0	1							
1	1	0							
1	1	1							

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

1st Iteration	
Inst.	Data
0xEBD8	
0xEBDC	
0xEBE0	0xAA18
0xEBE4	
0xFA10	0xBB94
0xFA14	
0xEBE0	0xAA1C
0xEBE4	

2nd Iteration	
Inst.	Data
0xEBD8	
0xEBDC	
0xEBE0	0xAA58
0xEBE4	
0xFA10	0xBBA4
0xFA14	
0xEBE0	0xAA5C
0xEBE4	

3rd Iteration	
	Inst.
0xEBD8	
0xEBDC	
0xEBE0	0xAA28
0xEBE4	
0xFA10	0xBBA4
0xFA14	
0xEBE0	0xAA2C
0xEBE4	

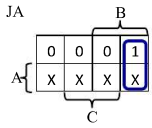
4th Iteration		
Data	Inst.	Dat
0xEBD8		
0xEBDC		
0xEBE0	0xAA68	
0xEBE4		
0xFA10	0xBBB4	
0xFA14		
0xEBE0	0xAA6C	
0xEBE4		

Answers

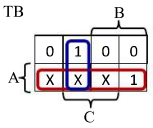
Q7. 0 1 3 2 6 0

(a)

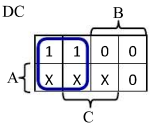
A	B	C	A^+	B^+		C^+	JA	KA	TB	DC
0	0	0	0	0	1	0	X	0	1	
0	0	1	0	1	1	0	X	1	1	
0	1	0	1	1	0	1	X	0	0	
0	1	1	0	1	0	0	X	0	0	
1	0	0	X(0)	X(1)	X(1)	X(0)	X(1)	X(1)	X(1)	
1	0	1	X(0)	X(1)	X(1)	X(0)	X(1)	X(1)	X(1)	
1	1	0	0	0	0	X(1)	1	1	0	
1	1	1	X(0)	X(0)	X(0)	X(0)	X(1)	X(1)	X(0)	



JA = $\bar{B}$ C

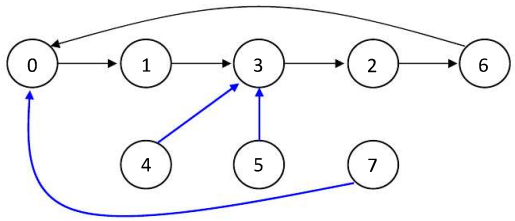


TB = $A\bar{C}$ + B



DC = B

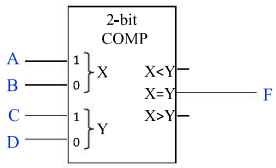
(b)



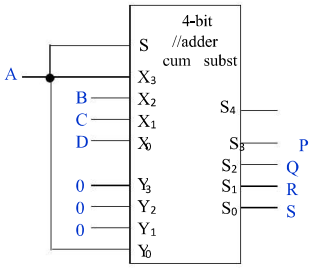
(c) There are no sink states because every state transits out of itself.

Q8. (a) Note that m(0, 5, 10, 15) ABCD = 0000, 0101, 1010 and 1111 A = C and B = D.

A single 2-bit magnitude comparator is used.



(b)



Q9. a) while ($s2 == 0 && $t0 != 0) { … }

b) Instruction C is at address
0xBEBE C0A8 = 101~~1 111~~0 1011 1110 1100 0000 1010 1000
encoding
= 000010 1110 1011 1110 1100 0000 1010 10
= 0000 1011 1010 1111 1011 0000 0010 1010
= 0x 0BAF B02A

c) $s2 = 0

d) $s2 = 1

e) Search for the 1 byte pattern in $s1 in $s0 on byte boundary (optionally: $s2 = 0 if not found; $s2 = 1 if found.)

Q10.
a)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
add	F	D	E	M	W															
add		F	D	E	M	W														
bne			F	D	E	M	W													
beq							F	D	E	M	W									
andi											F	D	E	M	W					
bne												F	D	E	M	W				
srl																F	D	E	M	W

20 cycles

b)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
add	F	D	E	M	W									
add		F	D	E	M	W								
bne			F	D	E	M	W							
beq				F	D	E	M	W						
andi					F	D	E	M	W					
bne						F	D	E	M	W				
addi							F	D	E	*	*			
srl										F	D	E	M	W

14 cycles

c) Swap instructions A and B to reduce the delay between instructions B and C. Alternative: Swap instructions C and D.

Q11.

a) 64 / 16 = 4 blocks 2 bits for index.
0xABCD EBD8 index = 01, offset = 1000

b) 3 misses.

1st Iteration		
Inst.	Index	
0xEBD8	01	M
0xEBDC	01	H
0xEBE0	10	M
0xEBE4	10	H
0xFA20	01	M
0xFA24	01	H
0xEBE0	10	H
0xEBE4	10	H

For c) and d) Data Accesses:

Access #	Full address	bits [31...6] tag	bits[5..4] set index
1	0x0000 AA18	0000 AA_00	01
2	0x0000 BB94	0000 BB_10	01
3	0x0000 AA1C	0000 AA_00	01
4	0x0000 AA58	0000 AA_01	01
5	0x0000 BBA4	0000 BB_10	10
6	0x0000 AA5C	0000 AA_01	01
7	0x0000 AA28	0000 AA_00	10
8	0x0000 BBA4	0000 BB_10	10
9	0x0000 AA2C	0000 AA_00	10
10	0x0000 AA68	0000 AA_01	10
11	0x0000 BBB4	0000 BB_10	11
12	0x0000 AA6C	0000 AA_01	10

00								
01			1	3		2		
10								
11								

c) First 3 accesses = 2 cache misses.

00								
01			1	3		2	4 (kills 2)	6
10		5 8	10 (kills 8)	12			7	9
11		11						

d) All 12 accesses = 7 cache misses.

For e) and f) index in bracket, bold = miss

1st Iteration		2nd Iteration		3rd Iteration		4th Iteration	
Inst.	Data	Inst.	Data	Inst.	Data	Data	Inst. Data
0xEBD8 (01)		0xEBD8 (01)		0xEBD8 (01)		0xEBD8 (01)	
0xEBDC (01)		0xEBDC (01)		0xEBDC (01)		0xEBDC (01)	
0xEBE0 (10)	0xAA18 (01)	0xEBE0 (10)	0xAA58 (01)	0xEBE0 (10)	0xAA28 (10)	0xEBE0 (10)	0xAA68 (10)
0xEBE4 (10)		0xEBE4 (10)		0xEBE4 (10)		0xEBE4 (10)	
0xFA10 (01)	0xBB94 (01)	0xFA10 (01)	0xBBA4 (10)	0xFA10 (01)	0xBBA4 (10)	0xFA10 (01)	0xBBB4 (11)
0xFA14 (01)		0xFA14 (01)		0xFA14 (01)		0xFA14 (01)	
0xEBE0 (10)	0xAA1C (01)	0xEBE0 (10)	0xAA5C (01)	0xEBE0 (10)	0xAA2C (10)	0xEBE0 (10)	0xAA6C (10)
0xEBE4 (10)		0xEBE4 (10)		0xEBE4 (10)		0xEBE4 (10)	

e) First iteration = 7 misses

f) All iterations = 28 misses