

**Answer**

NATIONAL UNIVERSITY OF SINGAPORE

**CS2100 – COMPUTER ORGANISATION**

(Semester 1: AY2019/20)

Time Allowed: 2 Hours

**INSTRUCTIONS TO CANDIDATES**

1. This question paper contains **FOURTEEN (14)** questions and comprises **TWELVE (12)** printed pages.
2. You are provided with:
  - a) An **Answer Booklet**, comprising **FOUR (4)** printed pages
  - b) An **OCR form**
3. Please write your Student Number on odd-numbered pages of the **ANSWER BOOKLET** provided **with a PEN**. Do not write your name.
4. **Shade and write** your Student Number on the OCR form with **at least 2B Pencil**.
5. Answer Question 1 to 10 by shading on the OCR Form with **at least 2B pencil**.
6. Answer Question 11 to 14 within the space provided on the Answer Sheet. You may write in pencil.
7. Submit only the Answer Booklet and the OCR form at the end of the test.
8. Maximum score is **100 marks**.
9. This is a **CLOSED BOOK** test. However, two sheets of double-sided A4 reference sheet is allowed.
10. Calculators and computing devices such as laptops and PDAs are not allowed.
11. The last three pages are for your rough work. They contain blank, K-maps and state table for your use.

**Questions 1 - 5:** Each question has only one correct answer. Write your answers in the boxes provided in the Answer Booklet. **Four marks** are awarded for a correct answer and no penalty for wrong answer.

1. If we have two processor chips P1 and P2 supporting the same ISA, with clock frequency 1GHz and 2GHz respectively, which of the following statement is TRUE?

- A. The same binary (executable) will run exactly twice as fast on P2.
- B. The CPI of P1 and P2 must be the same.
- C. P2 will executes half as many instructions as P1 for the same binary.
- D. The cycle time on P1 is 50% shorter than that of P2.
- E. **None of the above.**

**Commented [SYJ1]:** A and B are essentially talking about the same thing. P2 can use different implementation that changes the CPI (e.g. faster multiplication, faster add, etc).  
C is just random gibberish ☹️  
D is the reverse.

2. If we perform the following steps on a machine:

Register R = 0x1122 3344  
Store R into memory location at 0x4000  
Load byte at 0x4003 into register S

- A. If S contains 0x44, then this machine is a **little endian machine**.
- B. If S contains 0x4400 0000, then this machine is a **little endian machine**.
- C. If S contains 0x1100 0000, then this machine is a **big endian machine**.
- D. These steps give the same result in register S on **all machines**.
- E. **None of the above.**

**Commented [SYJ2]:** Big endian => S = 0x44  
Little endian => S = 0x11

3. Given the following MIPS code, what is the result produced in register \$t2 in iteration 3?

```
#Mem[0x1000] = 55 (decimal)
#Mem[0x1004] = 34 (decimal)
#$s0 = 0x1008
S:      #start of 1 iteration
lw  $t0, -4($s0)
lw  $t1, -8($s0)
sub  $t2, $t1, $0    #result produced
sw  $t2, 0($s0)
addi $s0, $s0, 4
...<stop condition omitted>...
j  S      #end of 1 iteration
```

- A. \$t2 = -55
- B. **\$t2 = 8**
- C. \$t2 = 13
- D. \$t2 = 21
- E. None of the above.

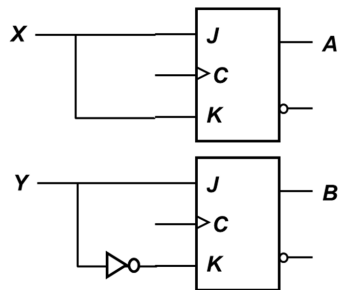
**Commented [SYJ3]:** Reverse Fibonacci,  
55, 34, 21, 13, 8  
So, 8 is the result in 3<sup>rd</sup> round.

4. Which of the following simplification step is a CORRECT application of the **consensus theorem**?

- A.  $A.B.C + A'.B'.D + C.D \Rightarrow A.B.C + A'.B'.D$   
 B.  $L'.M' + K'.L' + K.M' \Rightarrow K'.L' + K.M'$   
 C.  $X.Y + X'.Z' + Y.Z \Rightarrow X.Y' + X'.Z$   
 D.  $P'.Q' + P.R + Q'.R' \Rightarrow P'.Q' + P.R'$   
 E. None of the above.

Commented [SYJ4]: Ans

5. The follow sequential circuit generates the state AB as  $00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow 00 \rightarrow \dots$  (repeat). What is the correct Boolean expression for X and Y?



- A.  $X = A ; Y = A'$   
 B.  $X = A' ; Y = A$   
 C.  $X = B ; Y = B'$   
 D.  $X = B' ; Y = B$   
 E. None of the above.

Commented [SYJ5]: Effective make JA, KA a TA and JB, KB a DB

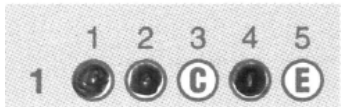
So, TA (X) = B; DB(Y) = B'

**Working:**

A	B	A+	B+	TA	DB
0	0	0	1	0	1
0	1	1	0	1	0
1	0	1	1	0	1
1	1	0	0	1	0

**Questions 6 – 10:** Each multiple response question has four options. Shade ALL options that you think are applicable to the question. [0-1 correct option = 0 mark, 2 correct options = 2 marks, 3 correct options = 3 marks, 4 correct options = 4 marks.]

Example:

1. Which number is a <b>prime number</b> ? A. 2 B. 3 C. 4 D. 5	Correct Answer: (A, B and D) 
--	---

6. Suppose the processor accessing the memory address **0xFEDCBA** triggers a cache miss that brings in memory block at **0xFEDCA0** to cache index **0x25**. We can deduce the following information about the cache setup:

- A. The tag of this memory block is 0x1FDB.
- B. The offset used on this machine is 4 bits.
- C. There are 32 total cache blocks.
- D. The tag of this memory block is 0xFED1.

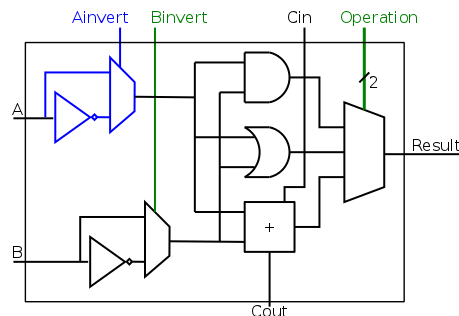
**Commented [SYJ6]:** 0xFEDCBA =  
1111 1110 1101 1100 1011 1010

0xFEDCA0 =  
1111 1110 1101 1100 1010 0000 (so offset = 5 bit)

Sidx = 0x25 = 10 0101 (so index = 6 bits)

So, the remaining bits (13 bits is the tag):  
1111 1110 1101 1  
= 1 1111 1101 1011

7. Which of the following statement is correct for the 32-bit ALU implementation? A 1-bit portion is shown below for your reference.



- a. For the 1-Bit ALU at the 0<sup>th</sup> position, i.e. for input A<sub>0</sub> and B<sub>0</sub>, the C<sub>in</sub> can be linked to Op<sub>1</sub> (i.e. using the same signal).
- b. The addition logic used in this implementation is the look-ahead adder.
- c. The NAND operation can be performed with the existing construction, but with new control signals.
- d. The XOR operation can be performed with the existing construction, but with new control signals.

**Commented [SYJ7]:** (a) False, Should be linked to Binvert. Counter example, add's Op is 10, which will add a spurious '1'.  
(b) False, a ripple-carry adder is used actually.  
(c) True, can invert both A, B and pass through OR  
(d) False. If FA is used for the XOR operation, the cin→cout connection between positions need to be severed.

8. Which of the following statements regarding Cache behaviour is **CORRECT**?

- A. Changing the cache block size can help to reduce the number of cold misses.
- B. N-Way Set-Associative cache with M cache Sets needs **M** number of cache tag comparator to detect cache hit/miss in 1 step. [Cache tag comparator compares the cache block tag with the tag of memory address]
- C. Fully associative cache with M total cache blocks needs **M** number of cache tag comparators to detect cache hit/miss in 1 step
- D. If we executes a statement like " $X[1] = X[1] + 5$ ", the "Write-Miss" policy is not applicable.

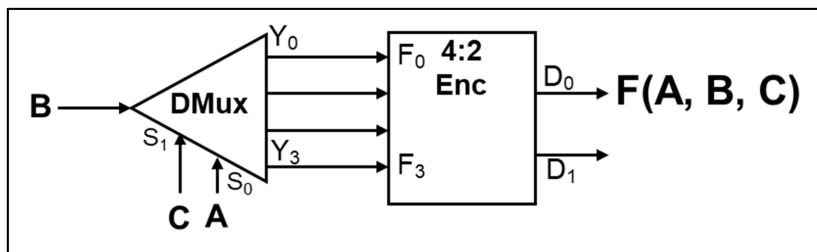
**Commented [SYJ8]:** (e) True  
 (f) Should be N to compare across all N cache block in the same set at once.  
 (g) True  
 (h) True,  $X[1] + 5$  will ensure  $X[1]$  is loaded as it is a read. Then the store for the assignment will not miss.

9. Which of the following statements regarding Control Dependency is **CORRECT**?

- A. Branch prediction and Delayed branch is similar in the sense that the instructions after the branch is fetched without waiting for branch outcome.
- B. The instructions in the delayed slot under Delayed Branch is never flushed.
- C. The early branching mechanism can utilize existing data forwarding paths implemented for RAW hazard.
- D. Only Branch prediction requires the help of compiler, i.e. compiler needs to be aware of this mechanism and generate the code accordingly.

**Commented [SYJ9]:** (a) True  
 (b) True  
 (c) False  
 (d) False

10. Given the following odd construction:



- A. Sum-of-Minterm for F includes **m7**. (i.e. m7 is one of the minterm(s))
- B. Product of Maxterm for F includes **M4**.
- C. Changing the 4:2 encoder to priority encoder will change the behaviour of F.
- D. F **cannot** be implemented with less than two 2-input standard logic gates (NOT, AND, OR) if **only non-complemented literals** are given.

**Commented [SYJ10]:**  $F = D0 \text{ (encoder)} = F1 + F3$

So,  $Y1 + Y3$  must be activated  
 $Y1 = B.C'.A = A.B.C'$   
 $Y3 = B.C.A = A.B.C$

For (a), (b)  
 $F = m6 + m7 = PM(0-5)$

(c), since DMux activate one line only, there wont be more than one 1 among  $F0-F3 \rightarrow$  no change.

(d)  
 F can be simplified to  $A.B$ , i.e. a single 2-input AND gate.

Only (a) and (b) are true.

## 11. [16 marks]

Design a sequential circuit that **cycles** through **Fibonacci numbers between 0 to 7**, i.e. **1, 2, 3 and 5**.

As control, the circuit takes a 1-bit input **X**. When **X = 0**, the circuit **cycles** through the Fibonacci number normally, i.e.

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 1 \dots \text{(repeats)}$$

When **X = 1**, the circuit cycles skip through those selected Fibonacci numbers **by 3 steps at a time**, i.e.

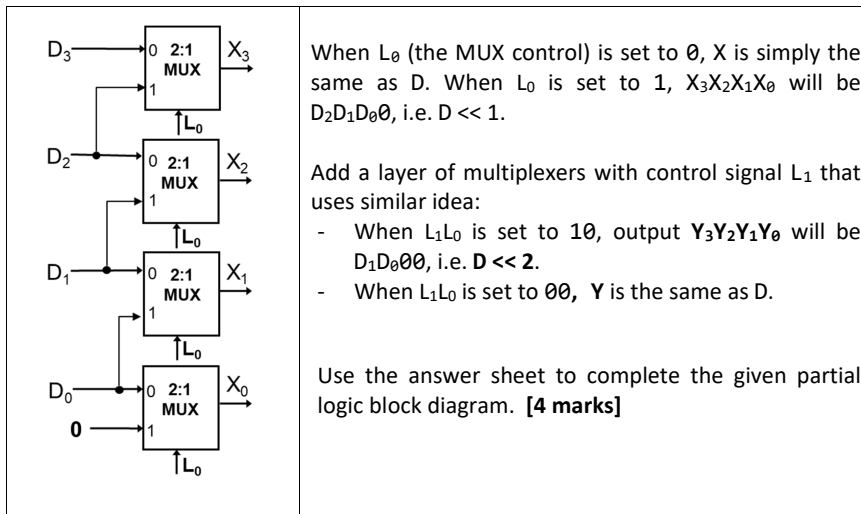
$$1 \rightarrow 5 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow \dots \text{(repeats)}$$

The states are represented by 3-bit values **ABC**. Implement the sequential circuit using a **T flip-flop for A**, a **JK flip-flop for B**, and a **D flip-flop for C**. Use the **partially filled state table given at the end of this paper to help**.

- Write out the **simplified SOP expressions** for the flip-flop input **TA** and **DC**. [4 marks]
- Give the **simplified POS** expression for **KB**. [2 marks]
- Implement **TA** using a **2x4 Decoder with active-low output**. The selector lines have been fixed to  $S_1 = X$  and  $S_0 = B$ , use **no more than three 2-input gates** (AND, OR, NAND, NOR). Only non-prime literals and constants (0, 1) are available. [4 marks]
- If the sequential circuit are implemented with **simplified SOP expression for all flip-flops inputs**, give any two unused states **when X = 1** and the state they transit to. Use the table in answer sheet to fill in your answers. [4 marks]
- Actually, the Fibonacci sequence should be 1, 1, 2, 3, 5, ..... (i.e. there should be another '1' in between of '1' and '2'). Unfortunately, we **cannot** use the setup given in this question to implement the correct sequence. Briefly explain why. [2 marks]

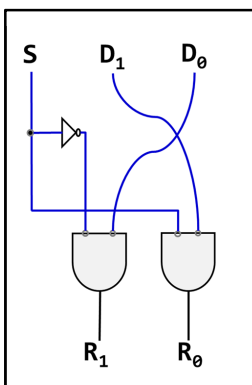
12. [14 marks] This question tackles the idea of **bit-shifting**. Please note that part (a) – (c) present one idea, while part (d) – (f) present another. Take time to read the context carefully.

a. Observe the interesting structure below:



- b. Following from (a), what should we do if we need to shift D by 3 places, i.e.  $D \ll 3$ ? [1 mark]
- c. Suppose we generalize this structure for 32 bits of input. If we add a **third layer using multiplexers with control  $L_2$** , what should be the behaviour of  $L_2L_1L_0 = 100$  (i.e. only third layer is activated) to give the most flexibility? Briefly explain. [3 marks]

----- Note: New Context for (d) to (e) -----



- d. The simple circuit on the left handles shifting of a **2-bit** input  $D_1D_0$  **to the left OR to the right by 1-bit**. The control S decide the direction. When  $S = 0$ , the input is shifted to the **left**; When  $S = 1$ , the input is shifted to the **right**.

Express the output bits  $R_1, R_0$  as Boolean expression. [2 marks]

- e. Suppose we want to upgrade the 2-bit circuit in (d) so that we can support an additional "no-shift" operation (i.e.  $R_1R_0$  is simply equal to  $D_1D_0$ ). As there are now 3 operations to support, we change the control signal  $S$  to a two-bit value  $S_1S_0$ . When  $S = 00$  (left shift);  $S = 11$  (right shift);  $S = 01$  or  $10$  (NO shift). Study and complete the partial diagram in the **ANSWER SHEET**. You are allowed to add no more than **three 2-input OR gates**. [4 marks]

13. [12 marks] As we know, the performance of program execution is governed by this formula:

$$\text{ExecutionTime} = \text{Instruction\_Number} * \text{Cycle\_Per\_Instruction} * \text{Cycle\_Time}$$

For each of the following change, indicate one factor (i.e. Instruction Number, CPI or Cycle Time) that is affected. Briefly explain in one sentence how is that factor affected. [Each part carries 3 marks, total = 4 x 3 marks = **12 marks**]

- Increase the CPU clock frequency.
- Use a better algorithm to solve the problem.
- Change the internal adder from parallel adder to look-ahead adder.
- Relocate the data in memory for better cache behaviour.



14. [18 marks] Below is a heavily simplified code:

```
for (i = 0; i < 36; i = i+1)
    ..... = A[i] * B[i];
```

Suppose A[] and B[] are both **double floating point arrays** (i.e. each element takes 8 bytes memory space) and A[0] is located at 0x20406080 and B[0] at 0xB0B0D0D0.

For part (a)-(e), we use a **Direct-Mapped cache with 128 bytes capacity with block size of 32 bytes**.

- a. Give the **cache index in hexadecimal** for the element A[0] and B[3]. [2 marks]
- b. Considering **only the accesses of A[] and B[]**, give the following metric for the **entire loop**:
  - i. The number of **cache miss** for accessing A[] elements. [3 marks]
  - ii. The number of **cache miss** for accessing B[] elements. [3 marks]
- c. Suppose the complete statement in the loop is:

```
result = A[i] * B[i];
```

- Give the **number of additional cache miss** the variable "result" can cause in the **worst case**. Also give the cache index of the variable "result" in order to cause such a scenario (you only need to give one answer if there are multiple answers). [3 marks]
- d. Move array B[] to a **new memory location that can cause a 100% cache miss-rate** with the original code. The new memory location **should be as close as possible to the original starting point of B[]**. Note that we consider only accesses of A[] and B[], i.e. just like part (b). Give your answer in hexadecimal. [2 marks]
  - e. Suppose the complete statement in the loop is actually: [3 marks]

```
C[i] = A[i] * B[i];
```

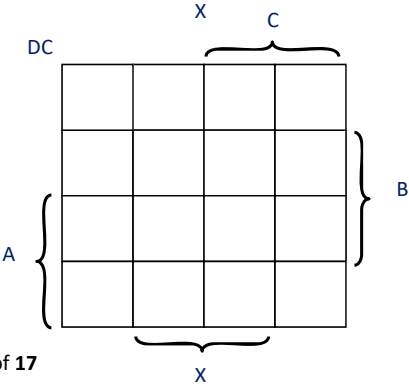
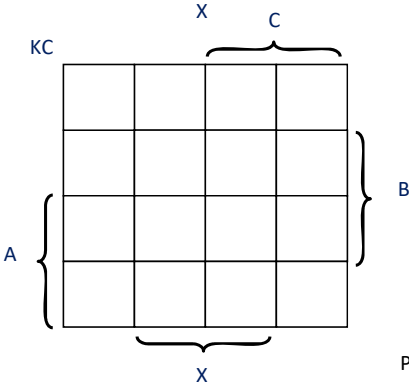
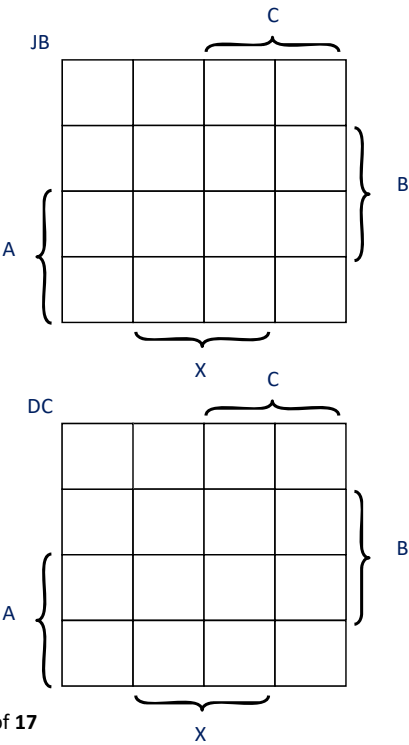
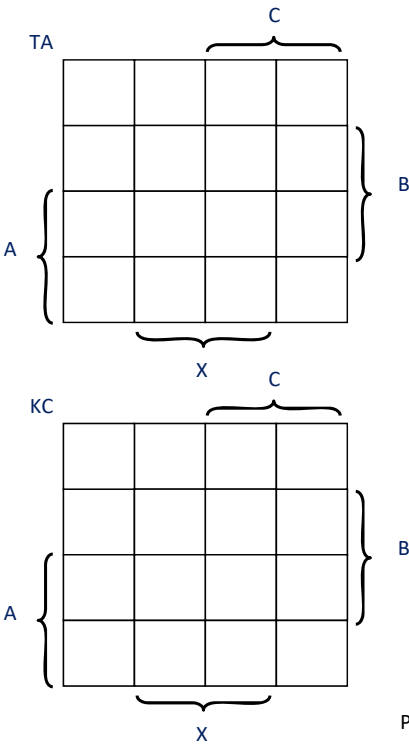
Which **write-miss policy** results in better cache behaviour? Briefly explain.

- f. If we consider only the accesses of A[] and B[], Ms.Titanica claims that with a **2-way Set-Associative cache of the same total size (i.e. total 128 bytes)**, this code will always give the best cache hit-rate **regardless of where A[] and B[] are placed in the memory**. Is she right? Briefly explain. [2 marks]

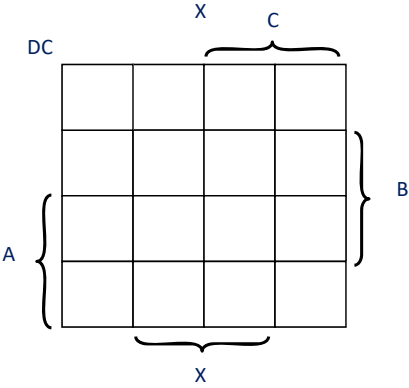
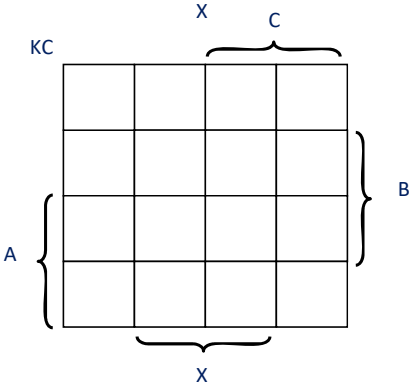
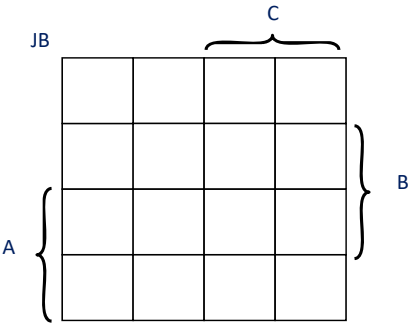
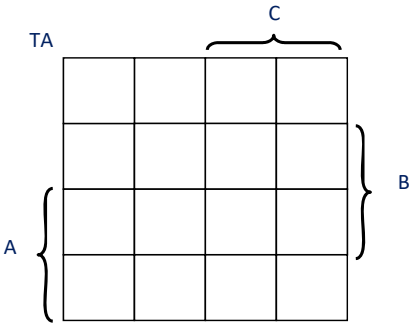
~~ END OF PAPER ~~

(Blank K-maps and partial state table are provided in next page.)

A	B	C	X	A+	B+	C+	TA	JB	KB	DC	
0	0	0	0	X	X	X					
0	0	0	1	X	X	X					
0	0	1	0	0	1	0					
0	0	1	1	1	0	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								



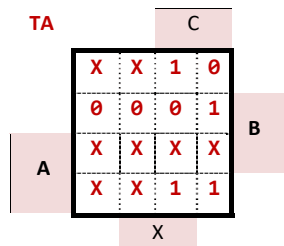
A	B	C	X	A+	B+	C+	TA	JB	KB	DC	
0	0	0	0	X	X	X					
0	0	0	1	X	X	X					
0	0	1	0	0	1	0					
0	0	1	1	1	0	1					
0	1	0	0								
0	1	0	1								
0	1	1	0								
0	1	1	1								
1	0	0	0								
1	0	0	1								
1	0	1	0								
1	0	1	1								
1	1	0	0								
1	1	0	1								
1	1	1	0								
1	1	1	1								



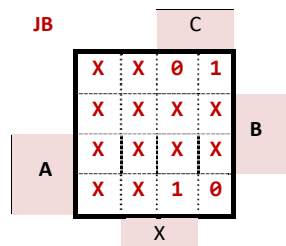
< Empty page for your rough work >

## -----ANSWER Break -----

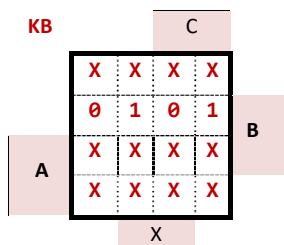
A	B	C	X	A+	B+	C+	TA	JB	KB	DC	
0	0	0	0	X	X	X	X	X	X	X	
0	0	0	1	X	X	X	X	X	X	X	
0	0	1	0	0	1	0	0	1	X	0	
0	0	1	1	1	0	1	1	0	X	1	
0	1	0	0	0	1	1	0	X	0	1	
0	1	0	1	0	0	1	0	X	1	1	
0	1	1	0	1	0	1	1	X	1	1	
0	1	1	1	0	1	0	0	X	0	0	
1	0	0	0	X	X	X	X	X	X	X	
1	0	0	1	X	X	X	X	X	X	X	
1	0	1	0	0	0	1	1	0	X	1	
1	0	1	1	0	1	1	1	1	X	1	
1	1	0	0	X	X	X	X	X	X	X	
1	1	0	1	X	X	X	X	X	X	X	
1	1	1	0	X	X	X	X	X	X	X	
1	1	1	1	X	X	X	X	X	X	X	



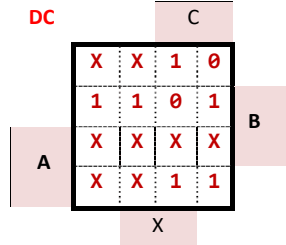
$$TA = A + B'.X + B.C.X'$$



$$JB = A'.X' + A.X$$



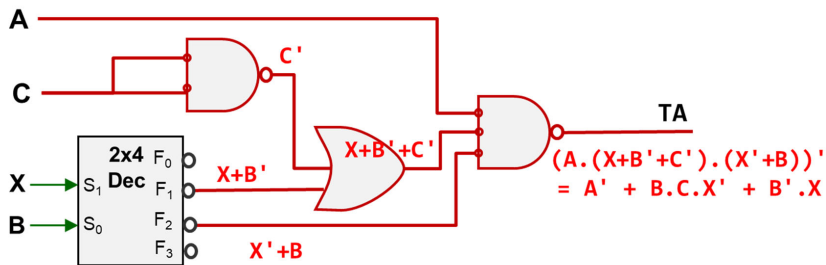
$$KB = C.X' + C.X$$



$$DC = A + C' + B'.X + B.X'$$

b. from K-map,  $KB' = C.X + C'.X' \rightarrow KB = (C' + X').(C + X)$

c.



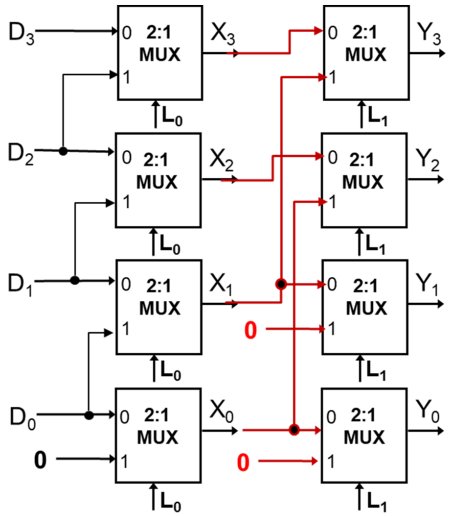
d. (yellow: When X = 0) (green: When X = 1), any two of the green answer will do.

A	B	C	X	A+	B+	C+
0	0	0	0	0	1	1
0	0	0	1	1	0	1
0	0	1	0	0	1	0
0	0	1	1	1	0	1
0	1	0	0	0	1	1
0	1	0	1	0	0	1
0	1	1	0	1	0	1
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	0	1	1
1	0	1	0	0	0	1
1	0	1	1	0	1	1
1	1	0	0	0	1	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	1

e. If the current state and external stimuli is the same, we will transit to the same next state. So, if there are two '1' state with different behaviour, we have to encode them differently, e.g. internally using a 4-bit state, to distinguish between the two '1's.

Q12.

a.



b.

Just set  $L_1L_0$  to 11

c.

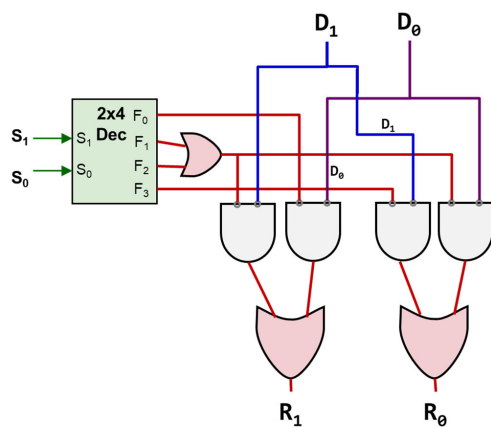
Should shift by 4, i.e.  $D \ll 4$ . This allows the 3 control bits  $L_2L_1L_0$  to provide shifting between 0 to 7 places.

d.

$$R_0 = S.D_1$$

$$R_1 = S'.D_0$$

e.



13. As we know, the

- a. Increase the CPU clock frequency.
- b. Use a better algorithm to solve the problem.
- c. Change the internal adder from parallel adder to look-ahead adder.
- d. Relocate the data in memory for better cache behaviour.

	Factor (Circle)	Brief Explanation
a.	<u>Inst#</u> CPI CycleTime	Higher frequency = shorter cycle time.
b.	<u>Inst#</u> <u>CPI</u> CycleTime	Better algorithm = lesser instructions
c.	Inst# <u>CPI</u> <u>CycleTime</u>	Better adder = less cycle for add = Lower CPI OR Better adder = lower cycle time for machine = lower Cycle Time
d.	Inst# <u>CPI</u> CycleTime	Better cache behaviour = lower cycle for memory instruction (avg) = lower CPI



Q14.

- a. Cache Idx for A[0] is 0, Cache idx for B[3] is 3
- b. i. A[], 9 misses (as A[0..35] spread across 9 blocks)  
ii. B[], 10 misses (as B[0..35] spread across 10 blocks)
- c. Cache Index = 0, As there are a total of 20 accesses for \$block 0, if "result" occupies the same block, it will cause all 20 to miss. Originally, there were 5 misses, so 15 extra misses.
- d. Just move B[0] to start at a block with \$idx 0, so 0xB0B0D100
- e. Write around. By avoiding C[] elements coming into cache, the original behaviour (which is as good as it gets) is preserved.
- f. Yes, she is right. As we only access A[i] and B[i] in order, as long as two cache blocks can co-exist, then the cache miss rate will be minimized.