

NATIONAL UNIVERSITY OF SINGAPORE**CS2100 – COMPUTER ORGANISATION**

(Semester 2: AY2020/21)

Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This assessment paper consists of **SIXTEEN (16)** questions (excluding question 0) in **THREE (3)** parts and comprises **ELEVEN (11)** printed pages.
2. Answer ALL questions.
3. This is an **OPEN BOOK** assessment.
4. The maximum mark of this assessment is 100.

Question	Max. mark
Q0	3
Part A: Q1 – 6	12
Part B: Q7 – 11	15
Part C: Q12	12
Part C: Q13	13
Part C: Q14	13
Part C: Q15	18
Part C: Q16	14
Total	100

5. You are to submit a single pdf file (size $\leq 20\text{MB}$) to your submission folder on LumiNUS.
6. Your submitted file should be named after your Student Number (eg: A1234567X.pdf) and your Student Number should be written on the first page of your file.
7. Do NOT write your name anywhere in your submitted file.

----- END OF INSTRUCTIONS -----

You do not need to write any answer for question 0.

0. (a) Submit a **single pdf file** into the submission folder. [1 mark]
 (b) Name your pdf file with your Student Number (eg: A1234567X.pdf). [1 mark]
 (c) Write your Student Number (not your name!) on the first page of your file. [1 mark]

Part A: Multiple-Choice Questions [Total: 6×2=12 marks]

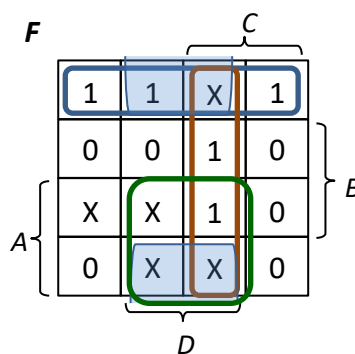
Each multiple-choice question (MCQ) is worth **TWO marks** and has exactly **one** correct answer. You may write your answers into the boxes in the Answer Sheets provided or if you are using your own paper, write your answers on a **single line** to conserve space. For example:

1. A 2. B 3. C 4. D ...

Please write in **CAPITAL LETTERS**.

1. Given the Boolean function $F(A,B,C,D) = \sum m(0,1,2,7,15) + \sum x(3,9,11,12,13)$ where x denotes don't-care, how many prime implicants (PIs) are there on the K-map of F ?
- A. 3
B. 4
 C. 5
 D. 6
 E. None of the above.
2. Given the same Boolean function in question 1 above, how many essential prime implicants (EPIs) are there on the K-map of F ?

- A. 0
 B. 1
C. 2
 D. 3
 E. None of the above.



PIs are $A'B'$, $C'D$, $B'D$, $A'D$. The EPIs are underlined.

3. A certain machine has 3 types of instructions: A , B and C , which have opcode of 3 bits, 5 bits, and 7 bits respectively. Assuming that each type must have at least one instruction, and the encoding space for opcode is completely utilized, what is the maximum total number of instructions using an expanding opcode scheme?
- A. 98
 B. 108
C. 110
 D. 120
 E. 128

A: 000
 B: 001 00
 C: 001 01 xx, 001 10 xx, 001 11 xx $\rightarrow 3 \times 4 = 12$
 $\{010 - 111\} \text{ xxxx} \rightarrow 6 \times 16 = 96$
 Total = 1 + 1 + (12 + 96) = **110**.

4. Study the MIPS code below. \$v0 contains a 7-bit value in its least significant bits; the rest of its bits are zeroes.

	addi \$s0, \$zero, 0
	addi \$t0, \$v0, 0
	addi \$t1, \$zero, 0
Loop:	andi \$t2, \$t0, 1
	xor \$t1, \$t1, \$t2
	addi \$s0, \$s0, 1
	slli \$t3, \$s0, 7
	beq \$t3, \$zero, out
	srl \$t0, \$t0, 1
	j Loop
Out:	sll \$t1, \$t1, 7
	or \$v0, \$v0, \$t1

What does the code do?

- A. Writes the number of 1's in the value in \$v0 to bit 7 of \$v0.
- B. Writes 0 to bit 0 of \$v0 if there are odd numbers of 1's in the value in \$v0, otherwise writes 1 to bit 0 of \$v0.
- C. Writes 0 to bit 0 of \$v0 if there are even numbers of 1's in the value in \$v0, otherwise writes 1 to bit 0 of \$v0.
- D. Writes 0 to bit 7 of \$v0 if there are odd numbers of 1's in the value in \$v0, otherwise writes 1 to bit 7 of \$v0.
- E. Writes 0 to bit 7 of \$v0 if there are even numbers of 1's in the value in \$v0, otherwise writes 1 to bit 7 of \$v0.**

The code generates even parity bit for \$v0.

5. How many instructions are executed by the code in question 4 when it completes?
- A. 12
 - B. 47
 - C. 52**
 - D. 54
 - E. 59

Working: $3 + (6 \times 7) + 5 + 2 = 52$

6. Given the Boolean expression below:

$$A \cdot B' \cdot (C + D' + E)' \cdot F + (C' + F')' + D \cdot F$$

Which of the following is equivalent to the above expression?

- A. F
- B. $D \cdot F$
- C. $(C' + D') \cdot F$
- D. $(C + D) \cdot F$**
- E. None of the above.

Part B: Multiple-Response Questions [Total: 5×3=15 marks]

Each multiple-response question (MRQ) is worth **THREE marks** and may have one answer or multiple answers. Write out **all** correct answers. For example, if you think that A, B, C are the correct answers, write A, B, C.

Only if you get all the answers correct will you be awarded three marks. **No partial credit will be given for partially correct answers.**

You may write your answers into the boxes in the Answer Sheets provided or if you are using your own paper, write your answers on a **single line** to conserve space. For example:

7. A,B 8. B,D 9. C 10. A,B,C,D 11. B,D,C

Please write in **CAPITAL LETTERS**.

7. Yucan wrote the value **66512** in his CS2100 exam, but did not specify the base. Choose from the list below all possible interpretations of this value.
- A. The 5-digit 7's complement of 155₇.**
 - B. The 5-digit 6's complement of 33156₆.
 - C. 66512₁₀.**
 - D. -33488₁₀ in 5-digit 10's complement of base 10.**
 - E. -22377₁₀ in 5-digit 9's complement of base 9.**

Answer: A,C,D,E

8. Which of the following statements are true about the MIPS datapath?
- A. The register file can only read one register at a time.
 - B. The register file can only write to one register at a time.**
 - C. A jump instruction (j) may not always be possible to jump to another instruction that is 16 instructions away from it.**
 - D. The target address of a branch instruction is given by PC + Immed x 4.
 - E. In the pipelined system, the PC contains the address of the instruction that is currently in the WB stage.

Answer: B,C

9. Which of the following MIPS instructions may be used to set the value of register **\$t2** to zero?

- A. **and \$t2, \$zero, \$zero**
- B. **andi \$t2, \$t2, 0**
- C. **xor \$t2, \$t2, \$t2**
- D. xori \$t2, \$t2, 0
- E. nor \$t2, \$zero, \$zero

Answer: A,B,C

10. If the datapath control generates the wrong signal for **MemToReg**, that is, it generates 0 when it should be 1, and 1 when it should be 0, which of the following instructions will be wrongly executed?

- A. bne
- B. sw
- C. **lw**
- D. **add**
- E. **ori**

Answer: C,D,E

11. Assuming that logical constants 0 and 1 are available but complemented literals are not, which of the following functions can be implemented using a single **4-bit magnitude comparator** with no additional logic gates?

- A. **$F1(A,B,C,D) = A \cdot B \cdot C$**
- B. **$F2(A,B,C,D) = \Sigma m(8,10,12,14)$**
- C. **$F3(A,B,C,D) = \Sigma m(0,15)$**
- D. $F4(A,B,C,D) = \Sigma m(1,2,3,4)$

Answer: A,B,C

A: 0ABC = 0111.

B: 00AD = 0010.

C: 0AAA = 0BCD.

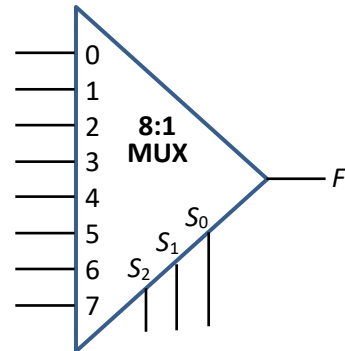
Q13. Combinational circuits [13 marks]

Note that logical constants 0 and 1 are available, but not complemented literals.

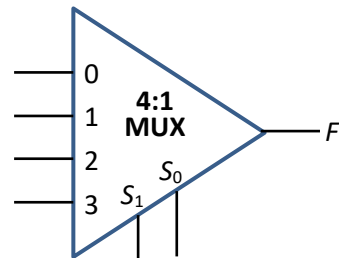
- (a) Given the following Boolean function

$$F(A,B,C,D) = \sum m(4, 6, 7, 9, 11, 12, 14, 15),$$

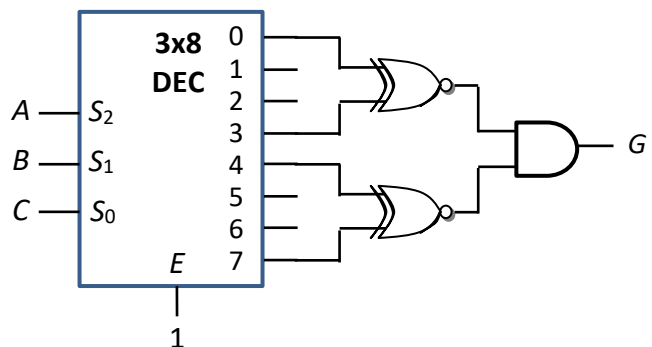
implement the function using a single 8:1 multiplexer as shown on the right without any additional logic gates. Once you have used a variable on a selector line, you should not use the same variable on the multiplexer inputs. [2]



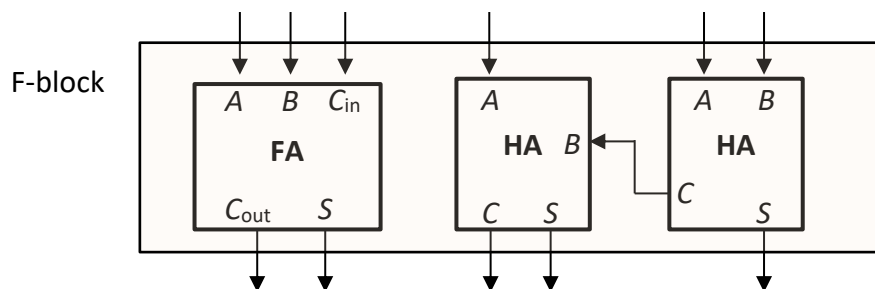
- (b) Given the same Boolean function in part (a) above, implement the function using a single 4:1 multiplexer as shown on the right without any additional logic gates. Once you have used a variable on a selector line, you should not use the same variable on the multiplexer inputs. [3]



- (c) A Boolean function $G(A,B,C)$ is implemented using a 3x8 decoder with 1-enable and active high outputs, 2 XNOR gates and an AND gate as shown on the right. Replace this circuit with a single logic gate. [4]



- (d) The F-block contains two half adders and a full adder as shown below. Using a single F-block, without any additional logic gates, design a circuit that takes in a 3-bit unsigned binary number $X = X_2X_1X_0$ and a one-bit value y to generate the output $Z_3Z_2Z_1Z_0$ which is a binary number representing the value $X+5y$. You may only connect inputs to the 6 inwards arrows and use the 5 outwards arrows for your outputs. [4 marks]



Q14. MIPS [13 marks]

Study the following MIPS code on integer arrays *A* and *B* which contain the same number of elements. The following are the variable mappings:

- $\$s0$ = base address of array *A*
- $\$s1$ = base address of array *B*
- $\$s2$ = *size* (number of elements in array *A*)
- $\$s5$ = *count*

```
.data
A: .word 10,21,12,17,9,1,20,33 # after: 10,18,12,2,7,-1,20,22
B: .word 100,3,20,15,2,2,65,11
size: .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
       # Box 1

# -----
      add    $s5, $0, $0      # I1
      add    $t0, $0, $0      # I2
loop: slt    $t8, $t0, $s2    # I3
      beq    $t8, $0, end      # I4
      sll    $t1, $t0, 2       # I5
      add    $t3, $t1, $s0     # I6
      lw     $s3, 0($t3)       # I7
      andi   $t9, $s3, 1       # I8
      beq    $t9, $0, skip     # I9
      add    $t4, $t1, $s1     # I10
      lw     $s4, 0($t4)       # I11
      sub    $s3, $s3, $s4     # I12
      sw     $s3, 0($t3)       # I13
      addi   $s5, $s5, 1       # I14
skip: addi   $t0, $t0, 1       # I15
      j      loop             # I16

# -----
end:   # Box 2

      syscall                # system call to print
      li     $v0, 10
      syscall                # system call to exit
```


Q14. (continue...)

Note that pseudo-instructions **la** and **li** are allowed, but not other pseudo-instructions.

- (a) Fill in Box 1 with MIPS instruction(s) to load the value of *size* into **\$s2**. [2]
- (b) Fill in Box 2 with MIPS instruction(s) to prepare for the printing of the value of *count* (**\$s5**). [2]
- (c) Using the variable names (*A*, *B*, *size*, *count*) shown in the variable mappings above, write an equivalent C code that corresponds to instructions I1 to I16 in the above MIPS code. You may use additional variable(s) if needed. You do not need to declare the variables in your C code. [3]
- (d) Write the instruction encoding of instruction I3 (`slt $t8, $t0, $s2`). Write your answer in hexadecimal. [2]
- (e) Write the instruction encoding of instruction I9 (`beq $t9, $0, skip`). Write your answer in hexadecimal. [2]
- (f) Write the instruction encoding of instruction I16 (`j loop`), assuming that instruction I1 (`add $s5, $0, $0`) is stored at address 0x10000C50. Write your answer in hexadecimal. [2]

Q15. Cache [18 marks]

Refer to the following MIPS code which is the same as the one in question 14. Here, we only look at instructions I1 to I16.

```

      add  $s5, $0, $0      # I1
      add  $t0, $0, $0      # I2
loop: slt  $t8, $t0, $s2    # I3
      beq  $t8, $0, end     # I4
      sll  $t1, $t0, 2      # I5
      add  $t3, $t1, $s0    # I6
      lw   $s3, 0($t3)      # I7
      andi $t9, $s3, 1      # I8
      beq  $t9, $0, skip    # I9
      add  $t4, $t1, $s1    # I10
      lw   $s4, 0($t4)      # I11
      sub  $s3, $s3, $s4    # I12
      sw   $s3, 0($t3)      # I13
      addi $s5, $s5, 1      # I14
skip: addi $t0, $t0, 1      # I15
      j    loop            # I16
end:

```

Q15. (continue...)

For parts (a) to (d): You are given a **direct-mapped data cache** with 128 words in total and each block contains 4 words. You may assume the following:

- The data cache is used for **lw** instructions but not for **sw** instructions.
- Array *A* starts at address **0xFF000C00** and all elements in array *A* are positive odd integers.
- Array *B* follows immediately after array *A* in the memory. That is, if the last element of array *A* is at address x , then the first element of array *B* is at address $x + 4$.

- (a) How many bits are there in the index field? In the byte offset field? [2]
- (b) Given that array *A* contains **3200** elements and array *B* contains the same number of elements, what is the hit rate of the data cache (i) for array *A* and (ii) for array *B*? Note that you need to consider only the **lw** instructions but not the **sw** instructions. [2]
- (c) Given that array *A* contains **3216** elements and array *B* contains the same number of elements, what is the hit rate of the data cache (i) for array *A* and (ii) for array *B*? Note that you need to consider only the **lw** instructions but not the **sw** instructions. [4]
- (d) Given that array *A* contains **3210** elements and array *B* contains the same number of elements, what is the hit rate of the data cache (i) for array *A* and (ii) for array *B*? Note that you need to consider only the **lw** instructions but not the **sw** instructions. [4]

For parts (e) to (g): You are given a **2-way set-associative instruction cache** with 16 words in total and each block contains 2 words, with LRU replacement policy. You may assume the following:

- There are 100 elements in array *A* and the same number of elements in array *B*.
- All elements in array *A* are positive odd integers.

- (e) How many bits are there in the set index field? In the byte offset field? [2]
- (f) Assuming that instruction *I1* (add \$s5, \$0, \$0) is stored at address **0x10000C50**, how many hits in total are there in the instruction cache in the execution of the code? Consider only instructions *I1* to *I16*. [2]
- (g) Assuming that instruction *I1* (add \$s5, \$0, \$0) is stored at address **0x10000C54**, how many hits in total are there in the instruction cache in the execution of the code? Consider only instructions *I1* to *I16*. [2]

Q16. Pipelining [14 marks]

Refer to the following MIPS code which is the same as the one in question 14. Here, we only look at instructions I1 to I16.

```

      add  $s5, $0, $0      # I1
      add  $t0, $0, $0      # I2
loop: slt   $t8, $t0, $s2    # I3
      beq  $t8, $0, end      # I4
      sll  $t1, $t0, 2       # I5
      add  $t3, $t1, $s0     # I6
      lw   $s3, 0($t3)       # I7
      andi $t9, $s3, 1       # I8
      beq  $t9, $0, skip     # I9
      add  $t4, $t1, $s1     # I10
      lw   $s4, 0($t4)       # I11
      sub  $s3, $s3, $s4     # I12
      sw   $s3, 0($t3)       # I13
      addi $s5, $s5, 1       # I14
skip: addi $t0, $t0, 1       # I15
      j    loop              # I16
end:

```

Assuming a 5-stage MIPS pipeline and all elements in array *A* are positive odd integers, answer the following questions. You need to count until the last stage of instruction I16.

- (a) How many cycles does this code segment take to complete its execution in the first iteration (I1 to I16) in an ideal pipeline, that is, one with no delays? [2]

For parts (b) to (d) below, given the assumption for each part, how many additional cycles does this code segment (I1 to I16) take to complete its execution in the first iteration as compared to an ideal pipeline? (For example, if part (a) takes 12 cycles and part (b) takes 20 cycles, you are to answer part (b) with the value 8 and not 20.)

- (b) Assuming without forwarding and branch decision is made at MEM stage (stage 4). No branch prediction is made and no delayed branching is used. [3]
- (c) Assuming without forwarding and branch decision is made at ID stage (stage 2). No branch prediction is made and no delayed branching is used. [3]
- (d) Assuming with forwarding and branch decision is made at ID stage (stage 2). Branch prediction is made where the branch is predicted not taken, and no delayed branching is used. [3]
- (e) Assuming the setting in part (d) above and you are not allowed to modify any of the instructions, is it possible to reduce the additional delay cycles in part (d) by rearranging some instructions, and if possible, by how many cycles? Explain your answer. (Answer with no explanation will not be awarded any mark.) [3]

=== END OF PAPER ===

Internal:

The following MIPS program is given to students.

Here, the array size for arrays A and B is 8. In the exam, I will change it to larger numbers.

```
# $s0 = array A; $s1 = arrayB; $s2 = size; $s3 = A[i]; $s4 = B[i]
# $s5 = count
# for (i = 0 to size-1) {
#     if (A[i] is odd) {
#         A[i] = A[i] - B[i];
#         count = count + 1;
#     }
# }
.data
A: .word 10,21,12,17,9,1,20,33 # after: 10,18,12,2,7,-1,20,22
B: .word 100,3,20,15,2,2,65,11
size: .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    $t0, size        # $t0 is the address of size
      lw    $s2, 0($t0)      # $s2 is the content of size
# -----
      add    $s5, $0, $0     # I1; count = 0
      add    $t0, $0, $0     # I2; i = 0
loop:  slt    $t8, $t0, $s2   # I3; i < size?
      beq    $t8, $0, end     # I4
      sll    $t1, $t0, 2     # I5; 4i
      add    $t3, $t1, $s0    # I6; addr of A[i]
      lw     $s3, 0($t3)      # I7; $s3 = A[i]
      andi   $t9, $s3, 1     # I8; extract LSB of A[i]
      beq    $t9, $0, skip    # I9; if LSB of A[i] is 0, go to skip
      add    $t4, $t1, $s1    # I10; addr of B[i]
      lw     $s4, 0($t4)      # I11; $s4 = B[i]
      sub    $s3, $s3, $s4    # I12; }
      sw     $s3, 0($t3)      # I13; } A[i] = A[i] - B[i]
      addi   $s5, $s5, 1     # I14; count = count + 1
skip:  addi   $t0, $t0, 1     # I15; i = i + 1
      j      loop            # I16
# -----
end:   li     $v0, 1          # system call code for print_int
      add    $a0, $s5, $0     # transfer $s5 to $a0 for printing
      syscall                # print $s5
      li     $v0, 10         # system call code for exit
      syscall
```

The following diagrams show the contents of arrays A and B.

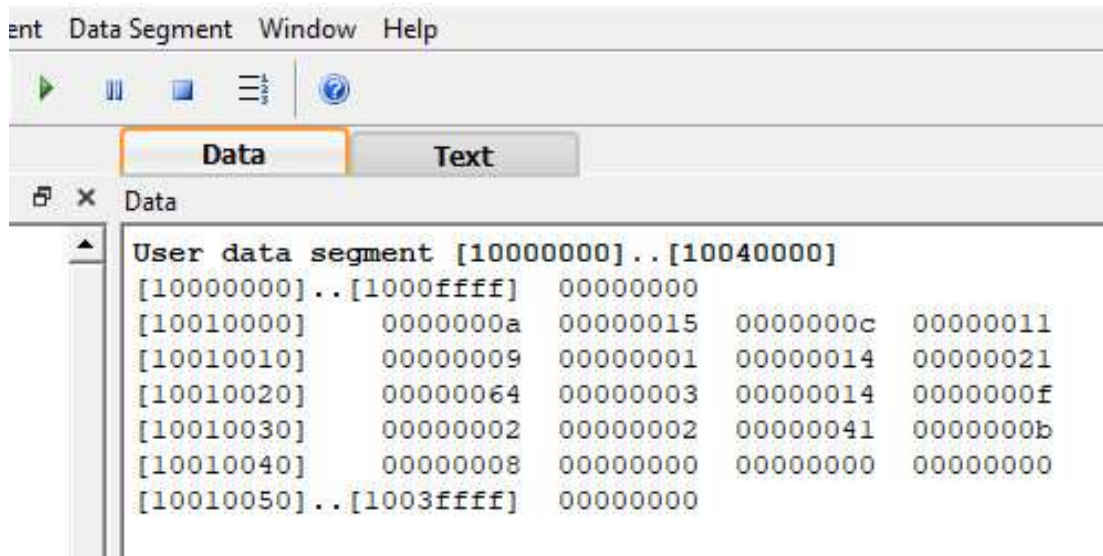
Array A (before): 10, 21, 12, 17, 9, 1, 20, 33

Array B: 100, 3, 20, 15, 2, 2, 65, 11

Array A (after): 10, 18, 12, 2, 7, -1, 20, 22

Data segment starts at address 0x10000000.

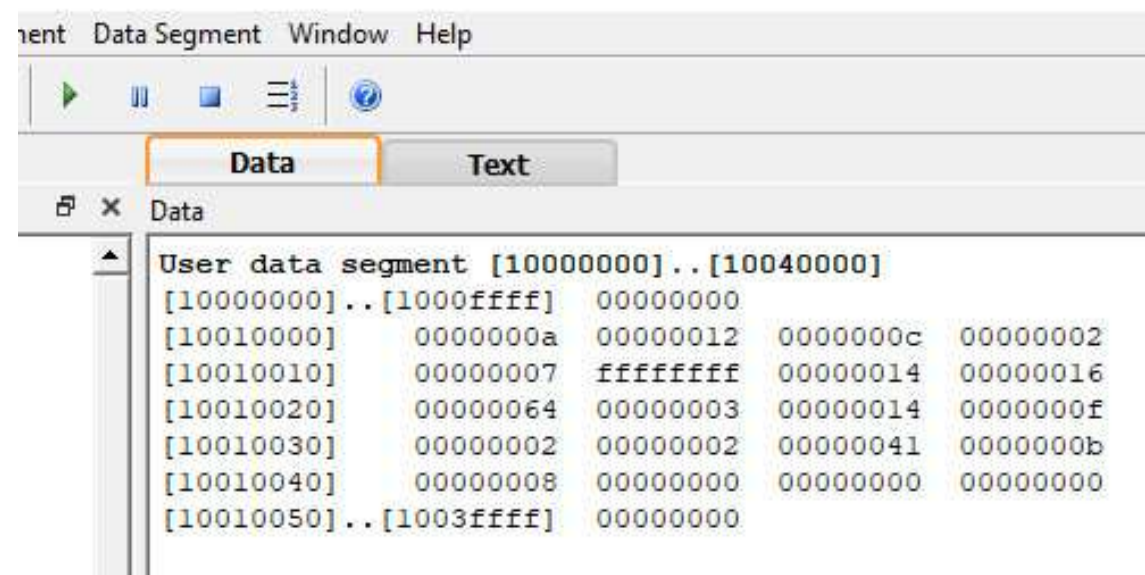
Before:



The screenshot shows the 'Data Segment' window with the 'Data' tab selected. The window title is 'Data'. The content displays the 'User data segment [10000000]..[10040000]'. The data is organized into columns representing memory addresses and their corresponding values. The values are mostly zeros, with some non-zero values in the second column.

Address	Value	Value	Value	Value
[10000000]..[1000ffff]	00000000			
[10010000]	0000000a	00000015	0000000c	00000011
[10010010]	00000009	00000001	00000014	00000021
[10010020]	00000064	00000003	00000014	0000000f
[10010030]	00000002	00000002	00000041	0000000b
[10010040]	00000008	00000000	00000000	00000000
[10010050]..[1003ffff]	00000000			

After:



The screenshot shows the 'Data Segment' window with the 'Data' tab selected. The window title is 'Data'. The content displays the 'User data segment [10000000]..[10040000]'. The data is organized into columns representing memory addresses and their corresponding values. The values are mostly zeros, with some non-zero values in the second column, including a large value 'ffffffff' at address [10010010].

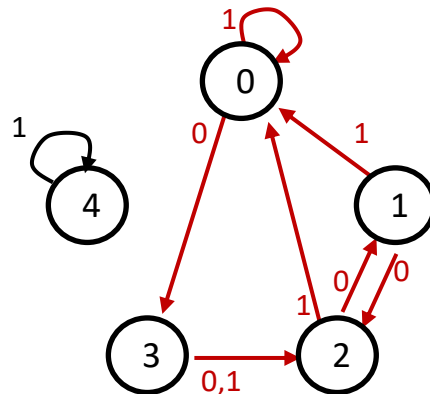
Address	Value	Value	Value	Value
[10000000]..[1000ffff]	00000000			
[10010000]	0000000a	00000012	0000000c	00000002
[10010010]	00000007	ffffffff	00000014	00000016
[10010020]	00000064	00000003	00000014	0000000f
[10010030]	00000002	00000002	00000041	0000000b
[10010040]	00000008	00000000	00000000	00000000
[10010050]..[1003ffff]	00000000			

Part C: Answers and Workings

Q12. Sequential Circuit (12 marks)

(a) $JA = Ax$; $KA = B$; $JB = x'$; $KB = C'$; $JC = x'$; $KC = 1$.

Present state			Input	Flip-flop A		Flip-flop B		Flip-flop C		Next state		
A	B	C	x	JA	KA	JB	KB	JC	KC	A ⁺	B ⁺	C ⁺
0	0	0	0	0	0	1	1	1	1	0	1	1
0	0	0	1	0	0	0	1	0	1	0	0	0
0	0	1	0	0	0	1	0	1	1	0	1	0
0	0	1	1	0	0	0	0	0	1	0	0	0
0	1	0	0	0	1	1	1	1	1	0	0	1
0	1	0	1	0	1	0	1	0	1	0	0	0
0	1	1	0	0	1	1	0	1	1	0	1	0
0	1	1	1	0	1	0	0	0	1	0	1	0
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	0	0	1	0	1	1	0	0
1	0	1	0	0	0	1	0	1	1	1	1	0
1	0	1	1	1	0	0	0	0	1	1	0	0
1	1	0	0	0	1	1	1	1	1	0	0	1
1	1	0	1	1	1	0	1	0	1	0	0	0
1	1	1	0	0	1	1	0	1	1	0	1	0
1	1	1	1	1	1	0	0	0	1	0	1	0



(b)

Present state		Input	Next State	
F	G	x	DF = F ⁺	DG = G ⁺
0	0	0	1	1
0	0	1	0	0
0	1	0	1	0
0	1	1	0	0
1	0	0	0	1
1	0	1	0	0
1	1	0	1	0
1	1	1	1	0

$$DF = F' \cdot x' + F \cdot G$$

$$DG = G' \cdot x'$$

DF

		G	
		0	1
F	1	0	1
	0	0	1
		x	

DG

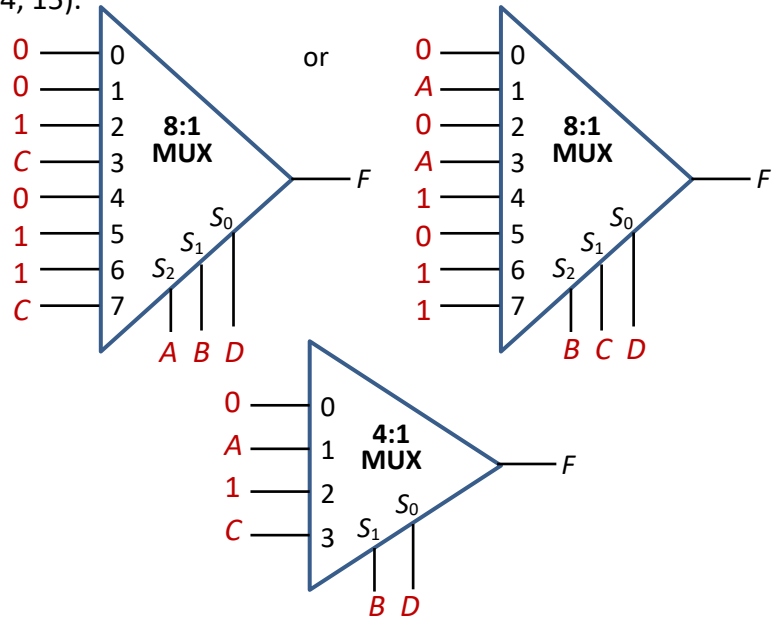
		G	
		0	0
F	1	0	0
	0	0	0
		x	

Q13. Combinational Circuits (13 marks)

(a) (2 marks)

$$F(A,B,C,D) = \sum m(4, 6, 7, 9, 11, 12, 14, 15).$$

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

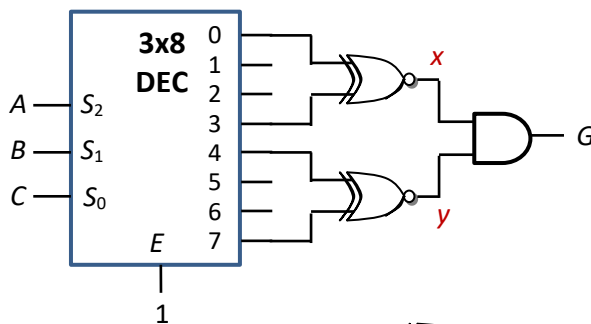


(b) (3 marks)

$$\text{From K-map, } F = B' \cdot D \cdot A + B \cdot D' + B \cdot C = B' \cdot D \cdot (A) + B \cdot D' \cdot (1) + B \cdot D \cdot (C)$$

(c) (4 marks)

Label the outputs from the XNOR gates as x and y.

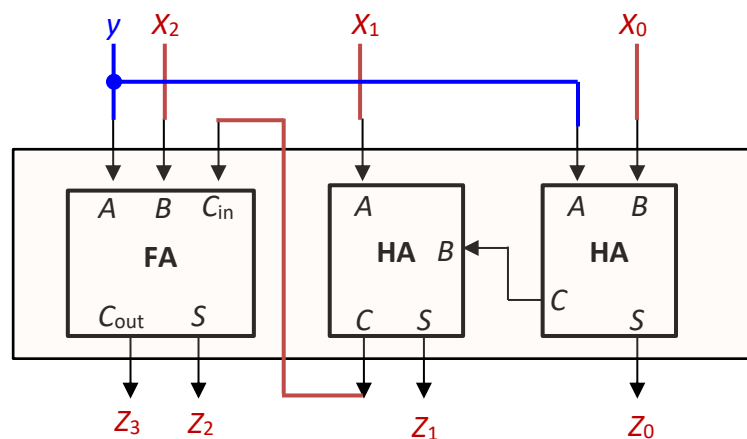


A	B	C	x	y	G
0	0	0	0	1	0
0	0	1	1	1	1
0	1	0	1	1	1
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	0	0

$$G = B \text{ XOR } C; G = B \oplus C$$



(d) (4 marks)



Q14. MIPS (13 marks)

(a) (2 marks)

```
la    $t0, size
lw    $s2, 0($t0)
```

(b) (2 marks)

```
li    $v0, 1
add   $a0, $s5, $0 # or addi $a0, $s5, 0
```

(c) (3 marks)

```
count = 0;
for (i = 0; i < size; i++) {
    if (A[i] % 2 == 1) { // or if (!(A[i] % 2))
        A[i] = A[i] - B[i];
        count = count + 1;
    }
}
```

(d) (2 marks) Answer: **0x0112C02A**

```
slt $t8, $t0, $s2
opcode = 0; funct = 0x2A = 0b101010;
rd = $t8 = $24 = 0b11000; rs = $t0 = $8 = 0b01000; rt = $s2 = $18 = 0b10010.
0b000000 01000 10010 11000 00000 101010 = 0x0112C02A
```

(e) (2 marks) Answer: **0x13200005**

```
beq $t9, $0, skip
opcode = 0x4 = 0b000100;
rs = $t9 = $25 = 0b11001; rt = $0 = 0b00000; skip = 5
0b000100 11001 00000 0000000000000101 = 0x13200005
```

(f) (2 marks) Answer: **0x08000316**

```
j loop
opcode = 0x2 = 0b000010;
Address at loop: 0x10000C58 = 0b 0001 0000 0000 0000 1100 0101 1000
0b000010 0001 0000 0000 0000 1100 0101 1000 = 0x08000316
```


Q15. Cache (18 marks)

- (a) (2 marks) Answers: index: **5**, byte offset: **4**.

$128/4 = 32$ blocks \rightarrow **5 bits** for index field; 16 bytes per block \rightarrow **4 bits** for byte offset.

- (b) (2 marks) Answers: **0%** for both arrays *A* and *B*.

As array *A* contains all positive odd integers, both **lw** instructions in the code are executed in each iteration.

Array *A* starts at $0xFF000C00 = 0b...110000000000$. Array *B* starts at $0xFF000C00 + (3200 \times 4) = 0xFF000C00 + 0x3200 = 0xFF003E00 = 0b...111000000000$.

A[0] and *B*[0] are both loaded into block 0 word 0. In fact, *A*[*i*] and *B*[*i*] ($0 < i < 3199$) are loaded into that cache at the same block same word, hence there will be cache thrashing, resulting in **zero hit rate** for both arrays *A* and *B*.

- (c) (4 marks) Answers: $\frac{3}{4}$ or **75%** for both arrays *A* and *B*.

Array *A* starts at $0xFF000C00$. Array *B* starts at $0xFF000C00 + (3216 \times 4) = 0xFF000C00 + 0x3240 = 0xFF003E40 = 0b...111001000000$.

A[0] is loaded into block 0 word 0 and *B*[0] into block 4 word 0. Hence for the first four iterations, *A*[0] will be a miss and *A*[1], *A*[2], *A*[3] are hits. Likewise for array *B*. This recurs for the rest of the array elements. Hence the hit rate for both arrays is $\frac{3}{4}$ or 75%.

- (d) (4 marks) Answers: **2407/3210** or **74.98%** for both arrays *A* and *B*.

Array *A* starts at $0xFF000C00$. Array *B* starts at $0xFF000C00 + (3210 \times 4) = 0xFF000C00 + 0x3228 = 0xFF003E28 = 0b...111000101000$.

A[0] is loaded into block 0 word 0 and *B*[0] into block 2 word 2.

The figure below shows the data cache for the first 8 iterations.

Block 0	<i>A</i> [0]	<i>A</i> [1]	<i>A</i> [2]	<i>A</i> [3]
Block 1	<i>A</i> [4]	<i>A</i> [5]	<i>A</i> [6]	<i>A</i> [7]
Block 2	<i>B</i> [0]	<i>B</i> [1]
Block 3	<i>B</i> [2]	<i>B</i> [3]	<i>B</i> [4]	<i>B</i> [5]
Block 4	<i>B</i> [6]	<i>B</i> [7]

:

The cache access pattern for array *A* is: (M,H,H,H), ..., M,H. The (M,H,H,H) is repeated 802 times. Therefore, hit rate = $2407/3210$ or 74.98%.

The cache access pattern for array *B* is: M,H,(M,H,H,H)... The (M,H,H,H) is repeated 802 times. Therefore, the hit rate is the same as array *A*.

(e) (2 marks) Answers: set index: **2**, byte offset: **3**.

$16/2/2 = 4$ blocks \rightarrow **2 bits** for set index field; 8 bytes per block \rightarrow **3 bits** for byte offset.

(f) (2 marks) Answer: **1396**

$0x10000C50 = 0b0001000..1100010**10000**$. So I1 is loaded into set 2 word 0.

Cache after the first iteration:

Set 0	I5	I6	I13	I14
Set 1	I7	I8	I15	I16
Set 2	I1	I2	I9	I10
Set 3	I3	I4	I11	I12

Hits in first iteration: 8.

Subsequent iterations, only I3 to I16 are executed

Hits in each of subsequent iterations: all $99 \times 14 = 1386$ hits.

Last two instructions I3 and I4 after last iteration: all 2 hits.

Therefore, total hits = $8 + 1386 + 2 = \mathbf{1396}$.

(g) (2 marks) Answer: **1395**

$0x10000C54 = 0b0001000..1100010**10100**$. So I1 is loaded into set 2 word 1.

Cache after the first iteration:

Set 0	I4	I5	I12	I13
Set 1	I6	I7	I14	I15
Set 2	I16	I1	I8	I9
Set 3	I2	I3	I10	I11

Hits in first iteration: 7.

Subsequent iterations, only I3 to I16 are executed

Hits in each of subsequent iterations: all $99 \times 14 = 1386$ hits.

Last two instructions I3 and I4 after last iteration: all 2 hits.

Therefore, total hits = $7 + 1386 + 2 = \mathbf{1395}$.

Q16. Pipelining (14 marks)(a) $16 + 5 - 1 = 20$ cycles. (2 marks)

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

(b) **+24** cycles. (3 marks)(c) **+20** cycles. (3 marks) (2 marks given if answer incorrect but = (b) – 4.)(d) **+4** cycles. (3 marks)

				(b)	(c)	(d)
	add	\$s5, \$0, \$0	# I1			
	add	\$t0, \$0, \$0	# I2			
loop:	slt	\$t8, \$t0, \$s2	# I3	+2	+2	
	beq	\$t8, \$0, end	# I4	+2	+2	+1
	sll	\$t1, \$t0, 2	# I5	+3	+1	
	add	\$t3, \$t1, \$s0	# I6	+2	+2	
	lw	\$s3, 0(\$t3)	# I7	+2	+2	
	andi	\$t9, \$s3, 1	# I8	+2	+2	+1
	beq	\$t9, \$0, skip	# I9	+2	+2	+1
	add	\$t4, \$t1, \$s1	# I10	+3	+1	
	lw	\$s4, 0(\$t4)	# I11	+2	+2	
	sub	\$s3, \$s3, \$s4	# I12	+2	+2	+1
	sw	\$s3, 0(\$t3)	# I13	+2	+2	
	addi	\$s5, \$s5, 1	# I14			
skip:	addi	\$t0, \$t0, 1	# I15			
	j	loop	# I16			
end:						
Total:				+24	+20	+4

(e) (3 marks) More than one possible answer. Example:

Move I14 (addi \$s5, \$s5, 1) to between I11 (lw \$s4, 0(\$t4)) and I12 (sub \$s3, \$s3, \$s4) to remove the 1 cycle of delay at I12.