CS2100 Computer Organisation
AY2021/22 Semester 1
Assignment 2

PLEASE READ THE FOLLOWING INSTRUCTIONS VERY CAREFULLY.

1. The **deadline** for this assignment is **18 October 2021, Monday, 1pm**. The submission folders will close shortly after 1pm and no submissions will be allowed after that. You WILL receive 0 mark if you do not submit on time.
2. You should complete this assignment **on your own** without collaborating or discussing with anyone. If you have been found to have cheated in this assignment, you will receive an F grade for this module as well as face other disciplinary sanctions. Take this warning seriously.
3. Please submit only **ONE pdf file** called **AxxxxxxxY.pdf**, where **AxxxxxxxY** is your student number.
4. Please keep your submission **short**. You only need to submit your answers; you do not need to include the questions.
5. Answers may be typed or handwritten. In case of the latter, please ensure that you use **dark ink** and your handwriting is **neat and legible**. Marks may be deducted for untidy work or illegible handwriting.
6. Upload your pdf file to LumiNUS > Files > Assignment 2 > (your tutorial group) > (your personal folder).
7. There are FIVE (5) questions on SIX (6) printed pages in this paper.
8. The questions are worth **40 marks** in total.
9. If you have any queries on this assignment, please post them on LumiNUS > Forum > Assignments. Queries posted elsewhere will not be answered.

=== END OF INSTRUCTIONS ===

Question 0. Submission instructions (3 marks)

You do not need to answer this question. Your tutor will award the marks accordingly.

(a) You have named your file with your student number, i.e., **AxxxxxxxY.pdf**. [1 mark]

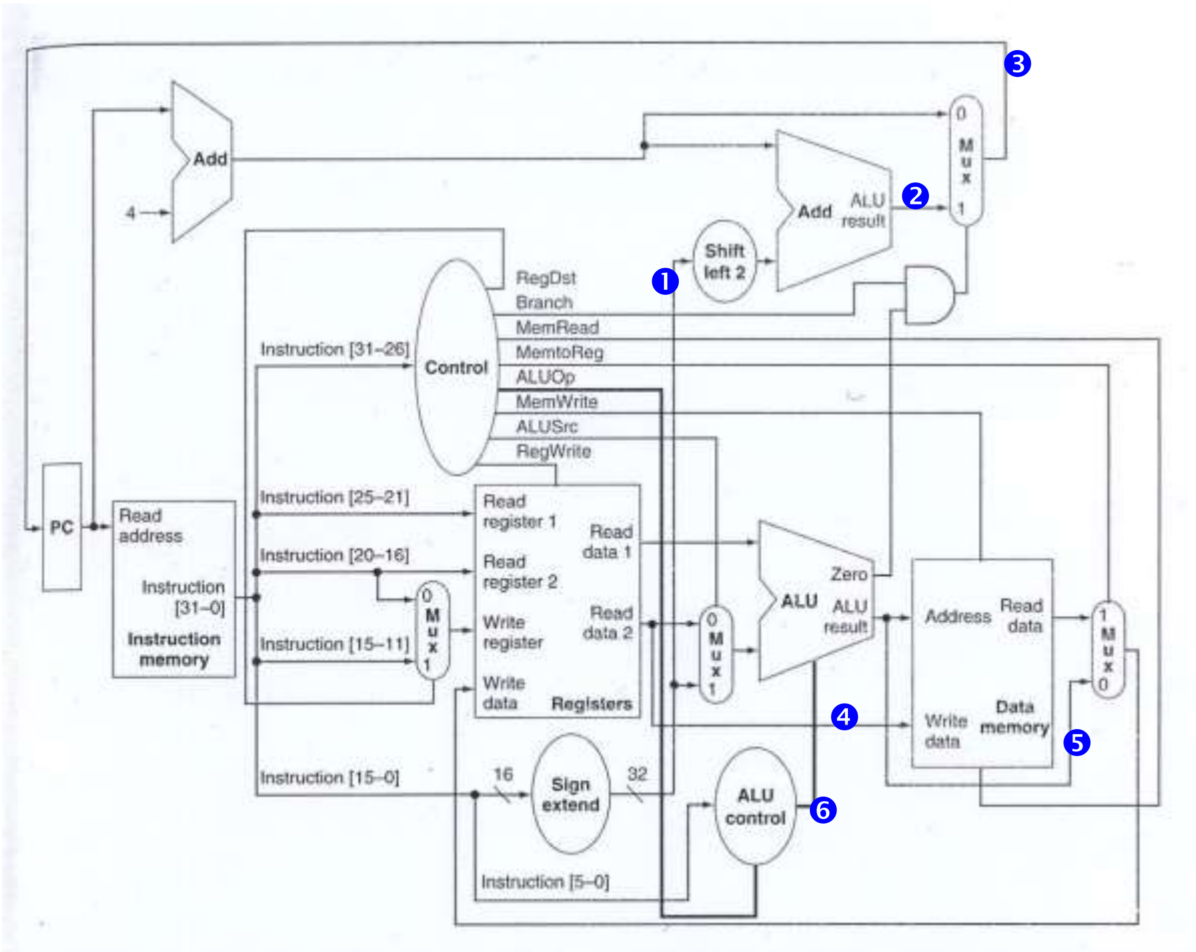
(b) You have submitted your assignment as a **single PDF file** and no multiple copies, i.e. there should only be ONE file in your submission folder. [1 mark]

(c) Your submission has your **tutorial group number**, **student number** and **name** at the top of the first page of your file. (All three items must be present.) [1 mark]

Question 1. Datapath (8 marks)

[Adopted from a past-year's exam question] Consider the MIPS datapath with the values of the registers shown below. All values are in hexadecimal.

```
R0  (r0) = 0x00000000 | R1  (at) = 0x00002000
R2  (v0) = 0x00000001 | R3  (v1) = 0x0000000a
R4  (a0) = 0x00000005 | R5  (a1) = 0x7ffff000
R6  (a2) = 0x7ffff004 | R7  (a3) = 0x000000b0
R8  (t0) = 0x00000001 | R9  (t1) = 0x00000c00
R10 (t2) = 0x0000c000 | R11 (t3) = 0xfffffff0
R12 (t4) = 0xf0000000 | R13 (t5) = 0x00000fff
R14 (t6) = 0x00006200 | R15 (t7) = 0x00000e00
R16 (s0) = 0x00300000 | R17 (s1) = 0x00000c00
R18 (s2) = 0x00040200 | R19 (s3) = 0x00011000
R20 (s4) = 0x00030200 | R21 (s5) = 0x10000000
R22 (s6) = 0x00055000 | R23 (s7) = 0xf0000000
R24 (t8) = 0x00000005 | R25 (t9) = 0x0000d000
R26 (k0) = 0x00000000 | R27 (k1) = 0x00000000
R28 (gp) = 0x10008000 | R29 (sp) = 0x7fffeff4
R30 (s8) = 0x1000000f | R31 (ra) = 0x00400018
```



Question 1. (continue…)

The current PC value is **160** and the instruction being executed is:

```
sw $s2, -40($sp)
```

Fill in the values of the fields in the table below. For rows marked with *, your answers must follow the base given (0b for binary, 0x for hexadecimal).

Field	**Value**
RegDst	
MemRead	
MemWrite	
ALUSrc	
RegWrite	
Instruction[31-26] *	0b
Instruction[25-21] *	0b
Instruction[20-16] *	0b
Instruction[15-11] *	0b
Instruction[5-0] *	0b
❶ * (output from sign-extend)	0x
❷ *	0x
❸ *	0x
❹ * (read data 2)	0x
❺ *	0x
❻ (ALU control output)	

Question 2. Simplification (14 marks)

(a) Simplify the following Boolean expression into its simplest **POS** expression, showing each step with justification (i.e. citing the law/theorem used). Marks will be deducted if a step is not justified or wrongly justified. [6 marks]

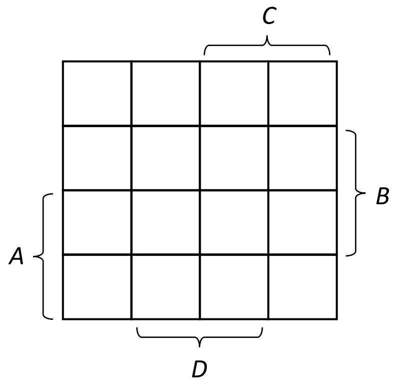
$$B \cdot Y \cdot E' \cdot (A' \cdot X + A \cdot X' + A \cdot X + A' \cdot X') + B' \cdot L \cdot U \cdot E' \cdot S' \cdot K \cdot Y + Y \cdot E' \cdot S'$$

You may skip obvious steps involving involution and commutative law, such as $(A \cdot B')'$ straight to $(A'+B)$ [De Morgan's theorem] instead of first to $(A'+(B')')$ [De Morgan's] then to $(A'+B)$ [involution]; or $(A' \cdot B + A)$ straight to $(A+B)$ [absorption theorem 2] instead of first to $(A + A' \cdot B)$ [commutative law] then to $(A+B)$ [absorption theorem 2].

Recall that the AND operator · must be present. Writing *AB* instead of $A \cdot B$ will incur penalty.

For each of the Boolean functions in parts (b) and (c) below, write (i) the number of PIs (prime implicants) in the K-map of the function, (ii) the number of EPIs (essential prime implicants) in the K-map of the function, (iii) its simplest SOP expression, and (iv) its simplest POS expression. If there are more than one simplest SOP/POS expression, you only need to write one. No partial credit will be given for incorrect answers for subparts (i) to (iv).

You may show your K-maps but they won't be graded. Your K-maps may be useful for us to locate your mistakes. If you do show your K-maps, please use the following layout. You may use X for don't cares.



(b) $F(A,B,C,D) = \Sigma m(3,6,7,11,14) + \Sigma X(1,5,9,13)$. [4 marks]

(c) $G(A,B,C,D) = \Pi M(0,1,5,7,13) \cdot \Pi X(4,9,11,12,15)$. [4 marks]

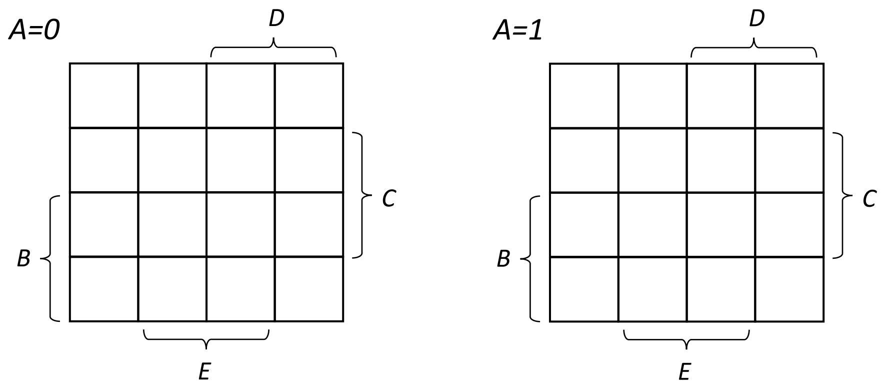
Question 3. Circuit Design (8 marks)

A combinational circuit takes in a 5-bit input *ABCDE* which is a value in excess 7 representation, and generates an output *M* such that *M*=1 if *ABCDE* represents a multiple of 2 or 5, or *M*=0 otherwise.

For example, if *ABCDE*=10001, then *M*=1; if *ABCDE*=10100, then *M*=0.

<u>You may assume that negative values will not be supplied to this circuit</u>.

(a) Draw a 5-variable K-map for *M* (draw one K-map with *A*=0 and another with *A*=1). You may use the K-maps below. [2 marks]



(b) Write out the simplified SOP expression for *M*. [3 marks]

(c) Let *V*(*A*,*B*,*C*,*D*,*E*) be the validity function, that is, *V* returns 1 if the input is valid, or 0 otherwise. Implement *V* using the fewest number of gates. Logic gates that are available for use include NOT, OR, AND, NOR, NAND, XOR and XNOR. Each gate (except NOT gate) must not have more than two inputs. Draw your circuit. [3 marks]

Question 4. Block-level design (7 marks)

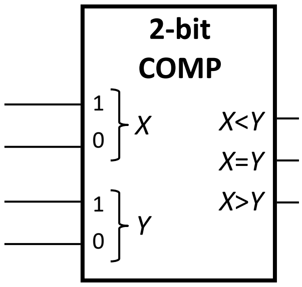
Note:

- Boolean constants (0 and 1) are always available;
- Complemented literals are not available.

(a) Implement the following Boolean function using a single 2-bit magnitude comparator with no additional logic gates.

$$F(A,B,C,D) = \Sigma m(1, 4, 5, 6, 7, 13)$$

The block diagram of a 2-bit magnitude comparator is shown below. [3 marks]



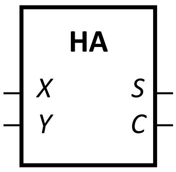
(b) Implement the following 3-variable Boolean functions *G* and *H*.

$$G(A,B,E) = \Sigma m(3,5)$$

$$H(A,B,E) = \Sigma m(1,2,4,6)$$

The block diagram for the half-adder is shown below. You are to implement the above two functions together using at most two half-adders and one logic gate. No mark will be awarded if you use more than two half-adders and one logic gate.

Logic gates that are available for use include NOT, OR, AND, NOR, NAND, XOR and XNOR. You are also to produce the truth table with the necessary columns for intermediate results to show that your circuit is correct. [4 marks]



A	*B*	*E*		...			
0	0	0		...			
0	0	1		...			
0	1	0		...			
0	1	1		...			
1	0	0		...			
1	0	1		...			
1	1	0		...			
1	1	1		...			

=== END OF PAPER ===