CS2100 Computer Organisation
2022/23 Semester I
Assignment 1

PLEASE READ THE FOLLOWING INSTRUCTIONS VERY CAREFULLY.

1. **DEADLINE:** The deadline for this assignment is **MONDAY 3 OCTOBER 2021, 11.59 PM**. Submission is **SOFTCOPY** in **PDF ONLY** on **LumiNUS**, NOT on Canvas! **No submissions will be accepted after the deadline.**

2. Fill your answers in the answer book called **CS2100Assg1AnsBk.docx** enclosed in the Assignment 1 ZIP file. Fill all your answers in this file and PDF it. Alternatively you may print out the answer book, write your answers manually, then scan your answers into a PDF file.

3. Name your file “AxxxxxxY.pdf”, where AxxxxxxY is your student ID. **3 marks will be deducted if you do not follow this naming convention.**

4. You should complete this assignment on your own without discussing with anyone.

5. All of you should be able to find CS2100 in your LumiNUS. Navigate to Files->Assignment 1 Submissions->Txx, where Txx is your tutorial group number. Submit your PDF file there.

6. Ensure that your submission contains your **tutorial group number, name and student ID. 3 marks will be deducted if you do not fill in this information.**

7. Answers may be typed or handwritten. If it is the latter, please ensure that it is neat and legible. Marks may be deducted for untidy work or illegible handwriting.

8. There are FOUR (4) QUESTIONS on FIVE (5) printed pages including this page. In addition there are two C source code files q1.c and q2.c in the ZIP file.

9. You are again reminded to i) Name your PDF file AxxxxxxY.pdf where AxxxxxxY is your student ID, and ii) tutorial group number, name and student ID in your answer book. You may lose up to 6 marks if you fail to do either or both of these.

10. You are likewise reminded to submit on LumiNUS and not on Canvas.

The questions are worth **40 marks** in total.

Question 1 (15 MARKS)

There is a source code file called “q1.c” that is enclosed in the Assignment 1 zip file. Answer the following questions:

a. What is the size, in bits, of each element of array t? (0.5 marks)
b. Write down the elements of array t in binary (2.5 marks mark)
c. For each of the following variables in the mystery function describe briefly the purpose of each variable. (8 marks)

Variable	**Purpose (in one short sentence)**
a	
b	
r	
d	

d. Describe in under 20 words what the mystery function does (5 marks)

Question 2 (5 MARKS)

In the tutorial we saw how to implement even parity in MIPS Assembly. In this question we will implement a function that calculates even *or* odd parity on an array of char of known length. You may assume that bit 7 of each element is initially 0.

We want to set bit 7 of **each** array element to either odd or even parity. In odd parity, bit 7 is set to 1 if there is an even number of 1's in bits 0 to 6 of that element and 0 otherwise. Conversely in even parity, bit 7 is set to 1 if there's an odd number of 1's in bits 0 to 6 of that array element, and 0 otherwise. The idea of course is that the **total** number of 1's in each element should be even for even parity, and odd for odd parity.

As an example, consider an array of 3 elements consisting of these elements (bit 7 is highlighted)

0b00011011, 0b00011001, 0b01010101

Suppose we want even parity, i.e. bit 7 is set so that the total number of 1 bits is even. Then the end result is:

0b00011011, 0b10011001, 0b01010101

You are given a file called q2.c. Implement the set_parity function to set bit 7 of each array element to either odd or even parity, depending on the value of "odd". When "odd: is 0, we set bit 7 of each array element according to the rules of even parity, and when "odd" is 1, we set bit 7 of each array element according to the rules of odd parity.

You may modify only the set_parity function. **You are not allowed to add any additional constants, functions or global variables.** You may, of course, add local variables within the set_parity function together with any code that you need **within** the function.

Cut and paste your code **only** for the set_parity function in your answer book. Do not paste in any additional code. (5 marks)

Question 3. (10 MARKS)

You are required to complete the following MIPS translation of a C-like program. Restriction: **you can only fill in the []**. If there is a letter infront of the [], it is hint on the starting letter of the instruction. You are not allowed to add new instructions / modify a given opcode. If the [] is empty, you can add any suitable instruction. **Note:** Fill your answers in the answer book, not here. **(9 x 1 mark = 9 marks)**

Variable Mappings	Comments
address of array "A[]" ➔ $s0 i ➔ $s1 (**initialized to 5**) j ➔ $s2	
`addi $s2, $zero, 1`	`j = 1`

`loop:`
`  slt  $t8, [                ]`
`  b[                    ], end`
	`while (j <= i) {`

`  [                ]`
`  [                ]`
`  [                ]`
`  lw   $t0, 0($t4)`
	`t0 = A[j-1]`

`  sll  $t3, $s2, 2`
`  add  $t5, $s0, $t3`
`  lw   $t1, 0($t5)`
	`t1 = A[j]`

`  s[                 ]`
`  b[                 ], skip`
	`if (a[j-1] > a[j] ) {`

`  [                ]`
`  [                ]`
	`//swap a[j-1] with a[j]`
	`}//end of if`

`skip:`
`  addi $s2, $s2, 1`
`  j loop`
	`j++`
	`} //end of while`

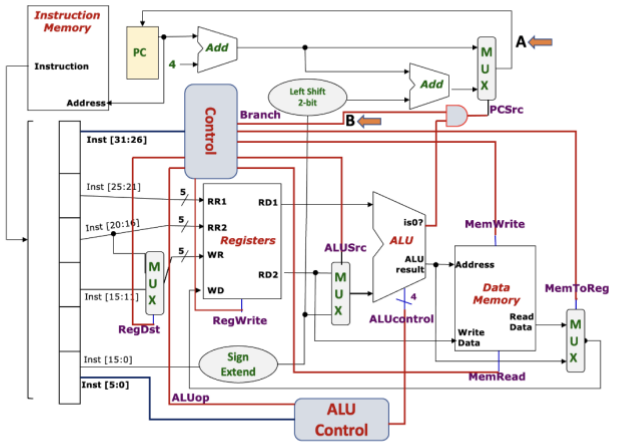
`end:`	

(b) Give the MIPS encoding for the following instruction in hexadecimal (1 mark)

`sll  $t3, $s2, 2  = 0x`

<u>Question 4</u> (10 MARKS)

Fill in the control signals and the values of the elements in your Datapath & Control path for the below branch (beq) instruction. You can assume the branch is taken and the PC is holding the address of beq instruction. The PC value is 0x200. You will also figure out the value of the registers when the branch is taken to fill the table.



<u>MIPS Program:</u>

```
                addi $s0, $s0, 0
                addi $t0, $t0, 0
                addi $t8, $t8, 100
                beq $t0, $t8, L1
                addi $t0, $t0, 1
                addi $t8, $t8, -1
                addi $s0, $t0, $t8
L1:
```

Fill in the values of the fields in the table in the answer book (not here!). For rows marked with *, your answers must follow the base given (0b for binary, 0x for hexadecimal). (0.5x20 = 10 marks)

RegDst		ALUOp	
RegWrite		isZero	
ALUSrc		Sign Extend Output*	0x
PCSrc		**A***	0x
ALUControl		**B**	
ALUResult*	0x	Inst[31-26]*	0b
MemRead		Inst[25:21]*	0b
MemWrite		Inst[20:16]*	0b
MemToReg		Inst[15:11]*	0b
WriteData*	0x	Inst[5:0]*	0b