CS2100 Computer Organization
AY2023/24 Semester I
Assignment 2
Due: Monday, 16 October 2023, 1pm

Instructions (PLEASE READ CAREFULLY!)

1. There are FOUR (4) questions in this assignment, totaling THIRTY-SEVEN (37) marks. Please do all parts of every question. Marks are indicated against each part.

2. An additional 3 marks is awarded for putting in your name, student ID and tutorial group number in your answers and for submitting in PDF format with the correct naming convention (see below). Thus, the total will be FORTY (40) marks. If you fail to do any of these, you will lose the 3 marks.

3. This assignment is due on **Monday, 16 October 2023, 1 pm.** You will be given until 1.15 pm to submit, **after which no submission will be accepted and you will receive ZERO for this assignment, regardless of how hard you've worked on it.**

4. **Plan to submit at least 2 hours early (i.e., by 11 am on 16 October)** in case there are technical issues with submission. If, by 1 pm on 16 October you are unable to submit due to technical reasons, please email your answers to Weng Fai at wongwf@nus.edu.sg before 1.15 pm. **No submission over email will be accepted after 1.15 pm.**

5. Complete your answers on the provided **CS2100Assg2AnsBk.docx** file. Save it as AxxxxxxY.pdf before submitting. For the programming part, you are required to describe the corresponding parts of your code both for documentation as well as assessing your thought process in designing it.

6. **Zip your AxxxxxxxY.pdf file together with your mips.c (your C code must use this name) file from the programming parts into a file called AxxxxxxxY.zip, and submit that on Canvas. Note that while the answer booklet is supplied as a Word document so that you have more flexibility space-wise, you must submit a PDF.** You will forfeit the 3 marks if you do not do all these, or if you fail to fill in your name, student ID and tutorial group number. **In addition if you do not submit your `mips.c` file, you will not receive marks for the programming portion of the Assignment.**

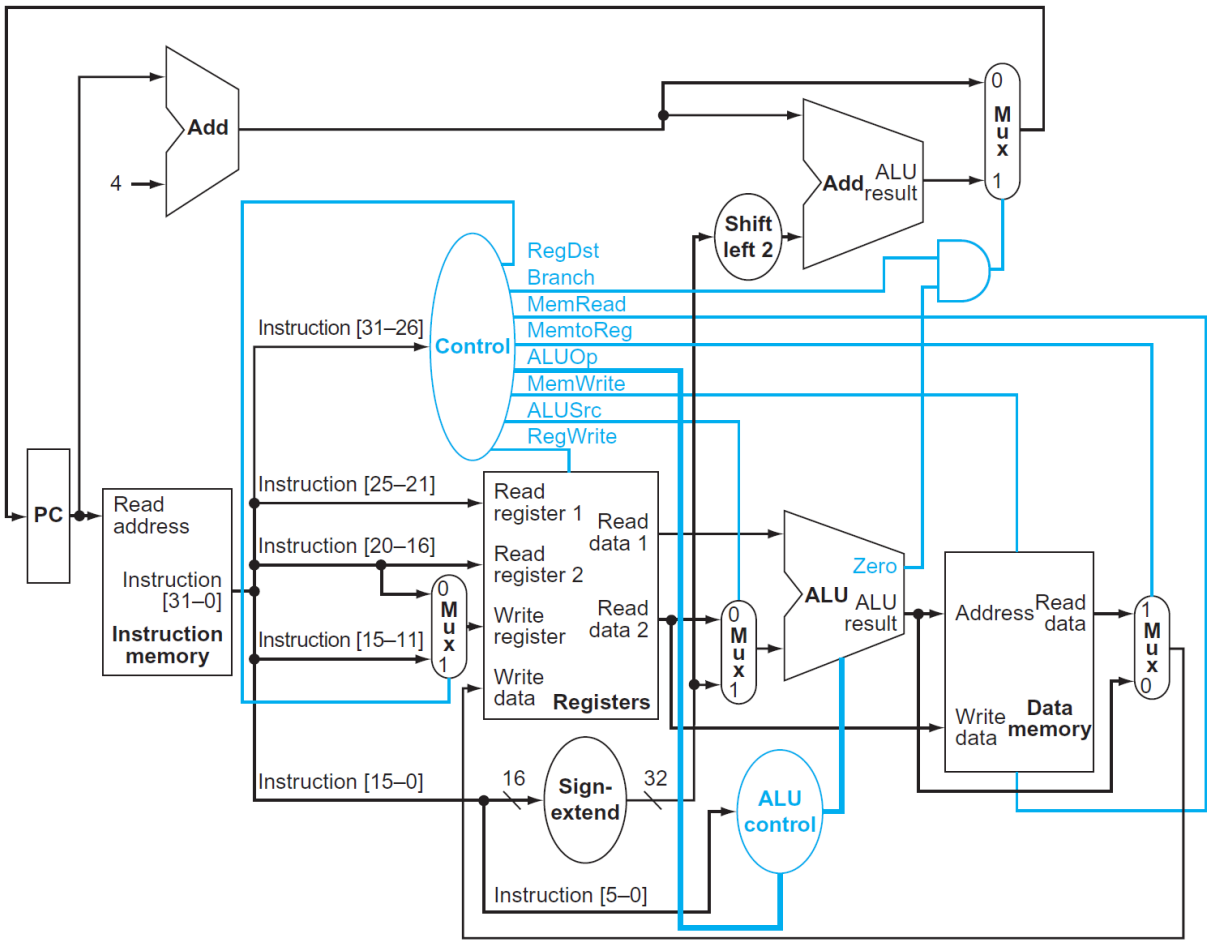
7. `mips.c` must be entirely compilable using one single command, namely:
   ```
   gcc -o mips main.c mips.c
   ```
   In particular, **no** additional file or library may be used.

8. You should do these assignments on your own. Do not discuss the assignment questions with others. Note that the NUS/SoC policy on plagiarism shall apply in full. Severe penalty may result in its violation.

9. Please use the Canvas Discussion Forums for clarifications.

Part A – Data and Control Paths

We shall finish what we started in Tutorial 5. In other words, you are to write a C simulator for the simple MIPS data and control path.



In particular, your implementation should conform to the following constraints:

1. Besides this question paper, you will be provided with three C files – **`main.c`**, **`mips.h`**, and **`mips.c`**. Modify/add to only the parts marked out in mips.c. All other parts of the code should not be modified. You will be penalized for touching parts you are not supposed to.
2. Your processor only needs to implement the following instructions:

   `add`, **`and`**, **`beq`**, **`lw`**, **`sw`**, **`nor`**, **`or`**, **`slt`**, **`sub`**.
3. Instead of the arbitrary **`intN_t`** and **`uintN_t`** we used in Tutorial 5, you can only use the bit lengths predefined in **`stdint.h`**, i.e., **`int8_t`**, **`uint8_t`**, **`int16_t`**, **`uint16_t`**, **`int32_t`**, **`uint32_t`**, as well as the **`bool`** type in **`stdbool.h`**.
4. Unlike in Tutorial 5, the data memory (instruction memory is not needed since our simulator for this assignment only works with a single instruction as input) is at most 1024 integers (4096 bytes). Any address that goes beyond is raised as an error.
5. The executable should be called with a single MIPS instruction in hexadecimal, for example:

   ```
   ./mips  0x014B4820
   ```

   which is "**`add $t1, $t2, $t3`**".

Question 1. (7 MARKS)

(a) Complete the implementations of the three multiplexor functions found in **`mips.c`**. They may be needed later. [3 marks]

(b) Complete the implementation of the **`void decode(uint32_t in, struct instr *insn)`** function found in **`mips.c`**. This function will take in a 32-bit unsigned integer encoding of a valid MIPS instruction and then fill out the **`struct insn`** whose address is also passed in as a pointer. Note that here (unlike in Tutorial 5) the purpose of this structure is to capture all the relevant information about instruction. Hence the binary will be decoded by interpreting the encoding in all three types of instructions. [4 marks]

Question 2. (10 MARKS)

(a) Complete the implementation for **`Control()`** function found in line 95 of the given (original skeleton) **`mips.c`**.. This should be simple given Tutorial 5. Just want to check that you were paying attention. [3 marks]

(b) Complete the implementation for **`ALUControl()`** function found in line 105 of the given (original skeleton) **`mips.c`**. [3 marks]

(c) Complete the implementation for the **`ALU()`** function found in line 114 of the given (original skeleton) **`mips.c`**. [4 marks]

Question 3. (10 MARKS)

Complete the implementation of **`execute()`** function found in line 123 of the given (original) **`mips.c`** which will complete your simulator. Given an instruction, **`execute()`** will simulate 4 of the 5 stages of instruction processing, naming decode (and operand fetch), execute, memory, and write-back. At the end of **`execute()`**, the contents of the register file, memory and PC should be as you would expect as per our lectures. Your code will be tested against a suite of 10 hidden tests consisting of instruction encoding for the instructions mentioned above. Note that not only will the contents of registers, memory and PC be checked against that expected of the test cases but also the structure of **`execute()`** should mimic the data and control path diagram given above. The idea being that you should be simulating, as faithfully as possible, how instructions are executed in this hardware. Hopefully this will complete your understanding of the various aspects of this (simple) processor.

Note that **`execute()`** does not return any result directly. Instead, it changes the global state of the program including the register file, memory and the PC. These will be checked – as a whole, not just the parts affected by the test case - for correctness during grading when we execute your code.

[10 marks]

Part B – Boolean Algebra

Question 4. (10 MARKS)

Note that unless otherwise stated, complemented literals are not available.

(a) What is the minimum number of 2-input NAND gates required to implement XOR(a,b), where a and b are Boolean variables? Illustrate your answer with a neatly drawn logic diagram, with inputs and output clearly labelled. Mark will be deducted if the inputs and output are not properly labelled. [2 marks]

(You do not need to provide the proof, but you are advised to trace your logic circuit to check its correctness yourself.)

(b) A chessboard consists of 8×8 squares, as shown in the diagram on the right where the squares are labelled from 0 through 63. We may represent the labels as 6-bit binary numbers.

0	1	2	3	4	5	6	7
15	14	13	12	11	10	9	8
16	17	18	19	20	21	22	23
31	30	29	28	27	26	25	24
32	33	34	35	36	37	38	39
47	46	45	44	43	42	41	40
48	49	50	51	52	53	54	55
63	62	61	60	59	58	57	56

In one move, a bishop can move any number of squares diagonally; up to the edge of the board. If it is on a grey square, it may land on any grey square, in the course of a game, but it will never land on a white square. Likewise, a bishop that is on a white square will never land on a grey square.

Assume that the binary numbers $A_5A_4A_3A_2A_1A_0$ and $B_5B_4B_3B_2B_1B_0$ represent two labels. Define a Boolean function $S_k(A_k, B_k)$ that outputs 1 if $A_k = B_k$, or 0 otherwise, where $0 \leq k \leq 5$. Write out the simplified Sum-of-Products (SOP) expression for S_k. [1 mark]

(Use the symbols we introduced in class: ’ for NOT, $\cdot$ for AND, $+$ for OR. Do not use other symbols.)

(c) Given two distinct labels $A_5A_4A_3A_2A_1A_0$ and $B_5B_4B_3B_2B_1B_0$, define a Boolean function $F(A_5A_4A_3A_2A_1A_0, B_5B_4B_3B_2B_1B_0)$ that outputs 1 if a bishop at $A_5A_4A_3A_2A_1A_0$ is able to move to $B_5B_4B_3B_2B_1B_0$ in two or fewer moves, or 0 otherwise.

Write out the simplified SOP expression for F in terms of S_k. [1 mark]

(d) Given the 4-variable Boolean function $G(P,Q,R,S) = \prod M(1,8,9,10,12,14) \cdot \prod X(2,5,6,15)$ where M are the maxterms and X the don’t-cares, what are the prime implicants on the K-map of G? Write out all the prime implicants. You do not need to show your K-map. [1 mark]

(e) Given the 4-variable Boolean function $G(P,Q,R,S) = \prod M(1,8,9,10,12,14) \cdot \prod X(2,5,6,15)$ in part (d) above, what are the essential prime implicants on the K-map of G? Write out all the essential prime implicants. You do not need to show your K-map. [1 mark]

(f) Given the 4-variable Boolean function $G(P,Q,R,S) = \prod M(1,8,9,10,12,14) \cdot \prod X(2,5,6,15)$ in part (d) above, write out the simplified Product-of-Sums (POS) expression for G. [2 marks]

(g) Given the 5-variable Boolean function

$$H(A,B,C,D,E) = \sum m(5,8,13,16,18,29) + \sum X(0,7,15,23,24,26,31)$$

where m are the minterms and X the don't-cares, write out the simplified Sum-of-Products (SOP) expression for H. You do not need to show the K-map or working. [2 marks]

=== END OF PAPER ===