COMP2611 Fall 2022 Homework #4

Note:

- **The deadline of this homework is at 11:55pm on Wednesday, 11th May 2022 (Hong Kong Time, UTC+8). NO late submissions will be accepted!**
- **This is an individual homework, all the answers submitted should be your own work.**
- **You can print the homework and work out the answers on paper. Then scan all the answer pages into a single pdf file "homework4_<studID>.pdf", or photo and zip them into a single zip file "homework4_<stdID>.zip". Alternatively, you can work with your iPad/laptop. Make sure every detail of the answers is clearly visible in your submission (verify this before submitting), otherwise marks may be deducted.**
- **We only accept e-submissions at the Canvas. Submit COMP2611 Canvas, Homework 4. You can upload for multiple times, only the most updated one before the deadline will be marked.**
- **Make sure you keep the original copy of your homework until the homework score is finalized.**

Name :

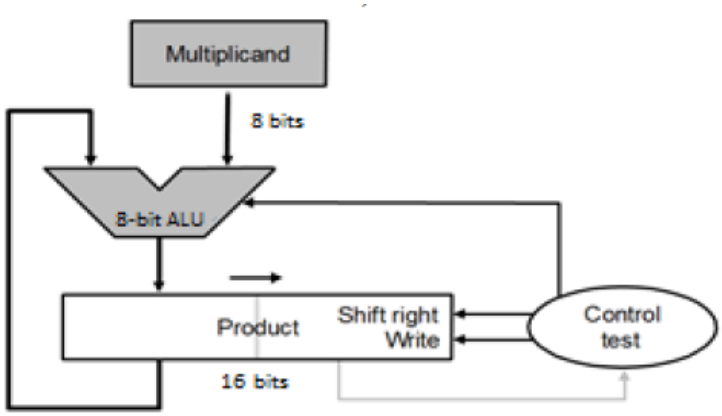
Student ID :

Email :

Question	Marks
1. Multiplication	/10
2. Division	/10
3. Single Cycle Datapath	/15
4. Single Cycle Datapath Control	/15
Total	/50

Question 1: Multiplication (10 points)

Given the **unsigned** binary multiplicand 01011010 and **unsigned** binary multiplier 01110001, compute the product with the optimized version of the multiplication hardware (slides 43 of Arithmetic note set) shown below.



a) Fill as many rows below as needed to calculate the product. Explain the operations in the "Action" column. Hint: Operations can be "**initial State**", "**no operation**", "**addition**", and "**shift right**", which carry the same meaning as in lecture notes. (8 points)

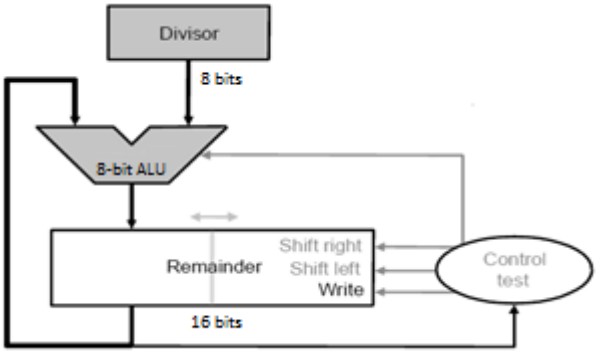
Iteration	Multiplicand (M)	Product (P)	Action
0	01011010	00000000 01110001	**Initial State**
1			
2			
3			
4			
5			

6			
7			
8			
9			

b) If this is an 8-bit **unsigned** number system, does the calculation in part (a) result in an overflow? Justify you answer, no mark will be given if you just answer "yes" or "no". (2 points)

Question 2: Division (10 points)

Given ***unsigned*** dividend 01110001, and ***unsigned*** divisor 00110001, compute the division result (quotient and remainder) with the optimized version of the division hardware (slides 59 of Arithmetic note set) shown below.



a) Fill as many rows below as needed to produce the result. Explain the operations in the "Action" column. Hint: Operations can be "**initial State**", "**subtraction**", "**shift left**", "**shift left, set LSb 0**", "**shift left, set LSb 1**", "**undo**" and "**adjust remainder**", which carry the same meaning as in lecture notes. LSB = least significant bit. (7 points)

Iteration	Divisor (D)	Remainder (R)	Action
0	00110001		
1			
2			
3			

4			
5			
6			
7			
8			
9			

The quotient is: _____

The remainder is: _____

b) By using the result in part (a) and also slide 60 of the note set (https://course.cse.ust.hk/comp2611/note/COMP2611_Arithmetic_Spring22.pdf), find the **quotient** and the **remainder** of -113/-49. (3 points)

Question 3: Single-cycle Datapath (15 points)

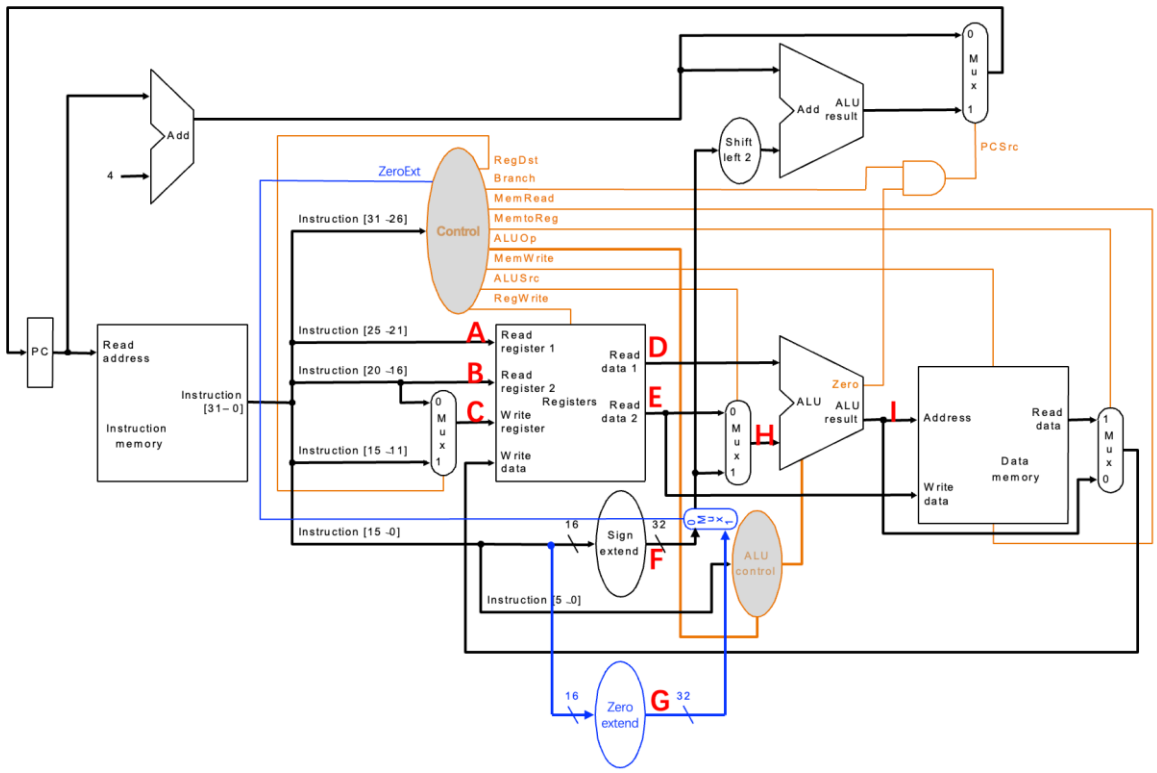
The following 32-bit MIPS instruction is fetched from the instruction memory:

001101 01010 01101 1010001110110100

Assume the data memory contains all zeros and the registers have the following values when the instruction is fetched ('0x' below means hex, otherwise when there is no '0x' it means decimal):

Register	`$at`	`$a1`	`$a3`	`$v0`	`$v1`	`$t2`	`$t5`	`$ra`
Value	0	0xB612	320	268695321	1092	0xDEADBEEF	933	0x40100000

a) Write the complete MIPS instruction (including all the involved registers, and the constant if there is any) that corresponds to the given 32-bit MIPS machine code. (4 points)



Refer to the **modified** Datapath above for the questions below. There are 2 major changes in this datapath. First, we added a "**Zero extend**" unit (oval) that is capable of zero extending the `imm` value. Second, we added a 1-bit "**ZeroExt**" control signal with a corresponding **multiplexer**. Assume we are executing the instruction fetched in part (a).

b) The labels A, B, C, D, E, F, G, and H in the datapath above indicate certain locations along the datapath. Find the data (**in binary format**) in the datapath for each of the labels. Note the data are with different bit widths (i.e. number of bits used) (8 points)

A =

B =

C =

D =

E =

F =

G =

H =

c) Derive the value of label **I** in above diagram. Show your step(s) clearly otherwise no point will be given. Write the value of label **I** in 32-bit binary format (3 points)

Question 4: Single Cycle Datapath Control (15 points)

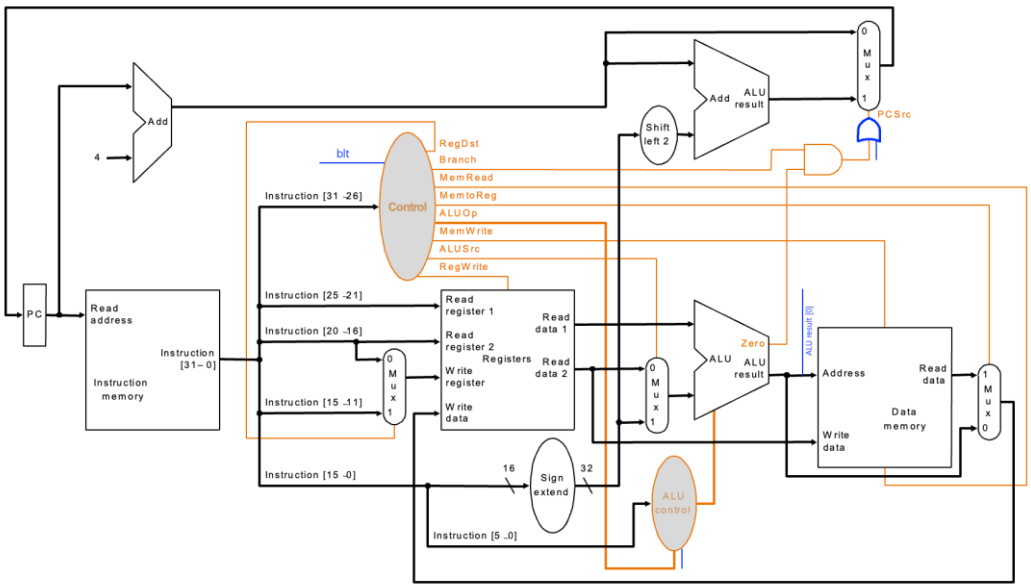
We wish to add the support to a pseudo MIPS instruction `blt` (Branch Less Than) as an `I-type` pure MIPS instruction in the **modified** single-cycle datapath below. The encoding of the `blt` instruction is exactly the same as the `beq` instruction. It stores the branch target using the **PC-relative** addressing mode exactly in the same way as the `beq` instruction, therefore the branch target calculation of `blt` is identical to that of the `beq` instruction.

The `blt` instruction compares the values in the `rs` and `rt` registers, if the value in `rs` is strictly less than the value in `rt` (i.e. `R[rs]<R[rt]`), it will put the branch target address into `PC`:

```
PC = PC + 4 + BranchAddr
```

Otherwise (i.e. `R[rs]>=R[rt]`), then:

```
PC = PC + 4
```



a) In the modified datapath, there is a newly added 1-bit control signal called `blt`, a new output that extract `bit 0` from the "`ALU result`", and an `OR` gate. All of them are marked in "blue" color. Make proper connections, so that this datapath will support the `blt` instruction correctly. You are just allowed to **make connections to the existing components/wires and add a single gate**, but not to expand or add any new MUX or other components. **Briefly explain how your modification would support running** `blt`. (7 points)

b) Complete the table below for control signals, so that the instructions can be executed correctly. If a control signal is a "don't care", **you must put x**, otherwise no mark will be given to it. (8 points)

Instr	RegDst	ALUSrc	Mem toReg	Reg Write	Mem Read	Mem Write	Branch	blt	ALU control
lw	0	1	1	1	1	0	0		0010
sw	x	1	x	0	0	1	0		
beq	x	0	x	0	0	0	1		
blt									