Lab 9. Arithmetic Logic Unit

Write code for Arithmetic Logic Unit (ALU). ALU performs all arithmetic and logical operations on 32 bit long operands. The result is also 32 bit long.

You are asked to write ALU that performs the following operations:

all ALU operations

I-type

Arithmetic, Logical Operation, Test-and-Set				
001 000		addi	addi *rt rs imm*	rt = rs + sxt(imm)
001 001		addiu	addiu *rt rs imm*	rt = rs + sxt(imm)
001 010		slti	slti *rt rs imm*	rt = (rs < sxt(imm) ? 1_{32} : 0_{32})
001 011		sltiu	sltiu *rt rs imm*	rt = (rs < sxt(imm) ? 1_{32} : 0_{32})
001 100		andi	andi *rt rs imm*	rt = rs ∧ zxt(imm)
001 101		ori	ori *rt rs imm*	rt = rs ∨ zxt(imm)
001 110		xori	xori *rt rs imm*	rt = rs ⊕ zxt(imm)
001 111		lui	lui *rt imm*	rt = imm0^{16}

Arithmetic, Logical Operation					
000000	100 000		add	add *rd rs rt*	rd = rs + rt
000000	100 001		addu	addu *rd rs rt*	rd = rs + rt
000000	100 010		sub	sub *rd rs rt*	rd = rs − rt
000000	100 011		subu	subu *rd rs rt*	rd = rs − rt
000000	100 100		and	and *rd rs rt*	rd = rs ∧ rt
000000	100 101		or	or *rd rs rt*	rd = rs ∨ rt
000000	100 110		xor	xor *rd rs rt*	rd = rs ⊕ rt
000000	100 111		nor	nor *rd rs rt*	rd = $\overline{\text{rs} \vee \text{rt}}$
Test-and-Set Operation					
000000	101 010		slt	slt *rd rs rt*	rd = (rs < rt ? 1_{32} : 0_{32})
000000	101 011		sltu	sltu *rd rs rt*	rd = (rs < rt ? 1_{32} : 0_{32})

R-type

Table 1

For additional Hints see the Lab 8 Hints.ppt file attached or read chapter 7 from Pr. Wolfgang's Book.

Your module should have the following inputs and outputs:

Inputs:

i - if it is 1, we have immediate type instruction.

SrcA - 32 bit long. This is the left operand

SrcB - 32 bit long. This is the right operand

af - 4 bit long. This is alu control signals. af decides which arithmetic or logic operation should be performed. Table 2 specifies values of af for corresponding instructions.

Outputs:

Alures – 32 bit long. This is the result of the arithmetic/logic operation perfomed on SrcA and SrcB.

Zero – This is zero flag. It is 1 if the alures is 0.

Neg – this is negative flag. It is 1 if alures is 0 and we have signed operation

ovfalu – this is overflow flag. It is 1 if operation caused overflow. And we had signed operation.

1) **I Type Instructions**: Write code that can perform all I type operations from Table 1
2) **R Type Instructions**: Write code that can perform all R type operations from Table 1
3) **Flags**: Write code that evaluates correct values for zero, neg and ovfalu flags.
4) **Simulation & Verification**: Write testbench for your design. Generate Waveforms and explain in your reports why do you think your design works correctly. Write Report in Template

If you want your module to be ELEGANT and worthy of international standards, I suggest using Figure 39, Lemma 39 and 40 from Pr. Wolfgang's Book (System Architecture An Ordinary Engineering Discipline by Wolfgang J. Paul).

Good Luck

ALU CONTROL SIGNAL (af) VALUES AND CORRESPONDING OPERATIONS

$af[3:0]$	i	$alures[n-1:0]$	$ovfalu$
0000	*	$a +_n b$	$[a]+[b] \notin T_n$
0001	*	$a +_n b$	0
0010	*	$a -_n b$	$[a]-[b] \notin T_n$
0011	*	$a -_n b$	0
0100	*	$a \wedge b$	0
0101	*	$a \vee b$	0
0110	*	$a \oplus b$	0
0111	0	$\overline{a \vee b}$	0
0111	1	$b[n/2-1:0]0^{n/2}$	0
1010	*	$0^{n-1}([a] < [b] \ ? \ 1:0)$	0
1011	*	$0^{n-1}(\langle a \rangle < \langle b \rangle \ ? \ 1:0)$	0

Table 2