Modules:

Module: _32bit_and

This module takes two 32 bit input and perform and operation for each bit and store them into result.

Module: _32bit_or

This module takes two 32 bit input and perform or operation for each bit and store them into result.

Module: _32bit_nor

This module takes two 32 bit input and perform nor operation for each bit and store them into result.

Module: _32bit_xor

This module takes two 32 bit input and perform xor operation for each bit and store them into result.

Module: half_adder

This module takes two 1 bit input and perform addition operation. Returns summation and carry out bit

Module: full_adder

This module takes two 1 bit input and carry in bit as an input and perform addition operation using half_adder module. Returns summation and carry out bit.

Module: _4bit_adder

This module takes two 4 bit input and carry in bit as an input and perform addition operation using four full adder module. Returns summation and carry out bit.

Module: _8bit_adder

This module takes two 8 bit input and carry in bit as an input and perform addition operation using 2 four bit adder module. Returns summation and carry out bit.

Module: _32bit_adder

This module takes two 32 bit input and carry in bit as an input and perform addition operation using 4 eight bit adder module. Returns summation and carry out bit.

Module: to32bit

This module takes one bit input and transform it to 32 bit output which contains itself on each bits. This module used as a helper module.

Module: _32bit_sub

This module takes two 32 bit input and carry in bit as an input. To perform this operation, _32bit_xor module used on second input. Than the one adds to result of the this operation. Finally result and first input adds to actual result to get subtraction and returns the output.

Module: _32bit_slt

This module takes two 32 bit input. To perform this operation, subtracts second input from first input and takes the most significant bit to get result.

Module: to32bit_zeros

This module takes one bit input and transform it to 32 bit output which contains itself on last bit. Rest of the bits are full of with zeros. This module used as a helper module.

Module: mux2x1

This module takes two 32 bit input and selection bit. _32bit_and operation performs on first input with s not and _32bit_and operation performs on second input with s and the results put in to or gate to find the actual result.

Module: mux4x1

This module takes four 32 bit input and 2bit selection bit. First and second input put into mux2x1 module with first selection bit and third and fourt input put into mux2x1 module with first selection bit. Result is found by putting the result into mux2x1 module with second selection bit.

Module: mux8x1

This module takes 8 32 bit input and 3 bit selection bit. First four input puts into mux4x1 module with first two selection bit and second four input puts into mux4x1 module with first two selection bit and the results are puts into mux2x1 module with last selection bit.

Module: alu32

This module takes 2 32 bit input and 3 bit selection bit. All implemented operations calculated and stored into 32 bit wires. And the result founded by the sending them into mux8x1 module with opcode and results. This is the inefficent way to perform arithmetic logic unit.

Test Results:

```
# time =  0, a =00000000000000000000000000000011, b=00000000000000000000000000000001, op=000, sum=00000000000000000000000000000100
# time = 20, a =11111000000000000000000000000011, b=11111111111111111111111111111111, op=001, sum=00000111111111111111111111111100
# time = 40, a =00000000000000000000000000000011, b=00000000000000000000000000000001, op=010, sum=00000000000000000000000000000010
# time = 60, a =00000000000000000000000000000000, b=00000000000000000000000000000001, op=100, sum=00000000000000000000000000000001
# time = 80, a =10000000000000000000000000000000, b=00000000000000000000000000000001, op=100, sum=00000000000000000000000000000000
# time = 100, a =10010101000000000000000000000000, b=11111100101000000000000000000001, op=101, sum=00000010010111111111111111111110
# time = 120, a =10010101000000000000000000000000, b=11111100101000000000000000000001, op=110, sum=10010100000000000000000000000000
# time = 140, a =10010101000000000000000000000000, b=11111100101000000000000000000001, op=111, sum=11111101101000000000000000000001
```