CSE 331 – COMPUTER ORGANIZATION HOMEWORK 2 REPORT

Name – Surname : Furkan OZEV

Number : 161044036

REQUESTED:

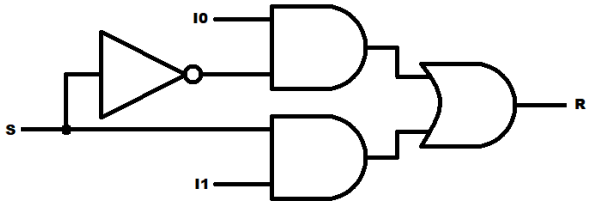
In this project, we are asked to design the same ALU as in the course notes. This ALU must support AND, OR, ADD, SUBTRACT and SET ON LESS THAN operations.

MODULES:

First of all, I designed the mux module which is necessary for selection in 1 bit ALU. I need a 4x1 mux module. Firstly, I designed a 2x1 mux module to use when designing 4x1 mux module.

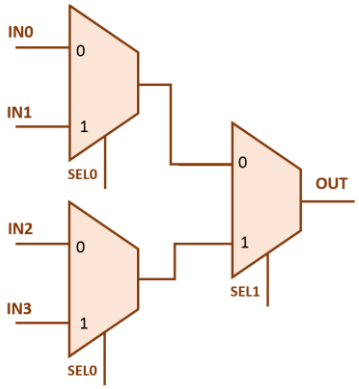
1.) Mux_2:

This module is 2x1 MUX module. It takes 2 bits as input and 1 bit as select bits. Selects one of the input bits according to the select bit as output bit. In this module 2 AND, 1 OR, 1 NOT gates is used. The design is as follows:



2.) Mux_4:

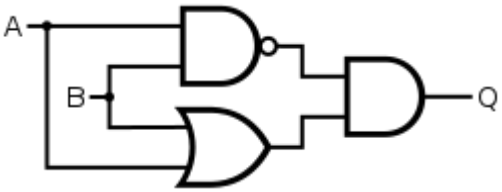
This module is 4x1 MUX module. It takes 4 bits as input and 2 bit as select bits. Selects one of the input bits according to the select bits as output bit. In this module 3 2x1 MUX modules is used. The design is as follows:



Then, I designed the xor module which is necessary in 1 bit ALU.

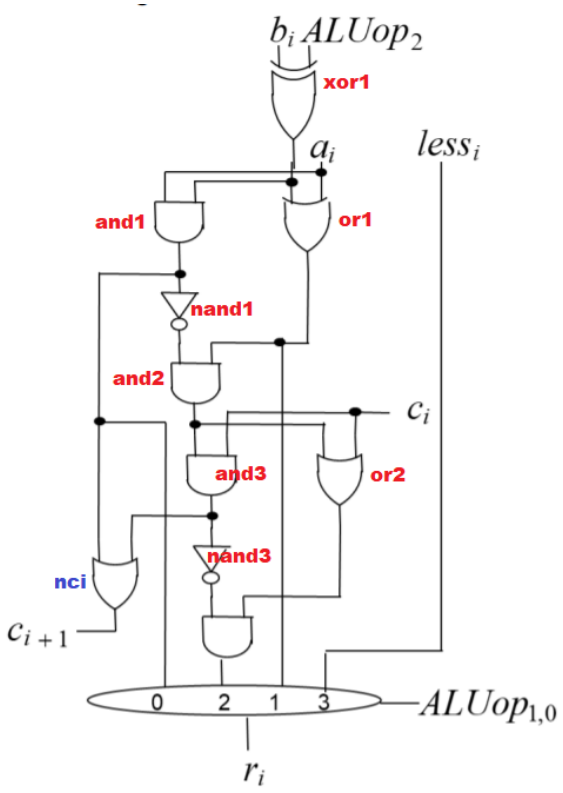
3.) Xor_:

This module is XOR module. It takes 2 bits as input. After logic operations, xor result is given as output bit. In this module 2 AND, 1 OR, 1 NOT gates is used. The design is as follows:



4.) Alu_1:

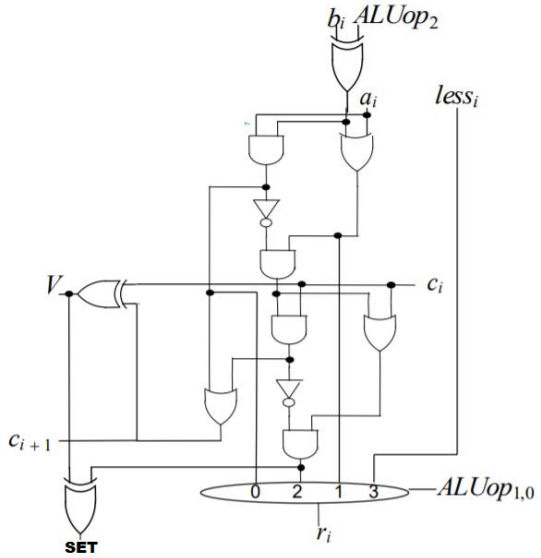
This module is 1 bit ALU module. This module is designed as given in the homework's pdf. This module takes 2 bits (as a and b), 3 bit opcode, carry bit, lessi bit as input. According to the opcode, the necessary logical operations are applied to these 2 bits (a and b) and result bit and carry bit are given as output. In this module 3 OR, 4 AND, 2 NOT gates is used. Also, in this module XOR_ module and MUX_4 module is used. This picture contains the names I used for the gates in the module. The design is as follows:



I needed a separate module for the set less than operation. For slt operation, we can find the result by giving the set bit of 32. bit as the less bit of the first bit. I needed this module to find the set bit of bit 32.

5.) Set_alu_1:

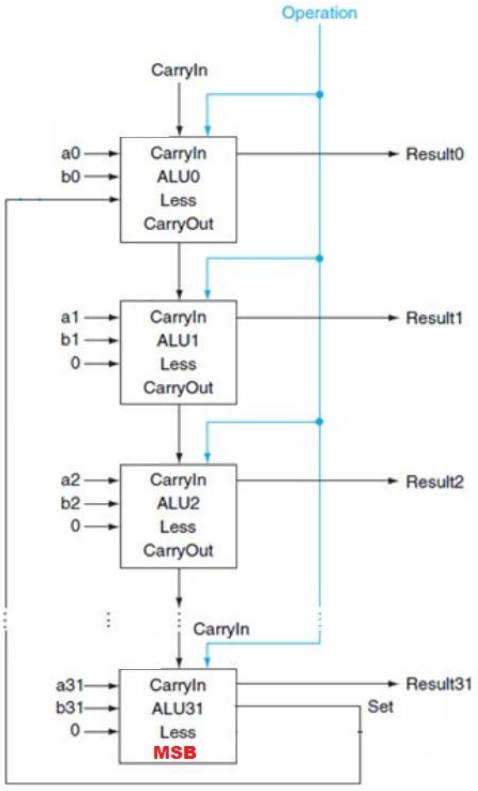
This module is 1 bit MSB ALU module. It will only be used for 32. bit. This module is designed as given in the lecture's slide. This module takes 2 bits (as a and b), 3 bit opcode, carry bit, lessi bit as input. According to the opcode, the necessary logical operations are applied to these 2 bits (a and b) and result bit, carry bit, SET bit are given as output. In this module 3 OR, 4 AND, 2 NOT gates is used. Also, in this module 2 XOR_ module and MUX_4 module is used. This picture contains the names I used for the gates in the module. The design is as follows:



For the SLT operation only the first bit is calculated and the other bits are 0. Therefore, the lessi bit is set to 0 when the alu modules are running. To find the lessi bit of the first module, the 32.bit 1 bit is executed in the msb module.

6.) Alu_32:

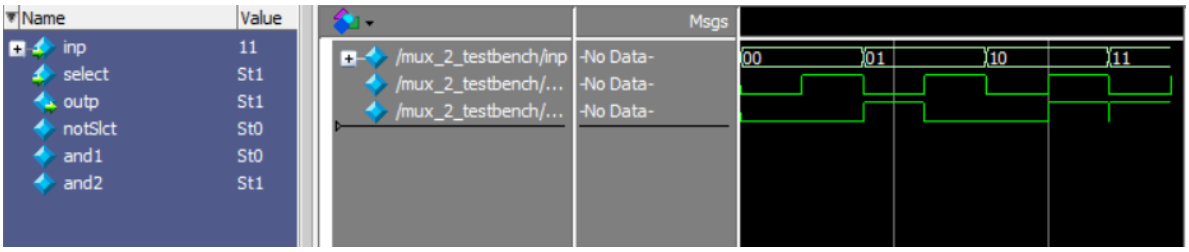
This module is 32 bit ALU module. This module takes 2 32-bit numbers (as a and b) and 3 bit opcode as input. It takes 2 32-bit numbers. According to the opcode given as input, logic operations are applied for these numbers and the output is given as 32-bit. First of all, if the opcode is 110, the initial value of carry in is set to 1, otherwise 0. Because these numbers are two's complement, the initial value of carry in must be 1 for subtraction. 1 bit alu module is executed for each bit in sequence. The carry bit of each bit is given as the carry in bit of the next bit. Unlike the other 31 bits, we run bit 32 in the module set_alu_1. Because we need to get the set bit from here. After receiving the set bit, we run the 1 bit alu module again for the first bit and give this set bit to as lessi bit of first module. This completes the 32-bit alu design. In this module 3 AND, 1 NOT gates is used. Also, in this module XOR_ module and 32 ALU_1 modules and 1 SET_ALU_1 module is used. The design is as follows:



TEST BENCHS:

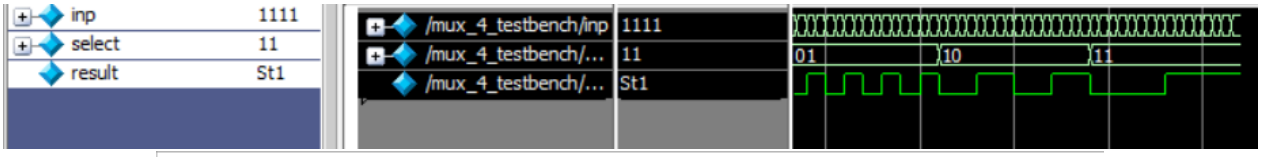
I added a screen shoot of only a short part of the outputs on the test bench. You can see the full part of test benches on ModelSim.

1.) Mux_2:



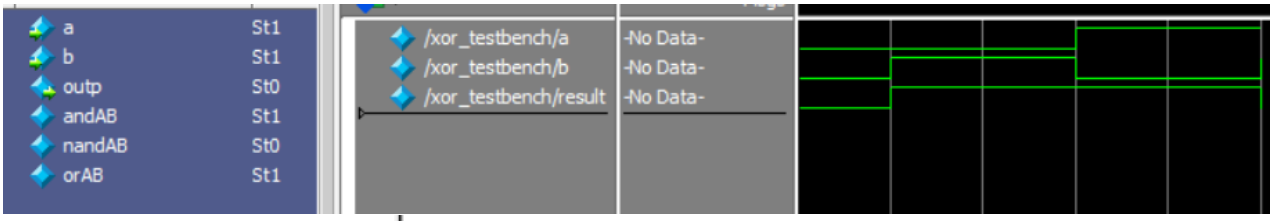

```
# time =  0, a=0, b=0, select=0, result=0
# time = 20, a=0, b=0, select=1, result=0
# time = 40, a=0, b=1, select=0, result=1
# time = 60, a=0, b=1, select=1, result=0
# time = 80, a=1, b=0, select=0, result=0
# time = 100, a=1, b=0, select=1, result=1
# time = 120, a=1, b=1, select=0, result=1
# time = 140, a=1, b=1, select=1, result=1
```

2.) Mux_4:



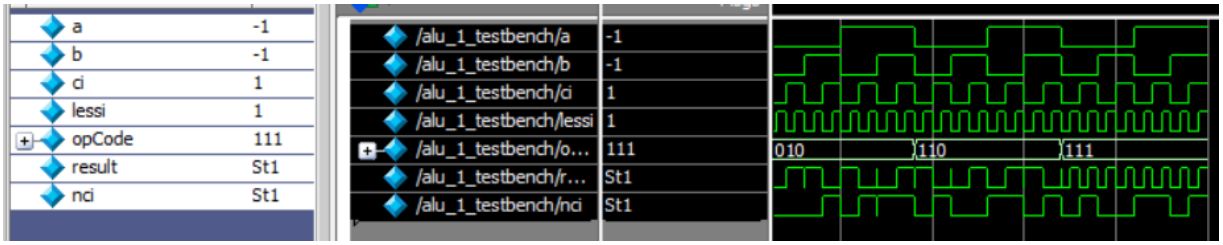

```
# time = 240, a=1, b=1, c=0, d=0, select_a=0, select_b=0, result=0
# time = 260, a=1, b=1, c=0, d=1, select_a=0, select_b=0, result=1
# time = 280, a=1, b=1, c=1, d=0, select_a=0, select_b=0, result=0
# time = 500, a=1, b=0, c=0, d=1, select_a=0, select_b=1, result=0
# time = 520, a=1, b=0, c=1, d=0, select_a=0, select_b=1, result=1
# time = 860, a=1, b=0, c=1, d=1, select_a=1, select_b=0, result=0
# time = 880, a=1, b=1, c=0, d=0, select_a=1, select_b=0, result=1
# time = 900, a=1, b=1, c=0, d=1, select_a=1, select_b=0, result=1
# time = 1080, a=0, b=1, c=1, d=0, select_a=1, select_b=1, result=0
# time = 1100, a=0, b=1, c=1, d=1, select_a=1, select_b=1, result=0
# time = 1120, a=1, b=0, c=0, d=0, select_a=1, select_b=1, result=1
```

3.) Xor_:



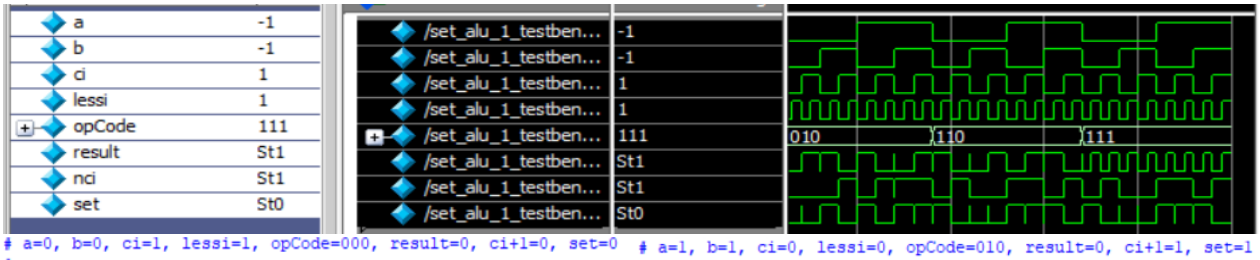

```
# time =  0, a=0, b=0, result=0
# time = 20, a=0, b=1, result=1
# time = 40, a=1, b=0, result=1
# time = 60, a=1, b=1, result=0
```

4.) Alu_1:




```
# a=0, b=0, ci=1, lessi=1, opCode=000, result=0, ci+1=0
#
# a=0, b=1, ci=0, lessi=0, opCode=000, result=0, ci+1=0
#
# a=0, b=1, ci=0, lessi=1, opCode=000, result=0, ci+1=0
#
# a=0, b=1, ci=1, lessi=0, opCode=000, result=0, ci+1=1
#
# a=0, b=1, ci=1, lessi=1, opCode=000, result=0, ci+1=1
#
# a=1, b=0, ci=0, lessi=0, opCode=000, result=0, ci+1=0
# a=1, b=1, ci=0, lessi=0, opCode=000, result=1, ci+1=1
#
# a=1, b=1, ci=0, lessi=1, opCode=000, result=1, ci+1=1
#
# a=1, b=1, ci=1, lessi=0, opCode=000, result=1, ci+1=1
# a=0, b=1, ci=0, lessi=0, opCode=001, result=1, ci+1=0
#
# a=0, b=1, ci=0, lessi=1, opCode=001, result=1, ci+1=0
#
# a=0, b=1, ci=1, lessi=0, opCode=001, result=1, ci+1=1
#
# a=0, b=1, ci=1, lessi=1, opCode=001, result=1, ci+1=1
# a=1, b=1, ci=1, lessi=0, opCode=001, result=1, ci+1=1
#
# a=1, b=1, ci=1, lessi=1, opCode=001, result=1, ci+1=1
# a=0, b=0, ci=0, lessi=0, opCode=111, result=0, ci+1=0
#
# a=0, b=0, ci=0, lessi=1, opCode=111, result=1, ci+1=0
#
# a=0, b=0, ci=1, lessi=0, opCode=111, result=1, ci+1=1
#
# a=0, b=0, ci=1, lessi=1, opCode=111, result=1, ci+1=1
# a=0, b=0, ci=0, lessi=0, opCode=010, result=0, ci+1=0
#
# a=0, b=0, ci=0, lessi=1, opCode=010, result=0, ci+1=0
# a=0, b=1, ci=1, lessi=0, opCode=010, result=0, ci+1=1
#
# a=0, b=1, ci=1, lessi=1, opCode=010, result=0, ci+1=1
#
# a=1, b=0, ci=0, lessi=0, opCode=010, result=1, ci+1=0
#
# a=1, b=0, ci=0, lessi=1, opCode=010, result=1, ci+1=0
#
# a=1, b=0, ci=1, lessi=0, opCode=010, result=0, ci+1=1
# a=0, b=1, ci=0, lessi=0, opCode=110, result=0, ci+1=0
#
# a=0, b=1, ci=0, lessi=1, opCode=110, result=0, ci+1=0
#
# a=0, b=1, ci=1, lessi=0, opCode=110, result=1, ci+1=0
#
# a=0, b=1, ci=1, lessi=1, opCode=110, result=1, ci+1=0
#
# a=1, b=0, ci=0, lessi=0, opCode=110, result=0, ci+1=1
#
# a=1, b=0, ci=0, lessi=1, opCode=110, result=0, ci+1=1
# a=1, b=1, ci=0, lessi=1, opCode=110, result=1, ci+1=0
#
# a=1, b=1, ci=1, lessi=0, opCode=110, result=0, ci+1=1
#
# a=1, b=1, ci=1, lessi=1, opCode=110, result=0, ci+1=1
# a=1, b=0, ci=1, lessi=0, opCode=111, result=0, ci+1=1
#
# a=1, b=0, ci=1, lessi=1, opCode=111, result=1, ci+1=1
#
# a=1, b=1, ci=0, lessi=0, opCode=111, result=0, ci+1=0
```

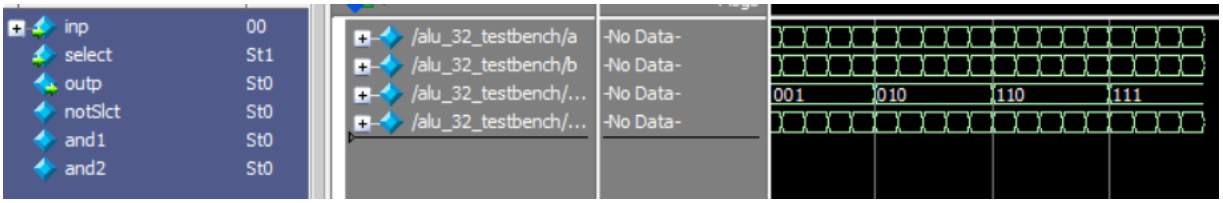
5.) Set_alu_1:




```
# a=0, b=0, ci=1, lessi=1, opCode=000, result=0, ci+1=0, set=0
#
# a=0, b=1, ci=0, lessi=0, opCode=000, result=0, ci+1=0, set=1
#
# a=0, b=1, ci=0, lessi=1, opCode=000, result=0, ci+1=0, set=1
#
# a=0, b=1, ci=1, lessi=0, opCode=000, result=0, ci+1=1, set=0
#
# a=0, b=1, ci=1, lessi=1, opCode=000, result=0, ci+1=1, set=0
#
# a=1, b=0, ci=0, lessi=0, opCode=000, result=0, ci+1=0, set=1
#
# a=1, b=0, ci=0, lessi=1, opCode=000, result=0, ci+1=0, set=1
# a=1, b=0, ci=1, lessi=0, opCode=001, result=1, ci+1=1, set=0
#
# a=1, b=0, ci=1, lessi=1, opCode=001, result=1, ci+1=1, set=0
#
# a=1, b=1, ci=0, lessi=0, opCode=001, result=1, ci+1=1, set=1
#
# a=1, b=1, ci=0, lessi=1, opCode=001, result=1, ci+1=1, set=1
#
# a=1, b=1, ci=1, lessi=0, opCode=001, result=1, ci+1=1, set=1
#
# a=1, b=1, ci=1, lessi=1, opCode=001, result=1, ci+1=1, set=1
#
# a=0, b=0, ci=0, lessi=0, opCode=010, result=0, ci+1=0, set=0
#
# a=0, b=0, ci=0, lessi=1, opCode=010, result=0, ci+1=0, set=0
# a=1, b=0, ci=0, lessi=0, opCode=010, result=1, ci+1=0, set=1
#
# a=1, b=0, ci=1, lessi=0, opCode=010, result=0, ci+1=1, set=0
#
# a=1, b=0, ci=1, lessi=1, opCode=010, result=0, ci+1=1, set=0
```

```
# a=1, b=1, ci=0, lessi=0, opCode=010, result=0, ci+1=1, set=1
#
# a=1, b=1, ci=0, lessi=1, opCode=010, result=0, ci+1=1, set=1
# a=0, b=1, ci=0, lessi=1, opCode=110, result=0, ci+1=0, set=0
#
# a=0, b=1, ci=1, lessi=0, opCode=110, result=1, ci+1=0, set=0
#
# a=0, b=1, ci=1, lessi=1, opCode=110, result=1, ci+1=0, set=0
#
# a=1, b=0, ci=0, lessi=0, opCode=110, result=0, ci+1=1, set=1
#
# a=1, b=0, ci=0, lessi=1, opCode=110, result=0, ci+1=1, set=1
#
# a=1, b=0, ci=1, lessi=0, opCode=110, result=1, ci+1=1, set=1
#
# a=1, b=0, ci=1, lessi=1, opCode=110, result=1, ci+1=1, set=1
# a=0, b=1, ci=1, lessi=0, opCode=111, result=0, ci+1=0, set=0
#
# a=0, b=1, ci=1, lessi=1, opCode=111, result=1, ci+1=0, set=0
#
# a=1, b=0, ci=0, lessi=0, opCode=111, result=0, ci+1=1, set=1
#
# a=1, b=0, ci=0, lessi=1, opCode=111, result=1, ci+1=1, set=1
#
# a=1, b=0, ci=1, lessi=0, opCode=111, result=0, ci+1=1, set=1
#
# a=1, b=0, ci=1, lessi=1, opCode=111, result=1, ci+1=1, set=1
#
# a=1, b=1, ci=0, lessi=0, opCode=111, result=0, ci+1=0, set=1
#
# a=1, b=1, ci=0, lessi=1, opCode=111, result=1, ci+1=0, set=1
#
# a=1, b=1, ci=1, lessi=0, opCode=111, result=0, ci+1=1, set=0
#
# a=1, b=1, ci=1, lessi=1, opCode=111, result=0, ci+1=1, set=0
```

6.) Alu_32:




```
# a = 10110011100100011011000001000100, decimal a=-1282297788, b = 00101100111000111011011001000010, decimal b=  753120834, OPCODE = 000, result = 00100000100000011011000001000000, decimal result=  545370176
#
# a = 00010011110010110000010001101011, decimal a=  332072043, b = 10101100100100101011010101011010, decimal b=-1399671462, OPCODE = 000, result = 00000000100000100000010001001010, decimal result=    8520778
#
# a = 00101111000101000001100110110110, decimal a=  789846454, b = 00111011101111101111110001111110, decimal b= 1002305790, OPCODE = 000, result = 00101011000101000001100010110110, decimal result=  722737334
#
# a = 11100100101111001101010101000010, decimal a= -457386686, b = 10101001001100101010001111001001, decimal b=-1456299063, OPCODE = 000, result = 10100000001100001000000101000000, decimal result=-1607433920
#
# a = 01011101011100101111100010110010, decimal a= 1566962018, b = 00100100111100100101011011101011, decimal b=  619861739, OPCODE = 000, result = 00000100011000000101000001100010, decimal result=   73420898
```

```
# a = 10110011100100011011000001000100, decimal a=-1282297788, b = 00101100111000111011011001000010, decimal b=  753120834, OPCODE = 001, result = 10111111111100111011011001000110, decimal result=-1074547130
#
# a = 00010011110010110000010001101011, decimal a=  332072043, b = 10101100100100101011010101011010, decimal b=-1399671462, OPCODE = 001, result = 10111111110110111011010101111011, decimal result=-1076120197
#
# a = 00101111000101000001100110110110, decimal a=  789846454, b = 00111011101111101111110001111110, decimal b= 1002305790, OPCODE = 001, result = 00111111101111101111110001111110, decimal result= 1069414910
#
# a = 11100100101111001101010101000010, decimal a= -457386686, b = 10101001001100101010001111001001, decimal b=-1456299063, OPCODE = 001, result = 11101101101111101111011111001011, decimal result= -306251829
#
# a = 01011101011100101111100010110010, decimal a= 1566962018, b = 00100100111100100101011011101011, decimal b=  619861739, OPCODE = 001, result = 01111101111101111111011111101011, decimal result= 2113402859
```

```
# a = 10110011100100011011000001000100, decimal a=-1282297788, b = 00101100111000111011011001000010, decimal b=  753120834, OPCODE = 010, result = 11100000011101010110011010000110, decimal result= -529176954
#
# a = 00010011110010110000010001101011, decimal a=  332072043, b = 10101100100100101011010101011010, decimal b=-1399671462, OPCODE = 010, result = 11000000010111011011100111000101, decimal result=-1067599419
#
# a = 00101111000101000001100110110110, decimal a=  789846454, b = 00111011101111101111110001111110, decimal b= 1002305790, OPCODE = 010, result = 01101010110100100001001010110100, decimal result= 1792152244
#
# a = 11100100101111001101010101000010, decimal a= -457386686, b = 10101001001100101010001111001001, decimal b=-1456299063, OPCODE = 010, result = 10001101111011110111100100001011, decimal result=-1913685749
#
# a = 01011101011100101111100010110010, decimal a= 1566962018, b = 00100100111100100101011011101011, decimal b=  619861739, OPCODE = 010, result = 10000010010110000100100001001101, decimal result=-2108143539
```

```
# a = 10110011100100011011000001000100, decimal a=-1282297788, b = 00101100111000111011011001000010, decimal b=  753120834, OPCODE = 110, result = 10000110101011011111101000000010, decimal result=-2035418622
#
# a = 00010011110010110000010001101011, decimal a=  332072043, b = 10101100100100101011010101011010, decimal b=-1399671462, OPCODE = 110, result = 01100111001110000100111100010001, decimal result= 1731743505
#
# a = 00101111000101000001100110110110, decimal a=  789846454, b = 00111011101111101111110001111110, decimal b= 1002305790, OPCODE = 110, result = 11110011010101100010000010111000, decimal result= -212459336
#
# a = 11100100101111001101010101000010, decimal a= -457386686, b = 10101001001100101010001111001001, decimal b=-1456299063, OPCODE = 110, result = 00111011100010100011000101111001, decimal result=  998912377
#
# a = 01011101011100101111100010110010, decimal a= 1566962018, b = 00100100111100100101011011101011, decimal b=  619861739, OPCODE = 110, result = 00111000011100111001101001110111, decimal result=  947100279
```

```
# a = 10110011100100011011000001000100, decimal a=-1282297788, b = 00101100111000111011011001000010, decimal b=  753120834, OPCODE = 111, result = 00000000000000000000000000000001, decimal result=          1
#
# a = 00010011110010110000010001101011, decimal a=  332072043, b = 10101100100100101011010101011010, decimal b=-1399671462, OPCODE = 111, result = 00000000000000000000000000000000, decimal result=          0
#
# a = 00101111000101000001100110110110, decimal a=  789846454, b = 00111011101111101111110001111110, decimal b= 1002305790, OPCODE = 111, result = 00000000000000000000000000000001, decimal result=          1
#
# a = 11100100101111001101010101000010, decimal a= -457386686, b = 10101001001100101010001111001001, decimal b=-1456299063, OPCODE = 111, result = 00000000000000000000000000000000, decimal result=          0
#
# a = 01011101011100101111100010110010, decimal a= 1566962018, b = 00100100111100100101011011101011, decimal b=  619861739, OPCODE = 111, result = 00000000000000000000000000000000, decimal result=          0
```