TEST CASES

ALU

ADDER

NAME	TEST_A	TEST_B	TEST_ALUFN	TEST_C	TEST_Z	TEST_V	TEST_N	Description
ADD1	0000000000000001	0000000000000001	000000	0000000000000010	0	0	0	Basic addition of 2 positive numbers: 1 + 1 = 2
ADD2	1111111111111111	1111111111111111	000000	1111111111111110	0	0	1	Addition of 2 negative numbers: (-1) + (-1) = (-2)
ADD3	0000000100000010	1111111011010101	000000	1111111111010111	0	0	1	Addition of positive and negative number: 258 + (-299) = -41
ADD4	1000000000000000	1000000000000000	000000	0000000000000000	1	1	0	Addition of 2 negative numbers, exceeds -32768: (-32768) + (-32768) = (-65536)
ADD5	0100000000000000	0100000000000000	000000	1000000000000000	0	1	1	Addition of 2 positive numbers, exceeds 32767: 16384 + 16384 = 32768
SUB1	0000000000000001	0000000000000001	000001	0000000000000000	1	0	0	Basic subtraction of 2 positive numbers: 1 – 1 = 0
SUB2	1111111111111111	1111111111111111	000001	0000000000000000	1	0	0	Subtraction of 2 negative numbers: (-1) - (-1) = 0
SUB3	0000110110101100	1111011000101000	000001	0001011110000100	0	0	0	Positive number subtract negative number: 3500 – (-2520) = 6020
SUB4	0100111000100000	1100010101101000	000001	1000100010111000	0	1	1	Positive number subtract negative number, exceeds 32767: 20000 – (-15000) = 35000
SUB5	1000101011010000	0001001110001000	000001	0111011101001000	0	1	0	Negative number subtract positive number, exceeds -32768: (-30000) – 5000 = (-35000)
NNUL	0000010000011010	0000011111010000	000010	0000101100100000	0	0	0	Multiplication of 2 16-bit positive numbers, lowest 16-bits: 1050 * 2000 = 2100000

COMPARE

NAME	TEST_A	TEST_B	TEST_ALUFN	TEST_C	TEST_Z	TEST_V	TEST_N	Description
EQ1	0000000000001011	0000000000011111	110011	0000000000000000	0	0	1	Compare equals 2 different numbers: 11 == 31 = FALSE
EQ2	0000000000011111	0000000000001011	110011	0000000000000000	0	0	0	Compare equals 2 different numbers, opposite: 31 == 11 = FALSE
EQ3	0000000000000000	0000000000000000	110011	0000000000000001	1	0	0	Compare equals 2 zeros: 0 == 0 = TRUE
EQ4	1111111111111111	1111111111111111	110011	0000000000000001	1	0	0	Compare equals the same negative number: (-1) == (-1) = TRUE
EQ5	0000000000001011	0000000000001011	110011	0000000000000001	1	0	0	Compare equals the same positive number: 11 == 11 = TRUE
EQ6	0000000000000001	0000000000000001	110011	0000000000000001	1	0	0	Compare equals 2 ones: 1 == 1 = TRUE
LT1	0000000000001011	0000000000011111	110101	0000000000000001	0	0	1	Compare less than 2 different numbers: 11 < 31 = TRUE
LT2	0000000000011111	0000000000001011	110101	0000000000000000	0	0	0	Compare less than 2 different numbers, opposite: 31 < 11 = FALSE
LT3	0000000000000000	0000000000000000	110101	0000000000000000	1	0	0	Compare less than 2 zeros: 0 < 0 = FALSE
LT4	1111111111111111	1111111111111111	110101	0000000000000000	1	0	0	Compare less than the same negative number: (-1) < (-1) = FALSE
LT5	0000000000001011	0000000000001011	110101	0000000000000000	1	0	0	Compare less than the same positive number: 11 < 11 = FALSE
LE1	0000000000001011	0000000000011111	110111	0000000000000001	0	0	1	Compare less than or equal to 2 different numbers: 11 <= 31 = TRUE
LE2	0000000000011111	0000000000001011	110111	0000000000000000	0	0	0	Compare less than or equal to 2 different numbers, opposite: 31 <= 11 = FALSE
LE3	0000000000000000	0000000000000000	110111	0000000000000001	1	0	0	Compare less than or equal to 2 zeros: 0 <= 0 = TRUE
LE4	1111111111111111	1111111111111111	110111	0000000000000001	1	0	0	Compare less than or equal to the same negative number: (-1) <= (-1) = TRUE
LE5	0000000000001011	0000000000001011	110111	0000000000000001	1	0	0	Compare less than or equal to the same positive number: 11 <= 11 = TRUE

BOOLEAN

NAME	TEST_A	TEST_B	TEST_ALUFN	TEST_C	TEST_Z	TEST_V	TEST_N	Description
AND1	0100011110001001	0001101000001110	011000	0000001000001000	0	0	0	Bitwise AND of 2 different numbers: 18313 & 6670 = 520
AND2	1111111111111111	0000000000001010	011000	0000000000001010	0	0	0	Bitwise AND of 2 different numbers: 65535 & 10 = 10
OR1	1000100010000000	1001100101001001	011110	1001100111001001	0	1	0	Bitwise OR of 2 different numbers: 34944 | 39241 = 39369
OR2	1111111111111111	0010000000001010	011110	1111111111111111	0	0	1	Bitwise OR of 2 different numbers: 65535 & 10 = 65535
XOR1	1001101000100110	1000000000001000	010110	0001101000101110	0	0	1	Bitwise XOR of 2 different numbers: 39462 ^ 32776 = 6702
XOR2	1101010100100100	1111111111100111	010110	0010101011000011	0	1	0	Bitwise XOR of 2 different numbers: 54564 ^ 65511 = 10947
A1	1001100110011001	0000000000001000	011010	1001100110011001	0	0	1	Asel of 2 different numbers: Asel(39321, 8) = 39321
A2	0000000100000001	0111001110010111	011010	0000000100000001	0	0	0	Asel of 2 different numbers: Asel(257, 29591) = 257
B	0000000100000001	0111001110010111	011100	0111001110010111	0	0	0	Bsel of 2 different numbers: Bsel(257, 29591) = 29591
NOR	0101010100100100	0010000000001010	010001	1000101011010001	0	0	0	Bitwise NOR of 2 different numbers: ~(21796 | 10) = 35537
NAND	0000000000000001	0000000000000011	010111	1111111111111110	0	0	0	Bitwise NAND of 2 different numbers: ~(1 & 3) = 65534

SHIFTER

NAME	TEST_A	TEST_B	TEST_ALUFN	TEST_C	TEST_Z	TEST_V	TEST_N	Description
SHL1	0000000000000001	0000000000000001	100000	0000000000000010	0	0	0	Shift left by 1: 1 << 1 = 2
SHL2	0000000000000001	0000000000001111	100000	1000000000000000	0	0	0	Shift left by 15: 1 << 15 = 32768
SHL3	1111111111111111	0000000000001010	100000	1111110000000000	0	0	0	Shift left by 10, exceeds 65535: 65535 << 10 = 64512
SHL4	1001100110011001	0000000000001000	100000	1001100100000000	0	0	1	Shift left by 8, exceeds 65535: 39321 << 8 = 39168
SHL5	1000100010000000	0000000000001001	100000	0000000000000000	0	0	1	Shift left by 9, exceeds 65535: 34944 << 9 = 0
SHR1	1000000000000000	0000000000000001	100001	0100000000000000	0	1	0	Shift right by 1: 32768 >> 1 = 16384
SHR2	1000000000000000	0000000000001111	100001	0000000000000001	0	1	0	Shift right by 15: 32768 >> 15 = 1
SHR3	1111111111111111	0000000000001010	100001	0000000000111111	0	0	1	Shift right by 10: 65535 >> 10 = 63
SHR4	1001100110011001	0000000000001000	100001	0000000010011001	0	0	1	Shift right by 8: 39321 >> 8 = 153
SHR5	0000000100010001	0000000000001001	100001	0000000000000000	0	0	0	Shift right by 9: 273 >> 9 = 0
SRA1	1000000000000000	0000000000000001	100011	1100000000000000	1	1	0	Shift right arithmetic by 1 with signed bit 1: -32768 >>> 1 = -16384
SRA2	0100000000000000	0000000000001110	100011	0000000000000001	1	0	0	Shift right arithmetic by 14 with signed bit 0: 16384 >>> 14 = 1
SRA3	1111111111111111	0000000000001010	100011	1111111111111111	1	1	0	Shift right arithmetic by 10 with signed bit 1: -1 >>> 10 = -1
SRA4	1001100110011001	0000000000001000	100011	1111111110011001	1	1	0	Shift right arithmetic by 8 with signed bit 1: -26215 >>> 8 = -103
SRA5	0000000100010001	0000000000001001	100011	0000000000000000	1	0	0	Shift right arithmetic by 9 with signed bit 0, bits discarded: 273 >>> 9 = 0
SRA6	0111111111111111	0000000000000111	100011	0000000011111111	1	0	0	Shift right arithmetic by 7 with signed bit 0, bits discarded: 32767 >>> 7 = 255

FAIL CASES

ALU

ADDER

NAME	TEST_A	TEST_B	TEST_ALUFN	TEST_C	TEST_Z	TEST_V	TEST_N	Description
ADD1	0000000000000001	0000000000000001	010000	0000000000000010	0	0	0	Basic addition of 2 positive numbers: 1 + 1 = 2 (Invalid ALUFN)
ADD2	1111111111111111	1111111111111111	000000	1111111111111110	0	0	0	Addition of 2 negative numbers: (-1) + (-1) = (-2) (Invalid N)
SUB1	0000000000000001	0000000000000001	000001	0000000000000000	1	1	0	Basic subtraction of 2 positive numbers: 1 – 1 = 0 (Invalid V)
SUB2	1111111111111111	1111111111111111	000001	0000000000000000	0	0	0	Subtraction of 2 negative numbers: (-1) - (-1) = 0 (Invalid Z)
NNUL	0000010000011010	0000011111010000	000010	0000101100100001	0	0	0	Multiplication of 2 16-bit positive numbers: 1050 * 2000 = 2100000 (Invalid C)

COMPARE

NAME	TEST_A	TEST_B	TEST_ALUFN	TEST_C	TEST_Z	TEST_V	TEST_N	Description
EQ1	0000000000001011	0000000000011111	110010	0000000000000000	0	0	1	Compare equals 2 different numbers: 11 == 31 = FALSE (Invalid ALUFN)
EQ2	0000000000011111	0000000000001011	110011	0000000000000000	0	0	1	Compare equals 2 different numbers, opposite: 31 == 11 = FALSE (Invalid N)
LT1	0000000000001011	0000000000011111	110101	0000000000000001	0	1	1	Compare less than 2 different numbers: 11 < 31 = TRUE (Invalid V)
LT2	0000000000011111	0000000000001011	110101	0000000000000000	1	0	0	Compare less than 2 different numbers, opposite: 31 < 11 = FALSE (Invalid Z)
LE1	0000000000001011	0000000000011111	110111	0000000000000000	0	0	1	Compare less than or equal to 2 different numbers: 11 <= 31 = TRUE (Invalid C)

BOOLEAN

NAME	TEST_A	TEST_B	TEST_ALUFN	TEST_C	TEST_Z	TEST_V	TEST_N	Description
AND1	0100011110001001	0001101000001110	011001	0000001000001000	0	0	0	Bitwise AND of 2 different numbers: 18313 & 6670 = 520 (Invalid ALUFN)
OR1	1000100010000000	1001100101001001	011110	1001100111001001	0	1	1	Bitwise OR of 2 different numbers: 34944 | 39241 = 39369 (Invalid N)
OR2	1111111111111111	0010000000001010	011110	1111111111111111	0	1	1	Bitwise OR of 2 different numbers: 65535 & 10 = 65535 (Invalid V)
XOR2	1101010100100100	1111111111100111	010110	0010101011000011	1	1	0	Bitwise XOR of 2 different numbers: 39462 ^ 32776 = 6702 (Invalid Z)
A2	0000000100000001	0111001110010111	011010	0000000100000000	0	0	0	Asel of 2 different numbers: Asel(257, 29591) = 257 (Invalid C)

SHIFTER

NAME	TEST_A	TEST_B	TEST_ALUFN	TEST_C	TEST_Z	TEST_V	TEST_N	Description
SHL1	0000000000000001	0000000000000001	100010	0000000000000010	0	0	0	Shift left by 1: 1 << 1 = 2 (Invalid ALUFN)
SHL2	0000000000000001	0000000000001111	100000	1000000000000000	0	0	1	Shift left by 15: 1 << 15 = 32768 (Invalid N)
SHR1	1000000000000000	0000000000000001	100001	0100000000000000	0	0	0	Shift right by 1: 32768 >> 1 = 16384 (Invalid V)
SHR2	1000000000000000	0000000000001111	100001	0000000000000001	1	1	0	Shift right by 15: 32768 >> 15 = 1 (Invalid Z)
SRA1	1000000000000000	0000000000000001	100011	1100000000000001	1	1	0	Shift right arithmetic by 1 with signed bit 1: -32768 >>> 1 = -16384 (Invalid C)

ERROR MODES

AUF1 (Automatic Fail 1)

NAME	Description
FAIL	The 16-bit output from the ALU module does not correspond with the expected value of the output from the test case

AUF2 (Automatic Fail 2)

NAME	Description
ERR1	The 16-bit output from the ALU module does not correspond with the expected value of the output from the test case
ERR2	The 1-bit output for zero condition Z from the ALU/adder unit does not correspond with the expected value of Z from the test case
ERR3	The 1-bit output for overflow condition V from the ALU/adder unit does not correspond with the expected value of V from the test case
ERR4	The 1-bit output for the most significant bit from the ALU/adder unit N does not correspond with the expected value of the most significant bit from the test case
ERR5	The 6-bit ALUFN input to the ALU module is not a valid opcode (no operation for that opcode has been implemented in the ALU)