**ALU**

**0 floating point –** 0 exponent and 0 fraction

**Infinity floating –** 2017 exponent and 0 fraction

1 10001000 01010100000000000000000

Exponent: 10001000 – 28 = 136 - 127 = 9

Fraction: 0.010101 and sign bit is 1

So:

(-1)1 x (1+ 0.0101012) x 29 = -1 x (1.0101012 x 29

= -1 x (10 1010 10002 x 20)

= -1 x (512 + 128 + 32 + 8)

= -680

**Guard bit –** 2nd to last digit in floating point

**Round bit –** last digit in floating point

**Sticky bit –** set if there is a non-zero bit shifted pass round bit

- Floating-point add is not associated

**Processor**

lw/sw, add, sub, and, or, slt, beq, j

**edge-triggered clocking –** allows memory read

and write in the same clock cycle

Data will be written only when clock is asserted 1

Output from memory will be updated only when a

Clock is deserted, changed from 1 to 0

Immediate value in I-type instructions is sign-extended

add/addi

sub

and/andi

or/ori

nor

slt/slti

beq/bne

lw/sw

**Grey code –** 00 01 11 10

**Subtraction –** x - y = x + (-y) **set b-negate**

- for slt, subtract a and b and check the MSB

If (a-b) < 0, then a < b

Subtract a and b, then check MSB

If MSB is 1, then a < b

**To detect overflow –** we need:

1. Sign bit of first operand
2. Sign bit of second operand
3. Sign bit of result

If overflow occurs, Set outphut must be inverted

**Combinational logic –** Given a set of inputs,

it always produces the same output.

**Sequential logic –** An output of a sequential logic

depends on its inputs and the

**content of its internal state**

1. Contains at least one state element
2. A state element has at least two inputs:
   1. Data to be written to the element
   2. Clock to determine when data is written
   3. e.g. Registers and memory

In C, there are no exceptions for

unsigned operations (addu, addiu, subu)

For multiplication, the size of the result is

The sum of the sizes of the operands

(two 4-bit operands will not exceed 8-bit output)

1. Sign extend both numbers to output bit-length
2. Multiply them with paper and pencil
3. Shift first operand left by 1
4. Add all product results for final output

Floats are **normalized** if they have a single nonzero digit to

the left of the decimal point

In 2.9979 x 105, 2.9979 is the **fraction** and

5 in 105 is the **exponent**

**Single-precision float –** 8-bit exponent and 23-bit fraction

**Double-precision float –** 11-bit exponent and 20-bit fraction

Value is (-1)5 x Fraction x 2exponent

**Floating point overflow –** exponent is too large to be represented

**Floating point underflow –** negative exponent is too large to fit

in exponent field