Tutorial 4.

① Unsigned Numbers. (all positive).

Ex: $(12)_{10} = (1100)_2$.

Addition:

Properties:

$$\begin{array}{r} 0 \\ +0 \\ \hline 0 \end{array} \quad \begin{array}{r} 0 \\ +1 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ +0 \\ \hline 1 \end{array} \quad \begin{array}{r} 1 \\ +1 \\ \hline 0 \end{array}$$; 1 carry out.

For unsigned numbers if
- carry out = 1 ⇒ overflows occurs
- carry out = 0 ⇒ NO overflow.

Ex:

$$\begin{array}{r} 1001 \\ 0011 \\ \hline 1100 \end{array}$$; carry out = 0. No overflow.

$$\begin{array}{r} (9)_{10} \\ +(3)_{10} \\ \hline (12)_{10} \end{array}$$

$$\begin{array}{r} 1101 \\ +0110 \\ \hline 0011 \end{array}$$; carry out = 1 overflow.

- if it's specify the # of bits used, when (Carry out = 1) overflow occurs ⇒ result is wrong.
- when no # of bits is specified, then the carry out must be part of the result.

Substraction:

$$\begin{array}{rr} 10001 & (17)_{10} \\ -00111 & (7)_{10} \\ \hline 01010 & (10)_{10} \end{array}$$

- when you borrow from one the next one has TWO "1".

Multiplication

```
      1110     (14)10   * Same as normal
     x0110     (6)10      multiplication *
  ---------
  0000000
  0011100
  0111000
  0000 00
  ---------
  1010100     (84)10.
```

Division

```
1001 ⟌ 11011
```

° first. try.
1 divided by 1001 = 0
then next.
11 divided by 1001 = 0.
next ; 110 by 1001 = 0
finally 1101 divide by 1001 = 1.

* divide each bit by the divisor until you can divide * after that normal division.

```
        00011
1001 ⟌ 11011
        1001 ↓
        ------
        01001
         1001
        ------
           00
```

② Signed Numbers (2.1 Signed Magnitude representation)

$(+7)_{10} \Rightarrow$ [0]111 [S]; 0 = positive
$(-7)_{10} \Rightarrow$ [1]111 1 = negative.

for Addition, example: add the magnitudes and use the sign of the bigger number.

2.2. 2's complement.

0011011			0011011
1100100	Invert (1's complement)	→	1100100
0000001	Add 1 ~~1's complement~~	→	0000001
1100101	→ 2's complement.	→	1100101

To convert to 2's complement directly.
- look at the least significant value.
  - if its 0 move to next
  - if its 1, take 1 and invert the rest.

ex:
- 0011011 ← least significant is 1
  ↓ invert the rest
  1100101
- 0001001000100000. ← 0's
  ↓
  1110110111110000
  → 1, invert rest

Also,
In addition

carry out.
↓
[Cn]

$$\begin{array}{r} a_{n-1} \ldots\ldots a_1 a_0 \\ + \; b_{n-1} \ldots\ldots b_1 b_0 \\ \hline c_{n-1} \ldots\ldots c_1 c_0 \end{array}$$

$V = C_n + C_{n-1}$
if $V = 0 \Rightarrow$ No overflo[w]
$V = 1 \Rightarrow$ ~~[illegible]~~ overflow

Exercise:

a) 26, -37, -123 as signed 10 bit binary numbers.

i) signed magnitude.

$(26)_{10} = (\boxed{0}\,0000011010)_2$.

$(-37)_{10} = (\boxed{1}\,0000100101)_2$

$(-123)_{10} = (\boxed{1}\,0011111011)_2$.

↓ sign. → magnitude.

ii) 2's complement.

$(26)_{10} = 0000011010$. ← Same magnitude, and now the sign is part of the number. (thats why tutor didn't box it).

$(-37)_{10}$ * keep the sign and convert only the magnitude.

$(-37)_{10} = 1111011011$ (1's complement)

$(-123)_{10} = 1110000101$ (1's complement).

* In 2's complement, positive numbers, results in same magnitude *

a2) 6 bits numbers in 2's complement.

a) 010110 $(22)_{10}$
$+\ 001001$ $(9)_{10}$
$\overline{\boxed{0}\ 011111}$ $(31)_{10}$.

Carry out ← 0

V = 0

No overflow.

b) 011001 $(25)_{10}$
$+\ 010000$ $(16)_{10}$
$\overline{\boxed{0}\ 101001}$ $(41)_{10}$.

V = 0 + 1 = 1

overflow

* overflow doesn't mean answer is wrong.

BUT

wrong answer means overflow *

in normal base: $-0 = -1$

c)
$$\begin{array}{rll} 0\,1\,0\,1\,1\,0 & (22)_{10} & 00 \\ -\;0\,1\,1\,1\,1\,1 & (31)_{10} & -01 \\ \hline 1\,1\,0\,1\,1\,1 & (-9)_{10} & (11) \end{array}$$

-1 to 2's complement = 11

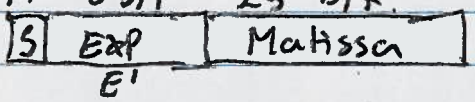
to check the answer, since its negative, we have to check its 2's complement, if its 9, then the result is correct.

110111

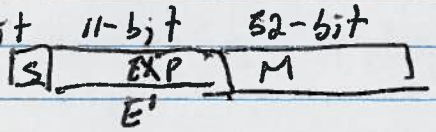
001001 → 9 ; then 110111 is correct.

Floating Point Numbers Representation

IEEE 754 single precission. (32-bit).



IEEE 754 double precission. (64-bit.



$E' = E + 127$ ← single precission.

bias (excess).

to compute bias.

$bias = 2^{(N-1)} - 1$; n = # of bits for exponents.

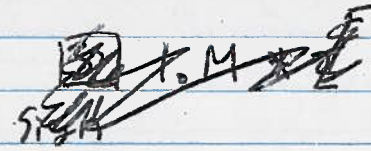
$= 2^{(8-1)} - 1$

$= 2^7 - 1$

$= 127$.

$(-1)^{sign} \; 1.M \times 2^{E}$

And the representation is:



Ex: express $(+7.5)_{10}$ in IEEE Single precission.

1-bit	8-bit	23-bit
S	E'	M

$$\text{bias} = 2^{N-1} - 1$$
$$= 2^{8-1} - 1$$
$$= 2^7 - 1$$
$$= 127.$$

First convert to binary.

$$(7.5)_{10} = (111.1)_2$$
$$2^2 2^1 2^0 . 2^{-1}$$

and $111.1 = 1.111 * 2^2$

$$(-1)^{\text{sign}} \; 1.111 * 2^2$$
$$(-1)^0 \; 1.\underbrace{111}_{M} * 2^{\textcircled{2}} \rightarrow E.$$

$$E' = E + 127 = 2 + 127 = 129. \Rightarrow (10000001)_2$$

Answer:

$$\underbrace{0}_{S} \; \underbrace{10000001}_{E'} \; \underbrace{111}_{M} \underbrace{0000000\ldots}_{20\ 0\text{'s}}$$

$$= 0\,10000001\,11100000000000000000000$$