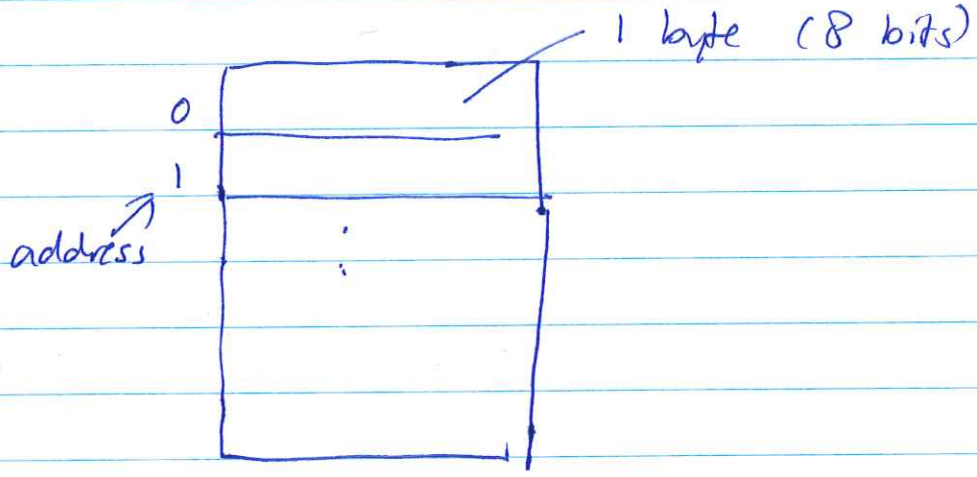
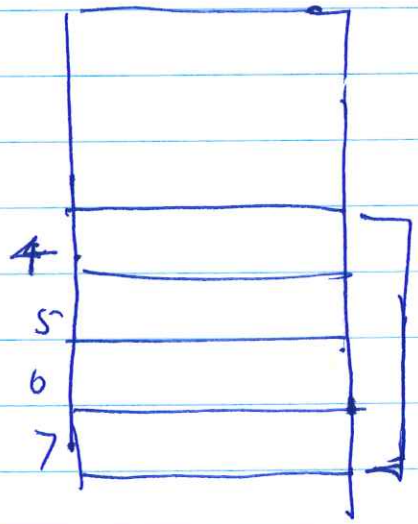
Representing "things" in computer

- Recall that computer memory is organized as follows:



- Consecutive bytes can be concatenated to form words. (long memory cells)

For example:



This word consists of 4 bytes.
A 4-byte word has $4 \times 8 = 32$ bits
and can hold $2^{32} = 4,194,304$ (over 4 million) diff. values (not at the same time!)

- Why do we need so many diff. values?

  → We need different values to represent diff. numbers in the computer and numbers can be quite large.

- What are the things that need to be represented inside the computer:

  (1) numerical quantities (numbers)
  (2) Text information (alphanumerical quantities)
  (3) Computer instructions (eg. add, sub, etc)

- Emphasize on representation:

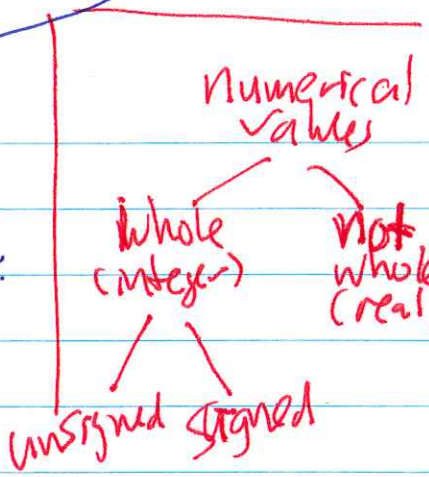
  representation = an agreement on how to represent certain things (values) by certain symbols.

  Also: agreement is "per set of things being represented"!

  → agreement on how to repr. (unsigned) numbers
  " " " " " signed "
  " " " " " Alphanum. quant
  " " " " " Comput. instr.

rules exists to repr:
data
(1) unsigned numbers
(2) signed numbers
(3) Alphanum. data (char)
(4) Computer instr.

integer = whole value. (0, 1, 2, ...)
NOT = decimal !!!



Representing numbers in computer

- There are 2 types of numbers:

  (1) integer numbers
  (2) real numbers.

- We can represent integer precisely (no round-off errors) in the computer.

  But real numbers can't be represented precisely because they may have infinite number of digits, eg:

  π = 3.14159265358979323846264338327950

  → approximation in computer.

- We will look at how integer numbers are represented.

Integer numbers

- There are 2 types of integers used in the computer.

  Signed and unsigned
  (ℕ (positive ∅ negative)) └ always ≥ ∅

Values and representation

- Overly used symbols:

  1 2 3 4 5 6 ...

  this is a representation of a value.

- ~~Value~~
  Numerical value has an intrinsic meaning.

Ideally, we represent numerical values as:

- ∅
- ∅ ∅
- ∅ ∅ ∅

Not as: 1, 2, 3 ... Why?

Scenario:

- You crash land in a remote island that has an entire diff. culture

- You want to trait your slaves for 3 coconuts.

- Can you use:

to represent the value 3 ??

No !!!

You will need to stick out

3 fingers !!!

- Fact:

  Numerical values are represented by diff. symbols in diffeent cultures.

World today: 1 2 3 4 . . .

Roman (World 2000 yrs ago): I II III IV V VI VII VIII IX X XI

China: 一 二 三 四 五

Ancient Egypt: | ∩ 𓆐

BTW:

Absolute representation (not positional)

When numbers are represented by:

I II III IV V VI

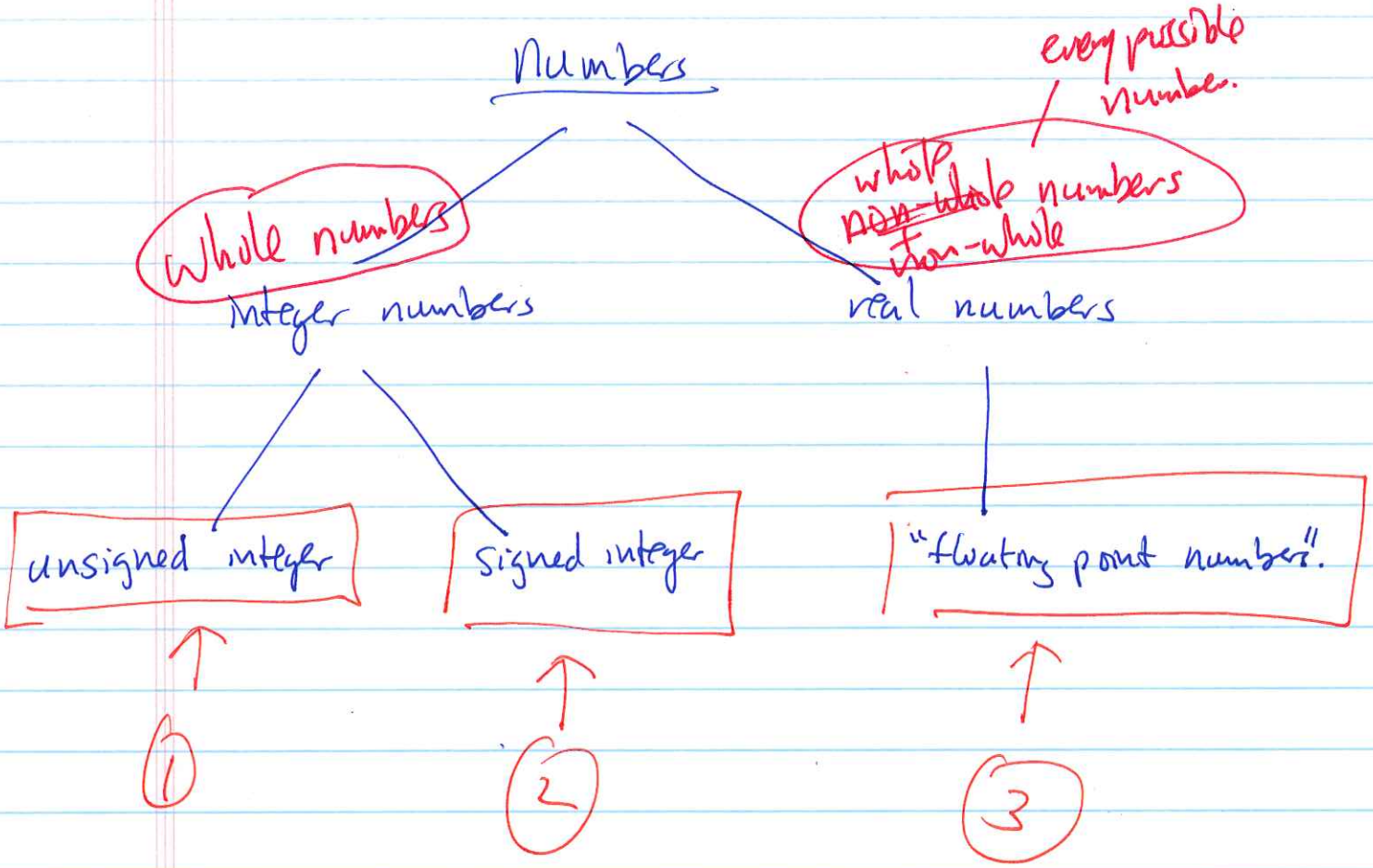
or — = ≡ ☐ ..

They are not useful to perform computation.

Romans and Chinese use

abacus

to perform computation (+, −, ...)



representation

Underlying goal for number systems:

Represent VALUES.

Romans: I II III IV V

Chinese: 一 二 三 四 五

Values →

etc. → Romans: I I .. V X ...

Representing unsigned integer values

- The most common representation system for unsigned integer values is the positional representation system.

- Non-positional representation system

  Each symbol has a fixed value.

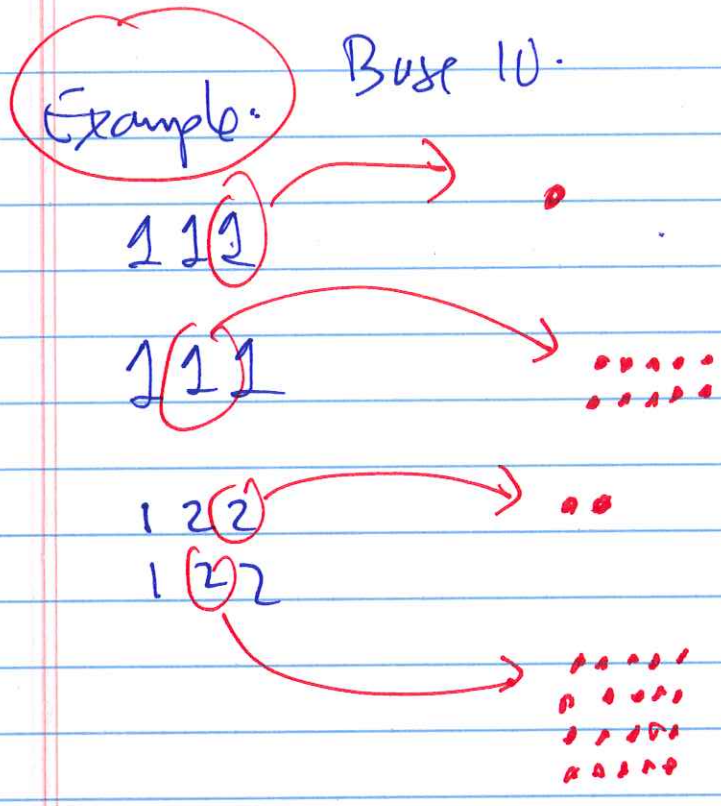
  eg: I •
  V •••••
  X ••••••••••

Positional representation system:

The value repr. by a symbol depends on:

(1) the symbol
(2) the position where the symbol occurs

Example. Base 10.

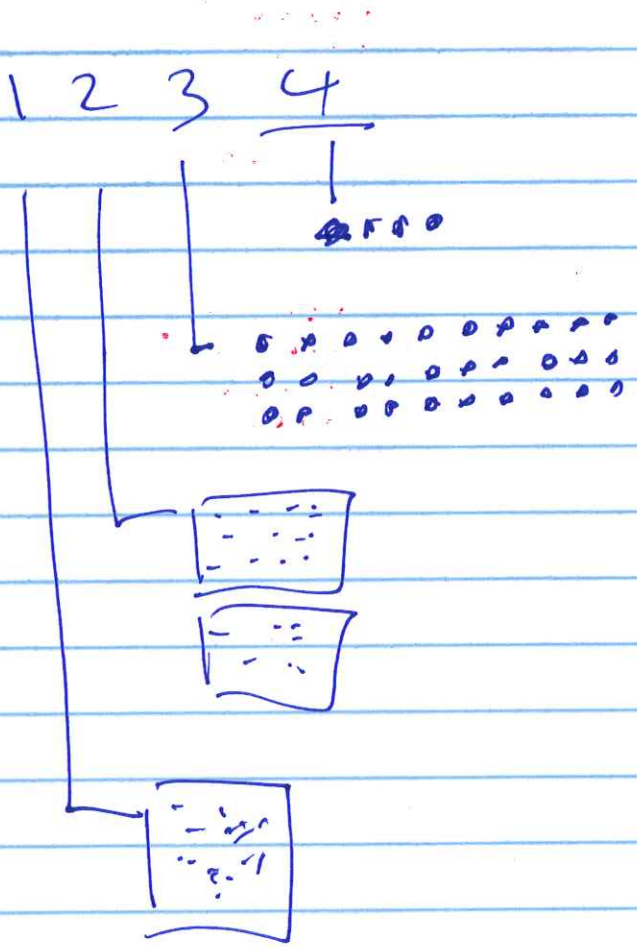


* Each positional representation has a base.

The value of each position is increased/decrease by one factor of the base when you move

← increase → decrease.

SU: 1234 base 10



total one thousand, two hundred thirty four dots.

In general; base b.

number: $a_n\, a_{n-1} \ldots a_1\, a_0$

value repr. $= a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + \ldots + a_1 \cdot b^1 + a_0 \cdot b^0$

Note: you need

b "symbols" (digits).

when you use base b.

$b = 10$ (decimal)

digits used: 0 1 2 3 4 5 6 7 8 9

$b = 2$ (binary)

digits used: 0 1.

Binary Integer number !!!

Binary numbers (and their represented values)

Notation: $(\ldots)_b$

b ← base

- Example 1:

$$(1000)_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$$

3 2 1 0

$$= 1 \cdot 8 + 0.4 + 0.2 + 0.1$$

$$= (8)_{10}.$$

Example 2:

$$(1010)_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$$

3 2 1 0

$$= 1.8 + 0.4 + 0.2 + 0.1$$

$$= 8 + 2 =$$

$$= (10)_{10}.$$

Joke:

There are 10 kinds of people:

those who understand binary numbers and those who don't.

Read as: $(10)_2 = 2$!!!

↓

binary number !!!

Binary Numbers and Computer Memory.

Recall: Computer memory (bunch of switches) can store bits.

- 1 memory cell stores 8 bits

Storing representations of unsigned values in memory:

Value 9

Binary number: 1001.

stored in 1 byte:

0000 1001

Store in 2 bytes:

0000 0000
0000 1001

↑

short variable.

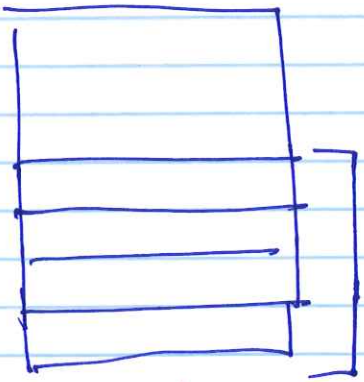
Store in 4 bytes.

↑

int variable.

0000 0000
0000 0000
0000 0000
0000 1001

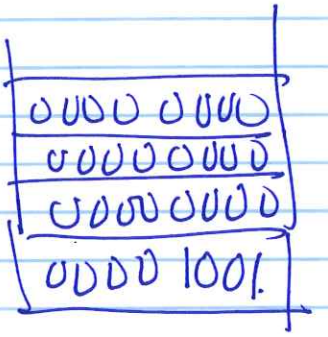
Quiz:

int x x →  Location reserved for variable x

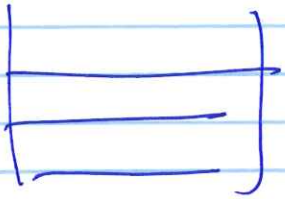
What is the effect of:

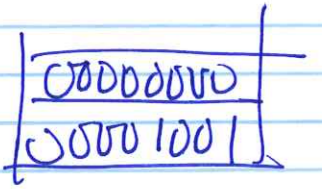
```
x = 9;
```

(in Java or any other progr. language)

Answer: x → 

---

short x; → x 

x = 9; 

Converting Numbers

Diff. repr. for SAME VALUE !!!

Converting binary ↔ decimal numbers

binary → decimal

- Given: $(1010.0110)_2 = (?)_{10}$

Solution:

$$(1010.0110)_2 = 2^7 + 2^5 + 2^2 + 2^1$$

$$= 128 + 32 + 4 + 2$$

$$= 166$$

decimal → binary

- Given: $(166)_{10} = (?)_2$.

Solution:

2	166	0
2	83	1
2	41	1
2	20	0
2	10	0
2	5	1
2	2	0
2	1	1
	0	

166 / 2 = 83, Remainder = 0.

binary number = 1010.0110

Remind them the context: BIT !!!

memory:

Advantage of positional repr:
We can do simpler arithmetic operations on these representations!

Arithmetic in the Binary number system

- Human are very good with arithmetic in the decimal (base 10) number system.

Memorize
Learn addition, subtraction, multiplication TABLES !!!

eg:

$$\begin{array}{r} 5 \\ +\ 7 \\ \hline 12 \end{array} \qquad \begin{array}{r} 7 \\ -\ 5 \\ \hline 2 \end{array}$$

We do it "instinctively" because we have MEMORIZED the addition, subtraction, multiplication tables in our childhood.

- What are these rules?
  - carry: when you add 2 digits and the sum goes over 9, you get a carry which worths a value of 10

carry

presentation: 7+4: 0 1 2 3 4 5 6 7 8 9

  - borrow: when you substract 2 digits and the first is less than the second, then you borrow 1 from the left digit and obtain a value of 10.

presentation: 14-7:

0 1 2 3 4 5 6 7 8 9

borrow

carry 1

eg: $12 + 9 = 21$

borrow 1

$12 - 9 = 3$

- The rules are the same in the binary number system, except:

  When you carry, the carry is worth a value of 2. (instead of 10 in decimal system).

  When you borrow, the borrow is also worth a value of 2.

- Let's do binary arithmetic.

$12 = (1100)_2$.

$2 \overline{)12}$	1
$2 \overline{)7}$	1
$2 \overline{)3}$	0
$2 \overline{)1}$	0
0	

$9 = (1001)_2$.

$2 \overline{)9}$	1
$2 \overline{)4}$	0
$2 \overline{)2}$	0
$2 \overline{)1}$	1
0	

presentation: 1+1

0 1 ← carry.

15
× 9
24

1100
+ 1001
11000 ← in binary

16 + 8 + 0 = 24. in decimal.

Another example:

00011.0111 → 32+16+4+2+1 = 55
+ 01001.0110 → 64+16+4+2 = 86 +
10000.1101 → 128+8+4+1 = 141

12
+ 9
21

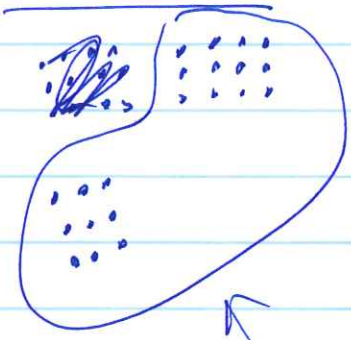
1100
+ 1001
10101

1 1 1 1 1
16 8 4 2 1

16+4+1 = 21.

Behind the scene:

Values are added:



Base 10 representation

$$\begin{array}{r} 12 \\ +\ 9 \\ \hline 21 \end{array}$$

Base 2 representation

$$\begin{array}{r} 1100 \\ +\ 1001 \\ \hline 10101 \end{array}$$

both symbol patterns represent the value (21)

Romans:

$$\begin{array}{r} XII \\ +\ IX \\ \hline XXI \end{array}$$

Key: (emphasis)

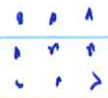
(1) number system represent VALUES

(2) No matter what number system is used; the following values operation is true:

and

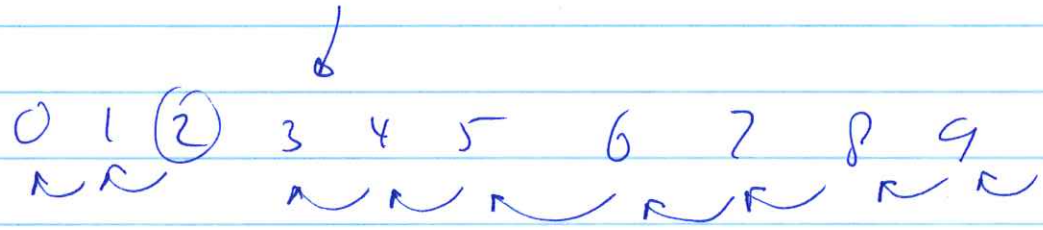
(3) You should not be surprised that arithmetic operations will work correctly

Example subtraction:

Value:	Decimal:	Binary
	12	1100
	− 9	1001
take away 	3.	0011

result:

Both patterns represent the value



Romans:

XII
− IX
III

X − X = 0.

+ II − (− I) = III !!!

Multiplication:

- One would wish people do arithmetic in binary!
- To do multiplication, one must learn the multiplication table by heart. ← elementary school

Recall:

$7 \times 1 = 7$

$7 \times 2 = 14$

$7 \times 3 = 21$

$7 \times 4 = 28$ etc.

To do multiplication in decimal, the tables that must learned are these:

X	1	2	3	4	5	9	10
0	0	0					0
1	1	2	3	4		9	10
2	2	4	6	8		18	·
3							
4							
5							
6							
7							
8							
9							

i.e.: the outcome of $d_1 \times d_2$ where $d_1, d_2 = 0, 1, \ldots, 9, 10$ must be learned.

Well, in binary, we must also learn a multiplication table, this one:

x	0	1
0	0	0
1	0	1

← multiplication table that "baby computers" must memorize !!!

Much easier than decmal.

So how do we do binary multiplication?

eg: $\begin{array}{r} 13 \\ \times\ 10 \\ \hline 130 \end{array}$ = ..01101 ← 13

x 01010 ← 10.

```
      0000
     1101
    0000
+  1101
-----------
 1000.0010
 1       1
 128     + 2  = 130.
```

Division

(1) Decimal division:

eg 1:

$$\begin{array}{r} 033 \\ 7 \overline{)\,233\,7} \\ 21\,\downarrow \\ \hline 27 \\ 21 \\ \hline 6 \end{array}$$

use multiplication table very effectively.

skip (waste of time).

use this

Suppose:

eg 2:

$$\begin{array}{r} 82 \\ 27 \overline{)\,2123\,7} \\ 216\,\downarrow \\ \hline 77 \\ 54 \\ \hline 23 \end{array}$$

??? very hard!

$8 \times 27 = 216$

we have to "guess" (try) one of 10 ~~digit~~ outcomes

0, 1, 2, ..., 9.

Binary:

eg 1: $(237)_{10} = (11101101)_2$

$(7)_{10} = (111)_2$

```
                32 16  8  4  2  1
              ---------------------
              | 0  0  1  0  0  0  1      = 33 (Quotient)
          ----|
      111 |   1  1  1  0  1  1  0  1
          ----|
              | 1  1  1  .  .  .  .  .
              ---------------------
                 0  0  1  1  0  1
                    ----------
                       1  1  1
                    ----------
                       1  1  0      = 6 (remainder).
```

---

eg 2:

$(27)_{10} = (11011)_2$

$(2237)_{10} = (100010111101)_2$

(division is on next page)

64 32 16 8 4 2 1

0 0 0 0 0 1 0 1 0 0 1 0 = 82

```
11011 | 1 0 0 0 1 0 1 1 1 0 1
      - 1 1 0 1 1
        0 0 1 1 0 1 1
            1 1 0 1 1
          -------------
              1 0 0 1 1 0
                1 1 0 1 1
              -----------
                  = 23
```

0 1 0 1 1 1

16 8 4 2 1

1	64
2	16
4	2
16	82
23	

do it

SKIP

Enter

Other number systems

- The rules of arithmetic are same in every number system.
- For example, let's do add & multiplication in the number system with base 5.

- Addition in base 5:

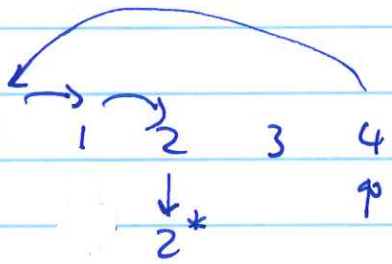
$$(234)_5 = 2 \times 5^2 + 3 \times 5^1 + 4 * 5^0$$
$$= 2 \times 25 + 3 \times 5 + 4 \times 1$$
$$= 50 + 15 + 4$$
$$= 69$$

$$(223)_5 = 2 \times 25 + 2 \times 5 + 2 \times 1$$
$$= 50 + 10 + 3$$
$$= 63$$

Base 5

$$\begin{array}{r} (234)_5 \\ +\ (223)_5 \\ \hline (1012)_5 \end{array}$$

4+3:



$125 + 5 + 2 = 132$

$$\begin{array}{r} 69 \\ 63 \\ \hline 132 \end{array}$$

correct.

Multiplication in base 5.

you need to "learn" the multiplication table first:

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	11	13
3	0	3	11	14	22
4	0	4	13	22	31

$(334)_5 = 3\times25 + 3\times5 + 4\times1$
$= 75 + 15 + 4 = 94$

$(31)_5 = 3\times5 + 1$
$= 15 + 1 = 16.$

$$\begin{array}{r} 334 \\ \times\ 31 \\ \hline 334 \\ +\ 21120 \\ \hline (22004)_5 \end{array}$$

625 125 25 5 1

$3\times4 = 22 \rightarrow$ carry 2
$3\times3 = 14 + 2 = 21 \rightarrow$ carry 2
$3\times3 = 14 + 2 = 21$

$2\times625 + 2\times125 + 4$
$= 1250 + 250 + 4 = 1504.$

$$\begin{array}{r} 94 \\ 16 \\ \hline 564 \\ 940 \\ \hline 1504 \end{array}$$

correct.

Other commonly used radices: 8 & 16.

Octal numbers: radix 8.

- uses the symbols: 0, 1, 2, 3, 4, 5, 6, 7.

Hexadecimal numbers: radix 16

- uses the symbols: 0, 1, 2, .., 9, A, B, C, D, E, F.
  '10' '11' '15'

- Q: if we write "10", can you tell me the value of this number?

  No.

- Q: what's missing.

  → Radix.

Binary "10" = 2
Decimal "10" = 10
Octal "10" = 8
Hexadec. "10" = 16.

Converting between octal ↔ decimal, hexadec. ↔ decimal

- Same technique is used.

octal → decimal

- $(354)_8 = (?)_{10}$.

- Answer:

$$(354)_8 = 3.(8^2) + 5.(8^1) + 4.(8^0)$$
$$= 3.(64) + 5(8) + 4.$$
$$= 192 + 40 + 4$$
$$= (236)_{10}$$

decimal → octal

- $(236)_{10} = (?)_8$.

$$8 \overline{)\ 236}\ 4$$
$$8 \overline{)\ 29}\ 5$$
$$8 \overline{)\ 3}\ 3.$$
$$0$$

$(354)_8$.

$$8 \overline{)\ 236}\ 4.$$
$$\underline{16}$$
$$76$$
$$\underline{72}$$
$$4.$$

$$8 \overline{)\ 29}\ 5.$$
$$\underline{24}$$
$$5.$$

$$8 \overline{)\ 3}\ 3.$$

Hex → dec

- $(AF4)_{16} = (?)_{10}.$

$$(AF4)_{16} = 10(16^2) + 15(16^1) + 4(16^0).$$

$$= 10(256) + 15(16) + 4.$$

$$= 2560 + 240 + 4$$

$$= 2804.$$

Dec → hex

- $(2804)_{16} = (?)_{10}.$

16 | 2804 — 4

16 | 175 — 15 . = F

16 | 10 — 10. = A

0

↑

$(AF4)_{(16)}.$

$$\begin{array}{r|l} & 175 \\ 16 & 2804 \\ & 16 \\ \hline & 120 \\ & 112 \\ \hline & 84 \\ & 80 \\ \hline & 4. \end{array}$$

$$\begin{array}{r|l} & 10 \\ 16 & 175 \\ & 16 \\ \hline & 15 \end{array}$$

Converting binary ↔ octal, binary ↔ Hexadec. numbers.

- Best group the bits.

$8^1 = 2^3$

1 digit in octal $\equiv$ 3 digits in binary

eg:

$$(1010.0110)_2 = (?)_8.$$

Group by 3 bits:

$$\underbrace{1010}_{\|}.\underbrace{0110}_{\|}$$
$$2 \quad . \quad 4 \quad 6$$

$110 = 2^2 + 2^1 = 4 + 2 = 6$

$100 = 2^2 = 4.$

$010 = 2^1 = 2.$

Thus:

$$(1010.0110)_2 = (246)_8.$$

Check:

$(1010.0110)_2 = 128 + 32 + 4 + 2 = 166$

$(246)_8 = 2(64) + 4(8) + 6.$
$= 128 + 32 + 6 = 166.$

Conversely:

$$(246)_8 = (?)_2. \quad \rightarrow \text{ expand to bits.}$$

$6 = 110$

$4 = 100$

$2 = 010.$

$$\Rightarrow \quad (010.100.110)_2.$$

- Same methode for Hexadecimal numbers:

$$(1010.0110)_2 = (?)_{16}.$$

Group by 4 bits:

$$\underbrace{1010}_{\text{A}}.\underbrace{0110}_{6}$$

$$A \quad 6. \quad \rightarrow \quad (A6)_{16}.$$

$0110 = 2^2 + 2^1 = 6.$

$1010 = 2^3 + 2^2 = 10. = A$

Conversely:

$$(A6)_{16} = (?)_2.$$

$A = 1010$

$6 = 0110.$

$$\rightarrow \quad (1010.0110)_2.$$