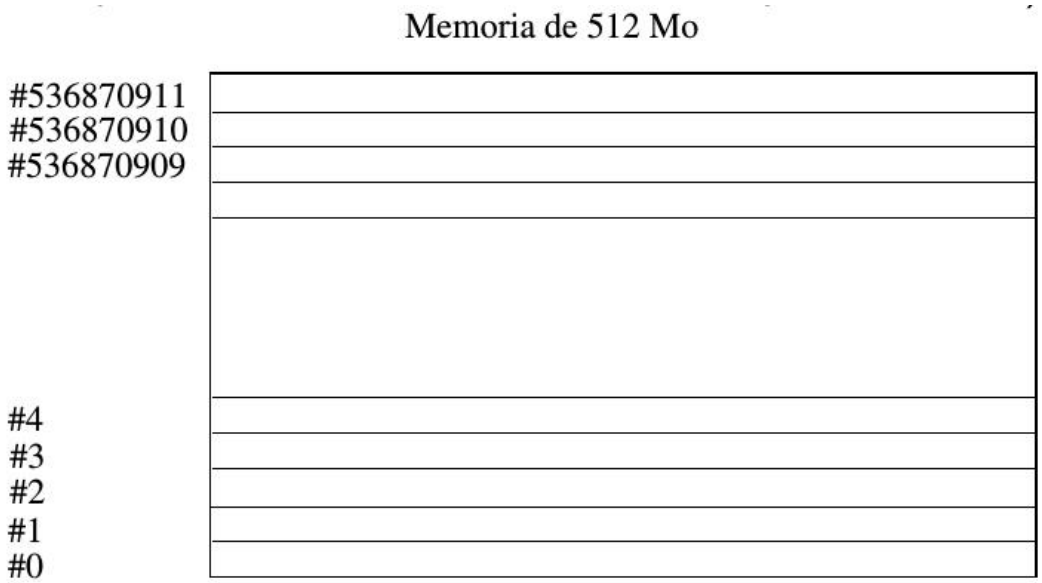
C Basics

Angel Noé Martínez González

June 16, 2015

Data Types I

The memory can be viewed as a bytes serie, directionable components; each byte has their unique direction in memory (32 bits in 32 bits machine)



Data Types II

- Generally speaking, a k-bits system has registers and buses of k-bits. We can have a system manipulator of 32 bits on a OS of 64 bits but not otherwise.
- A data type defines: number of bytes to use for a data and the way to use each byte.
- Elemental types: **characters**, **integers** and **floating points** (for real numbers).
- There is no standard in data types size but

$$1 == \text{sizeof(char)} \leq \text{sizeof(short)} \leq \text{sizeof(int)} \leq \text{sizeof(float)} \leq \text{sizeof(double)} \leq \text{sizeof(long double)}$$

sizeof(x) returns the bytes number of the variable x: variable type or only type.

Data Types III

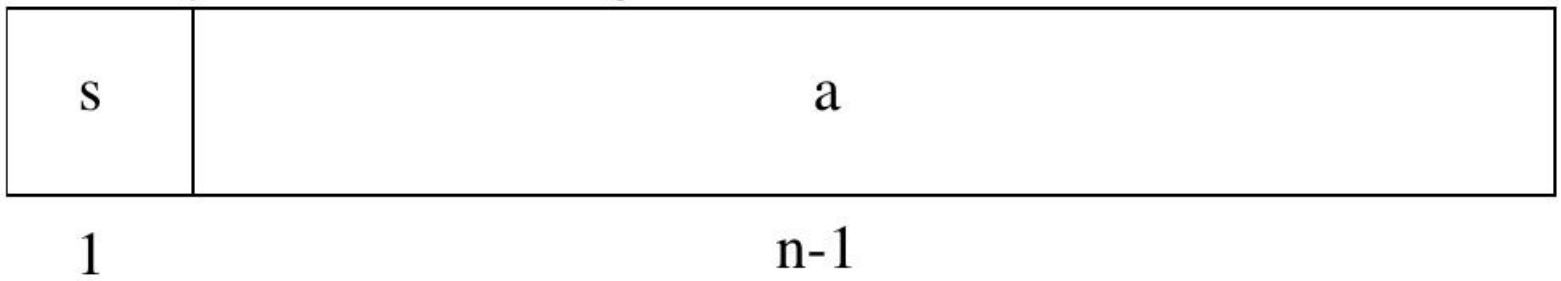
In a 32 bits machine

char	1	$[-128, 127]$
short	2	$[-32768, 32767]$
int	4	$[-2147483648, 2147483647]$
long	4	$[-2147483648, 2147483647]$
float	4	$[1.18 \times 10^{-38}, 3.4 \times 10^{38}]$
double	8	$[2.2 \times 10^{-308}, 1.8 \times 10^{308}]$
long double	10	$[1.18 \times 10^{-4932}, 3.4 \times 10^{4932}]$
apuntadores	4	$[0, 2^{32} - 1]$

unsigned of a type take only the positive values.

Integer Types I

To represent a subset on $\mathbb{N}$



For n bits to represent the number

- The most important bit is for the sign: $s = 0$ for positive
- A positive number presented in base 2 over $n - 1$ bits

$$a = \sum_{i=0}^{n-2} a_i 2^i$$

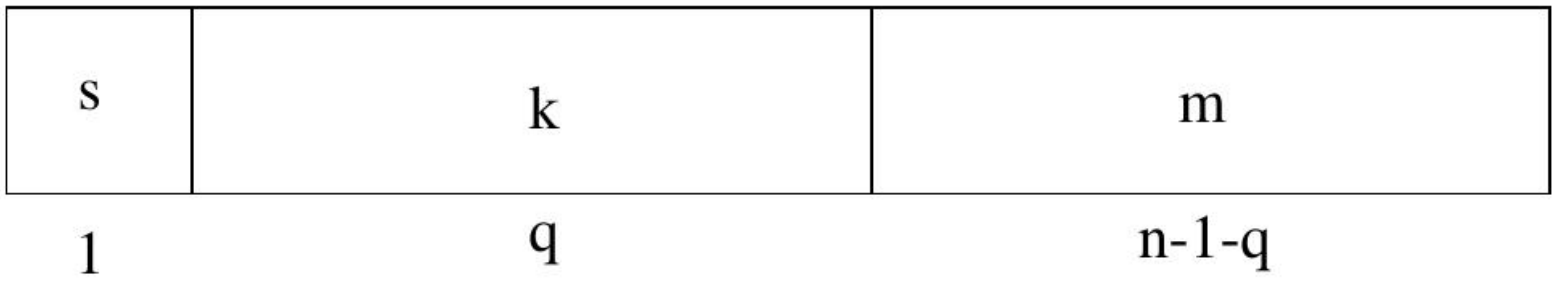
Integer Types II

Negative integers

- One's complement. Used to represent negatives but there are some issues.
- Two's complement. Used for a faster sum of numbers.
- Only one representation of 0.
- Basically is the one's complement plus 1

$$a = \sum_{i=0}^{n-1}(1 - a_i)2^i + 1 = 2^n - |a|$$

Floating Point Types I



- Standard IEEE 754.
- A number is represented as

$$f = s.m.b^e$$

$$e = k - (2^{q-1} - 1); 0 \leq m \leq b$$

- Floats: $m = 23, e = 8, s = 1$
- Doubles: $m = 52, e = 11, s = 1$

Floating Point Types II

Special Cases

- The standard also defines 3 extra numbers: NaN, $-$ inf and $+$ inf.
- It uses the two extremes of the exponent to represent them.

There are lot of more on the floating point numbers to study. *http://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html*. We may take one class specially for that topic if you're interested.

C programming Language I

- Born in 1972 at Bell labs by Dennis Richie and Ken Thompson.
- Initially to develop UNIX.
- Normalized by ANSI in 1989: C89 or ANSI C.
- Recovered by ISO in 1990: C90.
- C99 in 1999 with a lot of new stuff, very close to C++.
- New standard C11 in 2011 lots of new stuff.

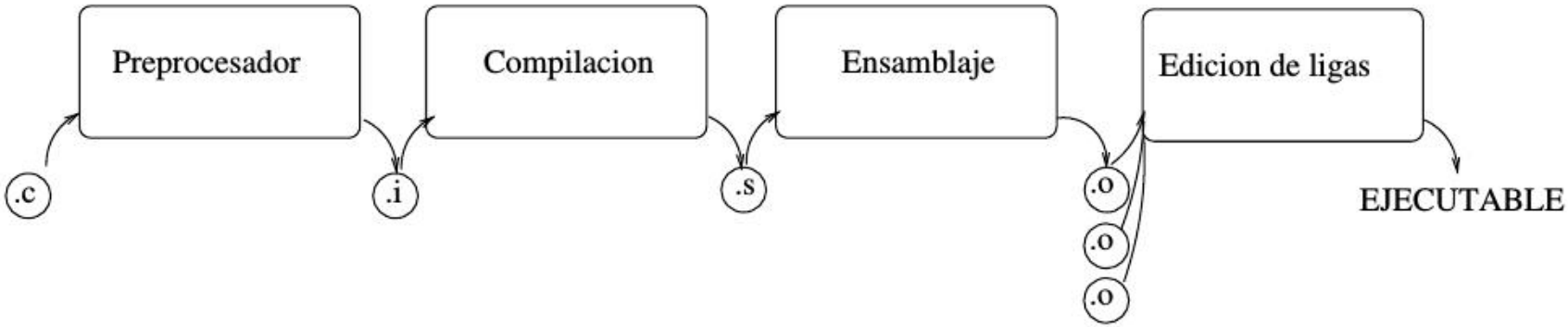
C programming Language II

Features

- Compiled language.
- Low level language.
- Efficient.
- Adapted to system operations: direct access to memory and control of low level process.
- Require large efforts from developer side.
- Reduced standard library.
- Not object oriented programming language.

C PROGRAMMING LANGUAGE III

Compiling



- Preprocessor: *#define*, *#include*.
- Compiling: transforms code into assembly code.
- Assembly transforms assembler code into machine code.
- Link editing combines object files into one executable.

C programming Language IV

Language elements

- Identifiers: variable names, struct names, labels, functions names, etc.
- Keywords: type names (*,&,...), control structures (while, if,...), types classifiers (int, float,...).
- Constant names and macros (from preprocessor) included by DEFINE.
- Strings.
- Operators (+,*,-,...).
- Punctuation(;).
- Comments.

C programming Language V

- Expressions: **int** a; **double** d;
- Variable definition: a=1; d = 2.1*a+a*a;
- Integer constants:
  - Decimal: **int** x =100;
  - Octal (introduced by 0): **int** x = 0144;
  - Hexadecimal (introduced by 0x); **int** x = 0x64;
- Flating constants. By default are double. We can modify them to **long double** (L) or **float** (F)
  - 12.34 **double**
  - 12.3e-4 **double**
  - 12.34F **float**
  - 12.34L **long double**

Functions

General form

```
type functionname(type1 arg1, type2 arg2,...){
    variable declaration scope;
    instructions;
    return somethingOfTypetype;
}
```

- Are the structuring base of a program.
- At running time they are inside the stack.
- main() is the topest function in the stack.

Operators I

Arithmetic

- +,-,*,/,%.
- lvalue: is a value to which we can assign a memory address.
- rvalue: are typically operation results.
- Relation operators: $>, <, <=, >=, ==, !=$

Operators II

Logic

- &&: AND, || : OR, !: NOT.
- The evaluation returns **int** the value of 1 means true, 0 for false.
- Left to right evaluation in case of a group of expressions.

```
if ( (i>=0) && (i<=9) &&
     !(a[i] == 0) || (a[i]>3) ){
     ....
```

Operators III

- Think about the order, better to stand in the following way to gain efficiency

```
if ( mostFrecuentlyViolatedCondition && ... &&
     lessFrecuentlyViolatedCondition ){
     ....
```

Operators IV

Bitwise logic operator

- &: AND, | : OR, $\sim$: NOT, ^ :XOR.
- $>>$, $<<$ shift over the bits to the left and to the right. Add zeros at the end or to the right.
- These are not boolean operators

Operators V

Applications

- Hint: use these operators only to unsigned type data.
- Multply or divide by power of two.