# Introduction to Programming

Variables, Assignment, Expressions, Logical Expressions

Hye-Chung Kum

Population Informatics Research Group

http://research.tamhsc.edu/pinformatics/

http://pinformatics.web.unc.edu/

**Course URL:**
http://pinformatics.tamhsc.edu/phpm672

POPULATION INFORMATICS RESEARCH GROUP

# Learn to fish

- Teach you how to fish
  - New software become available
  - Software version upgrade
- Give you good problems (lab & assignment) to learn to fish on your own
  - Basic tools only
  - You need to practice thinking in algorithms
- Available when you get stuck
- Reading: READ sections in the recommended book
- Top (problem) down(data) vs bottom up
  - Need to iterate

# Example mini-computer

| CPU (Processor) | RAM |
|---|---|
| | 00001000 |
| | 01110101 |
| | 10010001 |
| | ... |
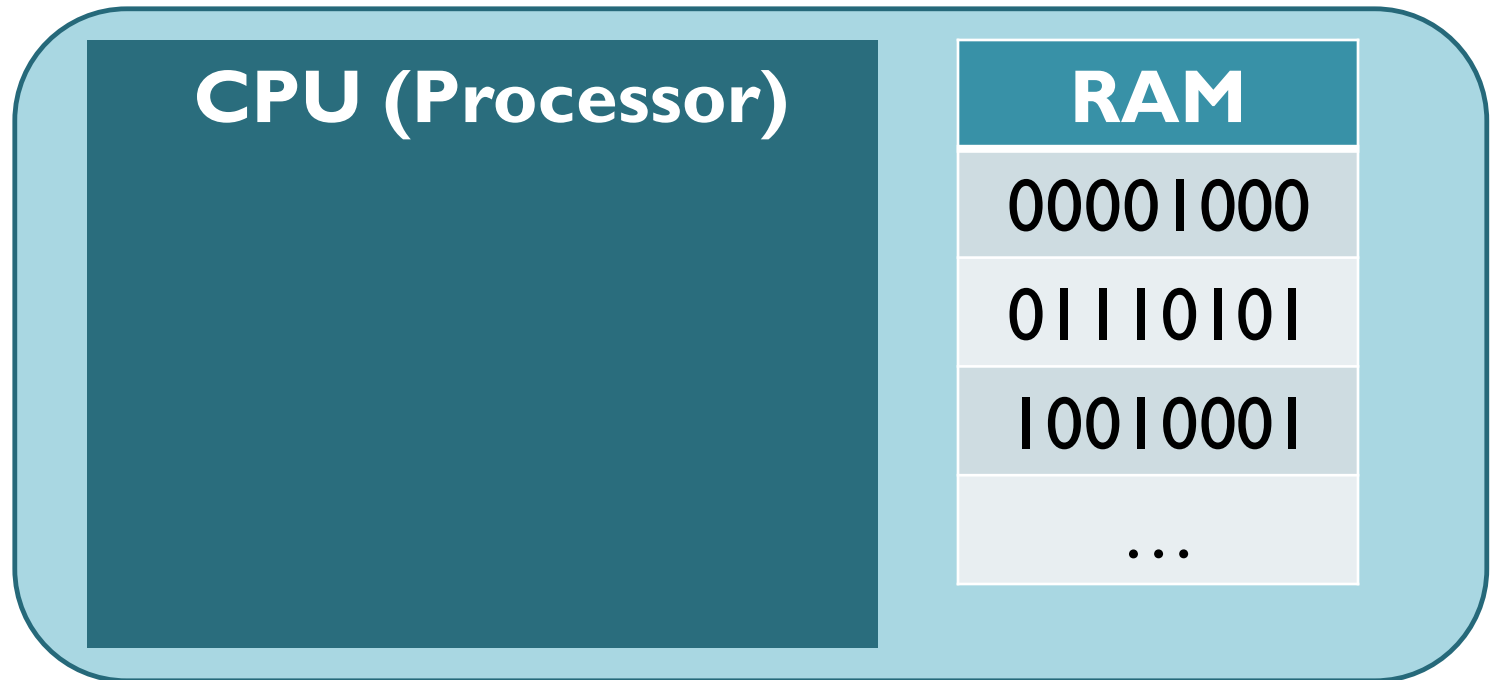
- CPU (Processor): Type of instructions it can run
  ◦ Example CPU:  Intel(R) Core(TM) i7 CPU Q720 @ 1.6 GHz
- RAM: memory
  ◦ 16 GB / 8GB / 4GB / 2GB
- Hard drive: permanent memory for storage

# Example mini-computer

**CPU (Processor)**

- Instruction set (2 bit)
  - 00: Save to
  - 01: Retrieve from
  - 10: Add
  - 11: Subtract

**RAM**

| |
|---|
| 00001000 |
| 01110101 |
| 10010001 |
| … |

- 5 * 3 = ?

# Example mini-computer

**CPU (Processor)**

- Instruction set (2 bit)
  - 00: Save to
  - 01: Retrieve from
  - 10: Add
  - 11: Subtract

**RAM**

| 00001000 |
| 01110101 |
| 10010001 |
| … |

- 5 * 3 = ?
  - Add 5
  - Add 5
  - Add 5

# Example mini-computer

**CPU (Processor)**

- Instruction set (2 bit)
  - 00: Save to
  - 01: Retrieve from
  - 10: Add
  - 11: Subtract

| RAM |
| :-: |
| 00001000 |
| 01110101 |
| 10010001 |
| ... |

- 5 * 3 = ?
  - Add 5
  - Add 5
  - Add 5

| Address | Instruction | Operand |
| :-: | :-: | :-: |
| 00 | 10 | 0101 |
| 01 | 10 | 0101 |
| 10 | 10 | 0101 |

# Example mini-computer

**CPU (Processor)**                    **RAM**

### Binary Numbers

1001            0001            0101
8 4 2 1          8 4 2 1          8 4 2 1

8*1+1*1=9        1*1=1        4*1+1*1=5

- 5 * 3 = ?
  ◦ Add 5
  ◦ Add 5
  ◦ Add 5

| Address | Instruction | Operand |
|---------|-------------|---------|
| 00      | 10          | 0101    |
| 01      | 10          | 0101    |
| 10      | 10          | 0101    |

# Example mini-computer

**CPU (Processor)**

- Instruction set (2 bit)
  - 00: Save to
  - 01: Retrieve from
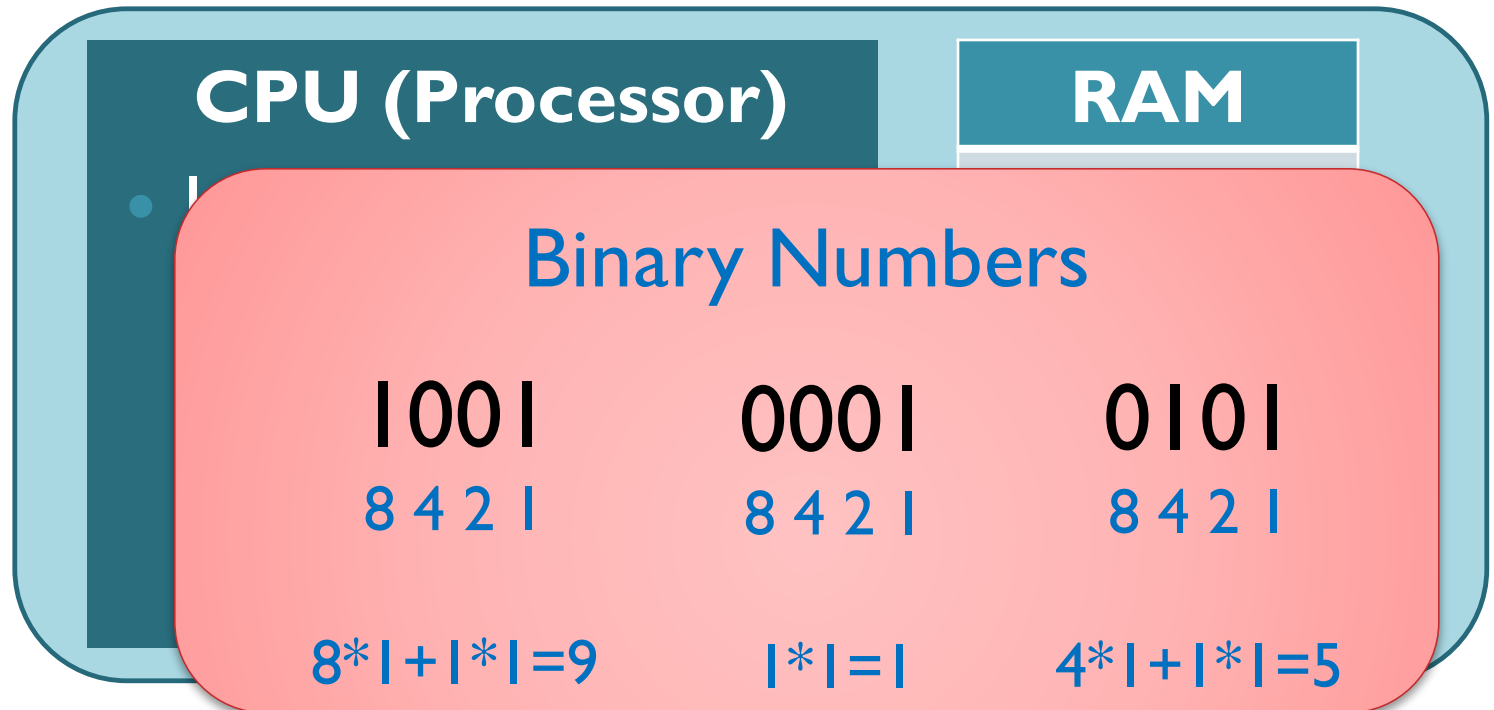  - 10: Add
  - 11: Subtract

| RAM |
|---|
| 00100101 |
| 01100101 |
| 10100101 |
| ... |

- 5 * 3 = ?
  - Add 5
  - Add 5
  - Add 5

| Address | Instruction | Operand |
|---|---|---|
| 00 | 10 | 0101 |
| 01 | 10 | 0101 |
| 10 | 10 | 0101 |

# Example mini-computer

**CPU (Processor)**

- Instruction set (2 bit)
  - 00: Save to
  - 01: Retrieve from
  - 10: Add
  - 11: Subtract

**RAM**

00100101

01100101

10100101

…

- 5 * 3 = ?
  - Add 5
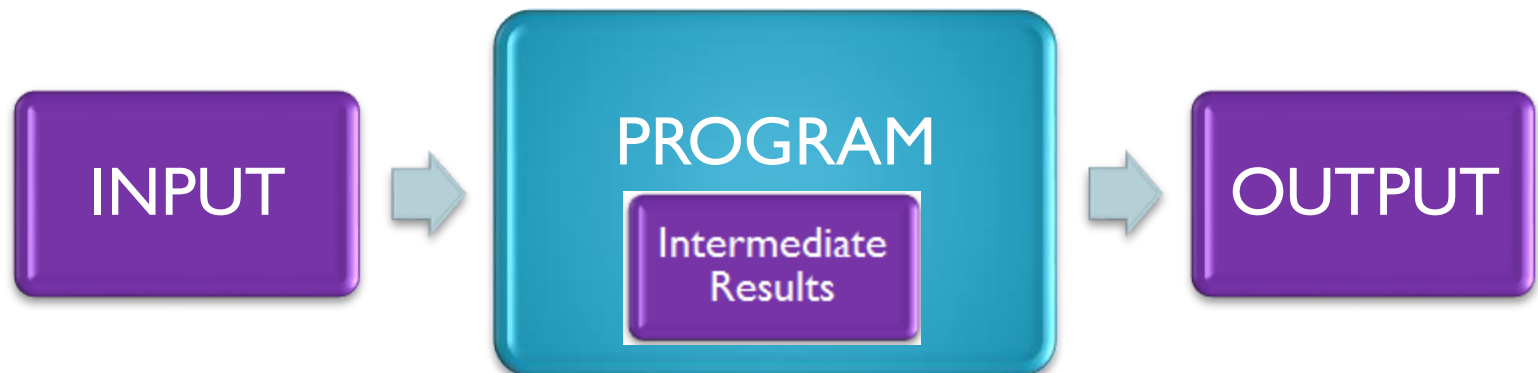  - Add 5
  - Add 5

Higher level language
Keyword
Vocabulary of language
SAS: proc/data/print

# Programming

- OUTPUT : Know what you want
- INPUT : what you have
- Intermediate results: What you need
- Program: change what you have (INPUT) to what you need (intermediate results. Often more than one level) to what you want (OUTPUT)

INPUT → PROGRAM (Intermediate Results) → OUTPUT

# Few tricks

- Divide & Conquer
  - Write code to do small things.
  - Combine the small pieces
- Look at memory (table) after each step
  - proc print data=fn(obs=10); where condition
- Become good with an editors
  - emacs, vi, internal editors
  - copy & paste/find/replace
- Regular expression/ wild card
  - *.sas; [optional]
- grep expression files: find things in text files
- diff fn1 fn2: compare two programs

# What we are going to learn

- Programming
  - Variables  Naming rules & Naming guidelines
  - Data Types  int double string binary
  - Expressions
  - Logical Expressions
- Operators
  - Logical  (~ / !), (& / and), (| / or)
  - Relational  <, <=, ==, >, >
- Learn Conditional programming
  - if then else end
- Common Pitfalls

# What is a **Variable**?

- A user defined name to represent a piece of memory for storing evaluated value(s). A variable consists of 5 items

*Name:*

meaningful human readable name

How the user refers to variable

*Data Type:*

How to interpret variable for data representation

*Size:*

How much storage memory is needed to store data value

Can be inferred from data type

**Value:**

Actual value associated with variable

stored in memory

*Storage location:*

Usually hidden from user by the interpreter or compiler

How the computer refers to a variable

*For Our Purposes: Columns*

Many variables. A columns of variables

# Variable, cont.

| Name | Data Type | Size | Memory Location (hidden from user) | Value |
|------|-----------|------|-----------------------------------|-------|
| Radius | float32 | 4 bytes | 0x1800F040 | 3.23 |
| currKey | char | 1 byte | 0x1800F049 | 'k' |
| firstName | string | 6 bytes | 0x1800B0E0 | "shawn" |
| width | int32 | 4 bytes | 0x1800CCE8 | 800 |
| type | int8 | 1 byte | 0x1800CCE7 | 27 |

- var label;
- value label (interpretation)
- SAS: proc contents

# **Naming Rules**
# Use **Valid** Names

- Length: reasonably short (8) but descriptive
- Syntax: similar to userid
  - ◦ Starts with a single letter followed by any number of letters, digits, or underscores.
  - ◦ Digits `[0-9]`, Letters `[a-zA-Z]`, Underscore '_'
  - ◦ Capitalization
    - STATA: differentiate
    - SAS: does not differentiate
    - Best to not use (too confusing for people)
- No spaces allowed
  - ◦ _ or camelCase

# Naming Rules, cont
## write program for people

- Avoid Keywords (if, else, while, for, …)
  - **Result:** Error / confusing
- Use **Meaningful** names
  - `currStudent` better than `fido`, `purpleSloth`, or `currItem`
- Write **readable** names
  - currStudent better than (cS, crSt, or crrStdnt)
- Convention
  - b_: binary (bincome, b_income, bIncome)
  - n_: number(nincome, n_income)
  - c_:string / character (cincome, c_income)
  - g_:groups (gincome)

# What is a **Data Type**?

- How to interpret a storage location to retrieve the correct value.
- Integer, Floating point, Logical, Char, Strings are typical data types
- Other languages require you to explicitly specify the data type of variables
- SAS implicitly infers the data type from the first initialization(use) via the specified expression.
    - Number/Char
    - String static

# Data Types : 8 bits = 1 bytes

| Data Type | Size (Bytes) | Min | Max | Notes |
|---|---|---|---|---|
| logical | 1 | 0 (false) | 1 (true) | |
| Int8 (sign bit + 7 bits) | 1 | $-128$ $=1111111=2^7$ | $+127$ $=1111111-1$ $=2^7-1$ | Numeric, integer, Exact |
| int16/long | 2 | $-32768$ | $+32767$ | Ditto |
| single | 4 | $-3.4028e+038$ | $+3.4028e+038$ | Numeric Real Approximate |
| double | 8 | $-1.7977e+308$ | $+1.7977e+308$ | Ditto |
| char | 2 | N/A | N/A | Encoded character |
| string | Varies len+1 | N/A | N/A | String of encoded characters |

# ASCII: character encoding

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | <NUL> | 32 | <SPC> | 64 | @ | 96 | ` | 128 | Ä | 160 | † | 192 | ¿ | 224 | ‡ |
| 1 | <SOH> | 33 | ! | 65 | A | 97 | a | 129 | Å | 161 | ° | 193 | i | 225 | · |
| 2 | <STX> | 34 | " | 66 | B | 98 | b | 130 | Ç | 162 | ¢ | 194 | ¬ | 226 | , |
| 3 | <ETX> | 35 | # | 67 | C | 99 | c | 131 | É | 163 | £ | 195 | √ | 227 | „ |
| 4 | <EOT> | 36 | $ | 68 | D | 100 | d | 132 | Ñ | 164 | § | 196 | ƒ | 228 | ‰ |
| 5 | <ENQ> | 37 | % | 69 | E | 101 | e | 133 | Ö | 165 | • | 197 | ≈ | 229 | Â |
| 6 | <ACK> | 38 | & | 70 | F | 102 | f | 134 | Ü | 166 | ¶ | 198 | Δ | 230 | Ê |
| 7 | <BEL> | 39 | ' | 71 | G | 103 | g | 135 | á | 167 | ß | 199 | « | 231 | Á |
| 8 | <BS> | 40 | ( | 72 | H | 104 | h | 136 | à | 168 | ® | 200 | » | 232 | Ë |
| 9 | <TAB> | 41 | ) | 73 | I | 105 | i | 137 | â | 169 | © | 201 | … | 233 | È |
| 10 | <LF> | 42 | * | 74 | J | 106 | j | 138 | ä | 170 | ™ | 202 | | 234 | Í |
| 11 | <VT> | 43 | + | 75 | K | 107 | k | 139 | ã | 171 | ´ | 203 | À | 235 | Î |
| 12 | <FF> | 44 | , | 76 | L | 108 | l | 140 | å | 172 | ¨ | 204 | Ã | 236 | Ï |
| 13 | <CR> | 45 | - | 77 | M | 109 | m | 141 | ç | 173 | ≠ | 205 | Õ | 237 | Ì |
| 14 | <SO> | 46 | . | 78 | N | 110 | n | 142 | é | 174 | Æ | 206 | Œ | 238 | Ó |
| 15 | <SI> | 47 | / | 79 | O | 111 | o | 143 | è | 175 | Ø | 207 | œ | 239 | Ô |
| 16 | <DLE> | 48 | 0 | 80 | P | 112 | p | 144 | ê | 176 | ∞ | 208 | – | 240 |  |
| 17 | <DC1> | 49 | 1 | 81 | Q | 113 | q | 145 | ë | 177 | ± | 209 | — | 241 | Ò |
| 18 | <DC2> | 50 | 2 | 82 | R | 114 | r | 146 | í | 178 | ≤ | 210 | " | 242 | Ú |
| 19 | <DC3> | 51 | 3 | 83 | S | 115 | s | 147 | ì | 179 | ≥ | 211 | " | 243 | Û |
| 20 | <DC4> | 52 | 4 | 84 | T | 116 | t | 148 | î | 180 | ¥ | 212 | ` | 244 | Ù |
| 21 | <NAK> | 53 | 5 | 85 | U | 117 | u | 149 | ï | 181 | µ | 213 | ' | 245 | ı |
| 22 | <SYN> | 54 | 6 | 86 | V | 118 | v | 150 | ñ | 182 | ∂ | 214 | ÷ | 246 | ^ |
| 23 | <ETB> | 55 | 7 | 87 | W | 119 | w | 151 | ó | 183 | Σ | 215 | ◊ | 247 | ~ |
| 24 | <CAN> | 56 | 8 | 88 | X | 120 | x | 152 | ò | 184 | Π | 216 | ÿ | 248 | ‾ |
| 25 | <EM> | 57 | 9 | 89 | Y | 121 | y | 153 | ô | 185 | π | 217 | Ÿ | 249 | ˘ |
| 26 | <SUB> | 58 | : | 90 | Z | 122 | z | 154 | ö | 186 | ∫ | 218 | ⁄ | 250 | ˙ |
| 27 | <ESC> | 59 | ; | 91 | [ | 123 | { | 155 | õ | 187 | ª | 219 | € | 251 | ˚ |
| 28 | <FS> | 60 | < | 92 | \ | 124 | | | 156 | ú | 188 | º | 220 | ‹ | 252 | ¸ |
| 29 | <GS> | 61 | = | 93 | ] | 125 | } | 157 | ù | 189 | Ω | 221 | › | 253 | ˝ |
| 30 | <RS> | 62 | > | 94 | ^ | 126 | ~ | 158 | û | 190 | æ | 222 | fi | 254 | ˛ |
| 31 | <US> | 63 | ? | 95 | _ | 127 | <DEL> | 159 | ü | 191 | ø | 223 | fl | 255 | ˇ |

# Data Types

| Type | Stored value | Interpreted value | Label Interpreted Value |
|---|---|---|---|
| int | 1000001 (65) | 65 | 65 or older |
| Char/string (ASCII) | 1000001 (65) | A | Asian |
| date | 1000001 (65) | 1960/3/6 (SAS) | |

- 1    0    0    0    0    0  1  =64+1=65
- 64  32  16    8    4    2  1

# Variable Type

- Number
  - Int (long), real (double, float), date time
- String/Character
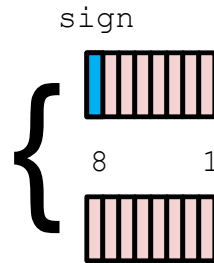  - Length matters
- Missing
  - . '.'
  - "
  - SAS: **.<0**

# **Integer** Number Representations

conversion functions `intmin, intmax`

**int8**
8-Bit Integer
**uint8**

sign

8    1

$[-2^7, +2^7-1] = [-128, +127]$

$[0, +2^8-1] = [0, +255]$

**int16**
16-Bit Integer
**uint16**

sign

16              1
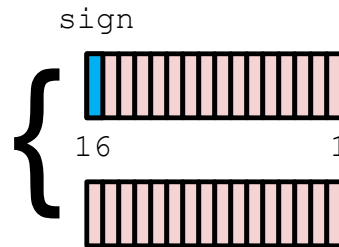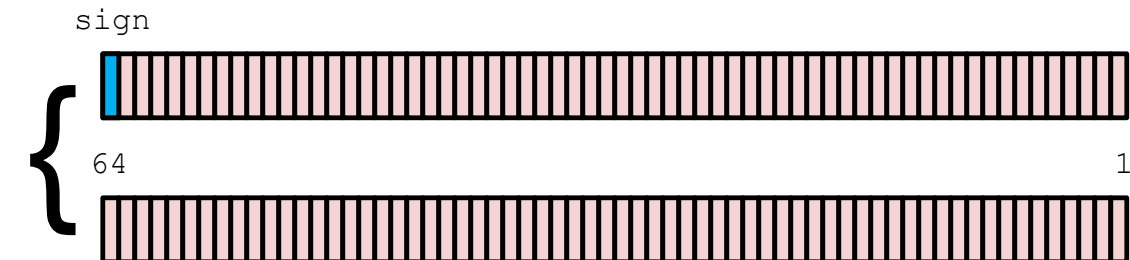
$[-32,768  +32,767]$

$[0  65,535]$

**int32**
32-Bit Integer
**uint32**

sign

32                          1
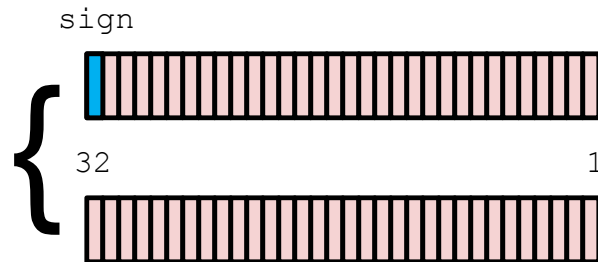
$[-2^{31}, +2^{31}-1]$

$[0, +2^{32}-1]$

**int64**
64-Bit Integer
**uint64**

sign

64                                                                1

# Integer Issues

- **Overflow,** expression tries to create an integer value larger than allowed valid range `[min,max]`
  - `x = int8( 127 ) + 1`
- **Truncation**, fractions not supported
  - `int16(23)/int16(5) = 5` **not** `4.6`
  - Rounds result to nearest whole number

# **Real** Number Representations
## IEEE 754 Floating point standard

- **Reals** (http://kipirvine.com/asm/workbook/floating_tut.htm)
  - Sign bit  *(1 bit) : + / -*
  - Exponent *(7 or 11 bits) : biased by 127 = exp-127*
  - Mantissa (fraction)  *(23 bits or 52 bits):* $\frac{1}{2}+\frac{1}{4}+\frac{1}{8}$ ...
  - *(+/-) (          127)*
    =
  - **Singl**
  - **Double**

### Binary Numbers

| 1001 | 1 0 0 1 |
|---|---|
| 8 4 2 1 | 1/2 1/4 1/8 1/16 |
| 8*1+1*1=9 | 0.5+0.0625=0.5625 |

63          52                                    0

# **Real** Number Representations
## IEEE 754 Floating point standard
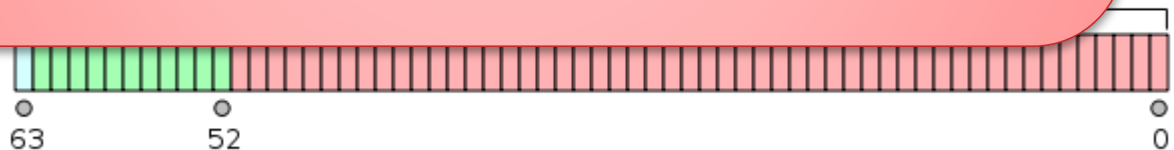
- **Reals** (http://kipirvine.com/asm/workbook/floating_tut.htm)
  - ◦ Sign bit *(1 bit) : + / -*
  - ◦ Exponent *(7 or 11 bits) : biased by 127 = exp-127*

Decimal fraction to Binary fraction
**Lose precision**

0.200000000000
= .00110011001100110011001
+ **remainder 0.000000071526**
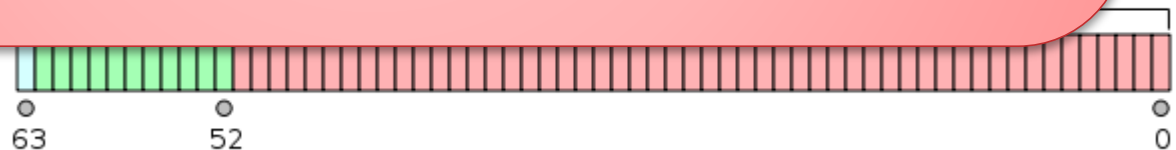
# Real Issues (single, double)

- **Precision Error**  $Error = |actual - representation|$
  - Most numbers don't get represented exactly
  - Finite precision of IEEE floating point
  - Represented by nearest real number
  - Separation between two closest numbers varies over entire range
- **Numeric Stability** (does error overwhelm?)
  - Truncation Errors  $Error = |true\_answer - computed|$
  - Accumulated error from repeated calculations
- **Don't compare real numbers**
  - 3.0 == 3.0 (NOT GOOD)

# Conversion between types

**Conversion:** Use cast function

- Upcast to larger data type, no issue
- Downcast to smaller data type,
  - Truncation & clamping problems
  - Conversion between signed and unsigned as an example
- Conversion from real to integer,
  - truncation to closest integer
- Conversion from integer to real,
  - approximation by nearest real
- Conversion from number to/from string
  - Pay attention

# Declare a variable

- Tell the computer I need room in memory for a certain variable
  - A certain length
  - A certain type
  - With a certain name
  - Optional: Set to an initial value (initialize)
- Length: static vs dynamic
- SAS
  - SAS: implicit when used for the first time
  - Not one variable, but column of variables

# What is **Assignment**?

**<variablename> = <expression>**

- Assigning a value of a specified data type to a storage location in computer memory.

- Variable name on left-hand side

- Expression on right-hand side
  - Expression is evaluated and reduced to a single value
  - Value is stored in storage location associated with variable name

# Assignment

SAS: **\<variablename\> = \<expression\>**

```
x=32;
y=7;
z=sqrt (x^2 + y^2);
```

# What is an **Expression**?

- A mathematical sequence of operators, function calls, variables, numbers, and parenthesis that evaluates to a value

- Examples:
  - `7`
  - `5*(4+3)`
  - `23 + sqrt( -1 ) / (4 – 4j)`

# Numbers and Strings

## SAS (Be careful of Strings getting cutoff)

```
data str2num;
str="123";
num=.;  * declare numeric variable;
num=str;
```

```
data num2str;
num=123;
Str=num;
str=put(num,$3.);
```
*This will cutoff the 4 at the end, because no space to store;
```
str="1234";
```

```
data test;
length str $10.;
set readin;
(or)
str="              ";
```

# SAS: Numbers and Strings

- Use **length**, or explicit declaration when needed
  - **num=.;**
  - **str="            ";**
- Be careful of white space
  - **compress()** will take out white space
- String: static length, so be careful not to cut off values when you get longer strings later.
  - NOTE: Invalid character data, i=110.00 , at line 15 column 10.
  - Must declare a new variable with longer length, then copy over all values
  - Try running string.sas (course website)

# Variables, Types, Assignments, Expressions



… Moving onto logical expressions
& conditional programming (after 10:15 next week)