The Journey of a Source Line (1)

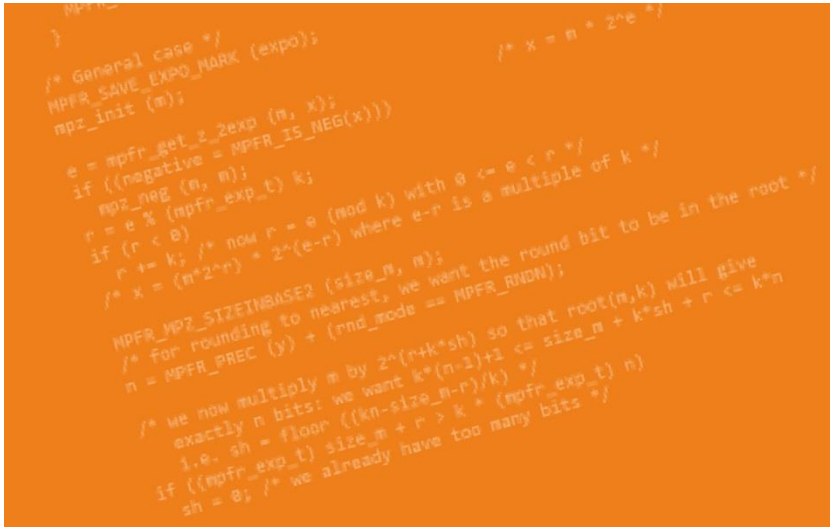
CERN Academic Training – Feb 2018

Andrzej Nowak

http://tik.services

tik. technology innovation knowledge

http://tik.services


/* General case */
MPFR_SAVE_EXPO_MARK (expo);
mpz_init (m);
e = mpfr_get_z_2exp (m, x);
if ((negative = MPFR_IS_NEG(x)))
mpz_neg (m, m);
r = e % (mpfr_exp_t) k;
if (r < 0)
r += k; /* now r = e (mod k) with 0 <= e < r */
/* x = (m*2^r) * 2^(e-r) where e-r is a multiple of k */
MPFR_MPZ_SIZEINBASE2 (size_m, m);
/* for rounding to nearest, we want the round bit to be in the root */
n = MPFR_PREC (y) + (rnd_mode == MPFR_RNDN);
/* we now multiply m by 2^(r+k*sh) so that root(m,k) will give
exactly n bits: we want k*(n-1)+1 <= size_m + k*sh + r <= k*n
i.e. sh = floor ((kn-size_m-r)/k) */
if ((mpfr_exp_t) size_m + r > k * (mpfr_exp_t) n)
sh = 0; /* we already have too many bits */

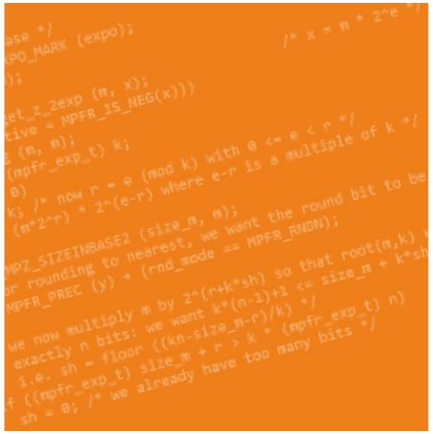

egoSpeed: 8.80 yawRate: -0.10 expDTime: 0.03 egoAccel: -0.24 Day gfi: 3
distance


T+ 00:08:01
STAGE 2
TELEMETRY
SPEED
ALTITUDE
23128
178
FALCON HEAVY TEST FLIGHT


AMD A10-7800 Series
AD780DYBI44JA
JA 1415SUN
9CR848IE40297
DIFFUSED IN GERMANY
MADE IN CHINA
AMD
© 2013 AMD

Image: AMD

Outline



Day 1: Mostly the upper layers & software

Day 2: Mostly the lower layers & hardware

The upper layers

What is a programming language?

Limited in scope

terms, grammar

"easy"

A simple language system designed to facilitate the writing of computer programs

build bigger things

Collins English Dictionary

Source code

```
#include <stdio.h>
int main(int argc, char *argv[]) {
        printf("Hello, world!\n"); return 0;
}

public class Hello {
        public static void main(String argv[]) {
                System.out.println("Hello, world!");
} }
```

~~print "Hello, world!"~~ print("Hello, world!")

What happens to code

Compilation

- Typically C, C++, Fortran, Pascal

Interpretation

- Typically Python, JavaScript, Smalltalk, Matlab, VBA

Recompilation

- VMs, Rosetta, Dynamo(RIO)

Transcompilation

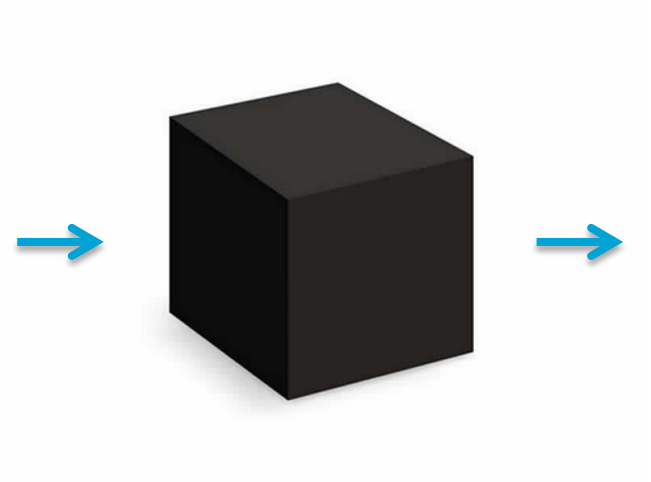
- E.g., C to HDL, ROSE

C


```
#include <stdio.h>

int main(int argc, char *argv[]) {
    printf(“Hello, world!\n”);
    return 0;
}
```

Compilers



Ahead-of-time compilation
OS view

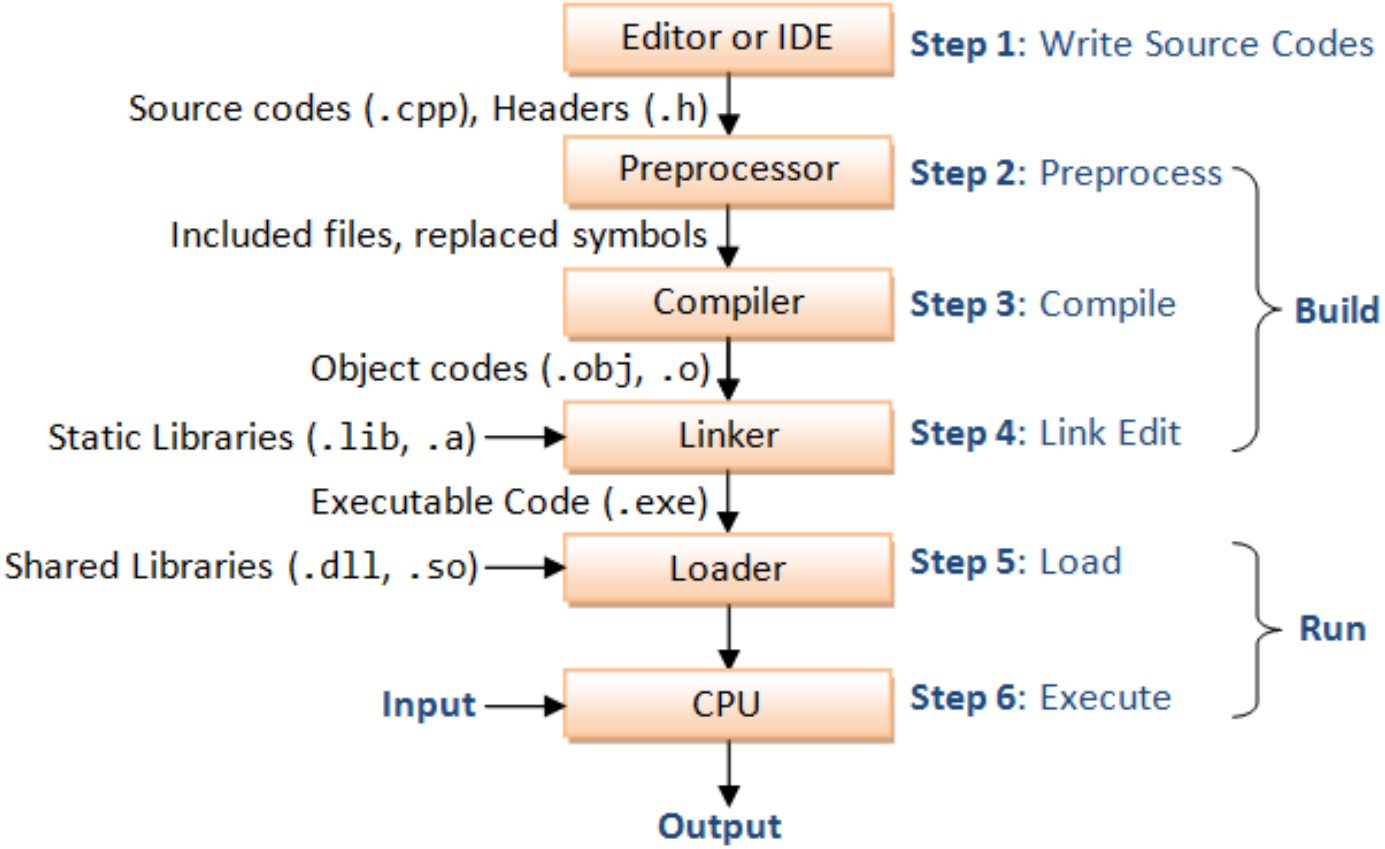


Image: NTU

Preprocessing

- The preprocessor modifies the source, in memory, as specified by preprocessor directives in the source
- Typical tasks:
  - Adding headers
  - Handling macros
  - Removing formatting, comments
- gcc -E

Compiler front- and back-ends

Front-end

- Language focus (Language-specific)
- Machine-independent

Back-end

- Language-independent
- Optimization
- Machine focus (Machine-specific)
  - Architectural analysis
  - Code generation

Ahead-of-time compilation

Compiler view

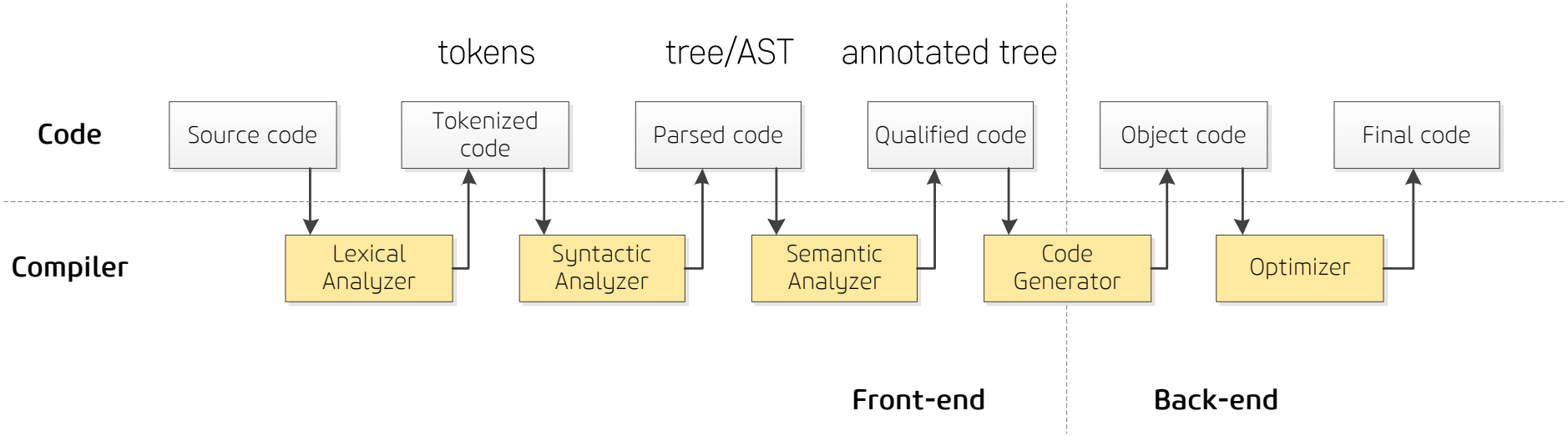


Image: NTU

Intermediate representations

Source file

```
int main () {
  int a,b,c,x,y,z;
  a = b+c;
  x = a+z+y+2;
}
```

Credit: cse.iitb.ac.in/~uday

Intermediate representations

AST (viz format)

```
graph: {
title: "main"
node: { title: "ENTRY" label: "ENTRY" }
node: { title: "EXIT" label: "EXIT" }
edge: { sourcename: "ENTRY" targetname: "0" linestyle: solid
priority: 100 }


node: { title: "0" label: "#0\nmodify_expr (11)\nreturn_expr
(13)"}
edge: { sourcename: "0" targetname: "EXIT" priority: 100
linestyle: solid }
}
```

Credit: cse.iitb.ac.in/~uday

Intermediate representations
GIMPLE

```
;; Function main (main)
main ()
{
  int z;
  int y;
  int x;
  int c;
  int b;
  int a;
  int D.1138;
  int D.1137;
  # BLOCK 0
  # PRED: ENTRY (fallthru)
  a = b + c;
  D.1137 = a + z;
  D.1138 = D.1137 + y;
  x = D.1138 + 2;
  return;
  # SUCC: EXIT
}
```

Credit: cse.iitb.ac.in/~uday

Simple RTL example


```
(set (reg:SI 140)
     (plus:SI (reg:SI 138)
          (reg:SI 139) ) )
```

Intermediate representations

RTL (>200 lines!)

```
(note 16 2 5 0 [bb 0] NOTE_INSN_BASIC_BLOCK)

(insn 5 16 6 0 (parallel [
            (set (reg/f:SI 7 sp)
                (and:SI (reg/f:SI 7 sp)
                    (const_int -16 [0xfffffff0])))
            (clobber (reg:CC 17 flags))
        ]) -1 (nil)
    (nil))

(insn 6 5 7 0 (set (reg:SI 61)
        (const_int 0 [0x0])) -1 (nil) (nil))  ......
```

Credit: cse.iitb.ac.in/~uday

Intermediate representations
ASM

```
main:
        pushl  %ebp
        movl   %esp, %ebp
        subl   $40, %esp
        andl   $-16, %esp
        movl   $0, %eax
        addl   $15, %eax
        addl   $15, %eax
        shrl   $4, %eax
        sall   $4, %eax
        subl   %eax, %esp
        movl   -16(%ebp), %eax
        addl   -20(%ebp), %eax
        movl   %eax, -24(%ebp)
        movl   -4(%ebp), %eax
        addl   -24(%ebp), %eax
        addl   -8(%ebp), %eax
        addl   $2, %eax
        movl   %eax, -12(%ebp)
        leave
        ret
```

Credit: cse.iitb.ac.in/~uday

Typical compiler optimizations

- Algebraic optimizations
- Dead code elimination
- Loop invariants
- Vectorization
- Register allocation optimization

...and many more

Advanced optimizations

LTO – Link-time optimization / IPO – Interprocedural optimization

- Analyzes calls program-wide
- E.g., variable sharing and modification, memory aliasing

FDO – Feedback-driven optimization / PGO – Profile-guided optimization

- Optimizes code based on runtime behavior
- Requires recompilation

Linking

- The linker joins object files to produce a final file
- Linkers can produce executable, library or object files
- Object files may be sourced from the same program or libraries

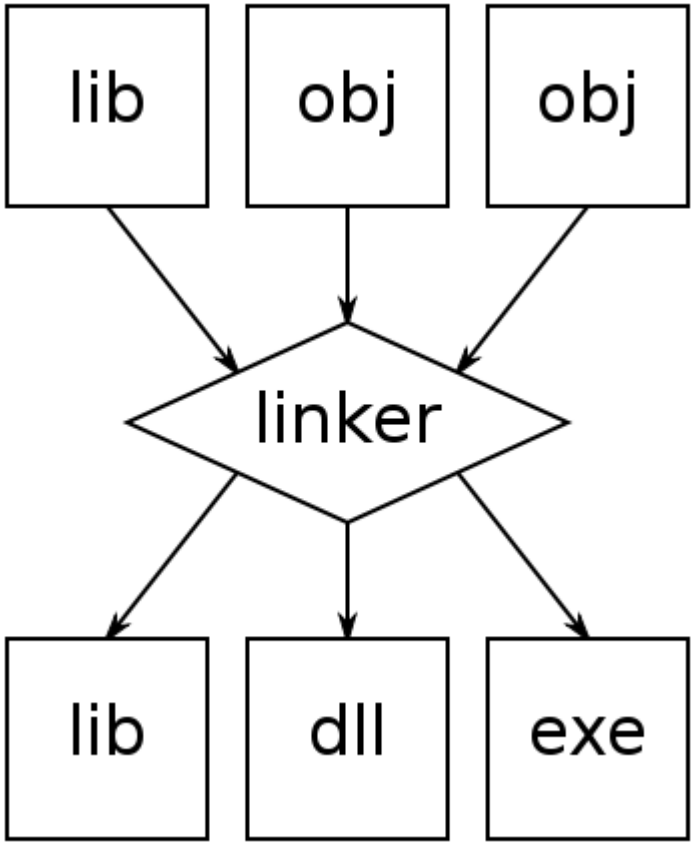
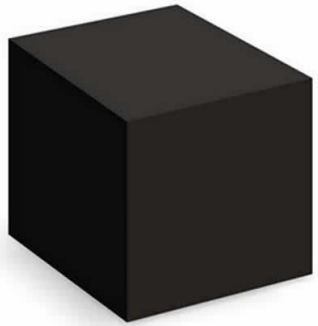
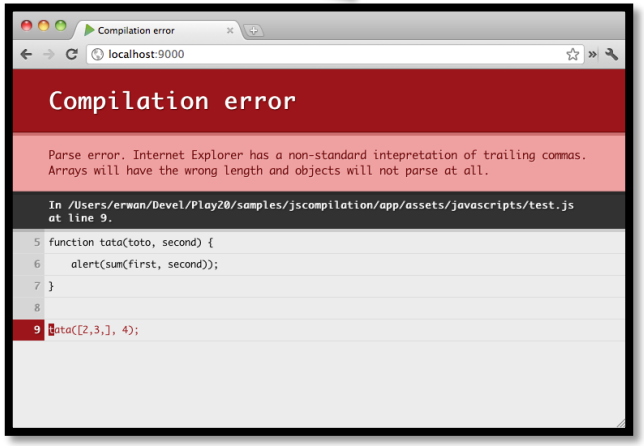
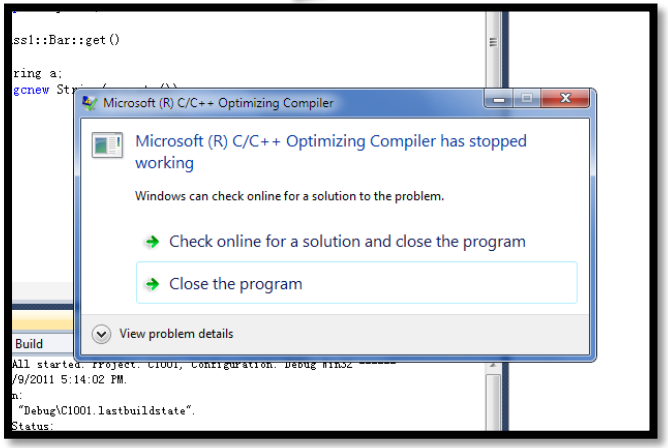


Image: Wikipedia

Compilers in practice







Be paranoid!

GCC Bugzilla – Bug List

Home | New | Browse | Search | Search [?] | Reports | Help | New Account | Log In | Forgot Password

User account creation filtered due to spam.

Tue Oct 18 2016 10:53:40 UTC

À la Sainte Véronique, je me mets à l'Informatique.

Hide Search Description

Priority: P1, P2, P3 **Summary:** (matches regular expression) \[([0-9./]*[/])*6[/][0-9./]*[Rr]egression *\] **Status:** UNCONFIRMED, NEW, ASSIGNED, SUSPENDED, WAITING, REOPENED
Target Milestone: 5.5, 6.3

201 bugs found.

ID	Product	Comp	Assignee	Status	Resolution	Summary	Changed
67807	gcc	libstdc+	unassigned	UNCO	---	[5/6/7 Regression] call to public member function catalog failed on Linux -std=c++03	2016-09-12
69188	gcc	lto	unassigned	UNCO	---	[5/6/7 Regression] ICE when linking objects at different optimization levels with LTO and profile generation enabled. (Works with 4.9.3.)	2016-09-12
70831	gcc	middle-e	unassigned	UNCO	---	[6/7 Regression] FTBFS: Build fails with bootstrap-lto and profiledbootstrap	2016-08-22
71399	gcc	target	unassigned	UNCO	---	[5/6/7 Regression] 5.3.0 bootstrap comparison failure on arm-linux-gnueabihf	2016-09-12
72707	gcc	c++	unassigned	UNCO	---	[5/6/7 regression] local anonymous union member hides names in the same scope	2016-08-03
72808	gcc	c++	unassigned	UNCO	---	[6/7 Regression] ICE on valid c++ code in verify_type (gcc/tree.c:14047)	2016-08-22
74563	gcc	target	unassigned	UNCO	---	[6/7 regression] Classic MIPS16 (non-MIPS16e) function return broken	2016-08-22
77459	gcc	libstdc+	unassigned	UNCO	---	[6/7 Regression] undefined reference to `snprintf' when building mingw-w64 cross-compiler	2016-09-29
77493	gcc	target	unassigned	UNCO	---	[6/7 Regression] -fcrossjumping (-O2) on ppc64le causes segfaults (jump to 0x0) (first bad r230091)	2016-09-16
77497	gcc	driver	unassigned	UNCO	---	[5/6/7 Regression] Setting DWARF level and debug level together has flag-ordering-dependent results	2016-09-08
77546	gcc	middle-e	unassigned	UNCO	---	[6/7 regression] C++ software renderer performance drop	2016-09-12
77849	gcc	c++	unassigned	UNCO	---	[5/6/7 Regression] Warning about deprecated enum even when "-Wdeprecated-declarations" is off	2016-10-05
70568	gcc	target	acsawdey	NEW	---	[5/6/7 regression] PowerPC64: union of floating and fixed doesn't use POWER8 GPR/VSR moves	2016-08-03
69633	gcc	rtl-opti	bernds	NEW	---	[6/7 Regression] Redundant move is generated after r228097	2016-08-22
65797	gcc	ipa	hubicka	NEW	---	[5/6/7 regression] IPA ICF causes function to be emitted with no debug line info	2016-09-22
69196	gcc	tree-opt	law	NEW	---	[5/6/7 Regression] code size regression with jump threading at -O2	2016-08-22
71947	gcc	tree-opt	law	NEW	---	[6 Regression] x ^ y not folded to 0 if x == y by DOM	2016-10-10
63191	gcc	rtl-opti	steven	NEW	---	[5/6/7 Regression] 32-bit gcc uses excessive memory during dead store elimination with -fPIC	2016-08-03

Java


```
public class Hello {
      public static void main(String argv[]) {
            System.out.println("Hello, world!");
      }
}
```

Just-in-time compilation
a.k.a. "dynamic translation"

- A program can be compiled to an intermediate representation and left until runtime
- Compilation during execution
  - From source
  - Or from "bytecode"
- Sometimes seen as a combination of ahead-of-time compilation and interpretation
- JIT compilers can handle files, functions or code fragments
- JIT compilers implement heavy caching

Bytecode

- Source code is translated/compiled to an intermediate representation called “bytecode”
- Bytecode can be interpreted or run by a virtual machine
- Bytecode is not machine code

A few Java bytecode ops
(operands not shown)

Mnemonic	Opcode (hex)	Stack change	Description
dup	59	value → value, value	duplicate the value on top of the stack
idiv	6c	value1, value2 → result	divide two integers
ifeq	99	value →	if value is 0, branch to instruction at branchoffset (branchbyte1 << 8 + branchbyte2)
isub	64	value1, value2 → result	int subtract
lstore_0	3f	value →	store a long value in a local variable 0

Java bytecode example

```
outer:
for (int i = 2; i < 1000; i++) {
    for (int j = 2; j < i; j++) {
        if (i % j == 0)
            continue outer;
    }
    System.out.println (i);
}
```

Wikipedia

Java bytecode dump example

```
0:   iconst_2
1:   istore_1
2:   iload_1
3:   sipush  1000
6:   if_icmpge 44
9:   iconst_2
10:  istore_2
11:  iload_2
12:  iload_1
13:  if_icmpge 31
16:  iload_1
17:  iload_2
18:  irem
19:  ifne 25
22:  goto    38
25:  iinc    2, 1
28:  goto    11
31:  getstatic     #84; // Field
java/lang/System.out:Ljava/io/Print
Stream;
34:  iload_1
35:  invokevirtual #85; // Method
java/io/PrintStream.println:(I)V
38:  iinc    1, 1
41:  goto    2
44:  return
```

Wikipedia

JVM
Java Virtual Machine

- An abstract computing machine
  - Operates on primitives – int/fp values and references
  - Garbage collection
- There are various implementations
- JRE – Java Runtime Environment
  - Typically contains a JVM implementation and a class library
  - Most well-known is HotSpot (Oracle, ex-Sun)
- JDK – Java Development Kit
  - Includes a compiler in addition to the above

JVM

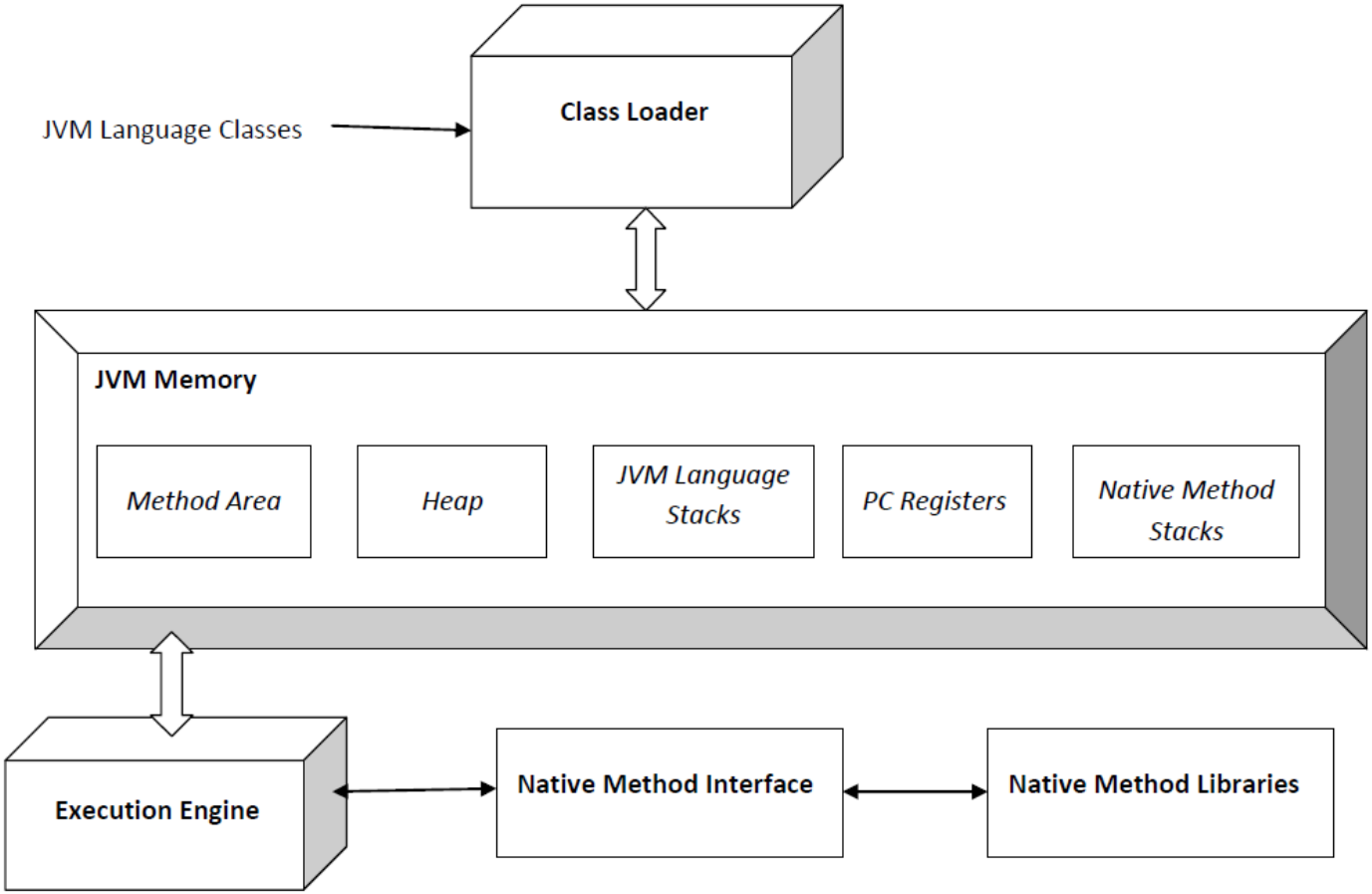
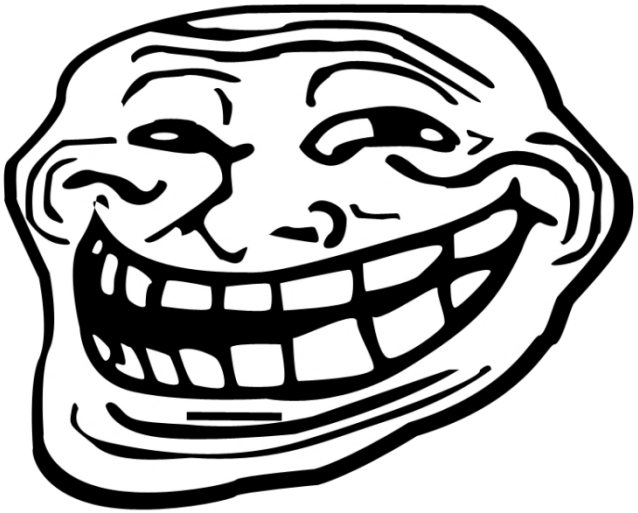


Image: Wikipedia



The TAQ

Trollingly Asked Questions

tik.

Is ICC faster than GCC?

Is C++ slower than C?

Why can't C/C++ be interpreted or JIT-ed?

CLING

- https://root.cern.ch/cling
- “interactive compiler”
  - Command line prompt and JIT compiler
- Designed for rapid turnaround and prototyping
- Jupyter Notebook front-ends exist

CLING + Jupyter

Interpreting the C++ programming language

`cling` has a broad support of the features of C++. You can define functions, classes, templates, etc ...

Functions

In [8]:
```
double sqr(double a)
{
    return a * a;
}
```

In [ ]:
```
double a = 2.5;
double asqr = sqr(a);
asqr
```

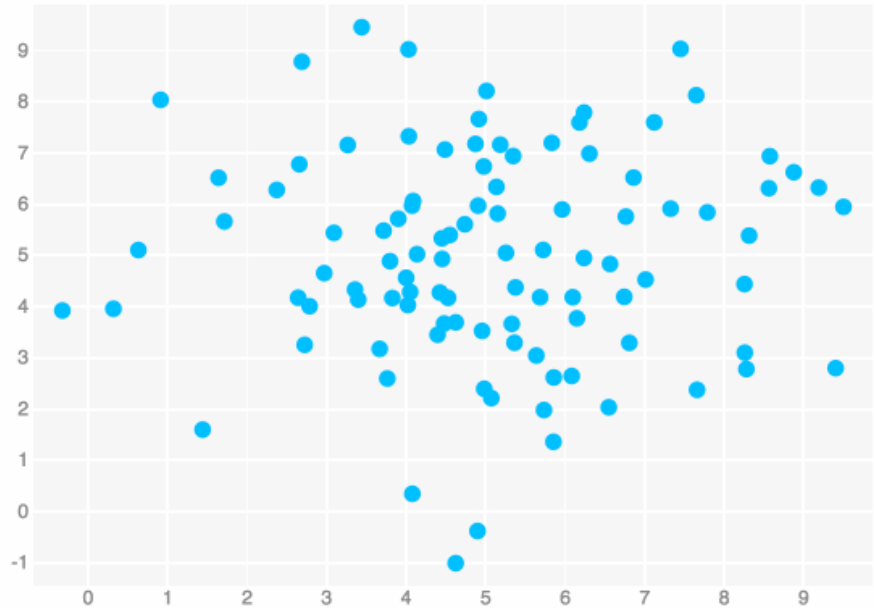
Classes

In [ ]:
```
class Foo
{
public:

    virtual ~Foo() {}
```

Image: Jupyter/QuantStack/
CLING team
07-Feb-18

CLING + Jupyter




```
In [13]: fig.animation_duration = 1000;

In [16]: scatter.x = randn(size);
         scatter.y = randn(size);

In [ ]:
```

Image: Jupyter/QuantStack/
CLING team
07-Feb-18

Recap: main types of code

```
template<class InputString,
bool unhexlify(const InputSt
  if (input.size() % 2 != 0)
    return false;
  }
  output.resize(input.size()
  int j = 0;
  auto unhex = [](char c) ->
    return c >= '0' && c <=
```

Source code

```
(set (reg:SI 140)
     (plus:SI (reg:SI 138)
              (reg:SI 139)))
```

Object code

Push Literal Variable 22	Push Literal Variable 23	Push Literal Variable 24
102	103	104
PopInto Receiver Variable 6	PopInto Receiver Variable 7	PopInto Temp 0
118	119	120
Push 1	Push 2	Return

Bytecode

```
401c3b:        31 ff
401c3d:        ba 05 0
401c42:        e8 e1 f
401c47:        48 89 d
401c4a:        48 89 c
401c4d:        bf 01 0
```

Machine code

```
.100 0       1       011111 010 0
                     t24q

tion SwapOps OpMode Op1    Uk2 PZSFlags
ooo x        m      111111 uuu f
1111 1       0      101001 100 0
                    regmd5

Oprn Condition SwapOps OpMode Op1    Uk2
     ccccc     x       m      111111 uuu
     00100     1       1      111001 1
```

Microcode

Thank you

e-mail: an@tik.services

http://tik.services