不深不淺，帶你認識 LLVM

Douglas Chen <dougpuob@gmail.com>

Found LLVM in our Life

Did you find it !

Douglas Chen <dougpuob@gmail.com>

陳鍵源 [Douglas Chen]
<dougpuob@gmail.com>

Life is short and we can’t change it.
But we can make it interesting.

Why I am HERE !

Because I believe the best way to learn something is sharing.

Not trying to teach you programming still, or show you how to use LLVM libraries. I just wanna introduce something new related to LLVM to you.

Agenda

1. Begin with a story
   - Before the story
   - The story
1. Free the Free
   - Self hosting?
   - What's diff btn LLVM&GCC?
1. Compiler
   - Understand the Magic
   - Optimation
   - LLVM
1. Go, let's find it (Products)
   - Apple's Projects
   - Google's Projects
   - Other Projects
4. Go, let's find it (JIT)
   - What is JIT?
   - JVM/GraalVM
   - Virtual Machine/QEMU
4. Go, let's find it (Web)
   - What is WebAssembly?
   - Project with WebAssembly
4. Q&A

Agenda

1.Begin with a story

1. Begin with a story

Before the story

What is compiler ?

Compiler is a **magic** making source code to application.

What is compiler ?

Source Code

Compiler

Application

1. Begin with a story

The story

?

Apple Computer & NeXT

Apple 1

Apple 2

Apple 3

Lisa

Macintosh

Objective-C

NeXTSTEP

Operating System

1976

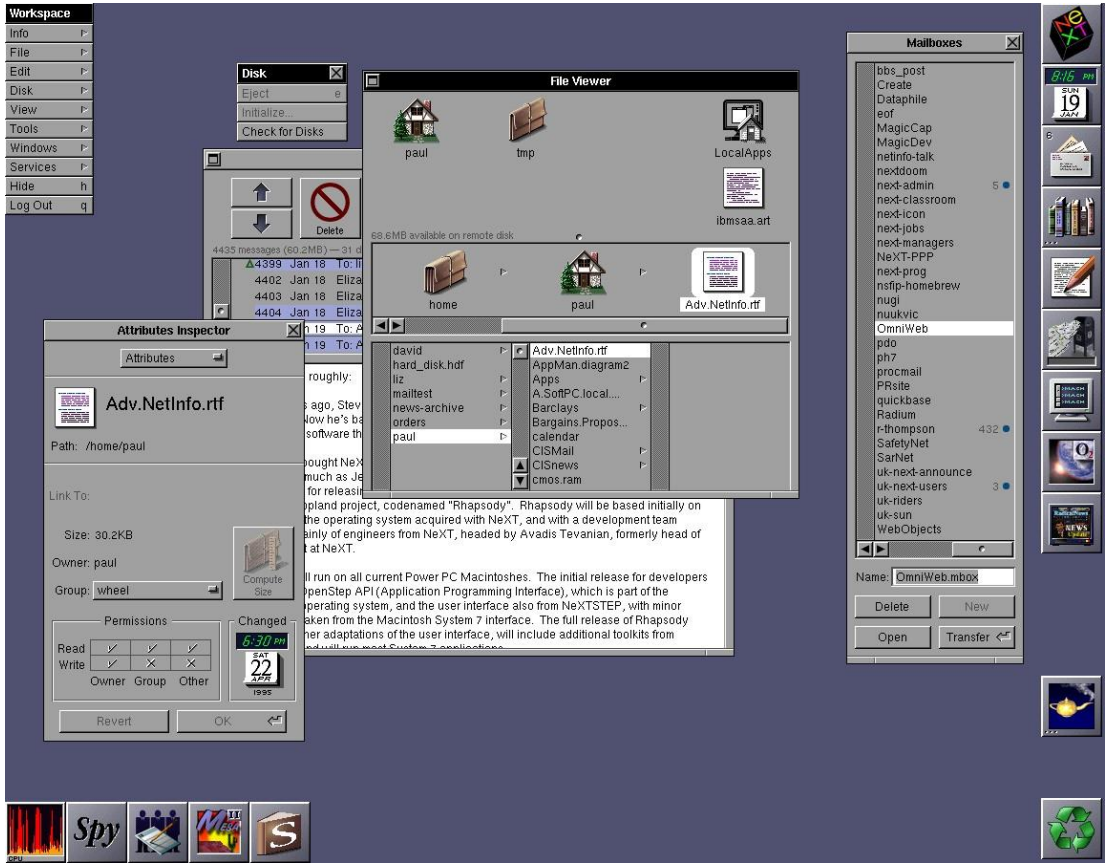
1985

1985

1988

1989

NeXT's NeXTSTEP OS



Return to Glory

1997

NeXTSTEP

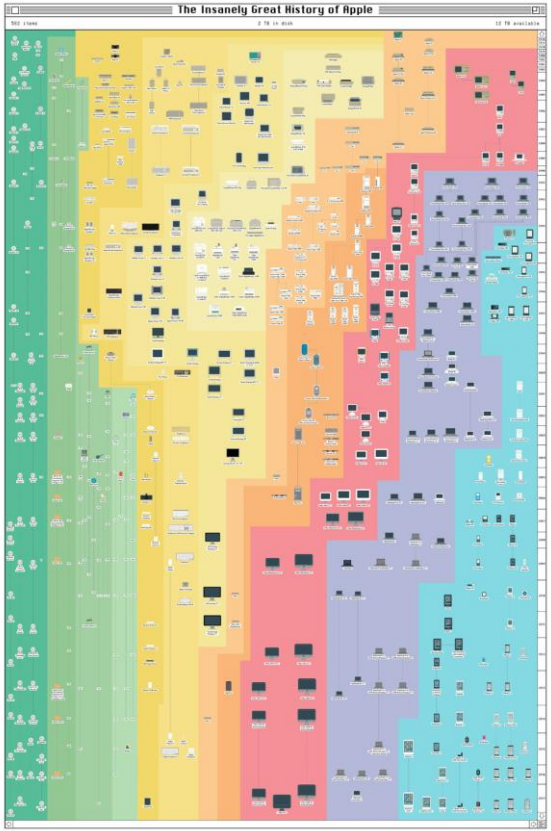
1997

mac OS

2001

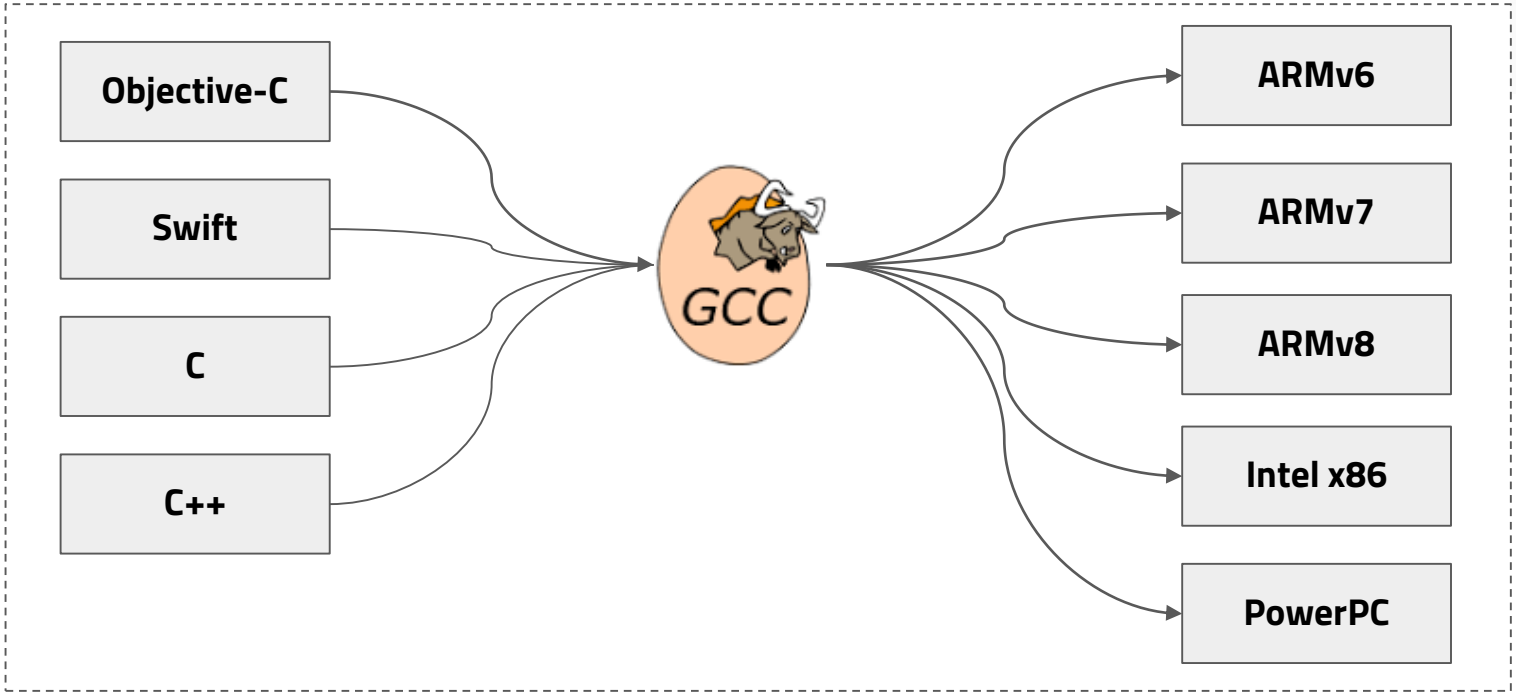
1998 Power Macintosh G3
1999 Power Macintosh G4
2000 PowerBook
2001 iPod
2002 iPod2
2003 iPod3
2004 iPod4 & Mini & Photo
2005 iPod5
iPod Shuffle
iPod Nano
Power Macintosh G5 (Intel)
2006 MacBook Pro
2007 Apple TV
iPhone
2008 MacBook Air
iPod Touch
iPhone 3G

Complicated Ecosystem



CPU	OS	Language
ARMv6	macOS	C
ARMv7	iOS	C++
ARMv8	watchOS	Objective-C
Intel x86	tvOS	Swift
PowerPC		

Complicated Ecosystem

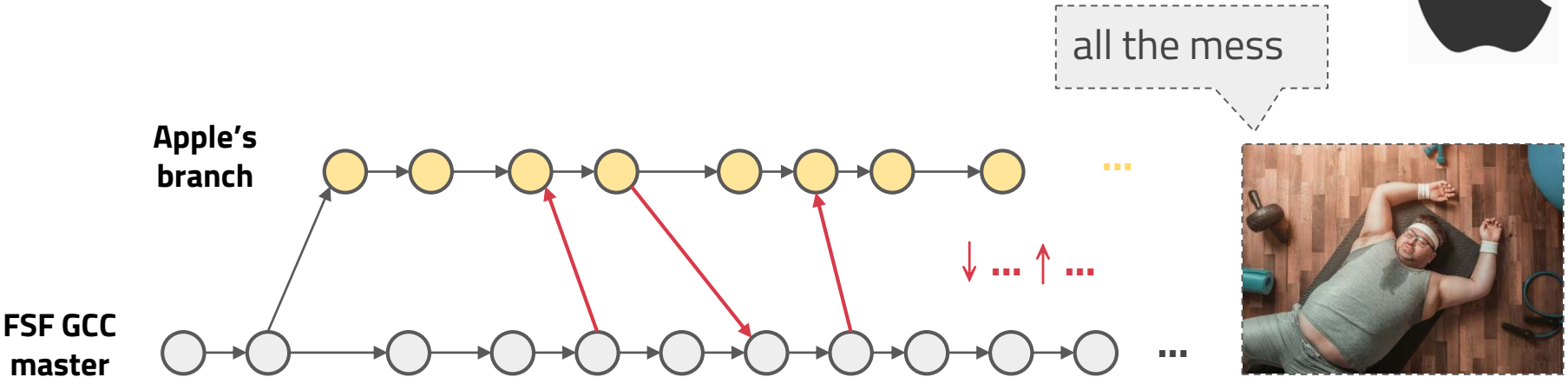


Application
Driver
OS

SDK

Xcode

Apple needs find a way out



GCC is developed for solving real problems, it has no time to make a good everything perfect.

Apple met LLVM

Chris Lattner

Twitter : https://twitter.com/clattner_llvm
Website : http://nondot.org/sabre/

Apple met LLVM

1997

NeXTSTEP

1997

mac OS

2001

2000

2005

2007 Xcode 3.x
2011 Xcode 4.x
2013 Xcode 5.x

2011 gcc > llvm 10%
2013 gcc ≈ llvm

(run-time performance)

1998 Power Macintosh G3
1999 Power Macintosh G4
2000 PowerBook
2001 iPod
2002 iPod2
2003 iPod3
2004 iPod4 & Mini & Photo
2005 iPod5
iPod Shuffle
iPod Nano
Power Macintosh G5 (Intel)
2006 MacBook Pro
2007 Apple TV
iPhone
2008 MacBook Air
iPod Touch
iPhone 3G

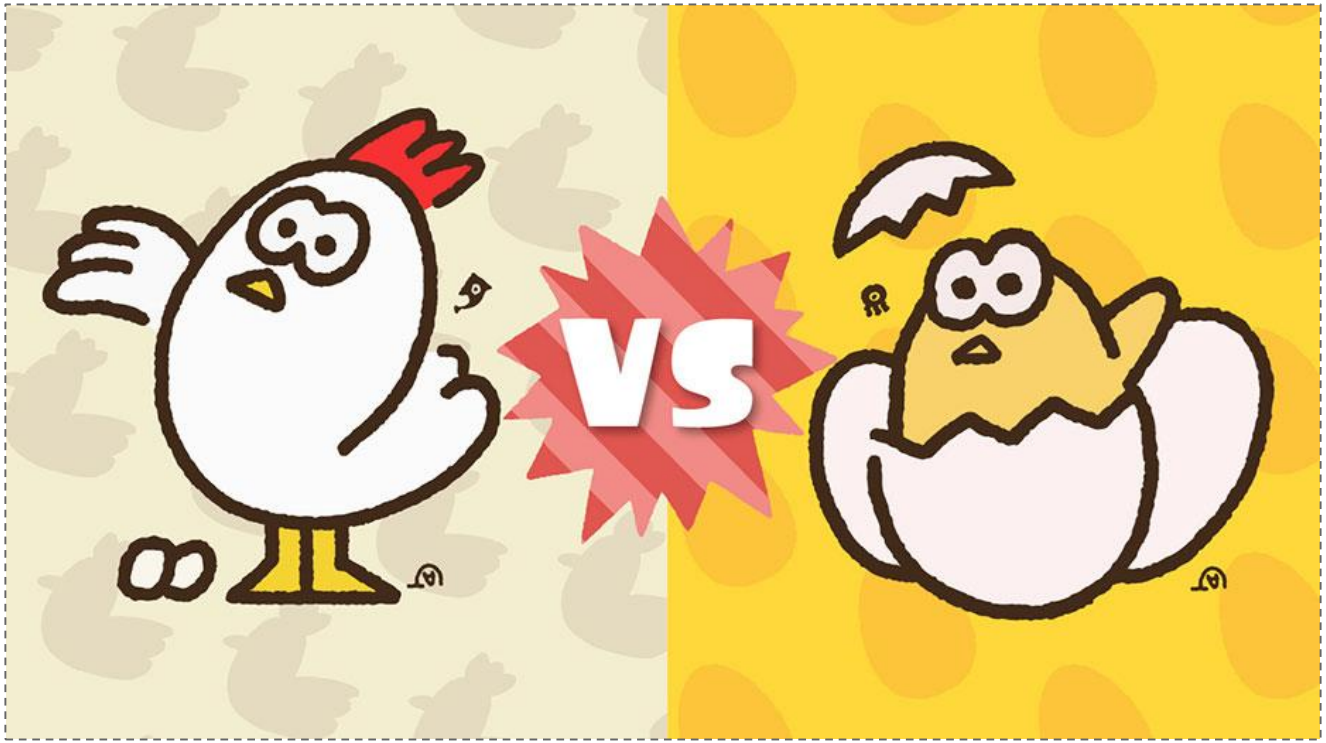
Agenda

2.Free the Free

2. Free the Free

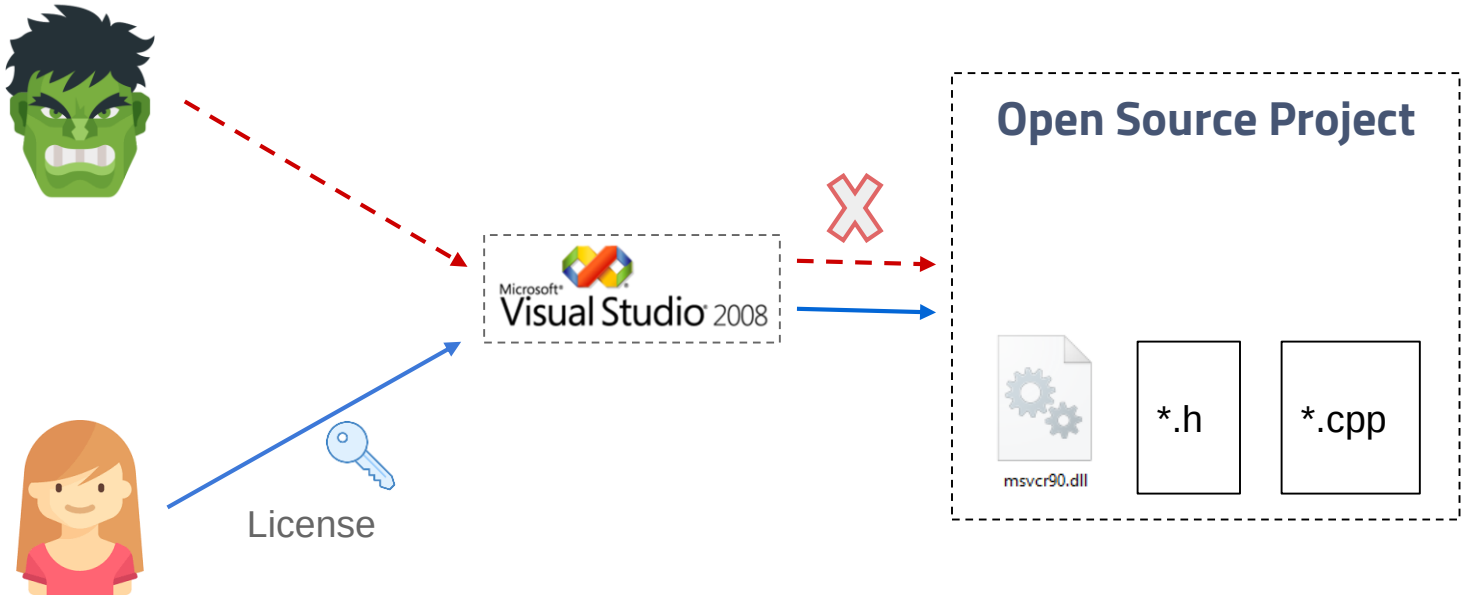
Why self hosting is important !

Why self-hosting is important !

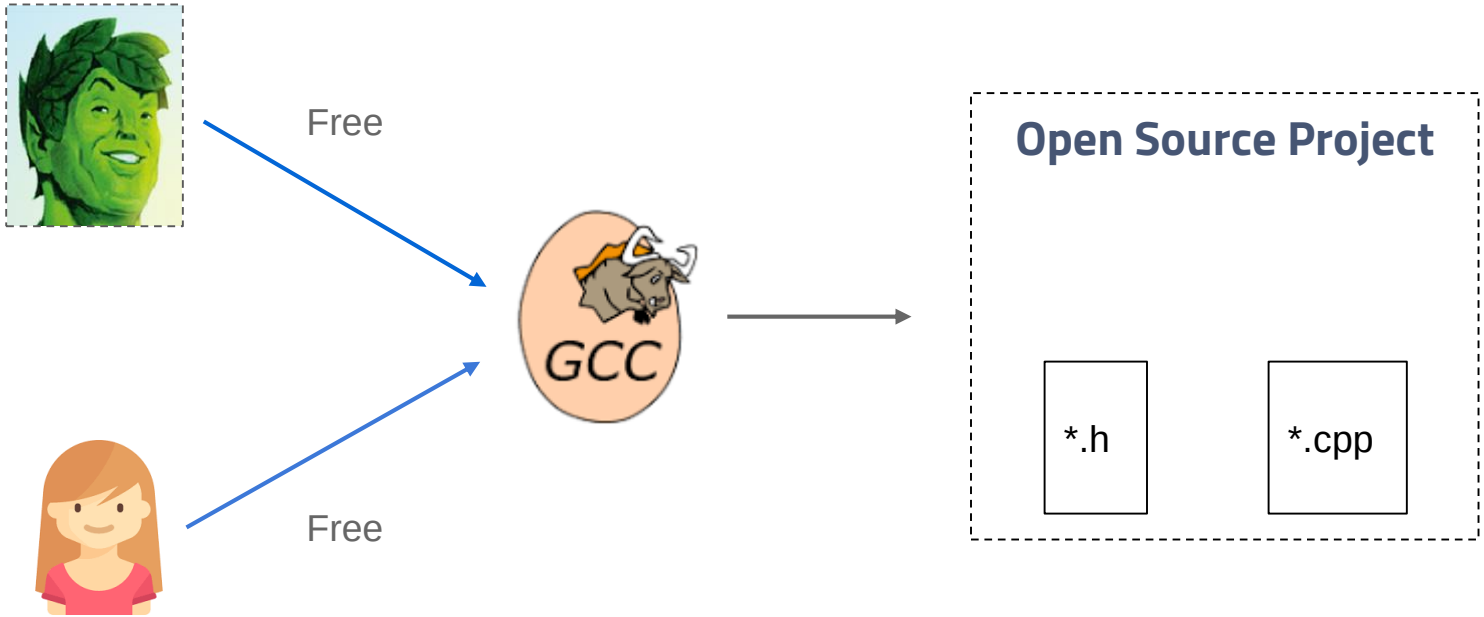


https://bbs.saraba1st.com/2b/thread-1375402-113-1.html

Why self hosting is important !



Why self hosting is important !



Why self hosting is important !

RMS

FSF

FREE SOFTWARE FOUNDATION

GNU

GCC

Richard M. Stallman

GNU's Not Unix!

GNU Compiler Collection (GNU C Compiler)

Why self hosting is important !

Free
the Free

GCC

Open Source Project

GCC

*.h

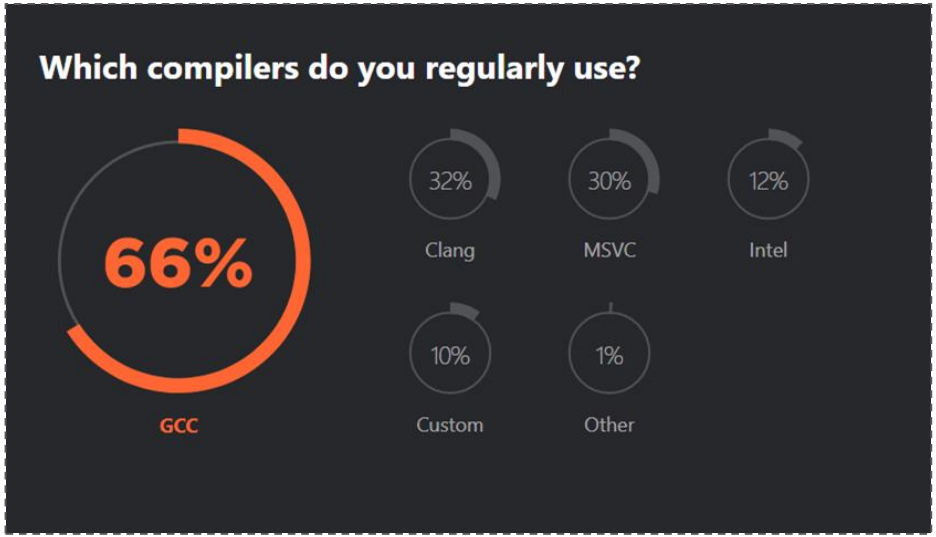
*.cpp

2. Free the Free

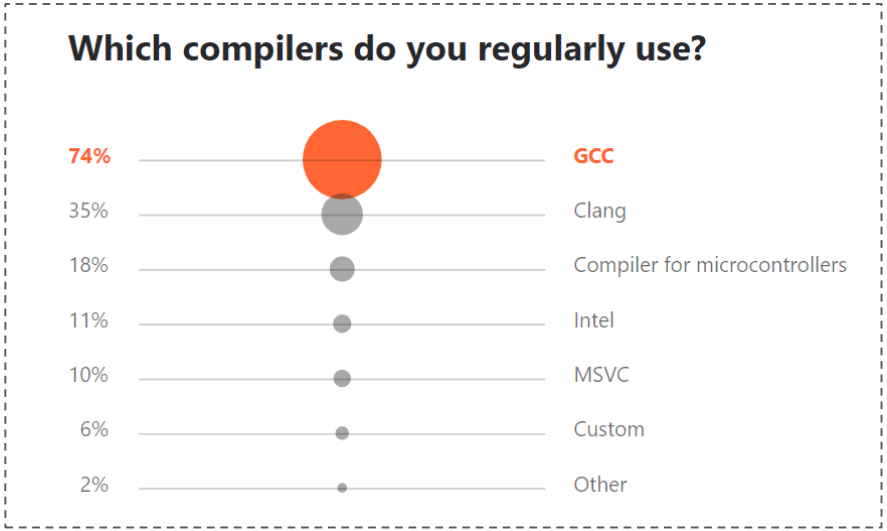
What’s the different between LLVM & GCC ?

Which compilers do your regularly use?

C++



C



https://www.jetbrains.com/lp/devecosystem-2019/cpp/

What is LLVM

1. LLVM is a Compiler 

2. LLVM is a Compiler Infrastructure 
3. LLVM is a series of Compiler Tools 
4. LLVM is a Compiler Toolchain 

5. LLVM is an open source C++ implementation 

Pro's of GCC vs Clang:

- GCC supports languages that Clang does not aim to, such as Java, Ada, FORTRAN, Go, etc.
- GCC supports more targets than LLVM.
- GCC supports many language extensions.

https://clang.llvm.org/comparison.html

Pro's of Clang vs GCC:

- The Clang ASTs and design are intended to be easily understandable by anyone.

- Clang is designed as an API from its inception, allowing it to be reused by source analysis tools, refactoring, IDEs (etc) as well as for code generation. GCC is built as a monolithic static compiler.

- Various GCC design decisions make it very difficult to reuse , ... Clang has none of these problems.

https://clang.llvm.org/comparison.html

Pro's of Clang vs GCC:

- Clang can serialize its AST out to disk and read it back into another program, which is useful for whole program analysis. GCC does not have this.
- Clang is much faster and uses far less memory than GCC.
- Clang has been designed from the start to provide extremely clear and concise diagnostics (error and warning messages).
- GCC is licensed under the GPL license. Clang uses a BSD license.

https://clang.llvm.org/comparison.html

What I see the different like this ...

GCC

LLVM

https://seriousplaypro.com/wp-content/uploads/2017/06/LEGO-Idea-House-26.jpg

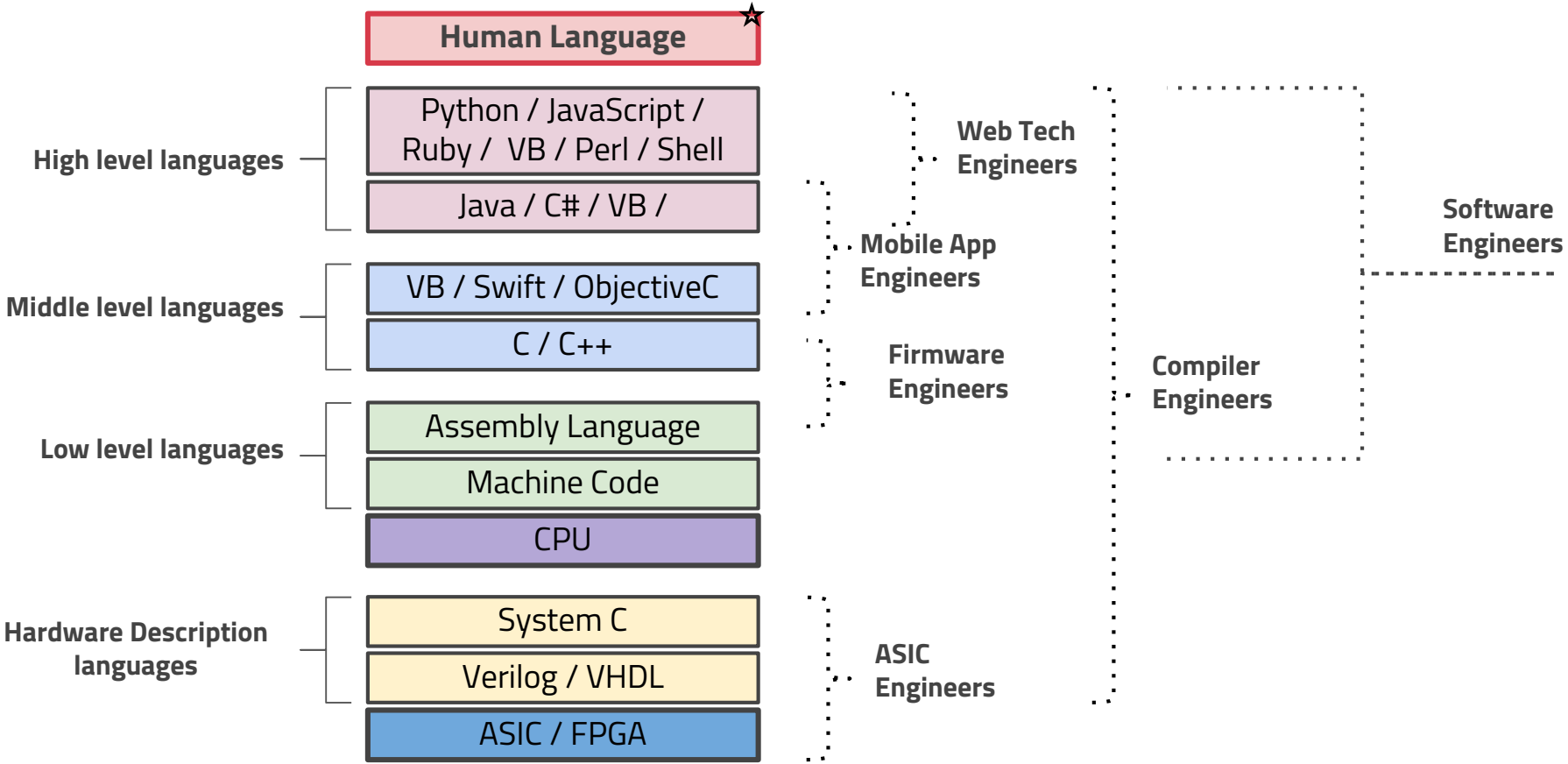
Agenda

3.Compiler

3. Compiler

Understand the magic

Computer Language stacks



Compiler


Source Code
Compiler
Exe Binary


Source Code
Front-End
Optimizer
Back-End
Machine Code

3. Compiler

Frontend

What is compiler ?

Compiler is a magic (making ...).

1 (token)	2 (token)	3 (token)	4 (token)	5 ... (tokens)
Compiler	is	a	magic	making ...
(S)	(V)	(C)	(C)	

Tokenization

```
// min.c
int min(int a, int b)
{
  if (a < b)
    return a;
  return b;
}
```

```
$ clang -cc1 -dump-tokens min.c
```

```
int         'int'    [StartOfLine]
identifier  'min'    [LeadingSpace]
l_paren     '('
int         'int'
identifier  'a'                 [LeadingSpace]
comma       ','
int         'int'    [LeadingSpace]
identifier  'b'                 [LeadingSpace]
r_paren     ')'
l_brace     '{'                 [LeadingSpace]
if          'if'     [StartOfLine]
[LeadingSpace]
l_paren     '('                 [LeadingSpace]
identifier  'a'
less        '<'                 [LeadingSpace]
identifier  'b'                 [LeadingSpace]
r_paren     ')'

return      'return' [StartOfLine] [LeadingSpace]
```

AST Dump

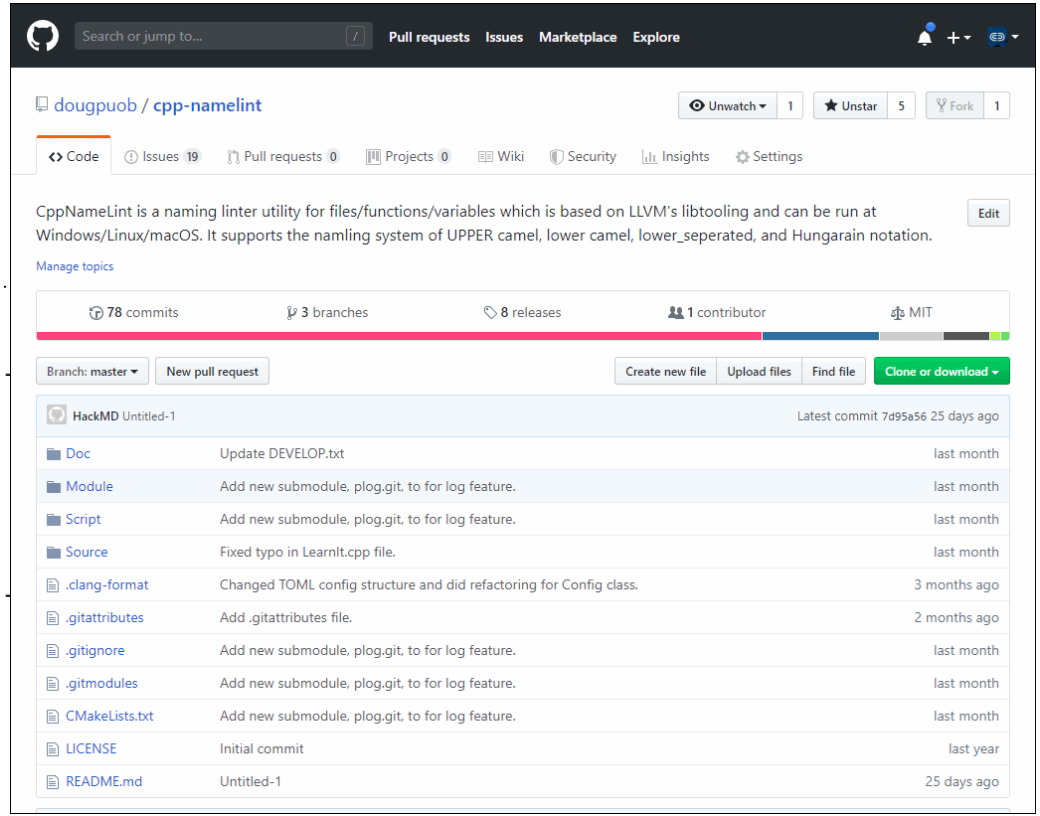
$ clang -cc1 -ast-dump min.c

```
// min.c
int min(int a, int b)
{
  if (a < b)
    return a;
  return b;
}
```

```
TranslationUnitDecl 0x2c8ce56b660 <<invalid sloc>> <invalid sloc>
`-FunctionDecl 0x2c8ce56be18 <min.c:2:1, line:6:1> line:2:5 min 'int
 |-ParmVarDecl 0x2c8ce56bcc0 <col:9, col:13> col:13 used a 'int'
 |-ParmVarDecl 0x2c8ce56bd38 <col:16, col:20> col:20 used b 'int'
 `-CompoundStmt 0x2c8ce56c0a0 <col:23, line:6:1>
   |-IfStmt 0x2c8ce56c018 <line:3:3, line:4:12>
   | |-<<<NULL>>>
   | |-BinaryOperator 0x2c8ce56bf98 <line:3:7, col:11> 'int' '<'
   | | |-ImplicitCastExpr 0x2c8ce56bf68 <col:7> 'int' <LValueToRValue>
   | | | `-DeclRefExpr 0x2c8ce56bf18 <col:7> 'int' lvalue ParmVar 0x2c8ce56bcc0 'a' 'int'
   | | `-ImplicitCastExpr 0x2c8ce56bf80 <col:11> 'int' <LValueToRValue>
   | |   `-DeclRefExpr 0x2c8ce56bf40 <col:11> 'int' lvalue ParmVar 0x2c8ce56bd38 'b' 'int'
   | |-ReturnStmt 0x2c8ce56c000 <line:4:5, col:12>
   | | `-ImplicitCastExpr 0x2c8ce56bfe8 <col:12> 'int' <LValueToRValue>
   | |   `-DeclRefExpr 0x2c8ce56bfc0 <col:12> 'int' lvalue ParmVar 0x2c8ce56bcc0 'a' 'int'
   | `-<<<NULL>>>
   `-ReturnStmt 0x2c8ce56c088 <line:5:3, col:10>
     `-ImplicitCastExpr 0x2c8ce56c070 <col:10> 'int' <LValueToRValue>
       `-DeclRefExpr 0x2c8ce56c048 <col:10> 'int' lvalue ParmVar 0x2c8ce56bd38 'b' 'int'
```

CppNameLint

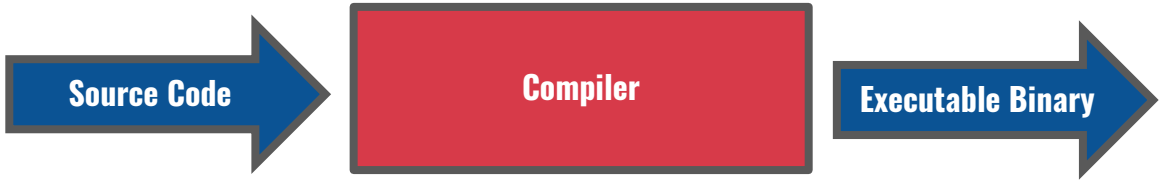
```
cppnamelint utility v0.2.5
-----------------------------------------------
 File    = Detection.cpp
 Config  = cppnamelint.toml
 Checked =   191  [File:0 | Func: 44 | Param:
 Error   =     7  [File:0 | Func:  0 | Param:
-----------------------------------------------
  <93,  5  > Variable :    wayToSort (auto)
  <93,  25 > Variable :    strA (string)
  <93,  38 > Variable :    strB (string)
  <168, 5  > Variable :    wayToSort (auto)
  <168, 25 > Variable :    strA (string)
  <168, 38 > Variable :    strB (string)
  <239, 9  > Variable :    nLowerPCount (size_t)
```



3. Compiler

Optimization

Compiler

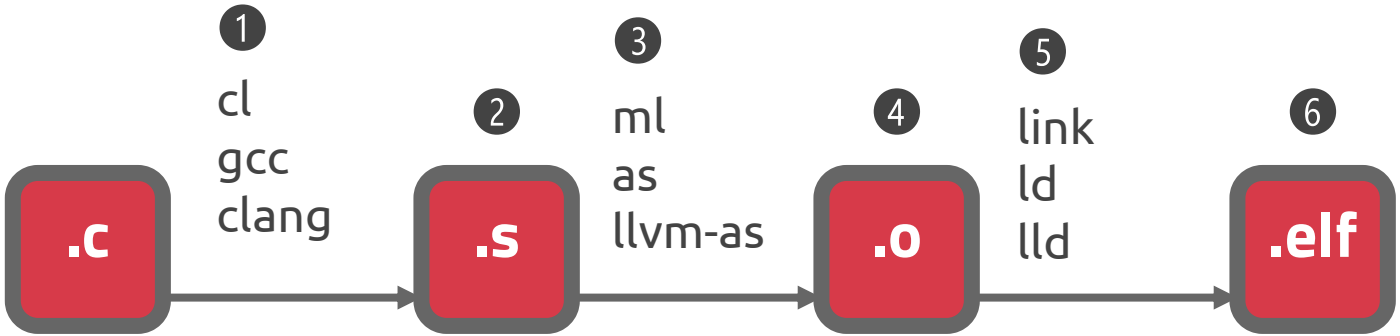


Optimize Here

Optimize Here

Optimize Here

❶--> compiler
❷--> assembly code(.s)
❸--> assembler
❹--> object file (.o)
❺--> linker
❻--> binary file
(.exe/.elf/.a)



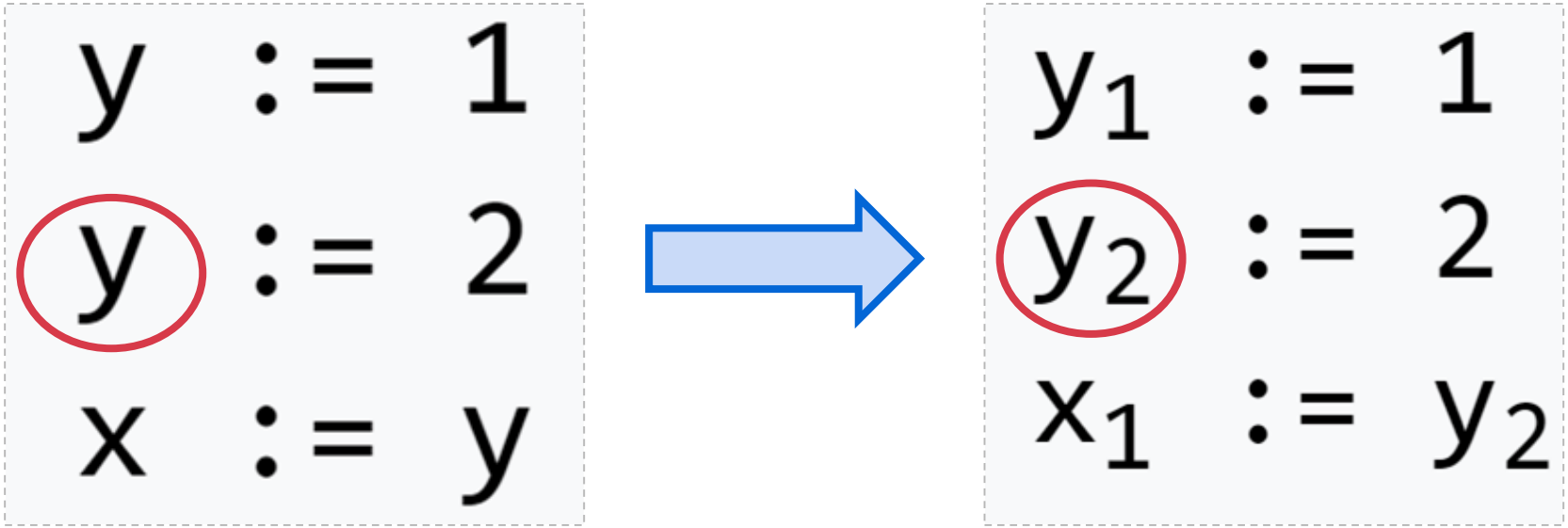
Optimization

- SSA (Static Single Assignment)
- Constant Propagation
- Dead Code Elimination
- Branch Free

SSA
(Static Single Assignment)

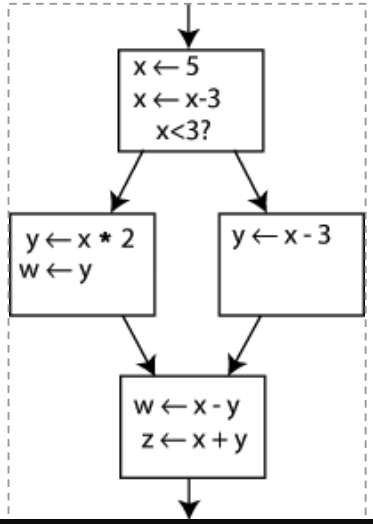
Optimization::SSA

SSA (Static Single Assignment)



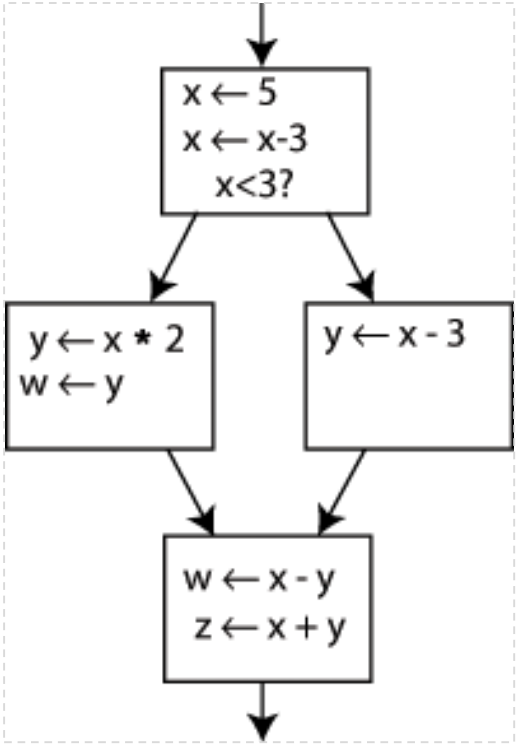
SSA (Static Single Assignment)

```
1   x = 5;
2   x = x - 3;
3   if (x < 3) {
4     y = x * 2;
5     w = y
6   } else {
7     y = x - 3;
8   }
9   w = x - y;
10  z = x + y;
```

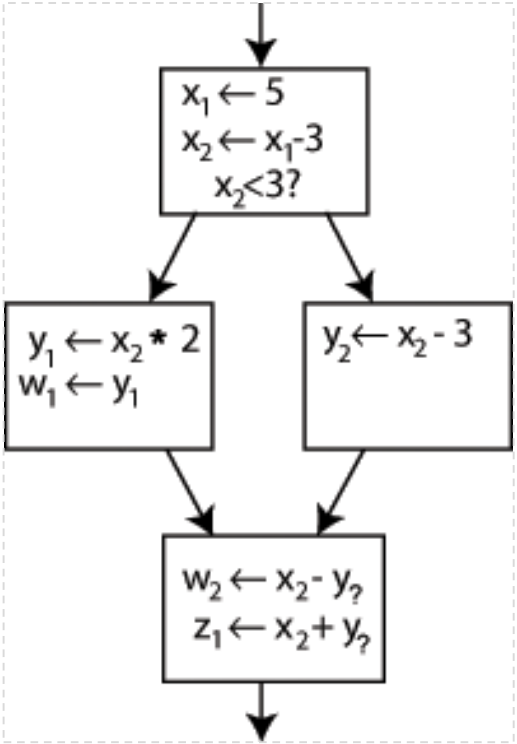



```
Function 'myfunc'
Lines executed:92.86% of 14
Branches executed:100.00% of 2
Taken at least once:50.00% of 2
No calls
```

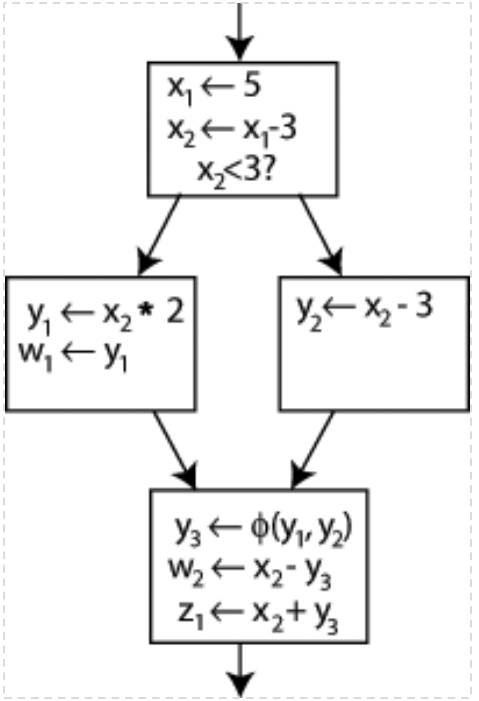
SSA (Static Single Assignment)



1



2



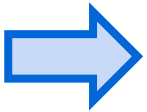
3

Constant Propagation
&
Dead Code Elimination

Constant Propagation & Dead Code Elimination

```
3   int GetValue()
4   {
5       int x = 14;
6       int y = 7 - x / 2;
7       return y * (28 / x + 2);
8   }
```

gcc -O0



```
1   GetValue():
2     push rbp
3     mov rbp, rsp
4     mov DWORD PTR [rbp-4], 14
5     mov eax, DWORD PTR [rbp-4]
6     mov edx, eax
7     shr edx, 31
8     add eax, edx
9     sar eax
10    neg eax
11    add eax, 7
12    mov DWORD PTR [rbp-8], eax
13    mov eax, 28
14    cdq
15    idiv DWORD PTR [rbp-4]
16    add eax, 2
17    imul eax, DWORD PTR [rbp-8]
18    pop rbp
19    ret
```

Constant Propagation & Dead Code Elimination

```
1   int GetValue()
2   {
3       int x = 14;
4       int y = 7 - x / 2;
5       return y * (28 / x + 2);
6   }
7
8   int GetValue1()
9   {
10      int x = 14;
11      int y = 7 - 14 / 2;
12      return y * (28 / 14 + 2);
13  }
14
15  int GetValue2()
16  {
17      int x = 14;
18      int y = 7 - 7;
19      return y * (2 + 2);
20  }
```

Constant Propagation

```
22  int GetValue3()
23  {
24      int x = 14;
25      int y = 0;
26      return 0;
27  }
28
29  int GetValue4()
30  {
31      return 0;
32  }
```

Dead Code Elimination

Optitmized

GetValue() = GetValue4()

Optimization

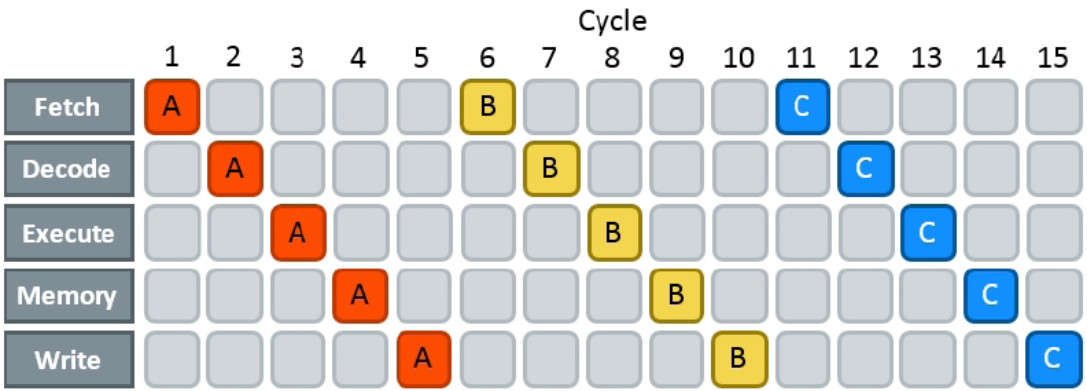
Branch Free

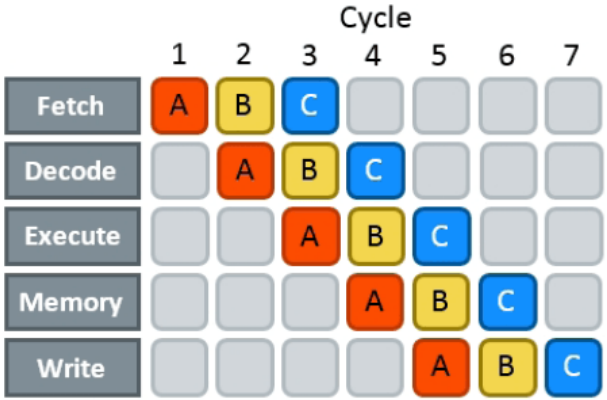
How to tell Branchs

```
 1  x = 5;
 2  x = x - 3;
 3  if (x < 3) {
 4     y = x * 2;
 5     w = y
 6  } else {
 7     y = x - 3;
 8  }
 9  w = x - y;
10  z = x + y;
```

```
Function 'myfunc'
Lines executed:92.86% of 14
Branches executed:100.00% of 2
Taken at least once:50.00% of 2
No calls
```

Instruction pipeline





https://techdecoded.intel.io/resources/understanding-the-instruction-pipeline/

Branch Free

Student Age

```
1   int tolower1(int ch)
2   {
3       if (ch >= 0x41/*A*/ && ch <= 0x5A/*Z*/)
4       {
5           ch += 32;
6       }
7       return ch;
8   }
```

Now

```
1   int tolower2(int c)
2   {
3       c += ((c-'A')<26U)<<5;
4       return c;
5   }
```

'A' → 'a'

...

'Z' → 'z'

...

'5' --> '5'

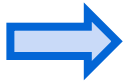
...

Branch Free (tolower1)

'A' → 'a'

```
1   int tolower1(int ch)
2   {
3       if (ch >= 0x41/*A*/ && ch <= 0x5A/*Z*/)
4       {
5           ch += 32;
6       }
7       return ch;
8   }
```

```
Function 'tolower1'
Lines executed:60.00% of 5
Branches executed:100.00% of 4
Taken at least once:50.00% of 4
No calls

Function 'tolower2'
Lines executed:100.00% of 3
No branches
No calls
```



```
1   tolower1: # @tolower1
2     push rbp
3     mov rbp, rsp
4     mov dword ptr [rbp - 4], edi
5     cmp dword ptr [rbp - 4], 65
6     jl .LBB0_3
7     cmp dword ptr [rbp - 4], 90
8     jg .LBB0_3
9     mov eax, dword ptr [rbp - 4]
10    add eax, 32
11    mov dword ptr [rbp - 4], eax
12  .LBB0_3:
13    mov eax, dword ptr [rbp - 4]
14    pop rbp
15    ret
```

Branch Free (tolower2)

'A' → 'a'

```
1    int tolower2(int c)
2    {
3        c += ((c-'A')<26U)<<5;
4        return c;
5    }
```

```
Function 'tolower1'
Lines executed:60.00% o   5
Branches executed:100   % of 4
Taken at least once:    .00% of 4
No calls

Function 'tolower2'
Lines executed:100.00% of 3
No branches
No calls
```

```
1    tolower2(int):
2            push    rbp
3            mov     rbp, rsp
4            mov     dword ptr [rbp - 4], edi
5            mov     eax, dword ptr [rbp - 4]
6            sub     eax, 65
7            cmp     eax, 26
8            setb    cl
9            and     cl, 1
10           movzx   eax, cl
11           shl     eax, 5
12           add     eax, dword ptr [rbp - 4]
13           mov     dword ptr [rbp - 4], eax
14           mov     eax, dword ptr [rbp - 4]
15           pop     rbp
16           ret
```

Branch Free  (CLANG -O3)

```
1   int tolower1(int ch)
2   {
3       if (ch >= 0x41/*A*/ && ch <= 0x5A/*Z*/)
4       {
5           ch += 32;
6       }
7       return ch;
8   }
```

```
1   int tolower2(int c)
2   {
3       c += ((c-'A')<26U)<<5;
4       return c;
5   }
```

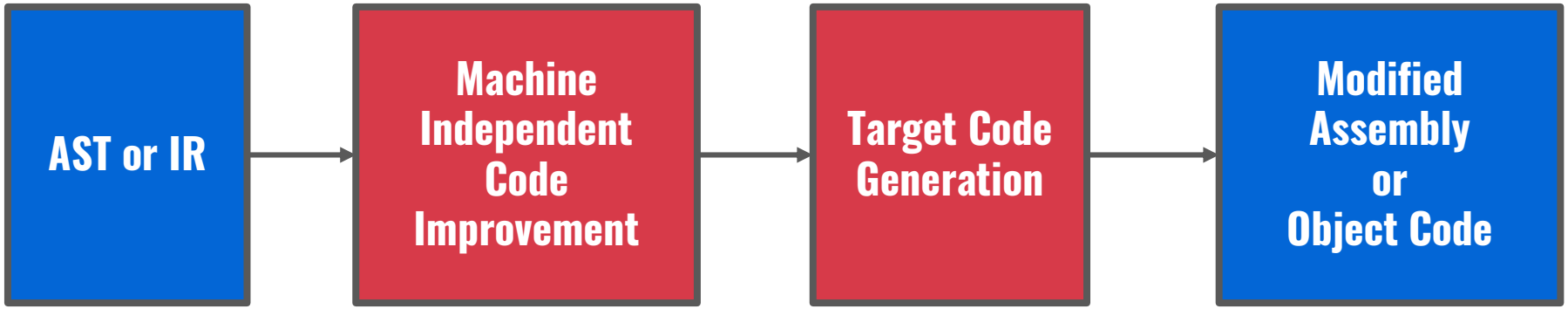
```
1   tolower1: # @tolower1
2     lea ecx, [rdi - 65]
3     lea eax, [rdi + 32]
4     cmp ecx, 26
5     cmovae eax, edi
6     ret
```

```
1   tolower2: # @tolower2
2     lea ecx, [rdi - 65]
3     xor eax, eax
4     cmp ecx, 26
5     setb al
6     shl eax, 5
7     add eax, edi
8     ret
```

3. Compiler

Backend

Backend synthesis



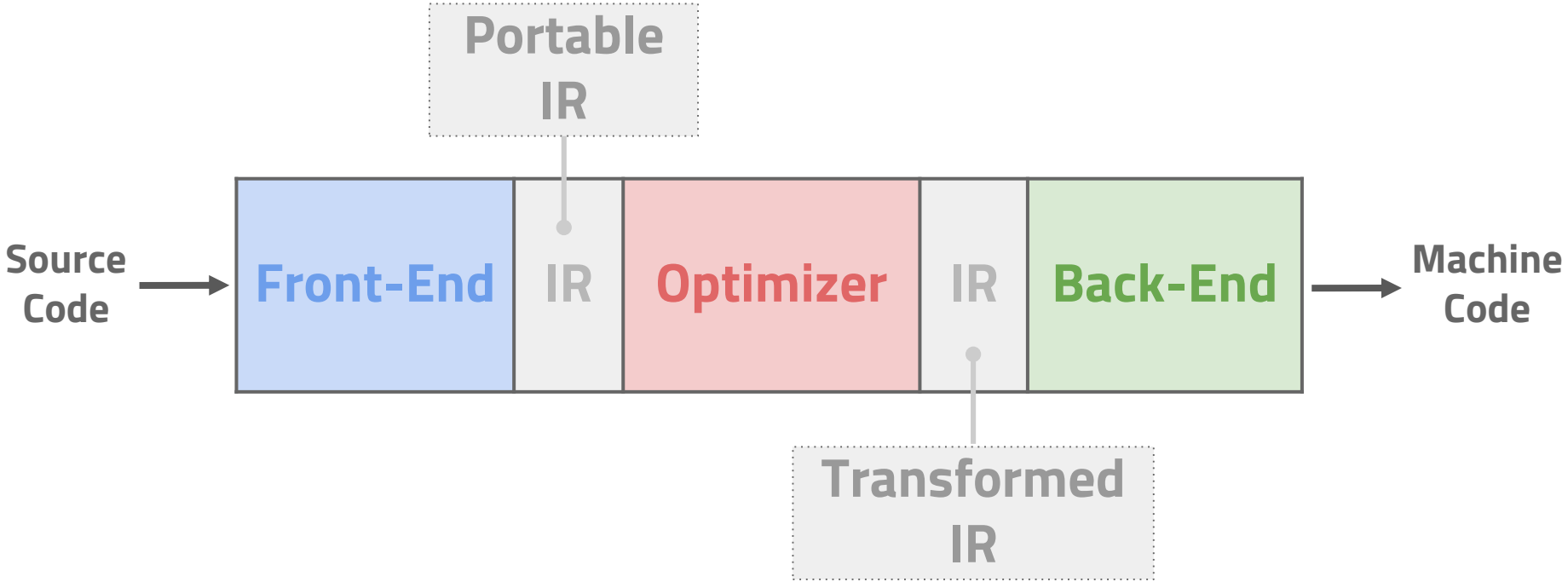
3. Compiler

LLVM

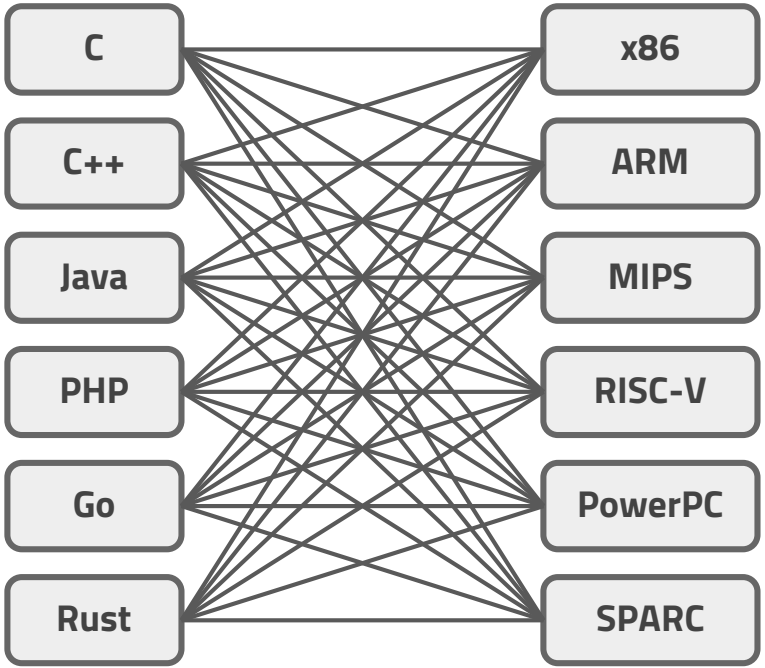
Traditional Compiler

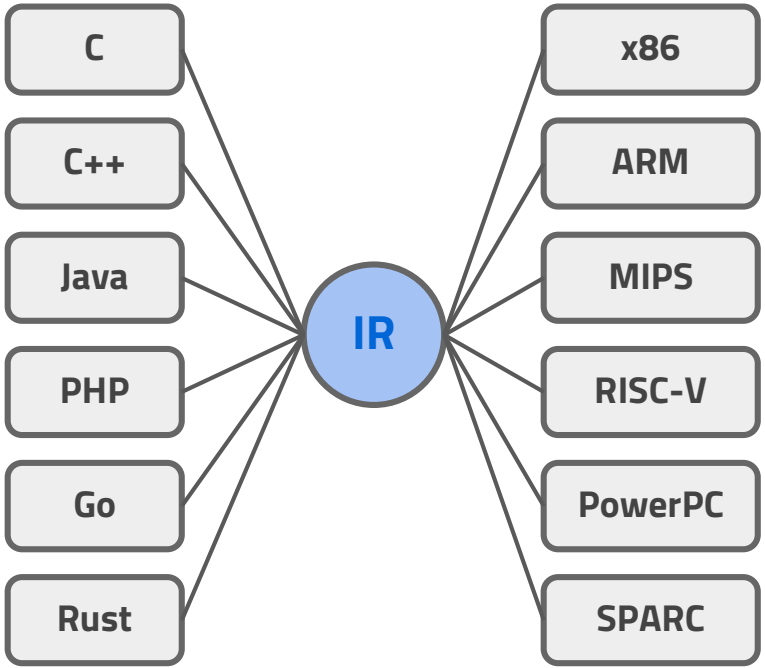
Traditional Compiler

LLVM

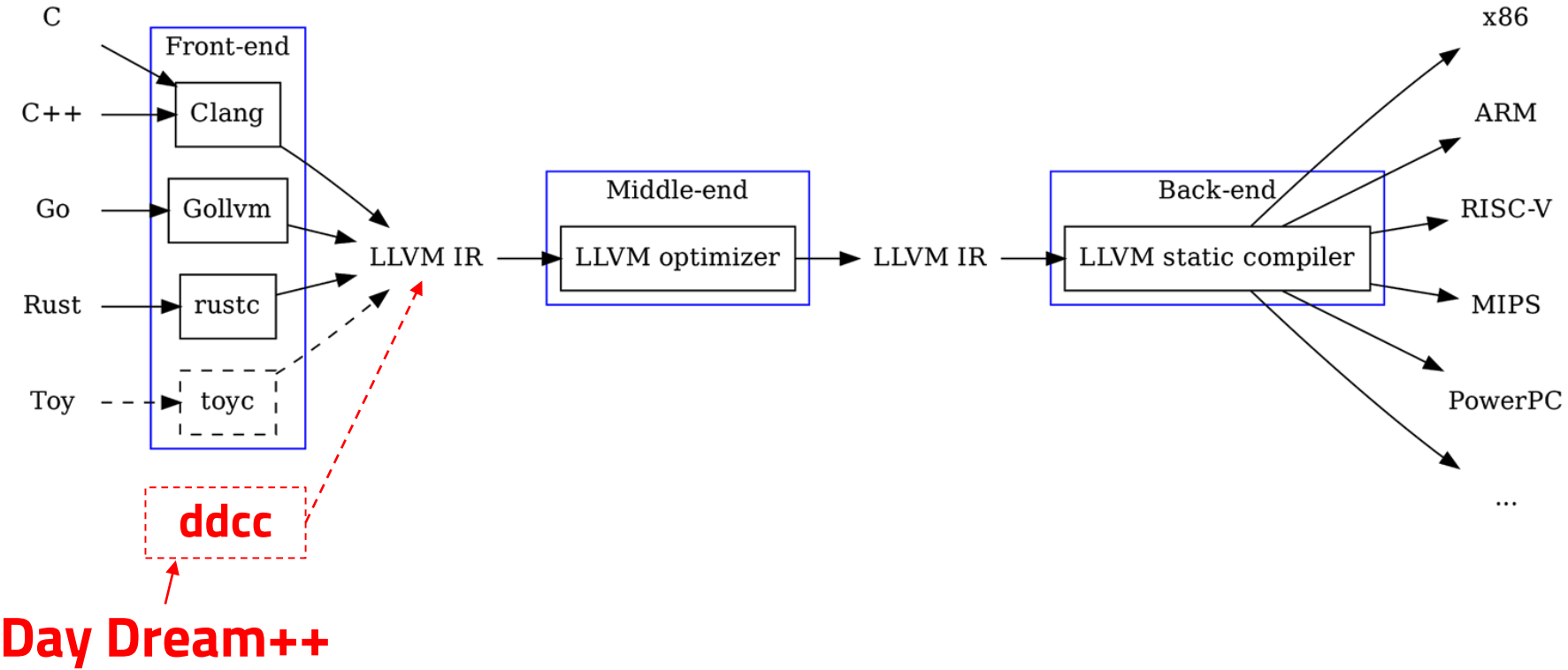

Portable
IR
Source
Code
Front-End
IR
Optimizer
IR
Back-End
Machine
Code
Transformed
IR

Traditional Compiler v.s. Modern Compiler

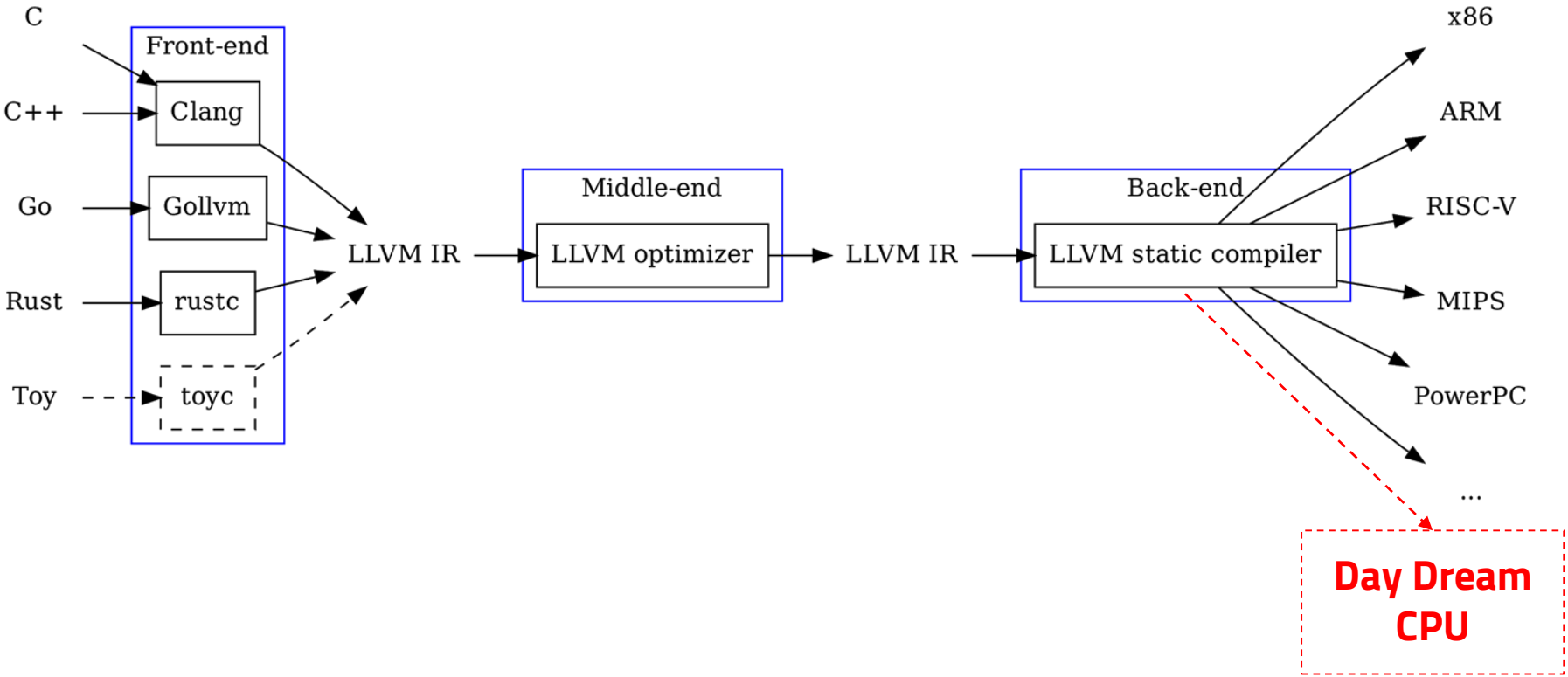




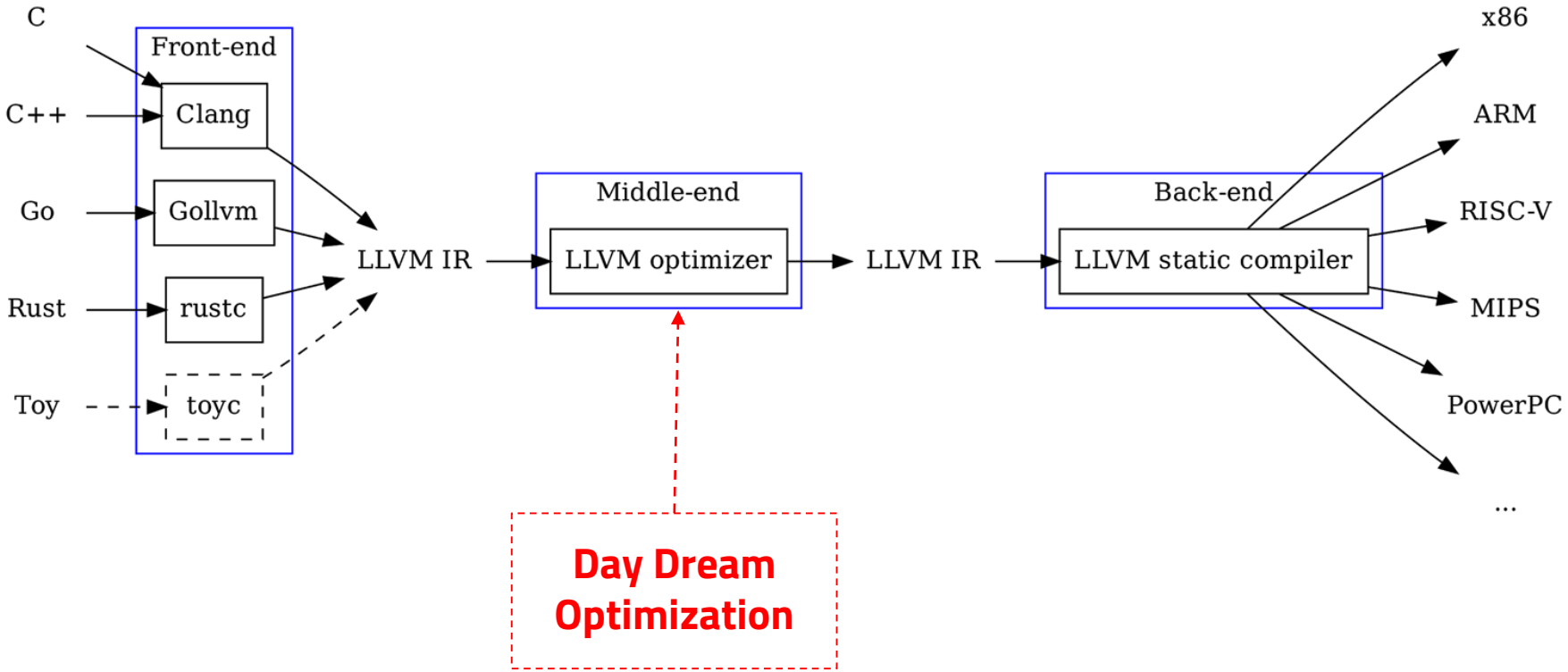
One Day I create a new language ...



One Day I create a new CPU/GPU/FPGA/ASIC ...



One Day I make a new OPTIMIZATION ...



Agenda

4.Go, let’s find it

Where to find it ?

a. Apple’s Projects

Xcode

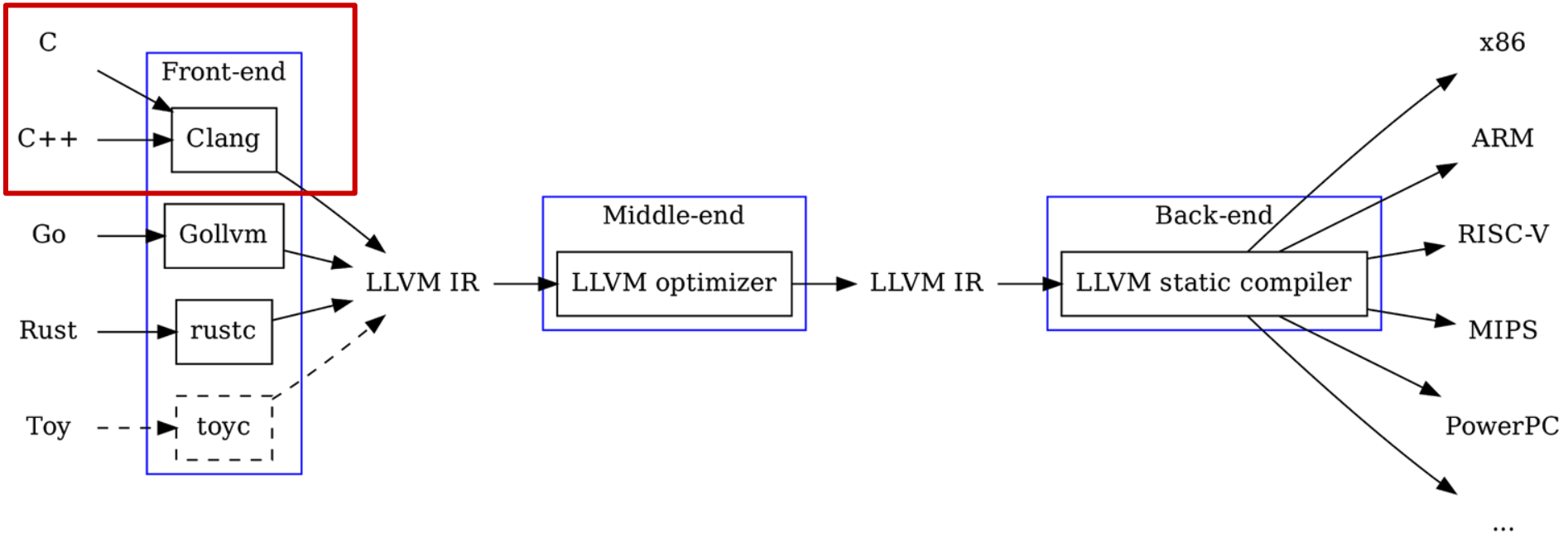
Objective-C

Xcode

Objective-C

Xcode Version	Release Date	Compilers
Xcode 2.0	2005/04/29	GCC
Xcode 3.x	2007/10/25	GCC & LLVM-GCC
Xcode 4.x	2011/03/09	LLVM-GCC
Xcode 5.x	2013/06/11	LLVM

https://xcodereleases.com/


C
C++
Front-end
Clang
Go
Gollvm
Rust
rustc
Toy
toyc
LLVM IR
Middle-end
LLVM optimizer
LLVM IR
Back-end
LLVM static compiler
x86
ARM
RISC-V
MIPS
PowerPC
...

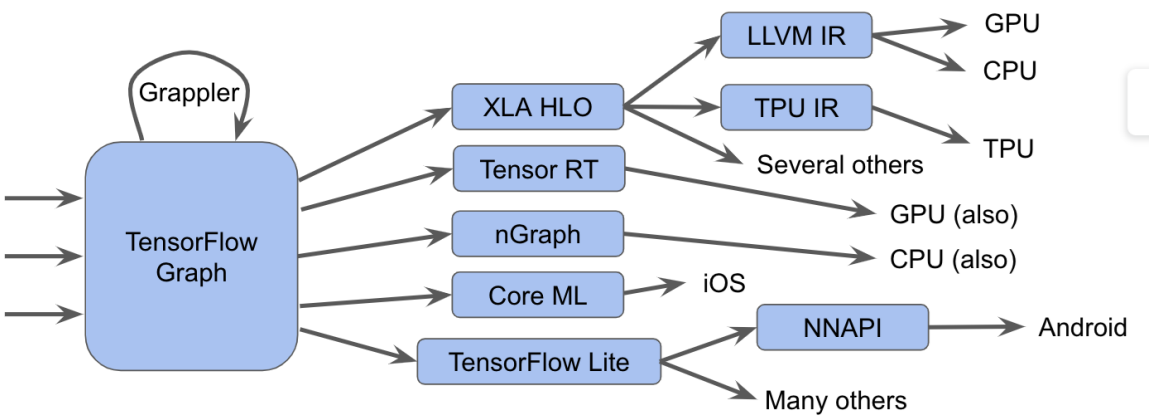
Where to find it ?

Google’s Projects

Google's Products

MLIR

MLIR: accelerating AI with open-source infrastructure

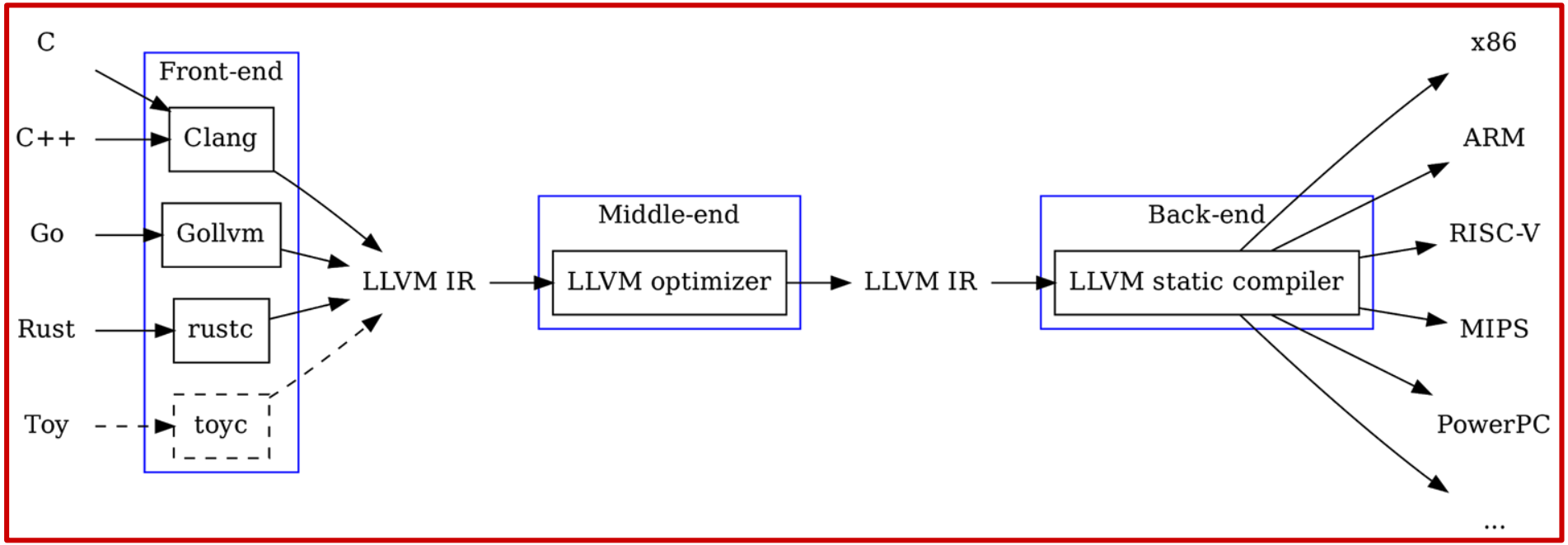


https://github.com/tensorflow/mlir

You might don't know about MLIR, but you MUST know this ...

MLIR

TensorFlow


C
C++
Go
Rust
Toy
Front-end
Clang
Gollvm
rustc
toyc
LLVM IR
Middle-end
LLVM optimizer
LLVM IR
Back-end
LLVM static compiler
x86
ARM
RISC-V
MIPS
PowerPC
...

Google’s Products

V8

V8: Emscripten is switching to the LLVM WebAssembly backend

V8 @v8js · Jul 1

🔥 Emscripten is switching to the **LLVM** WebAssembly backend, resulting in...

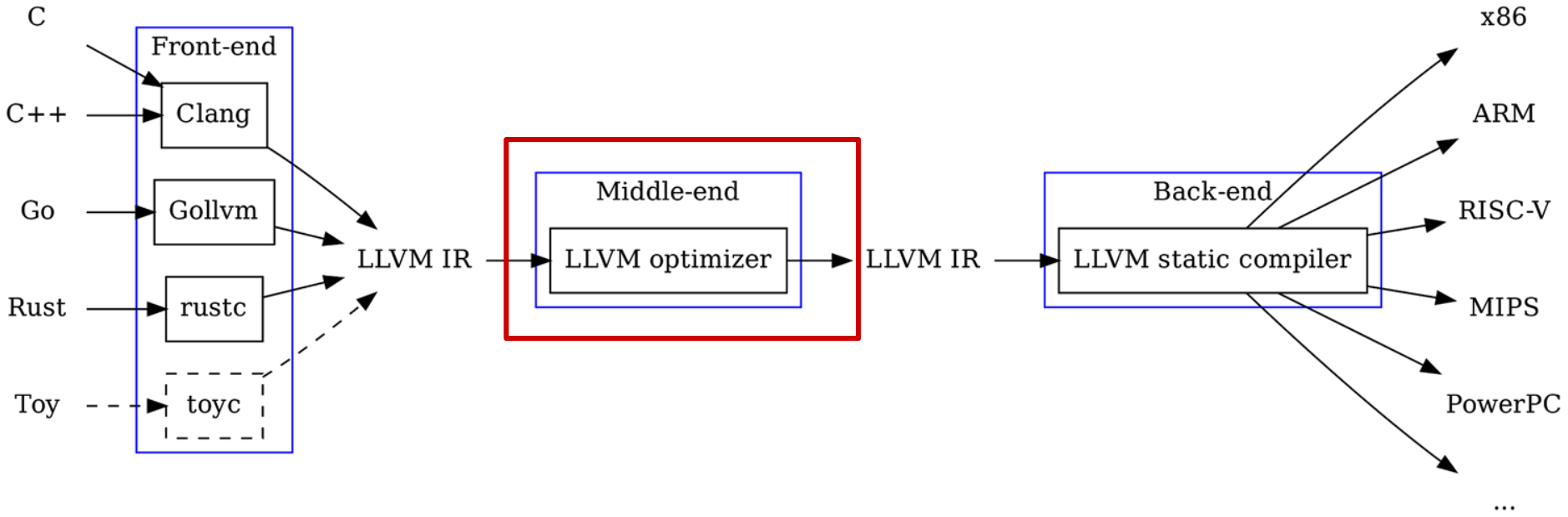
- much faster link times
- smaller and faster code
- support for all **LLVM** IR
- new WebAssembly features
- faster general updates from upstream **LLVM**

@kripken explains: v8.dev/blog/emscripte...

4 204 601

https://twitter.com/v8js/status/1145704863377981445

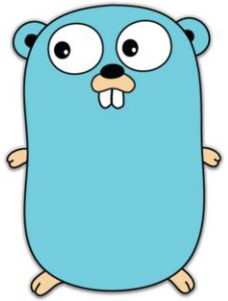
You might don’t know about V8, but you MUST know this ...


C
C++
Go
Rust
Toy
Front-end
Clang
Gollvm
rustc
toyc
LLVM IR
Middle-end
LLVM optimizer
LLVM IR
Back-end
LLVM static compiler
x86
ARM
RISC-V
MIPS
PowerPC
...

Google’s Products

gollvm

gollvm: an LLVM-based Go compiler



Go

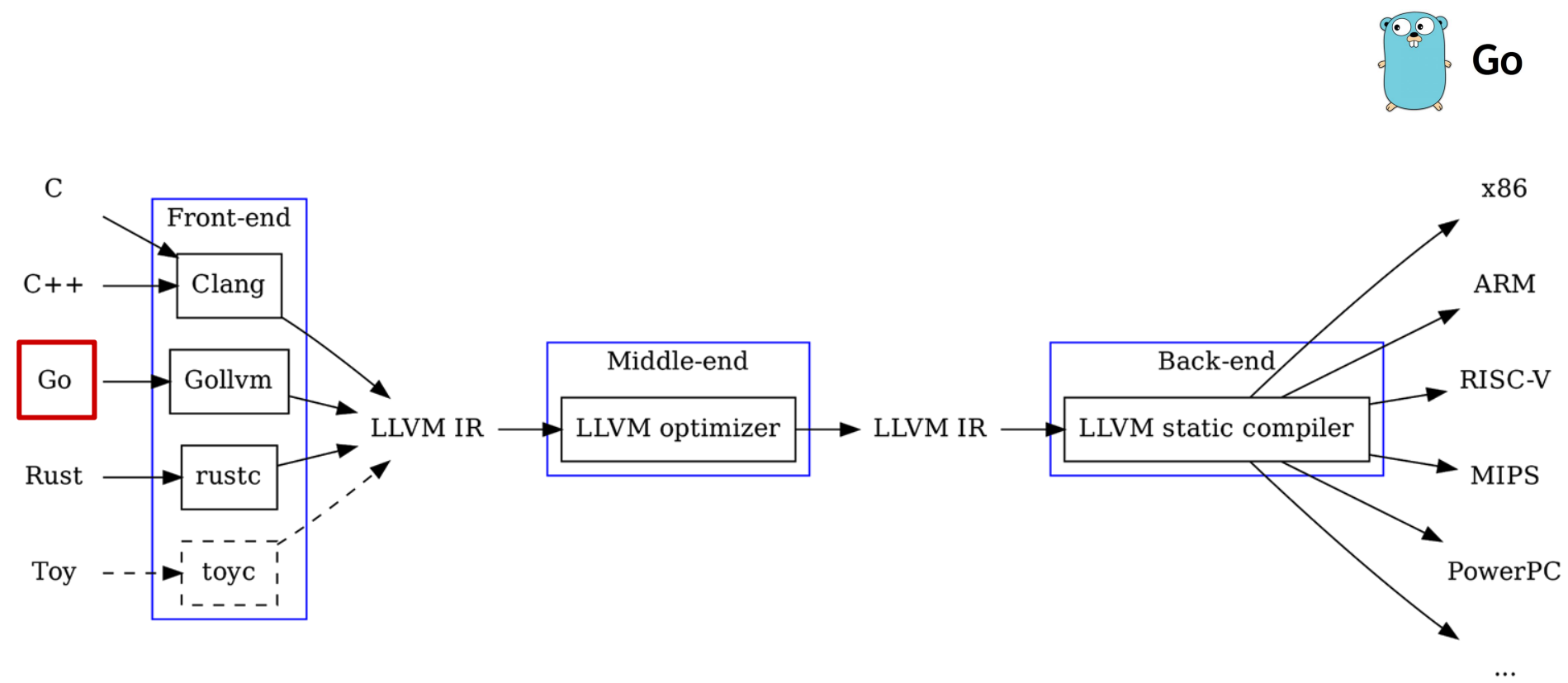
gollvm

Gollvm is an LLVM-based Go compiler. It incorporates “gofrontend” (a Go language front end written in C++ and shared with GCCGO), a bridge component (which translates from gofrontend IR to LLVM IR), and a driver that sends the resulting IR through the LLVM back end.

Gollvm (name still not finalized) is set up to be a subproject within the LLVM tools directory, similar to how things work for “clang” or “compiler-rt”: you check out a copy of the LLVM source tree, then within the LLVM tree you check out additional git repos.

You'll need to have an up-to-date copy of cmake on your system (3.6 vintage).

https://go.googlesource.com/gollvm/


Go
C
C++
Go
Rust
Toy
Front-end
Clang
Gollvm
rustc
toyc
LLVM IR
Middle-end
LLVM optimizer
LLVM IR
Back-end
LLVM static compiler
x86
ARM
RISC-V
MIPS
PowerPC
...

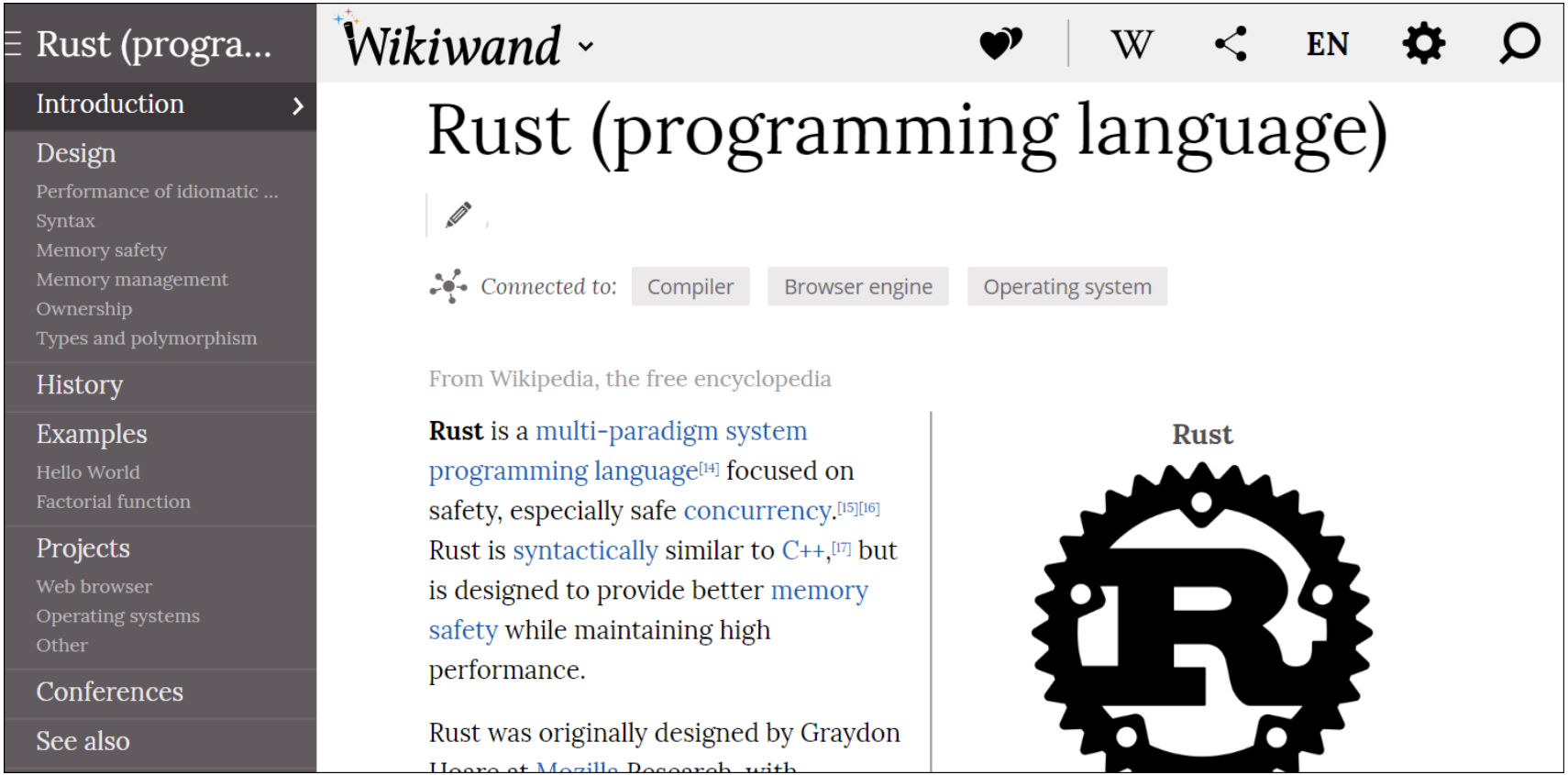
Where to find it ?

Other Projects

Other Products

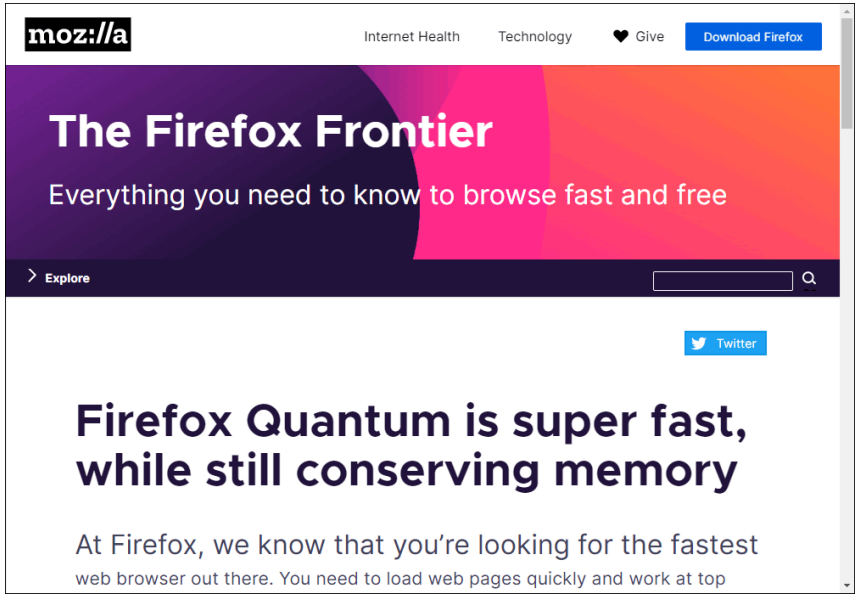
Rust Lang

Rust Lang


Rust (progra...
Introduction
Design
Performance of idiomatic ...
Syntax
Memory safety
Memory management
Ownership
Types and polymorphism
History
Examples
Hello World
Factorial function
Projects
Web browser
Operating systems
Other
Conferences
See also
Wikiwand
W
EN
Rust (programming language)
Connected to:
Compiler
Browser engine
Operating system
From Wikipedia, the free encyclopedia
Rust is a multi-paradigm system programming language[14] focused on safety, especially safe concurrency.[15][16] Rust is syntactically similar to C++,[17] but is designed to provide better memory safety while maintaining high performance.
Rust was originally designed by Graydon
Rust

You might don’t know about **Rust**, but you MUST know this ...

Firefox Quantum is super fast, while still conserving memory

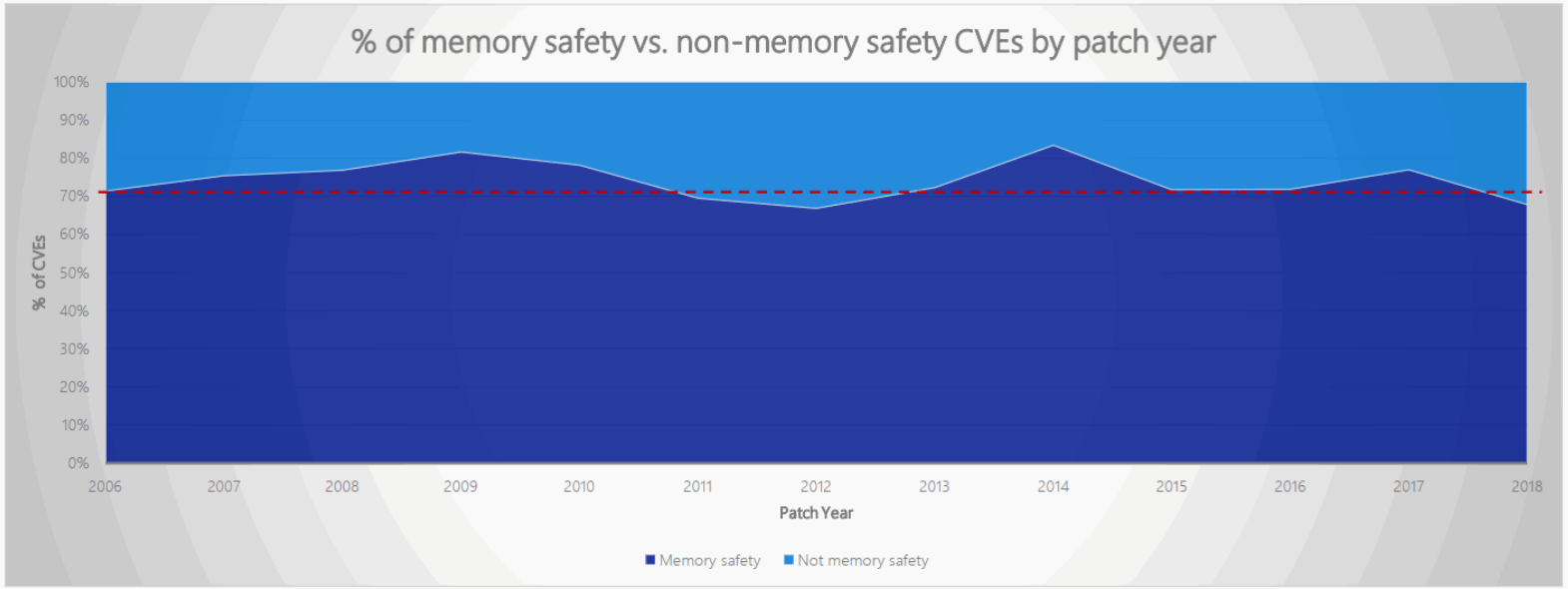


For instance, Quantum ships with a brand new CSS engine written in Mozilla's Rust programming language. Unlike other browsers, this can work in parallel across cores, rather than as one large process on a single core.

Microsoft to explore using Rust

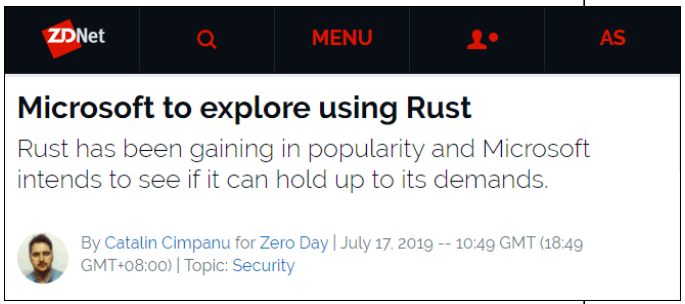
Memory safety issues remain dominant

We closely study the root cause trends of vulnerabilities & search for patterns



~70% of the vulnerabilities addressed through a security update each year continue to be memory safety issues

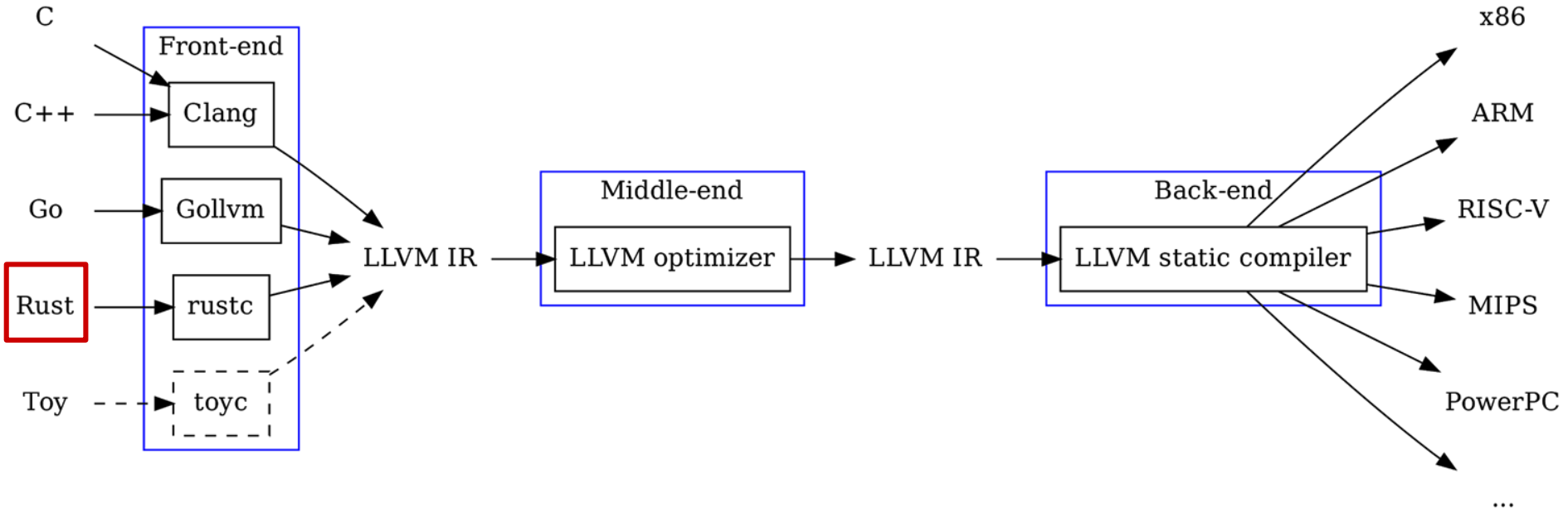
Microsoft to explore using Rust



Microsoft plans to explore using the Rust programming language as an alternative to C, C++, and others, as a way to improve the security posture of its and everyone else's apps.

The announcement was made yesterday by Gavin Thomas, Principal Security Engineering Manager for the Microsoft Security Response Center (MSRC).

"You're probably used to thinking about the Microsoft Security Response Center as a group that responds to incidents and vulnerabilities," Thomas said. "We are a response organization, but we also have a proactive role, and in a new blog series we will highlight Microsoft's exploration of safer system programming languages, starting with Rust."


C
C++
Go
Rust
Toy
Front-end
Clang
Gollvm
rustc
toyc
LLVM IR
Middle-end
LLVM optimizer
LLVM IR
Back-end
LLVM static compiler
x86
ARM
RISC-V
MIPS
PowerPC
...

Agenda

5. Go, let’s find it

(JIT)

4. Go, let's find it (Just-In-Time)

What is JIT?

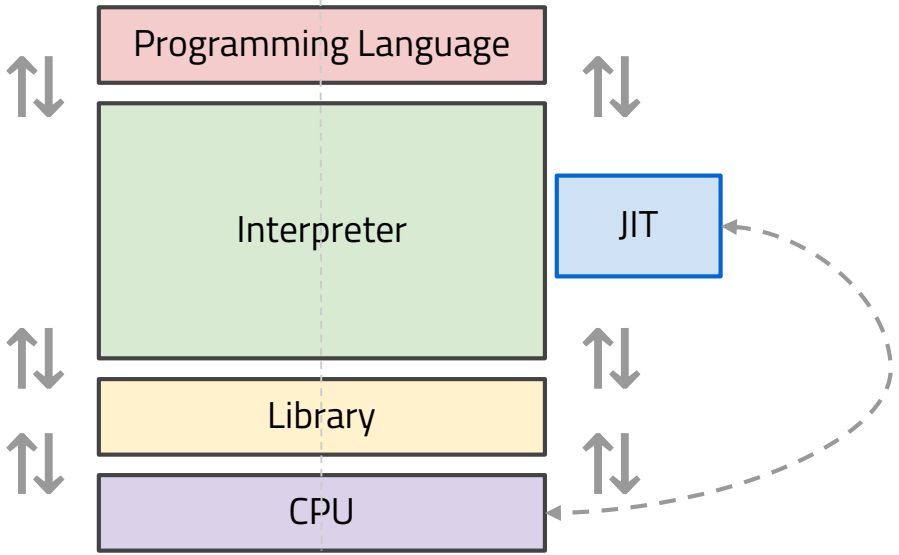
What is JIT?

Why we need JIT?

Develop FAST & Run FAST

What is JIT?

SLOWER



FASTER

python™ Ruby Java JS Microsoft® .NET Framework

How JIT works?

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>

// prints out the error and returns NULL.
void* alloc_executable_memory(size_t size)

{
  void* ptr = mmap(0, size, PROT_READ | PROT_WRITE | PROT_EXEC, MAP_PRIVATE |
MAP_ANONYMOUS, -1, 0);

  if (ptr == (void*)-1) {
    perror("mmap");
    return NULL;
  }

  return ptr;
}
```

https://eli.thegreenplace.net/2013/11/05/how-to-jit-an-introduction

How JIT works?

https://eli.thegreenplace.net/2013/11/05/how-to-jit-an-introduction

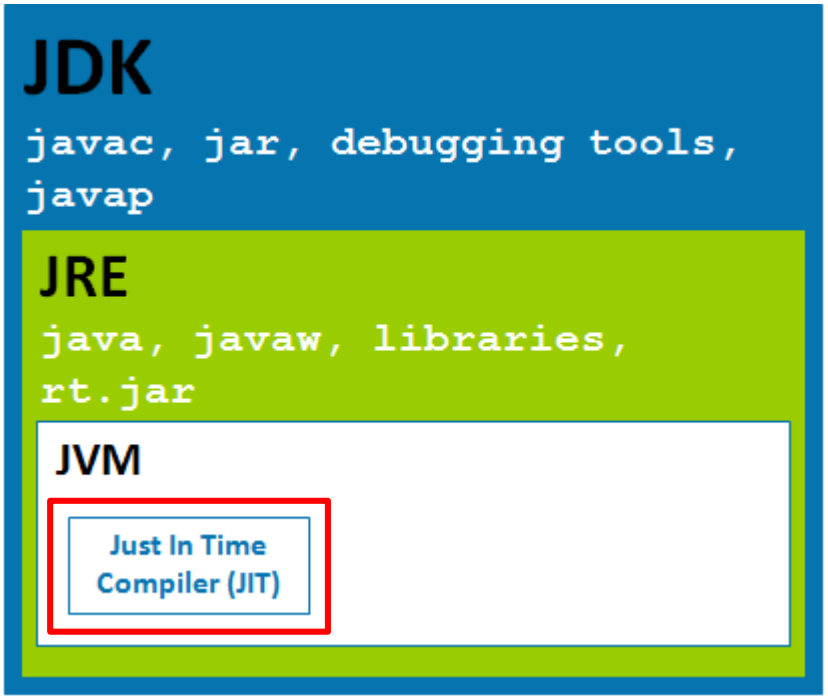
```
void emit_code_into_memory(unsigned char* m) {
  unsigned char code[] = {
    0x48, 0x89, 0xf8,                   // mov %rdi, %rax
    0x48, 0x83, 0xc0, 0x04,             // add $4, %rax
    0xc3                                // ret
  };  memcpy(m, code, sizeof(code));
}

const size_t SIZE = 1024;
typedef long (*JittedFunc)(long);
void run_from_rwx() {
  void* m = alloc_executable_memory(SIZE);
  emit_code_into_memory(m);
  JittedFunc func = m;
  int result = func(2);
  printf("result = %d\n", result);
}
```

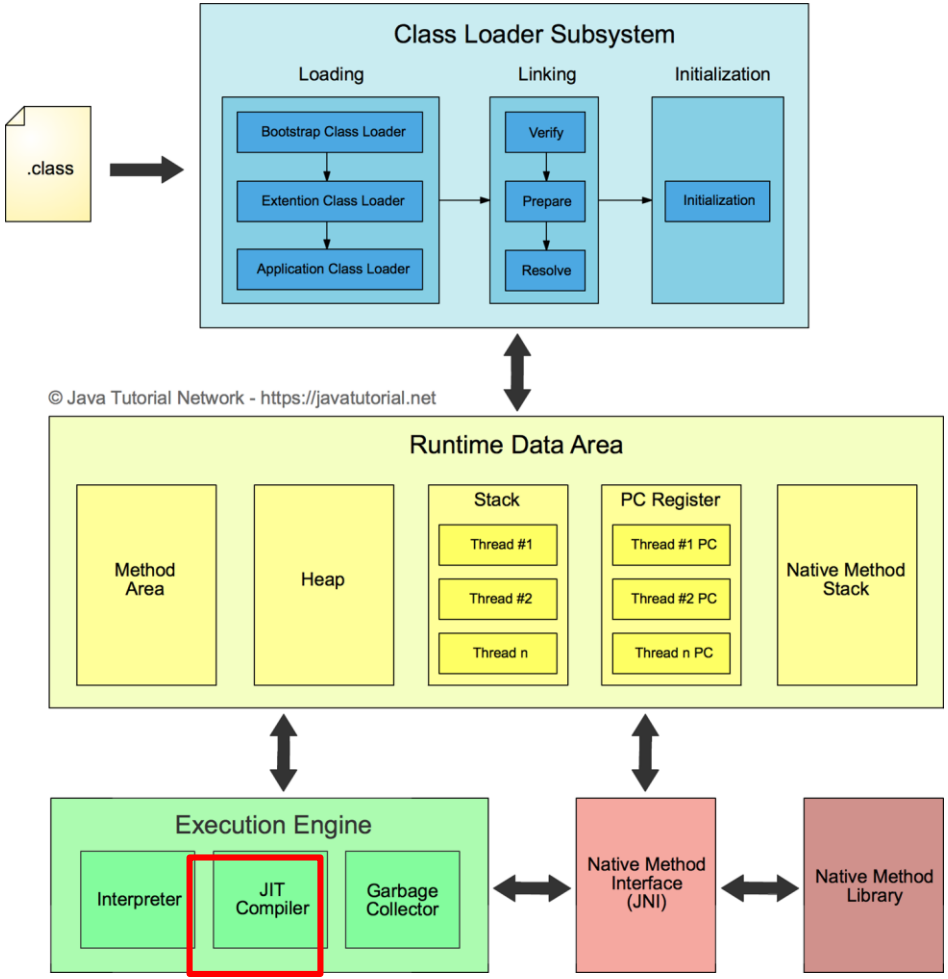
```
long add4(long num) {
  return num + 4;
}
```

Project with LLVM

JVM

Java Virtual Machine



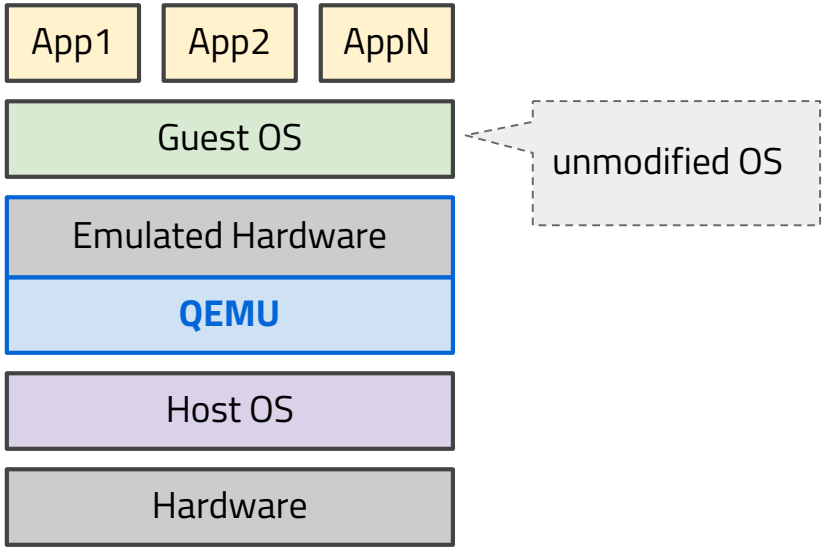


https://javatutorial.net/wp-content/uploads/2017/10/jvm-architecture-768x793.png

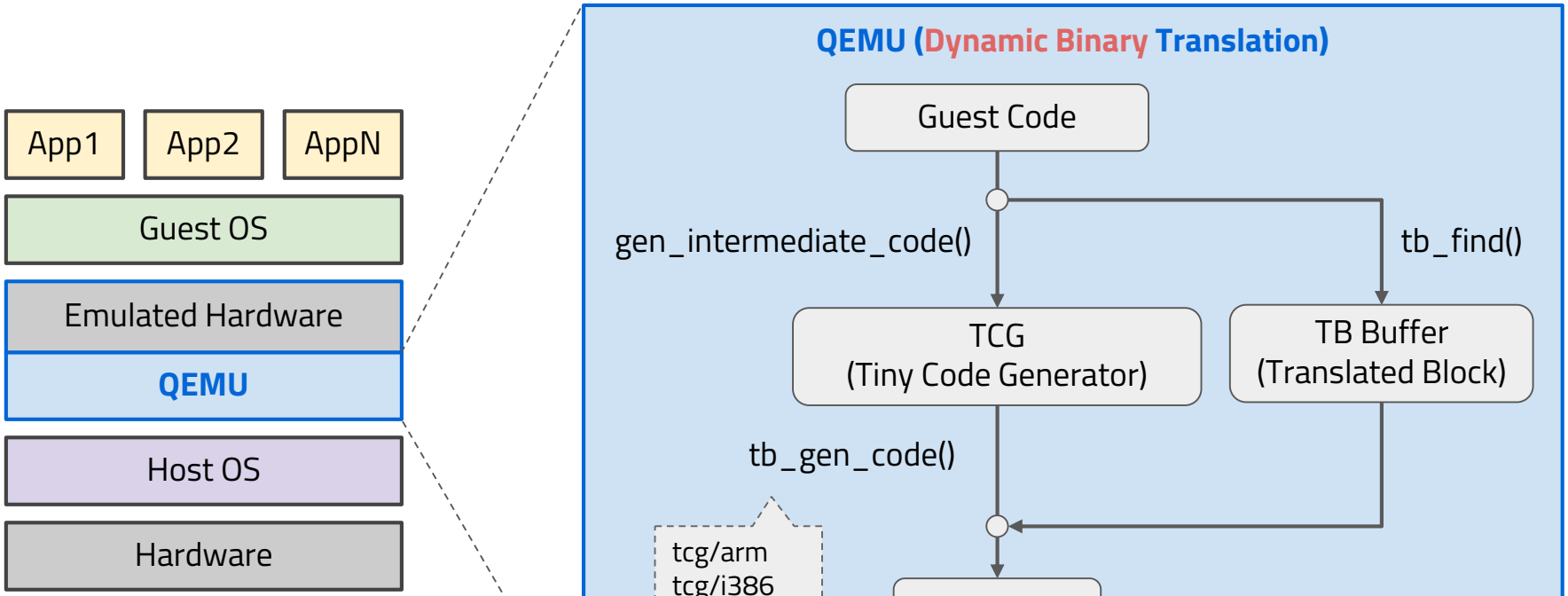
Project with JIT

QEMU / Virtual Box/ VMware

QEMU (Quick Emulator)



QEMU (Quick Emulator)



Agenda

6. Go, let’s find it

(Web Technology)

4. Go, let's find it (WebAssembly)

What is WebAssembly

Project with WebAssembly

JSLinux

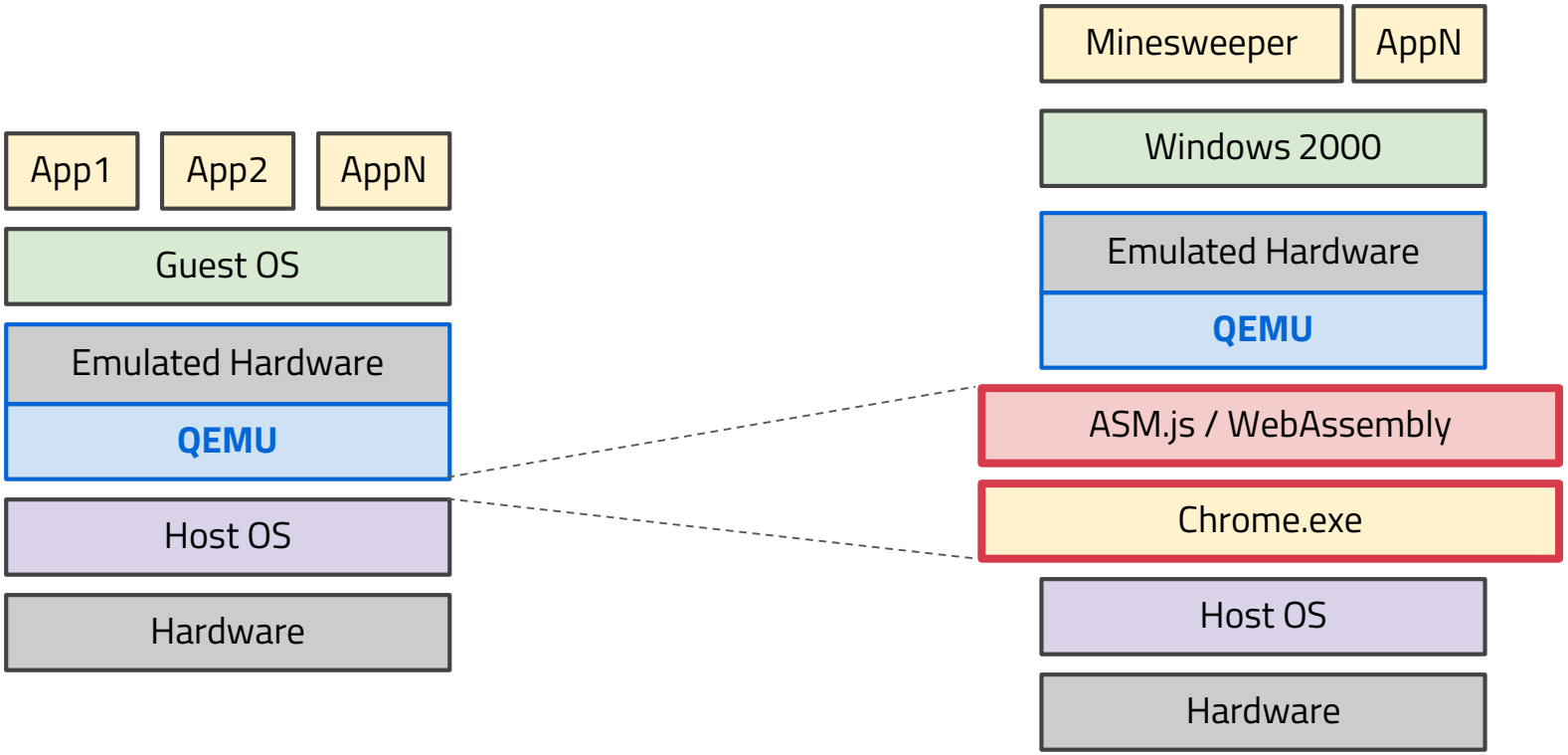
JSLinux

Run Windows 2000 on Web Browser

CPU	OS (Distribution)	User Interface	VFsync access	Startup Link	TEMU Config	Comment
x86	Linux 4.12.0 (Buildroot)	Console	Yes	click here	url	
x86	Linux 4.12.0 (Buildroot)	X Window	Yes	click here	url	Right mouse button for the menu.
x86	Windows 2000	Graphical	No	click here	url	Disclaimer.
x86	FreeDOS	VGA Text	No	click here	url	
riscv64	Linux 4.15.0 (Buildroot)	Console	Yes	click here	url	
riscv64	Linux 4.15.0 (Buildroot)	X Window	Yes	click here	url	Right mouse button for the menu.
riscv64	Linux 4.15.0 (Fedora 29)	Console	Yes	click here	url	Warning: longer boot time.
riscv64	Linux 4.15.0 (Fedora 29)	X Window	Yes	click here	url	Warning: longer boot time. Right mouse button for the menu.

https://bellard.org/jslinux/vm.html?url=https://bellard.org/jslinux/win2k.cfg&mem=192&graphic=1&w=1024&h=768

JSLinux

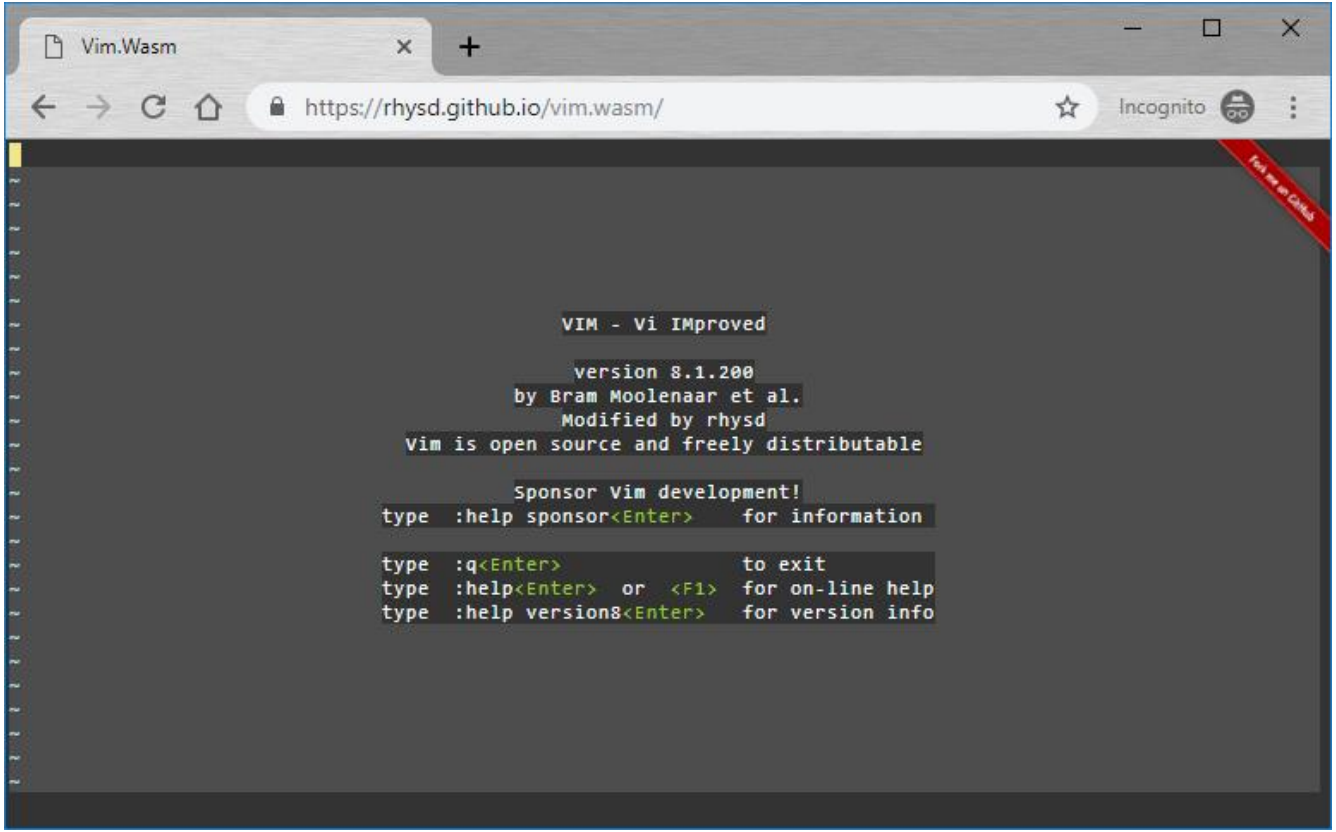

App1
App2
AppN
Guest OS
Emulated Hardware
QEMU
Host OS
Hardware
Minesweeper
AppN
Windows 2000
Emulated Hardware
QEMU
ASM.js / WebAssembly
Chrome.exe
Host OS
Hardware

Project with WebAssembly

vim.wasm

vim.wasm

https://rhysd.github.io/vim.wasm/



Project with WebAssembly

Google Earth

Google Earth

https://earth.google.com/web/

Project with WebAssembly

Others

Project with WebAssembly

Autodesk, June 8, 2019

Autodesk releases AutoCAD web app for Firefox

Autodesk today announced Firefox support for the AutoCAD Web App. AutoCAD is based on a codebase written decades ago. The Webassembly port demonstrates the potential of Webassembly to overcome barriers separating previously fragmented areas of software development divided by operating system, programming languages and frameworks.

Smilkov et al., February 4, 2019

TensorFlow developers looking into WebAssembly and WebGPU support

Two new web standards, WebAssembly and WebGPU, both have potential to improve TensorFlow.js performance. WebGPU is an emerging standard to express general purpose parallel computation on the GPU, enabling more optimised linear algebra kernels than those the WebGL backend can support today.

Ricardo Cabello (Mr. Doob), Corentin Wallez, May 9, 2019

Next-Generation 3D Graphics on the Web

This talk will cover the latest updates for adding 3D to your web site. You'll learn how to use <model-viewer> and new features for rendering, AR, and interactivity. You'll also see some phenomenal sites created with three.js. Then you'll get an overview of the WebGPU API that will provide modern features such as “GPU compute” as well as lower overhead access to GPU hardware and better, more predictable performance. Expect performance gains in Tensorflow.js up to 10x.

Dropbox, June 19, 2018

Dropbox's new compression algorithms written in Rust

DivANS is written in the Rust programming language, since Rust has guarantees about safety, security, and determinism in the safe subset of the language. Rust can also be as fast as hand-tuned C code and doesn’t need a garbage collector. Rust programs embed well in any programming language that support a C foreign function interface (FFI) and even allow runtime selection of the memory allocator through that C FFI. These properties make it easy to embed the DivANS codec in a webpage with WASM.

https://webassembly.eu/

Agenda

7. Q&A

END

HackMD Note
http://bit.ly/369THkW

Twitter
https://twitter.com/dougpuob