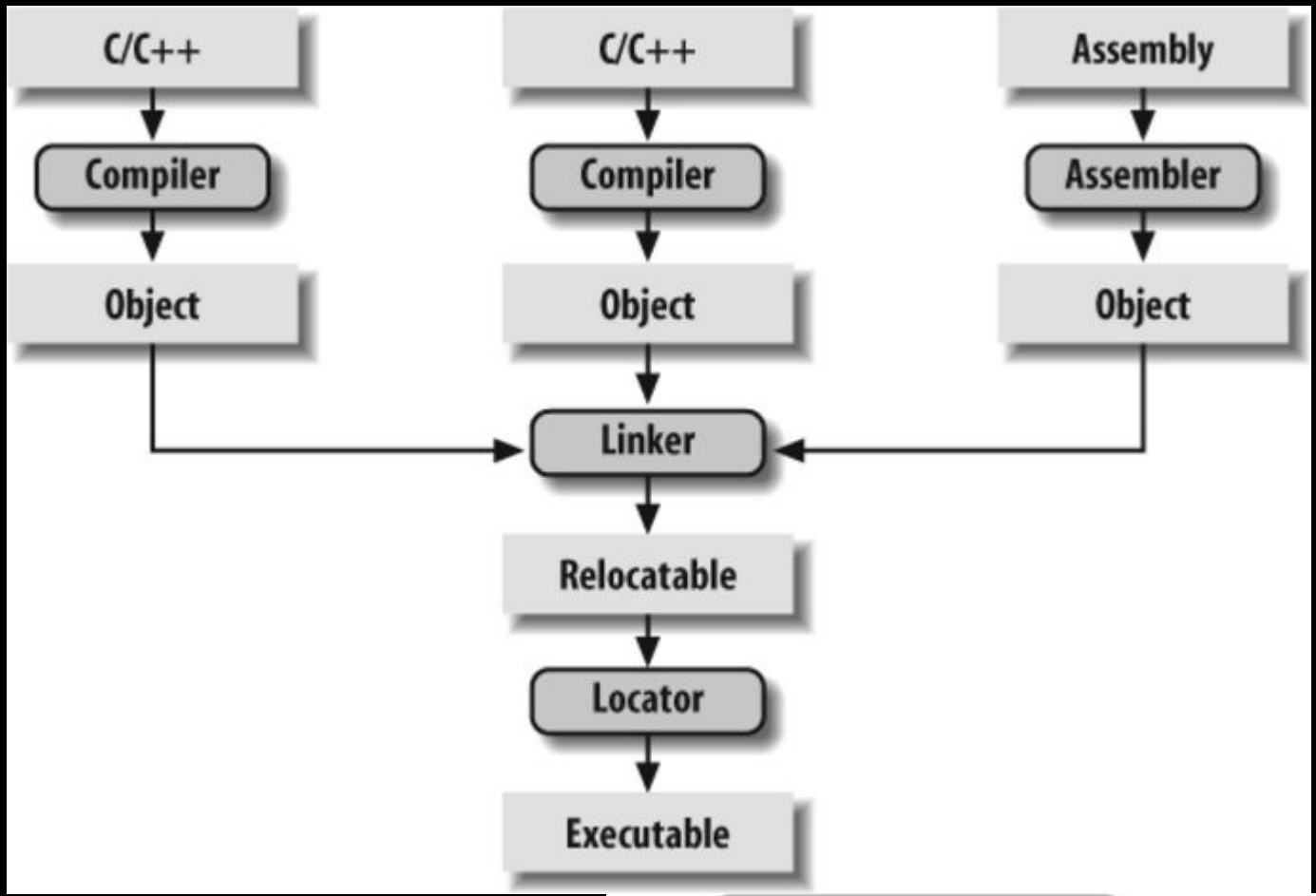
窮得只剩下 Compiler

Jim Huang(黃敬群) “jserv”
website: http://jserv.sayya.org/
blog: http://blog.linux.org.tw/jserv/
Apr 19, 2009 @ OSDC.tw

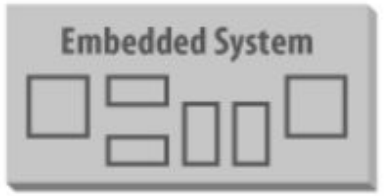
十字架 = cross

cross compiler

corss compiler 在新時代中，被賦予新的意義



The development tools that build the embedded software run on a general-purpose computer



Target

The embedded software that is built by those tools runs on the embedded system

簡單的 C 語言程式

```
void rewind(FILE *pLove);

int main(int argc, char **argv)
{
    FILE *p = fopen("/dev/heart", "w+");
    rewind(p);
    fclose(p);
    return 0;
}
```

```
void rewind(FILE *pLove);

int main(int argc, char **argv)
{
    FILE *p = fopen("/dev/heart", "w+");
    rewind(p);
    fclose(p);
    return 0;
}
```

Compiler

宅色夫 @OSDC.tw# 癡傻 % 愚昧 & 宅 *** 最後一次見到你的路口我現在才明白那原來是一條河或是一道地層下陷從那裡開始時間有了不同的轉速我們再也不站立在同一個地面了從軌道最靠近交錯的那一點逸出朝向全然不同的宇宙逐步擴張的距離我曾經以爲會是荒涼的而今竟令我心安所謂錯過並不是什麼「如果那時再努力一點」或「要是做了另一個決定」就好的事從來都不是那是兩個星系不同的軌道與規則 .....

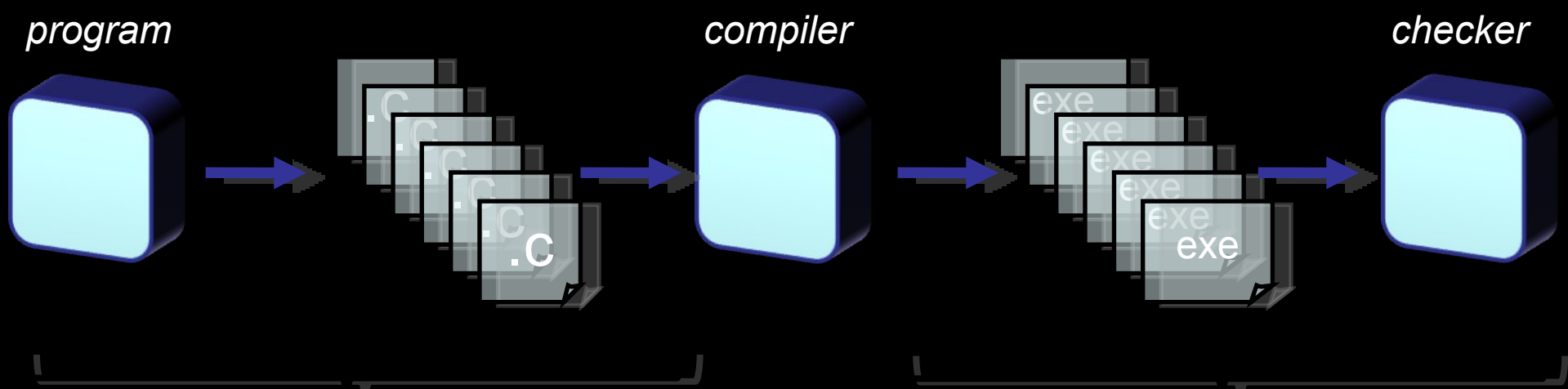
探索人生的進入點
打開心扉，強制撕裂著自己接受
愛回到最初，不過是場空
於是，我悄悄閉塞內心的思緒
終了，就回到原點吧

Interpreter

探索人生的進入點，打開心扉，強制撕裂著自己接受曾經以爲會是荒涼的而今竟令我心安所謂錯過並不是什麼「如果那時再努力一點」或「要是做了另一個決定」 ...

JIT compiler

回顧編譯器流程



解析原始程式碼，建立 AST (Abstract Syntax Tree)
依據 AST ，生成中間的 object ，稍後作 linking
執行生成的機械碼

編譯器離我們好遠，是嗎？

運算模式已大幅改觀
  Framework-driven
  SIMD/vectorization, Cell, muticore/SMP, ...

虛擬化 (Virtualization) 技術的時代：更多元、更安全、更有效率地使用硬體

資訊技術的雜交 (cross-over)

LLVM 正夯！

提綱

隱藏在我們周遭的 Compiler 技術

回首 Compiler 已遠：從 A0 到 LLVM

透過 LLVM 建構虛擬機器

技術大融通與未來展望

隱藏在我們周遭的編譯器

Java/.Net(虛擬機器 +Just-In-Time compiler)

網路瀏覽器

- Mozilla/Firefox (ActionMonkey/Tamarin)
- WebKit (SquirrelFish)
- Google Chrome (V8 engine)

Web 應用程式： JSP/Servlet, SilverLight/.Net

手機平台： Java ME, Android, iPhone

繪圖軟體： Adobe PixelBender, Shader

3D 高品質圖形處理： Gallium3D / OpenGL / Direct3D

Apple Inc. 是 LLVM 開發最主要的贊助廠商，僱用 Chris Lattner

回首 Compiler 已遠
從 A0 到 LLVM

Grace Hopper (1906-1992)
美國電腦科學家、海軍軍官
1951-1952 年間，爲 UNIVAC I 開發 A-0 system (Arithmatic Language version 0) 編譯器，具備 loader/linker

多才多藝的

LLVM

Google

llvm

搜尋圖片 搜尋所有網站 進階圖片搜尋 | 使用偏好

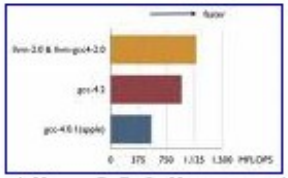
已使用「中度篩選」的「安全搜尋」

圖片 顯示： 所有大小的圖片 任何內容

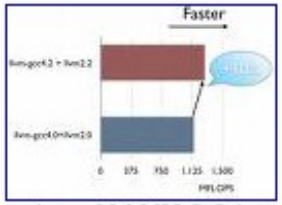
搜尋結果第 **1** - **20** 項，共約 **10,900** 項 (需時 0.08 秒)

... LangImpl5-cfg.png ...
423 x 315 - 38k - png
llvm.org
[更多來自 https://llvm.org 的資訊]

Like SproutCore, **LLVM** is neither new ...
395 x 277 - 140k - jpg
www.roughlydrafted.com

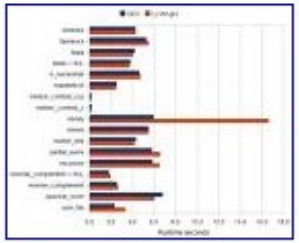


... and **llvm**-2.0 & **llvm**-gcc4-2.0.
550 x 335 - 50k - png
lucille.atso-net.jp
[更多來自 lucille.atso-net.jp 的資訊]

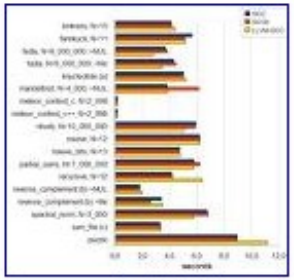


(The value of **LLVM** 2.0 is taken from ...
493 x 355 - 61k - png
lucille.atso-net.jp

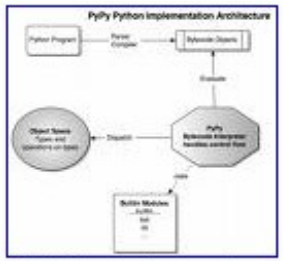
JITTutorial1.html · JITTutorial2-1. ...
828 x 452 - 58k - png
llvm.org



... between the **LLVM** compiler and ...
661 x 539 - 9k - png
leonardo-m.livejournal.com
[更多來自 www.fantascienza.net 的資訊]



Compilers used: **LLVM**-gcc V. 2.5 ...
620 x 596 - 11k - png
leonardo-m.livejournal.com



... **LLVM** (since August last year) ...
499 x 465 - 44k - png
codespeak.net

... 臨時，唱出來的才有說服力。
423 x 447 - 58k
lunayuan.spaces.live.com
[更多來自 blufiles.storage.live.com 的資訊]

很有耶誕節FU的巴黎
533 x 400 - 60k
lulu0318.spaces.live.com

LLVM =

Low Level Virtual Machine

三特性、三元素、三頭六臂

LLVM =

Low Level Virtual Machine

此 VM 非彼 VM

“LLVM does not imply things that you would expect from a high-level virtual machine. It does not require garbage collection or run-time code generation (In fact, LLVM makes a great static compiler!). Note that optional LLVM components can be used to build high-level virtual machines and other systems that need these services.”

http://www.llvm.org/

LLVM 三大特性

Low-Level VM

完整的編譯器基礎建設

- 可重用的、用以建構編譯器的軟體元件（compiler compiler 顯然不足以應付）
- 允許更快更完整的打造新的編譯器
- static compiler, JIT, trace-based optimizer, ...

開放的編譯器框架

- 多種程式語言支援
- 高彈性的自由軟體授權模式 (BSD License)
- 豐富的編譯輸出： C, machine code (Alpha, ARM, x86, Sparc, PowerPC, Cell SPU, 台灣心 Andes)

LLVM 三大元件

RISC 式虛擬指令集 (instruction set)

多種語言適用、與硬體架構無關的 IR(Intermediate Representation)

完整的高度整合函式庫與編譯器服務

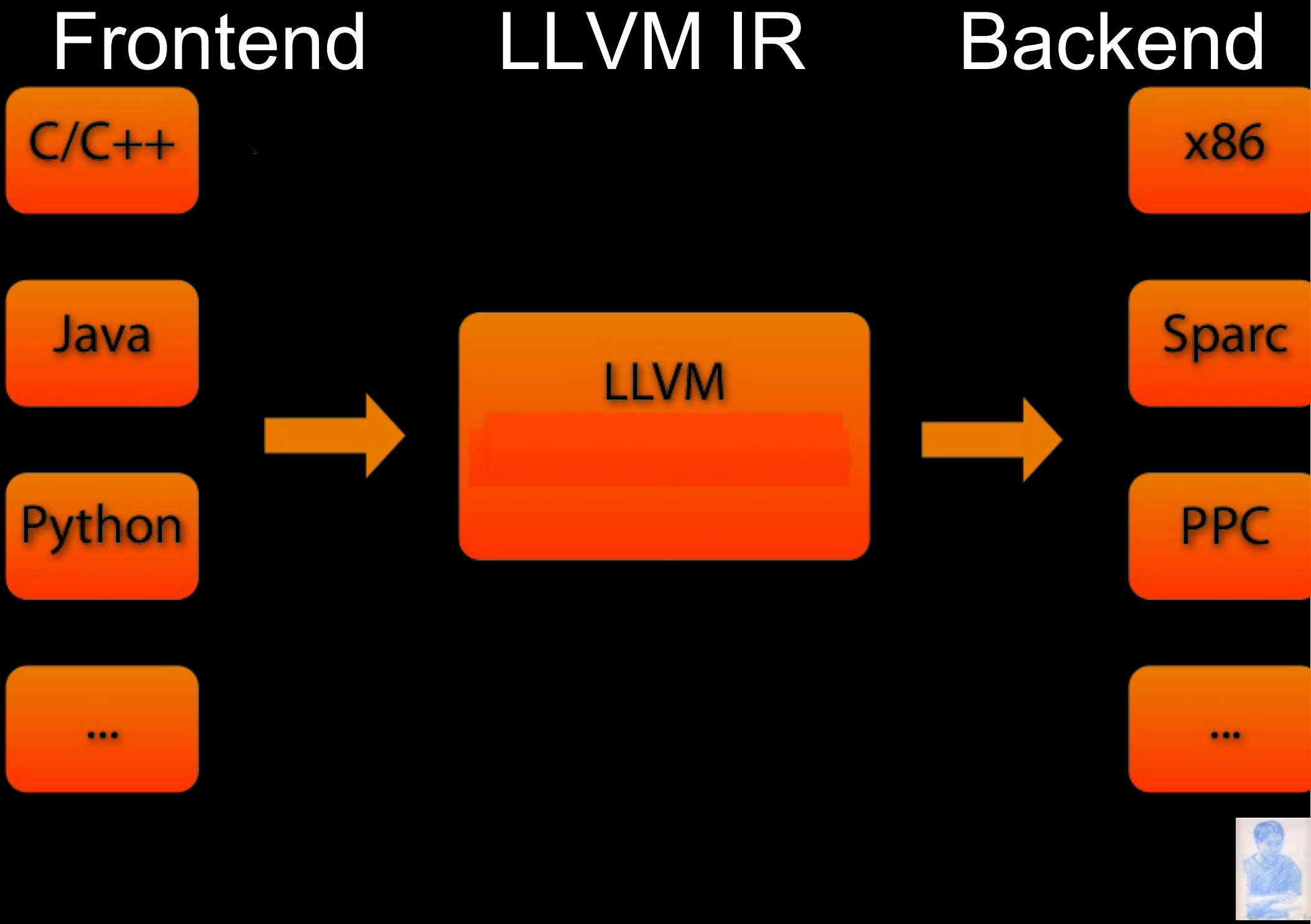
Analyses, optimizations, code generators, JIT compiler, garbage collection support, profiling, …

豐富的工具集

Assemblers, automatic debugger, linker, code generator, compiler driver, modular optimizer, …

LLVM:

三頭六臂


Frontend
LLVM IR
Backend
C/C++
Java
Python
...
LLVM
x86
Sparc
PPC
...

Frontend LLVM IR Backend

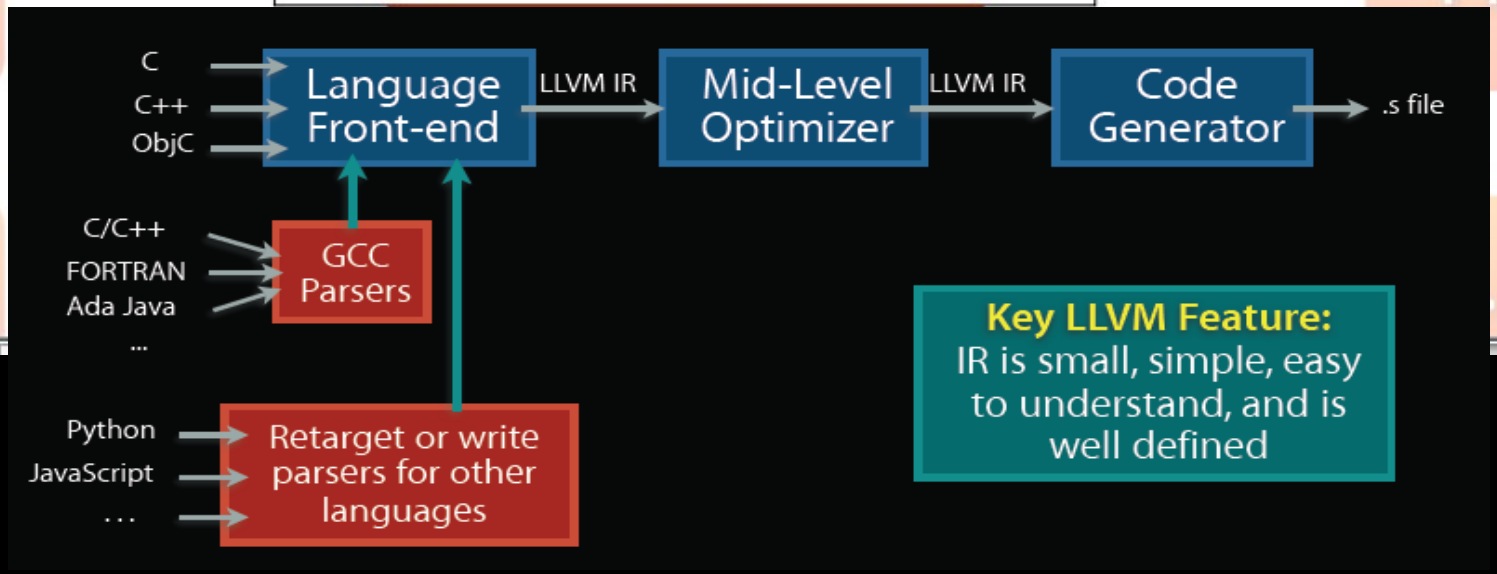
C/C++

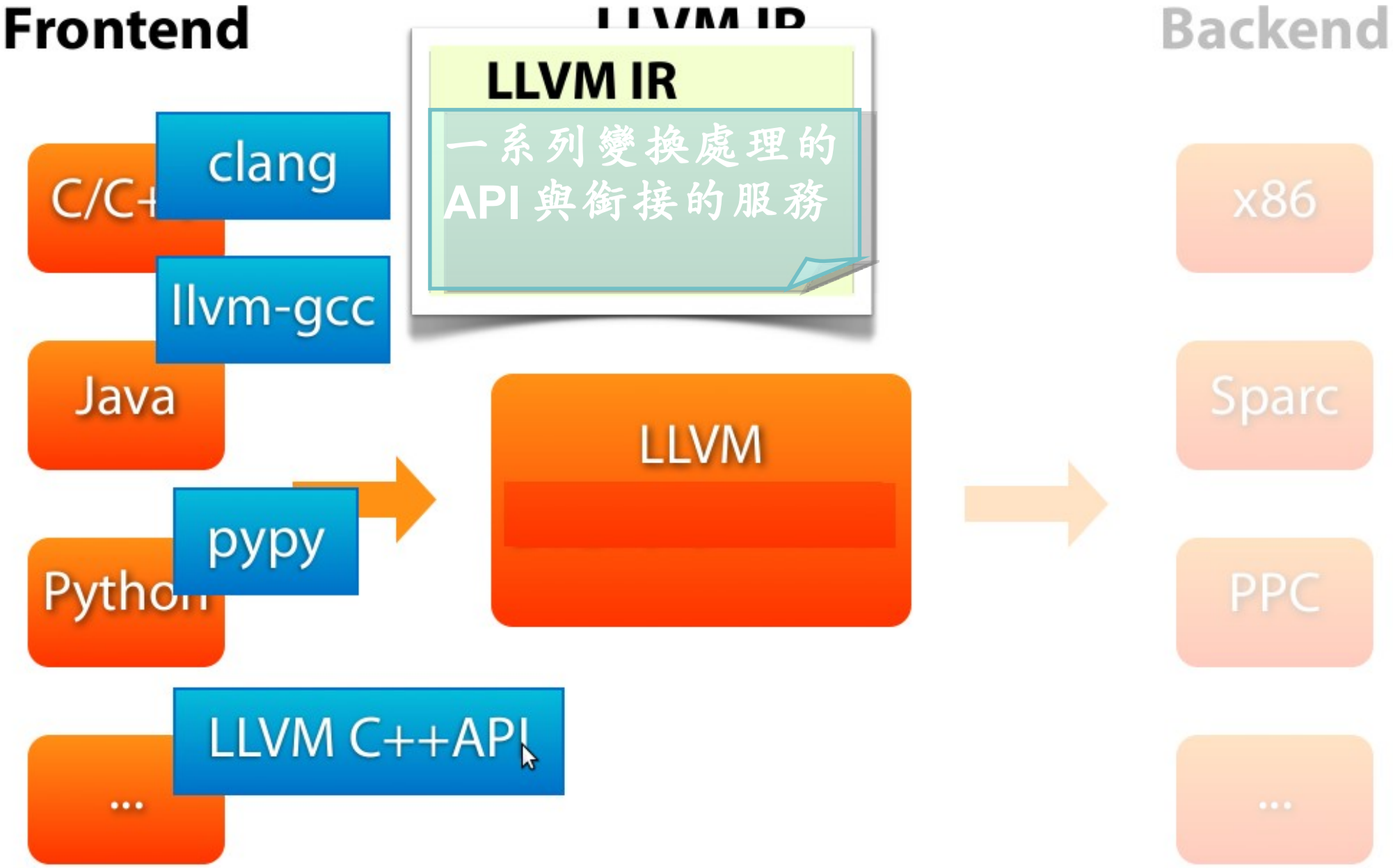
x86

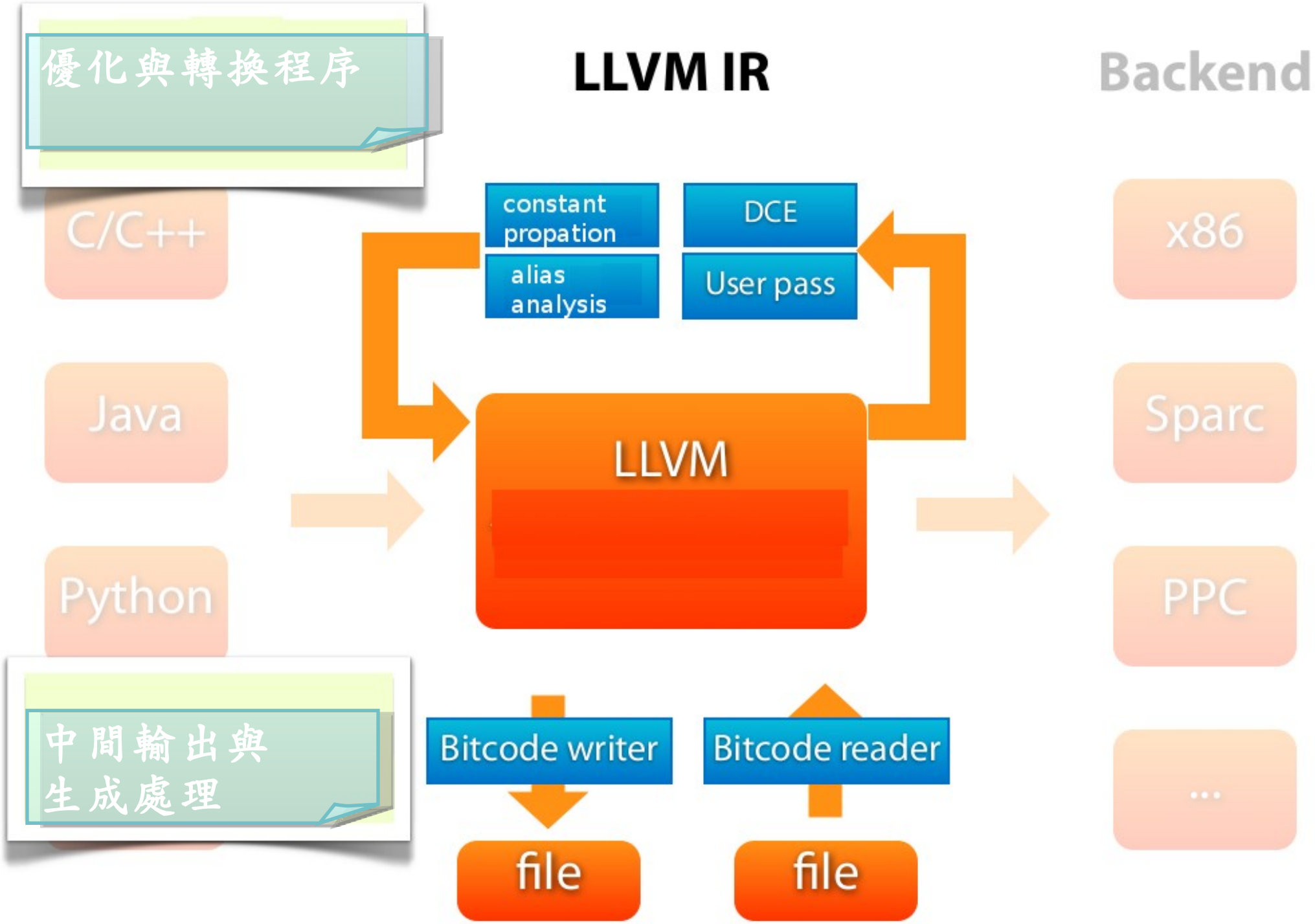
```
int add_func(
  int a, int b)
{
  return a + b;
}
```

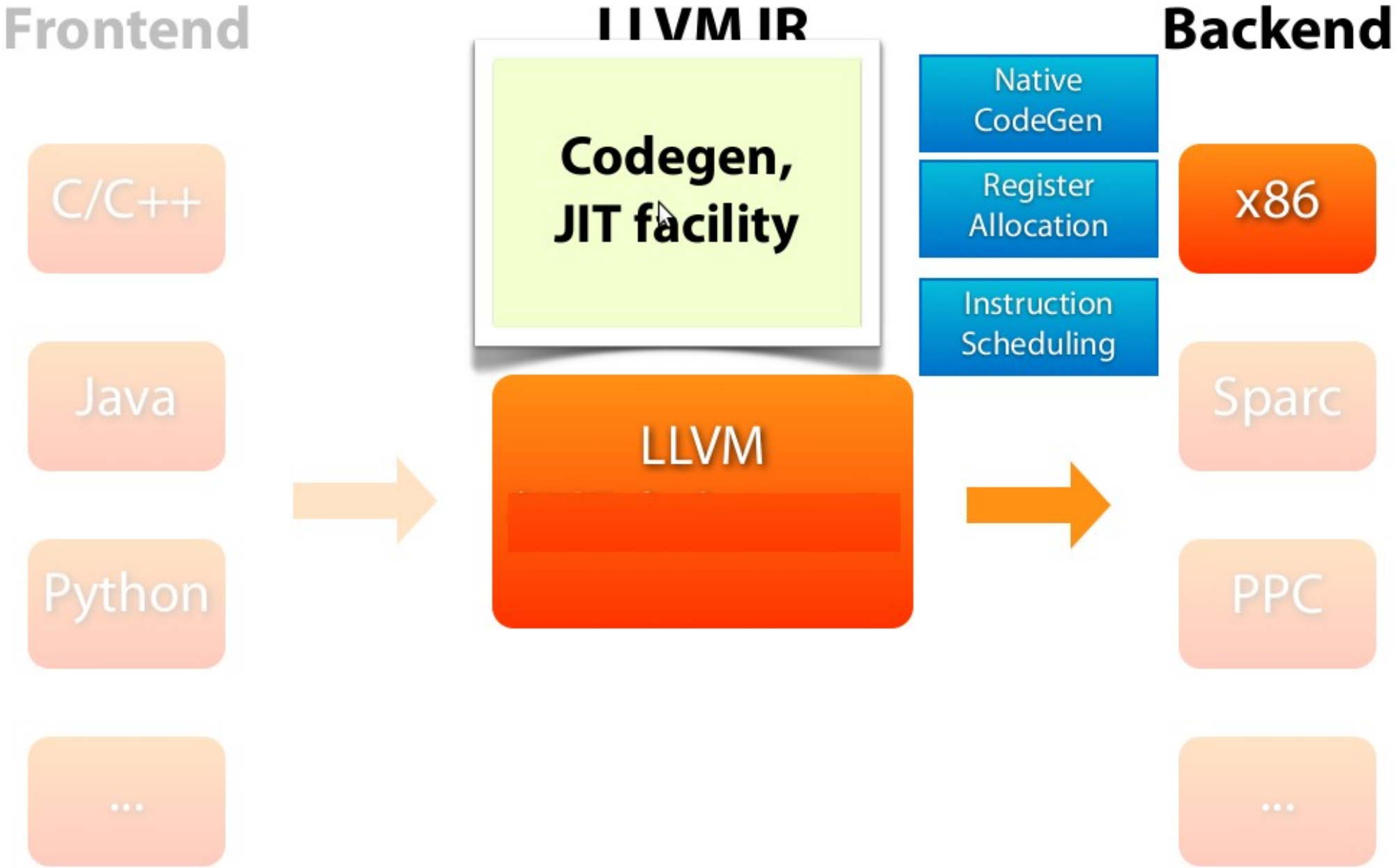
```
define i32 @add_func(i32 %a, i32 %b) {
entry:
     %tmp3 = add i32 %b, %a
     ret i32 %tmp3
}
```

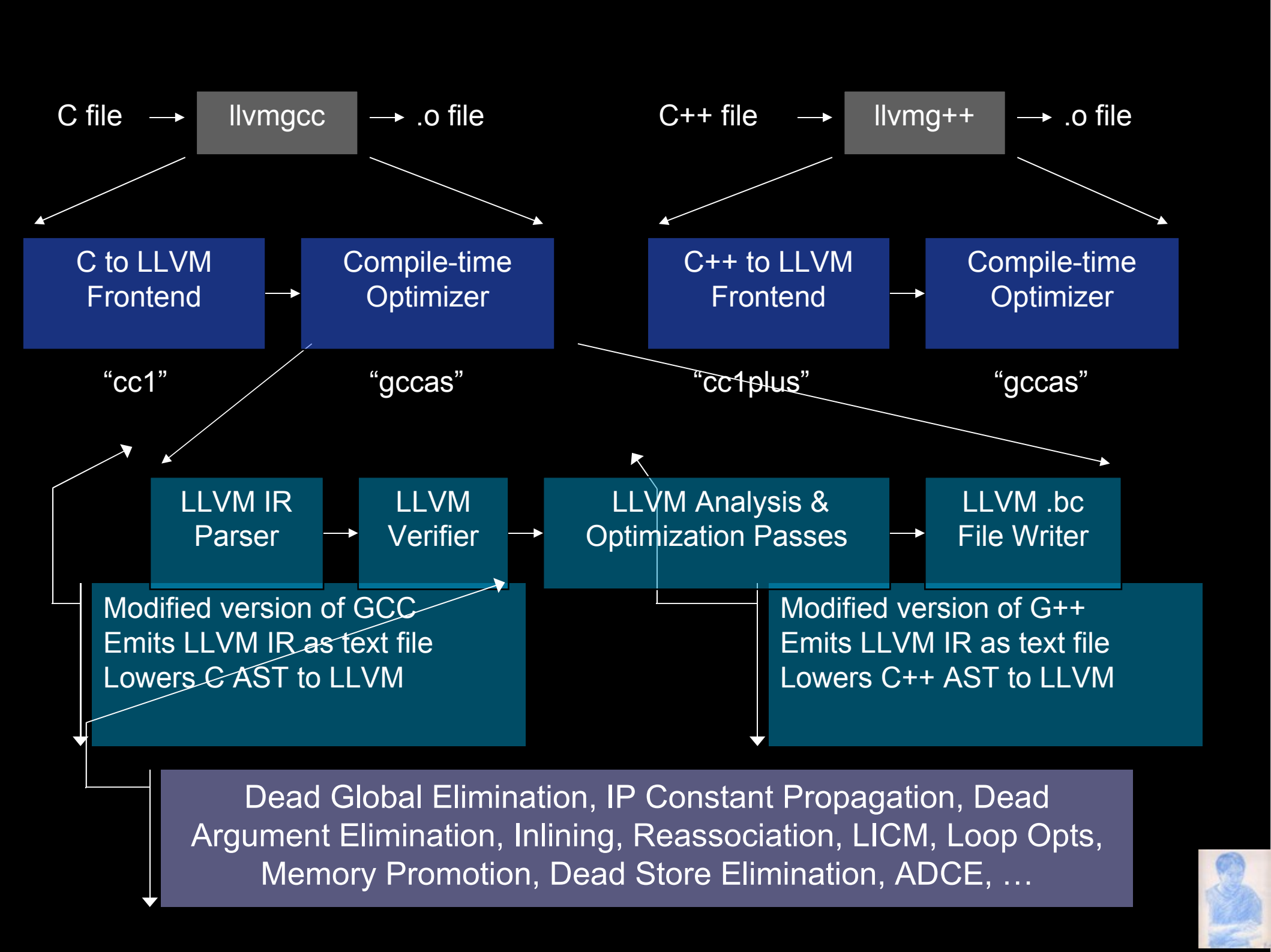
```
_add_func:
  movl 8(%esp), %eax
  addl 4(%esp), %eax
    ret
```

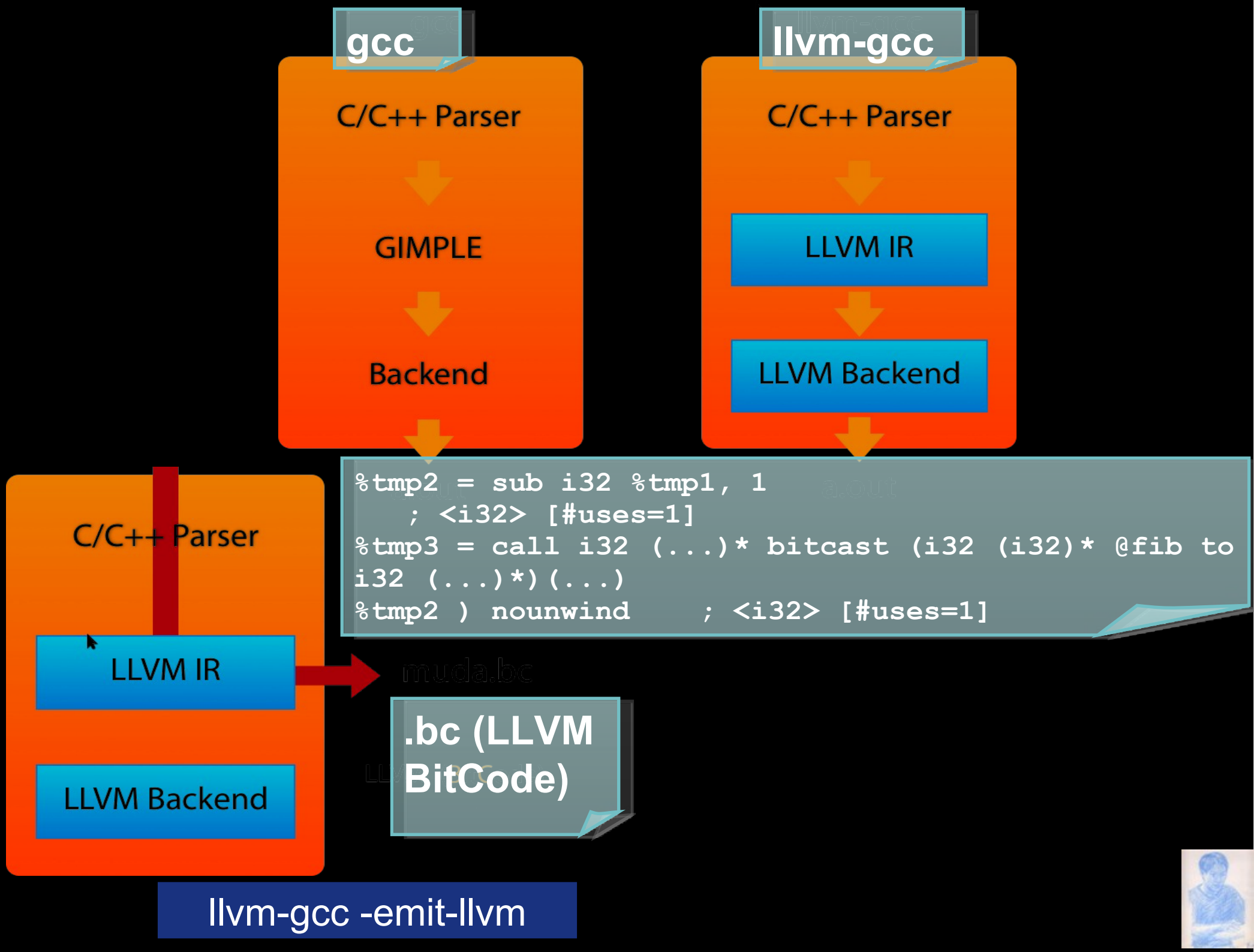



Frontend
LLVM IR
Backend
LLVM IR
一系列變換處理的
API 與銜接的服務
clang
C/C+
llvm-gcc
Java
pypy
Python
LLVM C++API
...
LLVM
x86
Sparc
PPC
...


優化與轉換程序
LLVM IR
Backend
C/C++
Java
Python
constant propation
alias analysis
DCE
User pass
LLVM
x86
Sparc
PPC
...
中間輸出與生成處理
Bitcode writer
Bitcode reader
file
file


Frontend
LLVM IR
Backend
C/C++
Java
Python
...
Codegen,
JIT facility
LLVM
Native
CodeGen
Register
Allocation
Instruction
Scheduling
x86
Sparc
PPC
...


C file
llvmgcc
.o file
C++ file
llvmg++
.o file
C to LLVM Frontend
Compile-time Optimizer
C++ to LLVM Frontend
Compile-time Optimizer
“cc1”
“gccas”
“cc1plus”
“gccas”
LLVM IR Parser
LLVM Verifier
LLVM Analysis & Optimization Passes
LLVM .bc File Writer
Modified version of GCC
Emits LLVM IR as text file
Lowers C AST to LLVM
Modified version of G++
Emits LLVM IR as text file
Lowers C++ AST to LLVM
Dead Global Elimination, IP Constant Propagation, Dead Argument Elimination, Inlining, Reassociation, LICM, Loop Opts, Memory Promotion, Dead Store Elimination, ADCE, …


gcc
C/C++ Parser
GIMPLE
Backend
llvm-gcc
C/C++ Parser
LLVM IR
LLVM Backend
%tmp2 = sub i32 %tmp1, 1
; <i32> [#uses=1]
%tmp3 = call i32 (...)* bitcast (i32 (i32)* @fib to
i32 (...)*)(...)
%tmp2) nounwind ; <i32> [#uses=1]
C/C++ Parser
LLVM IR
LLVM Backend
muda.bc
.bc (LLVM
BitCode)
llvm-gcc -emit-llvm

窮！

變！

http://sakura690606.pixnet.net/blog/post/23225124

「娘，我也想要性感的編譯器」

透過 LLVM 建構虛擬機器

這是一個到處都有虛擬機器 (VM) 的時代

（VM之）道在屎溺

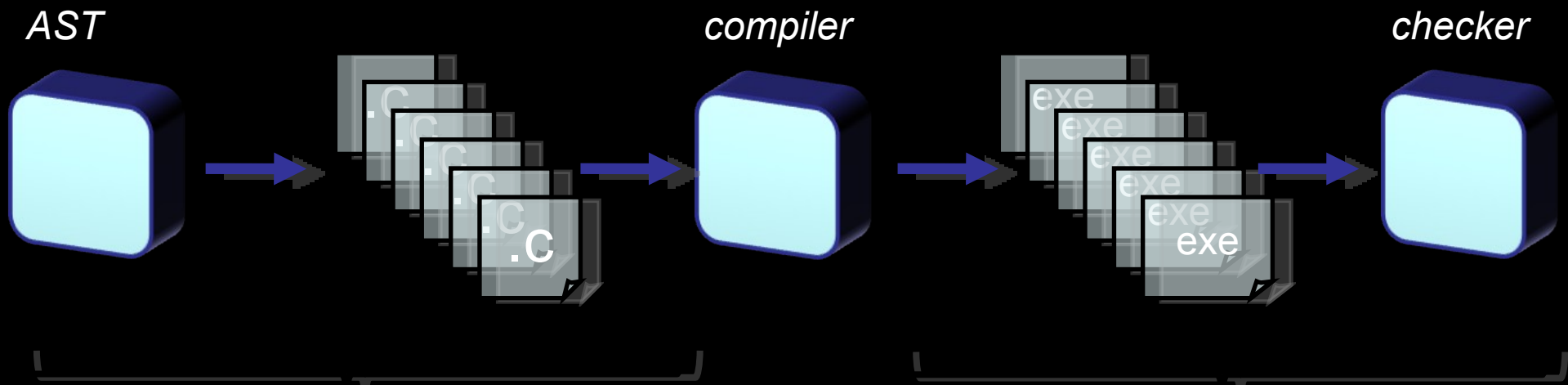
Shader / Raytracing (OpenGL/DirectX)
Web Browser / Adobe Flash
Android Dalvik VM
…

手法：

→**JIT : Just-In-Time** (Jizz In [My Pants]Time?!) **compiler**

→**IR to backend translation / code generation**

編譯器流程::JIT



建立 IR
載入必要的函式庫
連結程式模組
Optimizations + Transforms
codegen

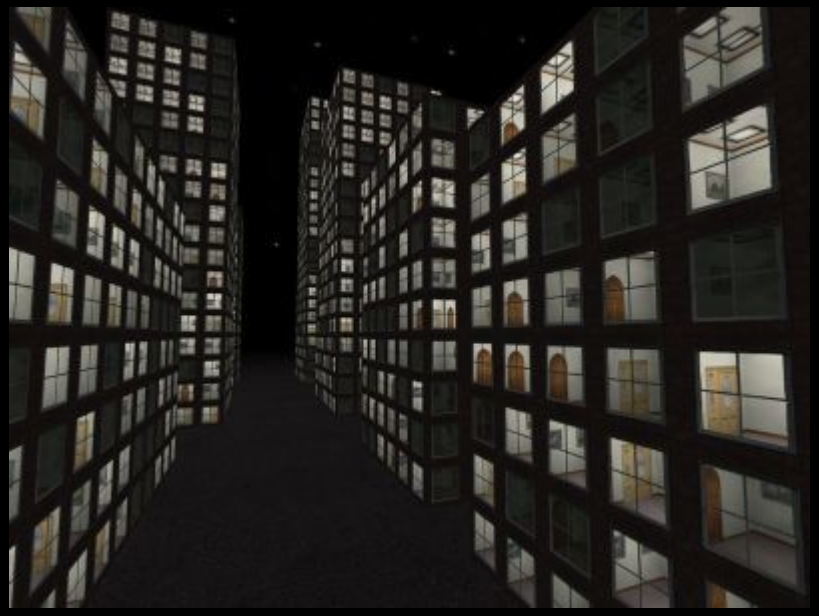
當 LLVM 煞到 OpenGL Shader

動態編譯器就在眼前

Pixel shader

PouetLink: www.pouet.net/prod.php?which=50865

Cocoon website: cocoon.planet-d.net



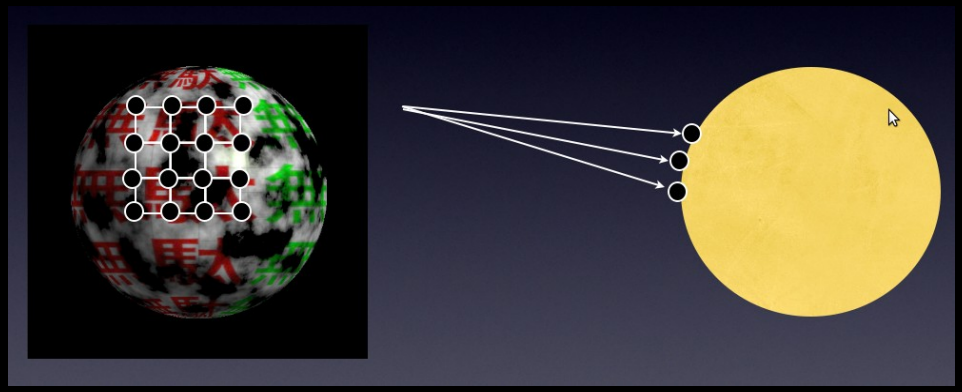
3D/Shader 的考量點

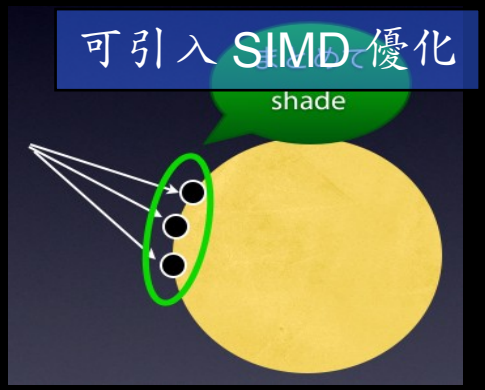
光

反射

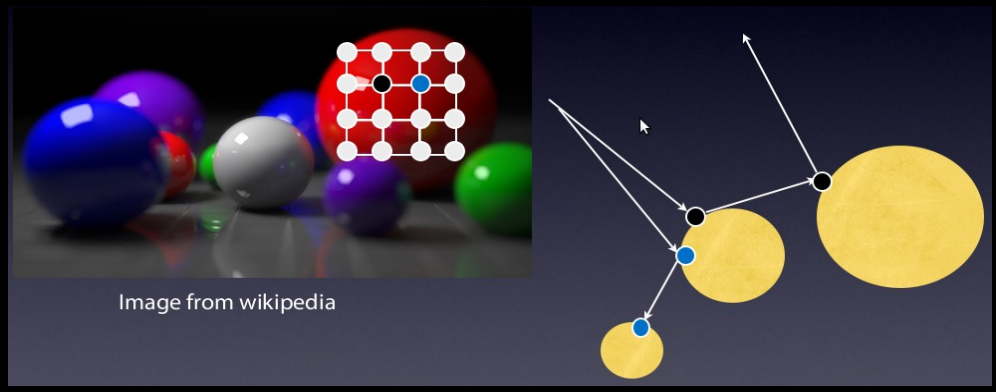
移動的演算法

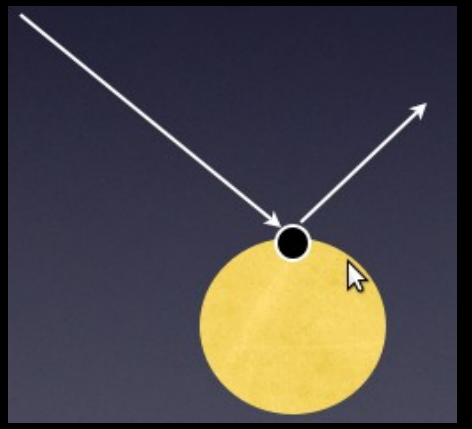
Reyes(scanline,polygon)





Raytracing

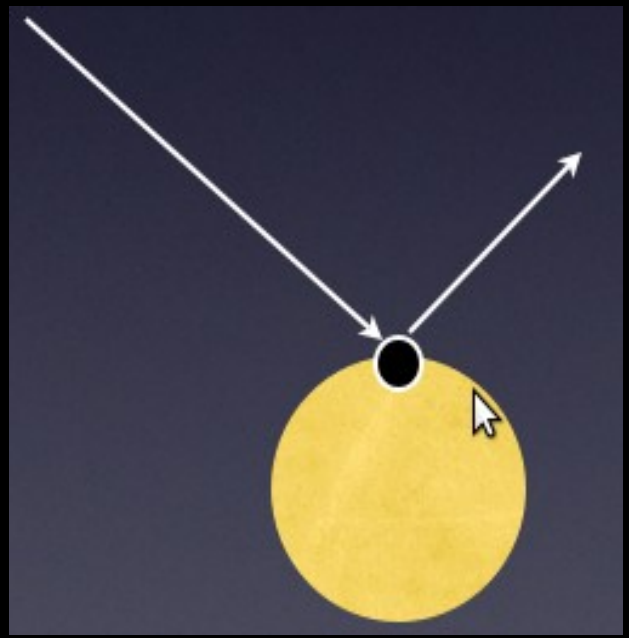




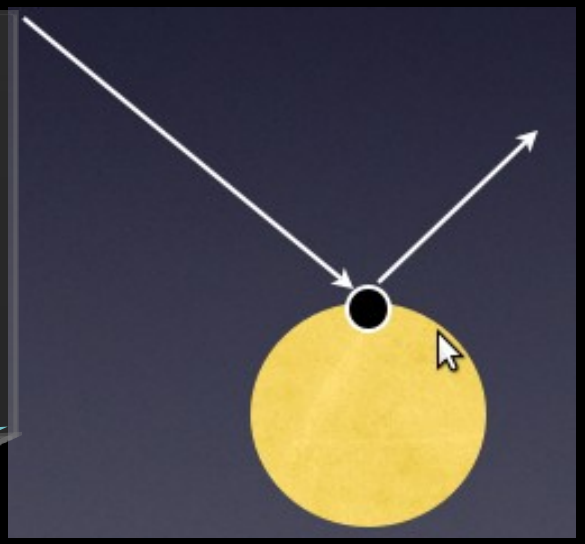
```
surface
matte(float Ka = 1.0; float Kd = 1.1;)
{
  normal Nf = faceforward(
                normalize(N), I);
  Oi = Os;
  Ci = Os * Cs * (Ka * ambient() +
                  Kd * diffuse(Nf));

}
```

push N
normalize
push I
faceforward
...



```
surface
matte(float Ka = 1.0; float Kd = 1.1;)
{
  normal Nf = faceforward(
              normalize(N), I);
  Oi = Os;
  Ci = Os * Cs * (Ka * ambient() +
                  Kd * diffuse(Nf));
}
```



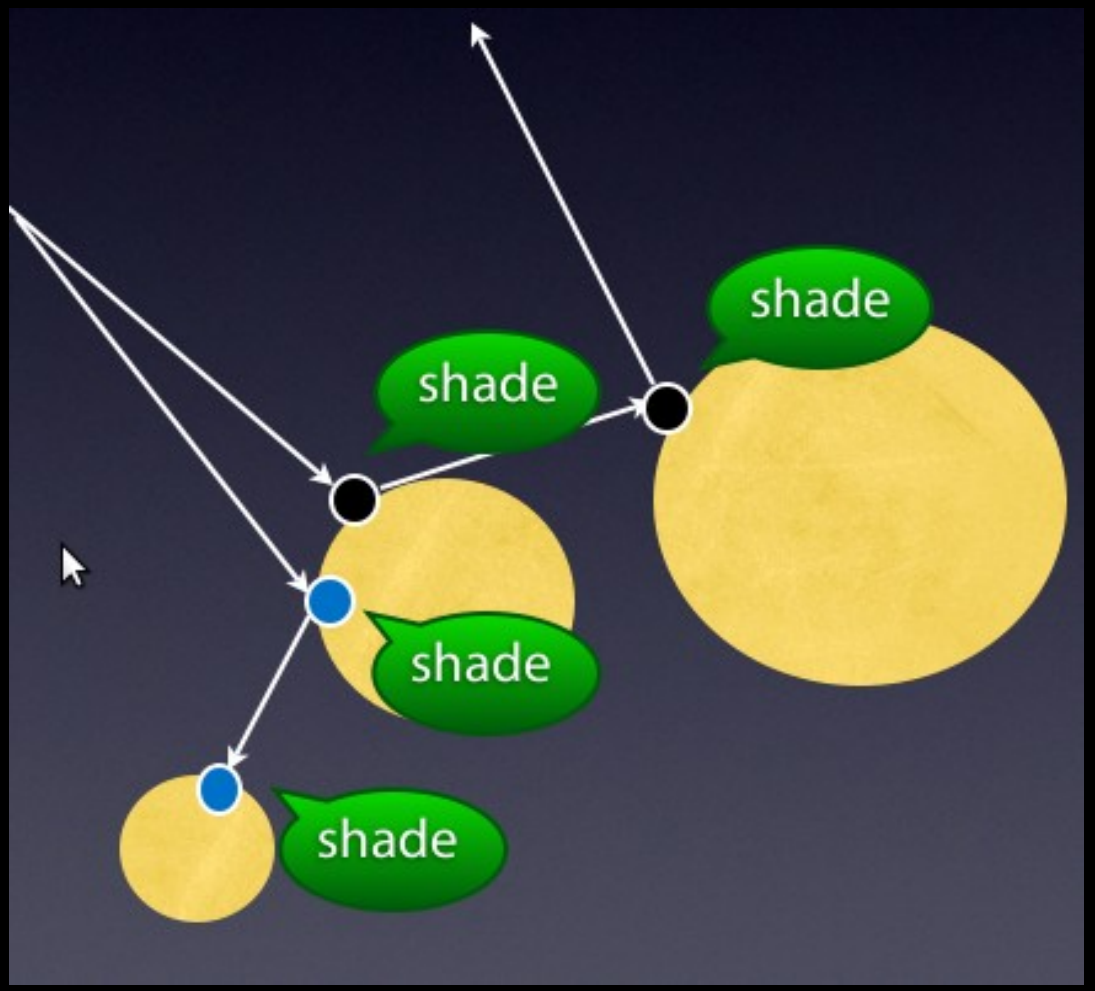
push N
normalize
push I
faceforward
...

Raytracing 的難題

無法善用 SIMD

運算相依性高且繁瑣

需要動態調整快速運算的路徑



Specialize 技巧

以 color space 轉換來說，相當大量且繁瑣的運算，如
BGRA 444R --> RGBA 8888

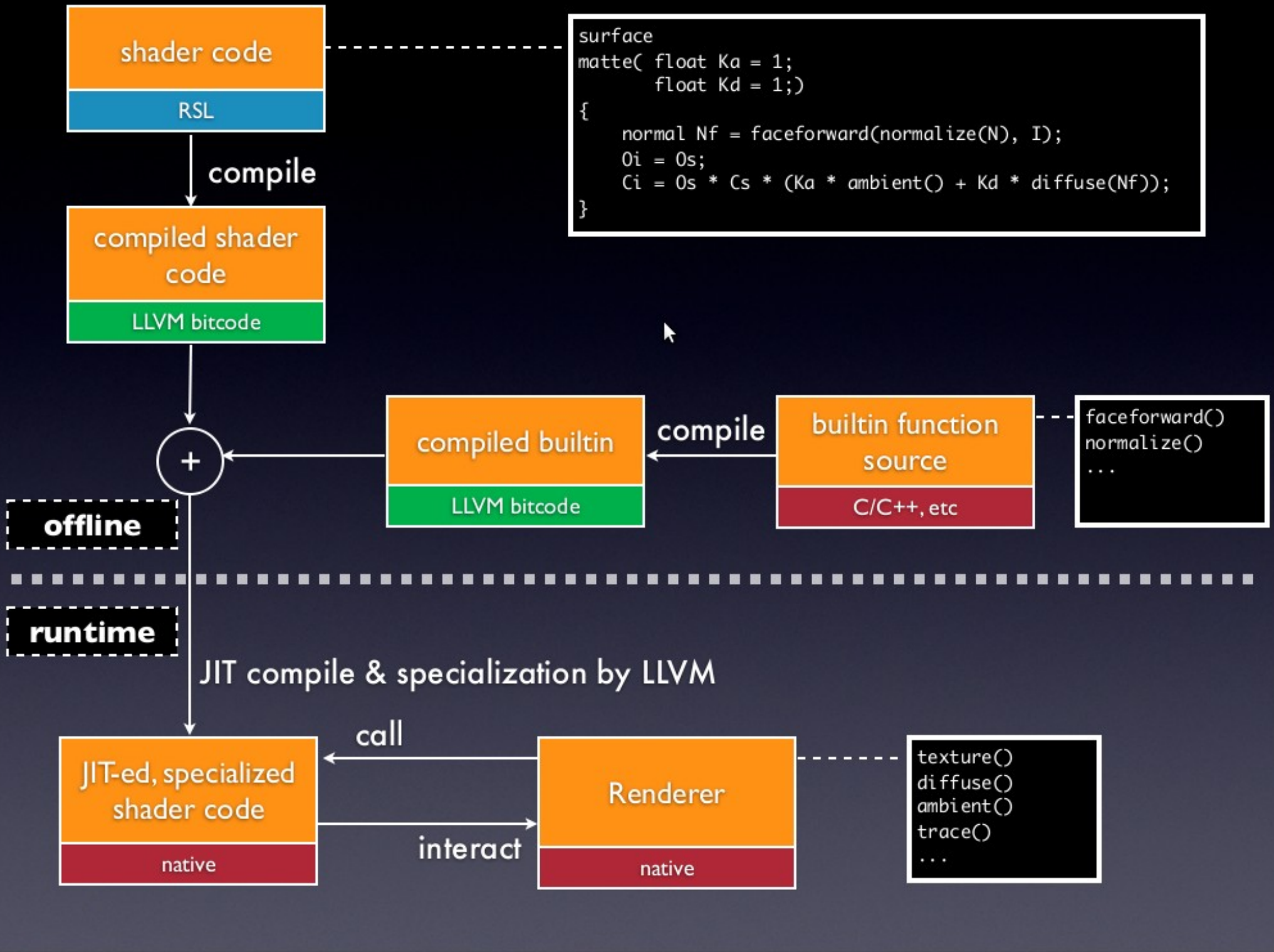
```
for each pixel {
  switch (infmt) {
  case RGBA 5551:
    R = (*in >> 11) & C
    G = (*in >> 6) & C
    B = (*in >> 1) & C
  ... }
  switch (outfmt) {
  case RGB888:
    *outptr = R << 16 |
              G << 8 ...
  }
}
```

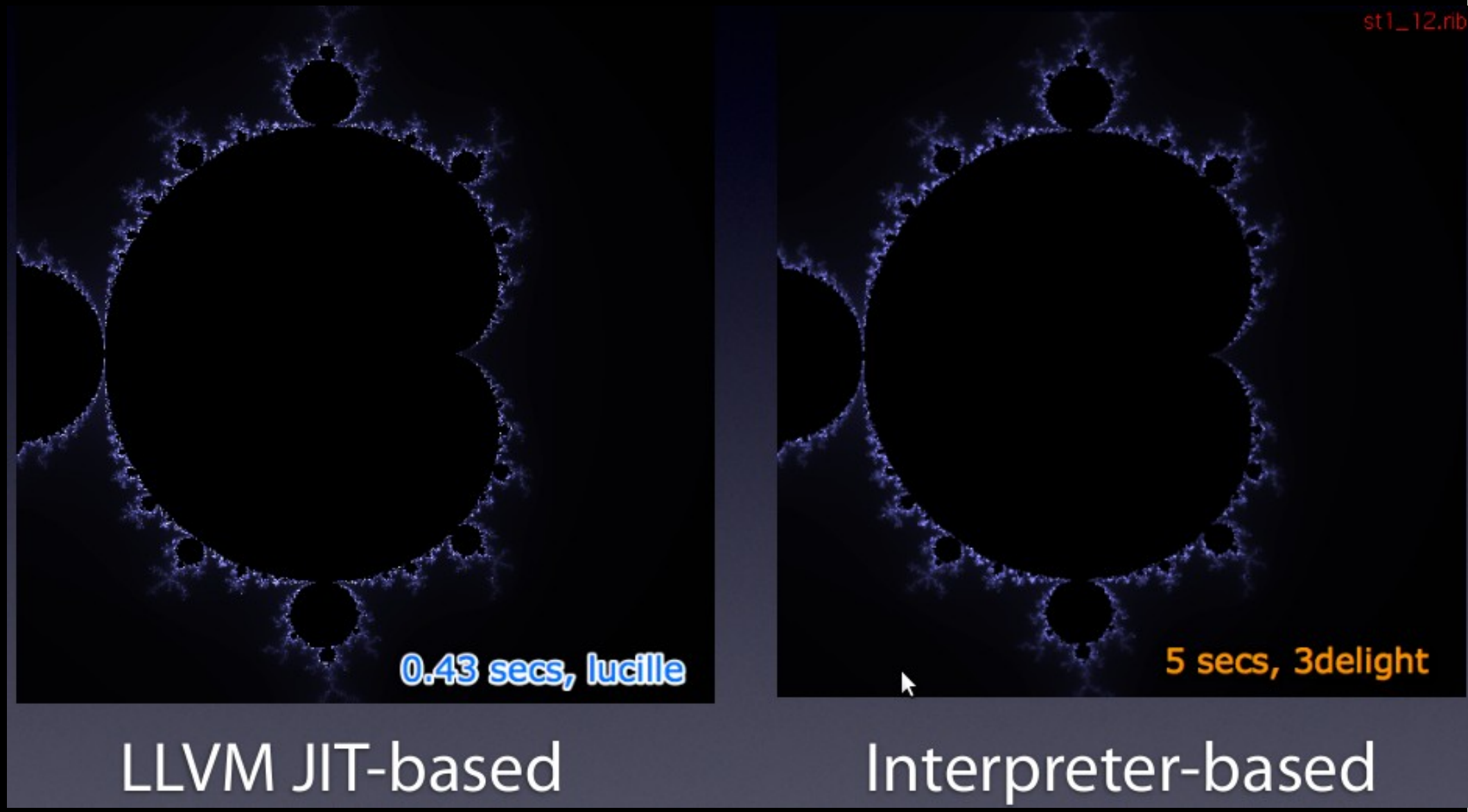
Run-time specialize

```
for each pixel {
  R = (*in >> 11) & C;
  G = (*in >> 6) & C;
  B = (*in >> 1) & C;
  *outptr = R << 16 |
            G << 8 ...
}
```

Compiler optimizes shifts and masking

Speedup depends on src/dest format:
– 5.4x speedup on average, 19.3x max speedup: (13.3MB/s to 257.7MB/s)


shader code
RSL
compile
compiled shader code
LLVM bitcode
surface
matte(float Ka = 1;
float Kd = 1;)
{
normal Nf = faceforward(normalize(N), I);
Oi = Os;
Ci = Os * Cs * (Ka * ambient() + Kd * diffuse(Nf));
}
+
compiled builtin
LLVM bitcode
compile
builtin function source
C/C++, etc
faceforward()
normalize()
...
offline
runtime
JIT compile & specialization by LLVM
call
JIT-ed, specialized shader code
native
interact
Renderer
native
texture()
diffuse()
ambient()
trace()
...



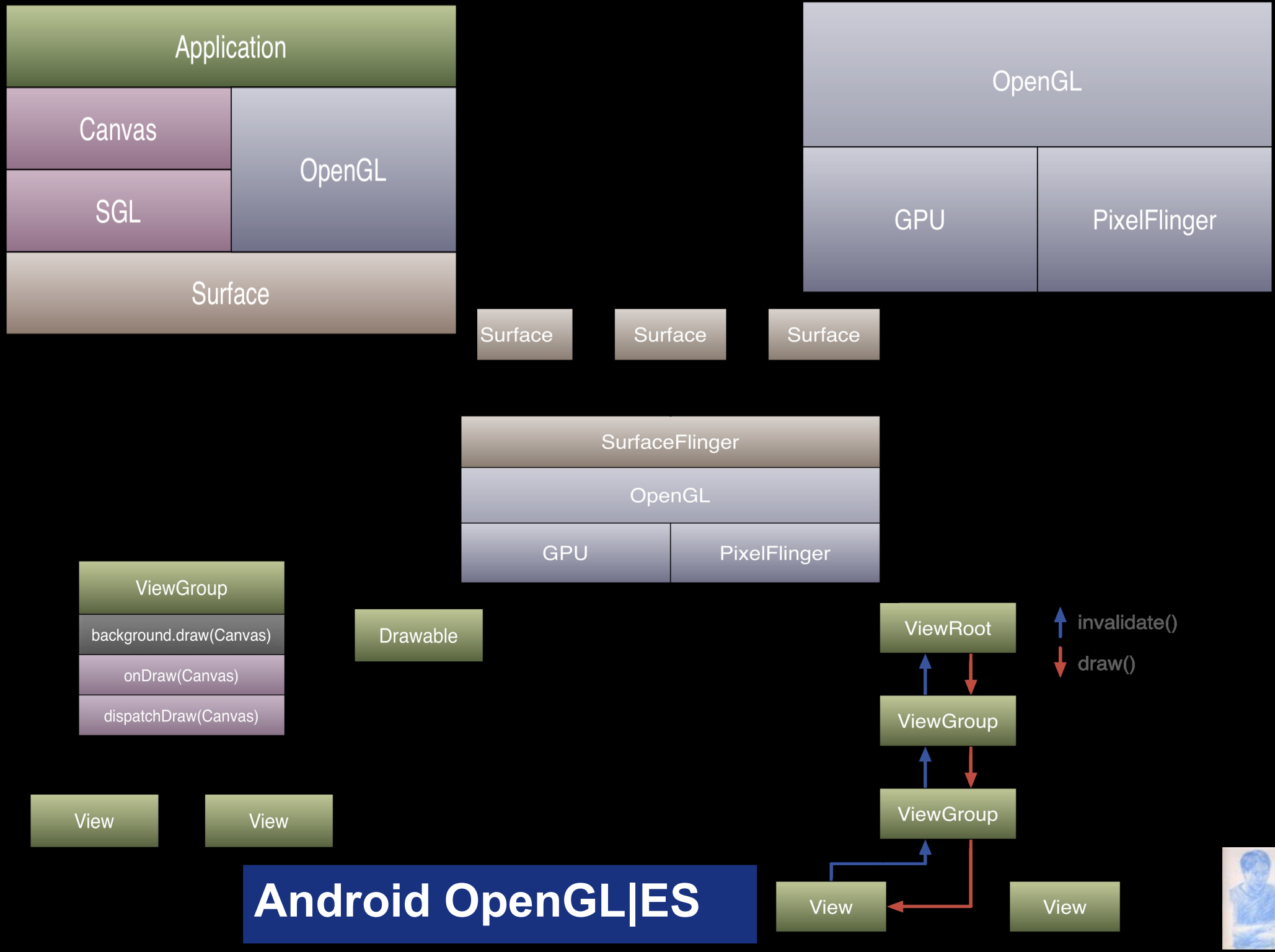
Mandelbort 碎形運算透過 LLVM JIT 後，
提昇效能達到 11 倍

LLVM 與繪圖處理的應用

手機平台： Android PixelFlinger

繪圖軟體： Adobe PixelBender, Shader

3D 高品質圖形處理： Gallium3D / OpenGL


Application
Canvas
SGL
OpenGL
Surface
OpenGL
GPU
PixelFlinger
Surface
Surface
Surface
SurfaceFlinger
OpenGL
GPU
PixelFlinger
ViewGroup
background.draw(Canvas)
onDraw(Canvas)
dispatchDraw(Canvas)
Drawable
ViewRoot
invalidate()
draw()
ViewGroup
ViewGroup
View
View
View
View
Android OpenGL|ES

當 LLVM 煞到 Adobe Flash

安全多元的享受 C/C++ 的美好

有時，你不會想看到遙遙 ...

（不安全、不舒服的 C 語言）

COM11 - Tera Term VT

File Edit Setup Control Window Help

```
  Object 0xc6e78ac0:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78ad0:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78ae0:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78af0:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78b00:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78b10:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78b20:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78b30:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78b40:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78b50:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78b60:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78b70:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78b80:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78b90:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78ba0:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78bb0:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b kkkkkkkkkkkkkkkk
  Object 0xc6e78bc0:  6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b 6b a5 kkkkkkkkkkkkkkk?
 Redzone 0xc6e78bd0:  bb bb bb bb                                       遙遙
 Padding 0xc6e78bf8:  5a 5a 5a 5a 5a 5a 5a 5a                           ZZZZZZZZ
Call Trace:
[<c008039e>] print_trailer+0xfe/0x110
[<c00805c6>] check_bytes_and_report+0x76/0x98
[<c008077c>] check_object+0xb0/0x198
[<c008140c>] __slab_alloc+0x484/0x50c
[<c00814cc>] kmem_cache_alloc+0x38/0x84
[<c003b918>] copy_process+0x74/0xb98
[<c003c526>] do_fork+0xea/0x220
[<c0025f0e>] sys_clone+0x32/0x44
[<c00225fa>] ret_fast_syscall+0x0/0xa8

FIX task_struct: Restoring 0xc6e78930-0xc6e78a33=0x6b

FIX task_struct: Marking all objects used
float_bessel    0  INFO  :  Initial thread: Waiting for 100 threads to finish
```

跟遙遙一起研究 Linux Kernel吧

「可是，男人都想要當慣 C」

承認吧，慣 C 有很多好處，所以 Adobe 建立 Alchemy(FlaCC) 專案

當然啦，慣 C 的風險也很高

「學長，我不敢修電腦」

虛擬機器：AVM2

Adobe Alchemy

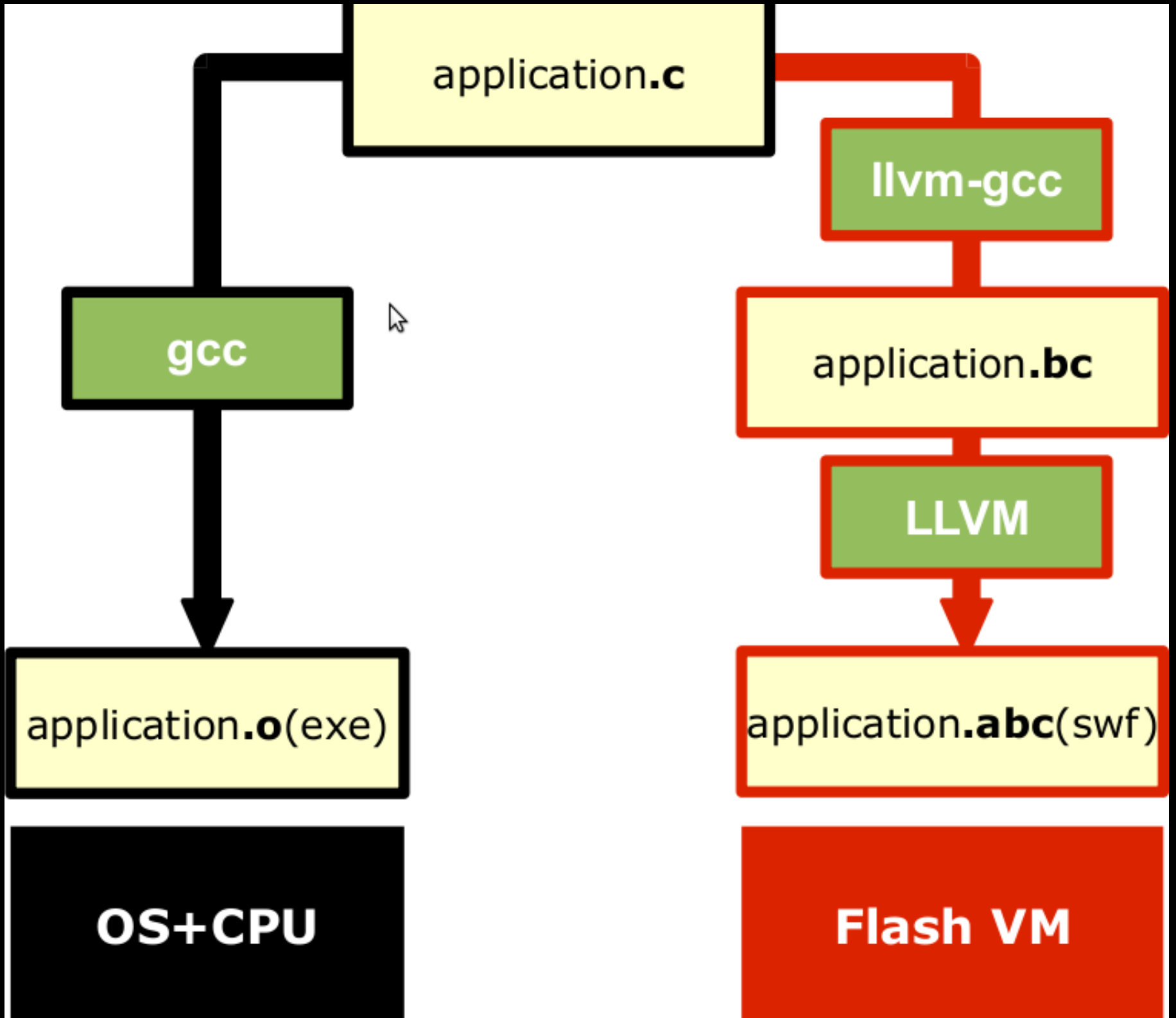
實驗性的 LLVM Backend

LLVM → ABC (ActionScript Bytecode)

ABC 類似精簡的 x86 指令集

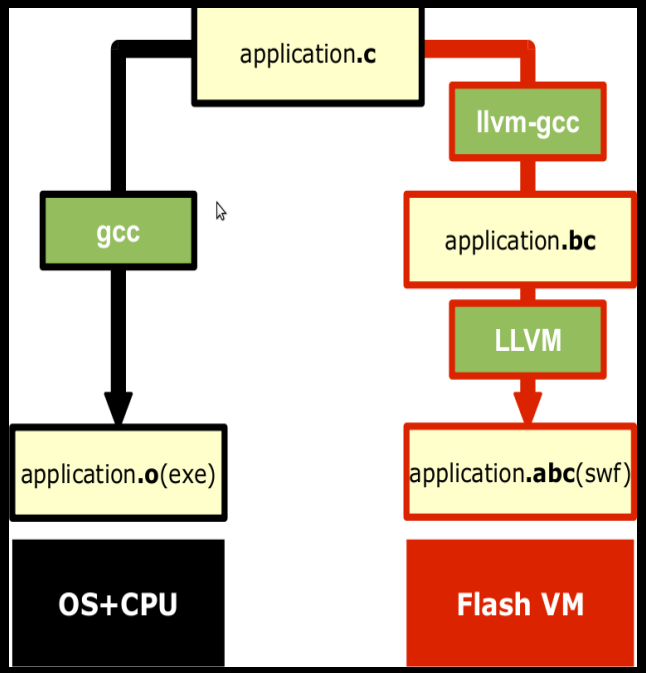
提供 POSIX 模擬層與完整的 BSD libc + GNU ISO C++ library

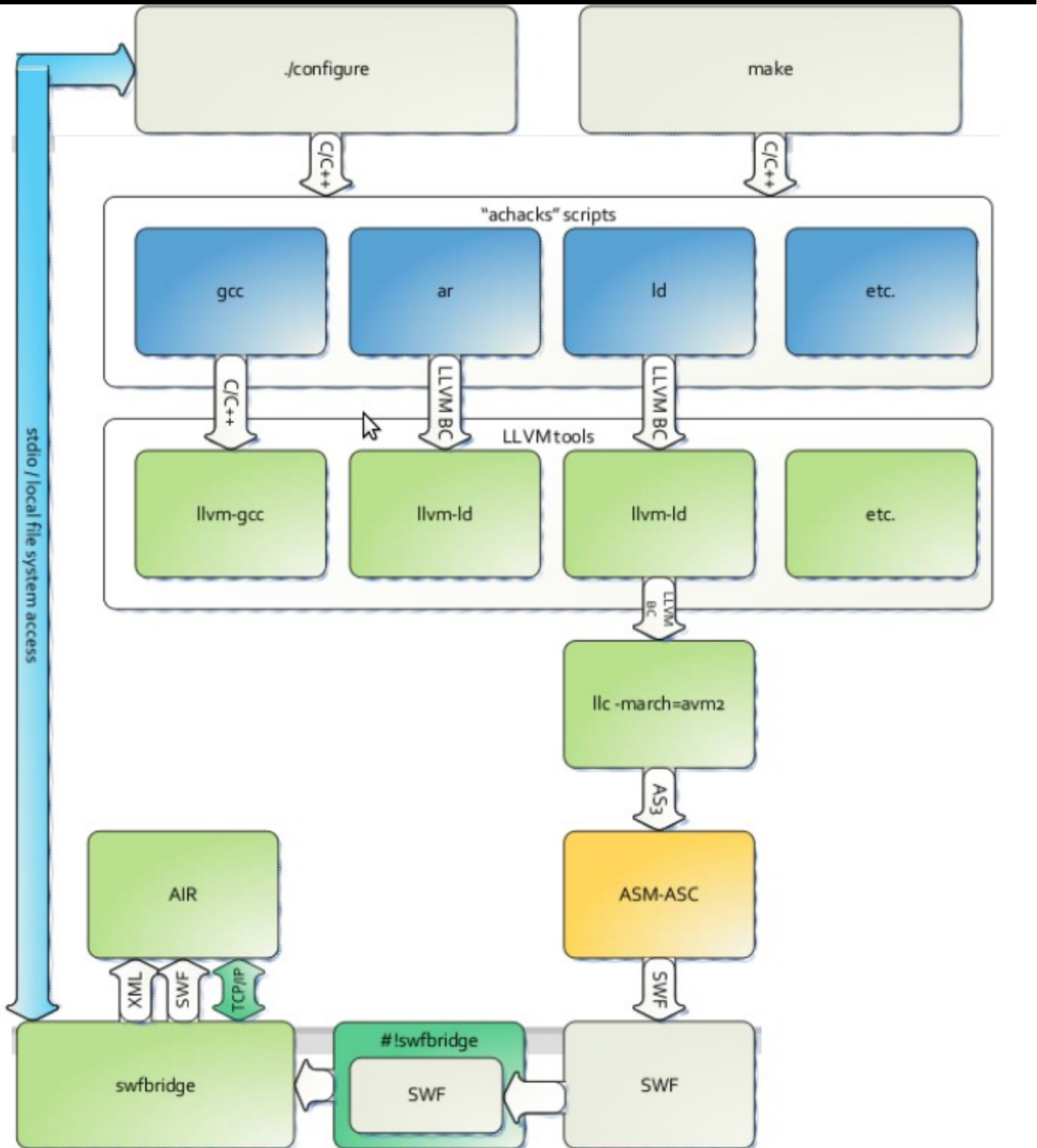
足以執行若干重要的應用程式


application.c
gcc
application.o(exe)
OS+CPU
llvm-gcc
application.bc
LLVM
application.abc(swf)
Flash VM

Example generated AS3

```
    __asm(lbl("___vfprintf_state0"))
    __asm(lbl("___vfprintf__XprivateX__BB79_0_F"))
        mstate.esp -= 4; __asm(push(mstate.ebp), push(mstate.esp),
op(0x3c), stack(-2))
        mstate.ebp = mstate.esp
        mstate.esp -= 2640
        i0 = (0)
        i1 = ((__xasm<int>(push((mstate.ebp+16)), op(0x37))))
        __asm(push(i1), push((mstate.ebp+-84)), op(0x3c), stack(-2))
        __asm(push(i0), push((mstate.ebp+-86)), op(0x3a), stack(-2))
        i0 = ((__xasm<int>(push((mstate.ebp+8)), op(0x37))))
        i1 = ((__xasm<int>(push((mstate.ebp+12)), op(0x37))))
        __asm(push(i1), push((mstate.ebp+-2295)), op(0x3c), stack(-2))
        i1 = ((__xasm<int>(push(___mlocale_changed_2E_b), op(0x35))))
        i2 = ((mstate.ebp+-1504))
        i3 = ((mstate.ebp+-1808))
        __asm(push(i3), push((mstate.ebp+-2259)), op(0x3c), stack(-2))
        i3 = ((mstate.ebp+-1664))
        __asm(push(i3), push((mstate.ebp+-2097)), op(0x3c), stack(-2))
        i3 = ((mstate.ebp+-304))
        __asm(push(i3), push((mstate.ebp+-2115)), op(0x3c), stack(-2))
        i3 = ((mstate.ebp+-104))
        __asm(push(i3), push((mstate.ebp+-2277)), op(0x3c), stack(-2))
        __asm(push(i1!=0), iftrue,
target("___vfprintf__XprivateX__BB79_2_F"))
    __asm(lbl("___vfprintf__XprivateX__BB79_1_F"))
        i1 = (1)
        __asm(push(i1), push(___mlocale_changed_2E_b), op(0x3a),
stack(-2))
    __asm(lbl("___vfprintf__XprivateX__BB79_2_F"))
        i1 = ((__xasm<int>(push(___nlocale_changed_2E_b), op(0x35))))
        __asm(push(i1!=0), iftrue,
target("___vfprintf__XprivateX__BB79_4_F"))
    __asm(lbl("___vfprintf__XprivateX__BB79_3_F"))
        i1 = (1)
        __asm(push(i1), push(_ret_2E_993_2E_0_2E_b), op(0x3a), stack(-
2))
        __asm(push(i1), push(_ret_2E_993_2E_2_2E_b), op(0x3a), stack(-
2))
        __asm(push(i1), push(___nlocale_changed_2E_b), op(0x3a),
stack(-2))
    __asm(lbl("___vfprintf__XprivateX__BB79_4_F"))
        i1 = (__2E_str1881)
        i3 = ((__xasm<int>(push(_ret_2E_993_2E_0_2E_b), op(0x35))))
        i4 = ((__xasm<int>(push((i0+12)), op(0x36))))
        i1 = ((i3!=0) ? i1 : 0)
        __asm(push(i1), push((mstate.ebp+-2124)), op(0x3c), stack(-2))
        i1 = (i0 + 12)
        __asm(push(i1), push((mstate.ebp+-2025)), op(0x3c), stack(-2))
        i1 = (i4 & 8)
        __asm(push(i1==0), iftrue,
target("___vfprintf__XprivateX__BB79_7_F"))
    __asm(lbl("___vfprintf__XprivateX__BB79_5_F"))
        i1 = ((__xasm<int>(push((i0+16)), op(0x37))))
        __asm(push(i1!=0), iftrue,
target("___vfprintf__XprivateX__BB79_9_F"))
    __asm(lbl("___vfprintf__XprivateX__BB79_6_F"))
        i1 = (i4 & 512)
        __asm(push(i1!=0), iftrue,
target("___vfprintf__XprivateX__BB79_9_F"))
    __asm(lbl("___vfprintf__XprivateX__BB79_7_F"))
        mstate.esp -= 4
```


application.c
llvm-gcc
gcc
application.bc
LLVM
application.o(exe)
application.abc(swf)
OS+CPU
Flash VM


./configure
make
C/C++
C/C++
"achacks" scripts
gcc
ar
ld
etc.
C/C++
LLVM BC
LLVM BC
LLVM tools
llvm-gcc
llvm-ld
llvm-ld
etc.
LLVM BC
llc -march=avm2
AS3
AIR
ASM-ASC
XML
SWF
TCP/IP
SWF
stdio / local file system access
swfbridge
#!swfbridge
SWF
SWF

當 LLVM 煞到 Android

安全多元的享受 C/C++ 的美好

0xlab LLVM hacks

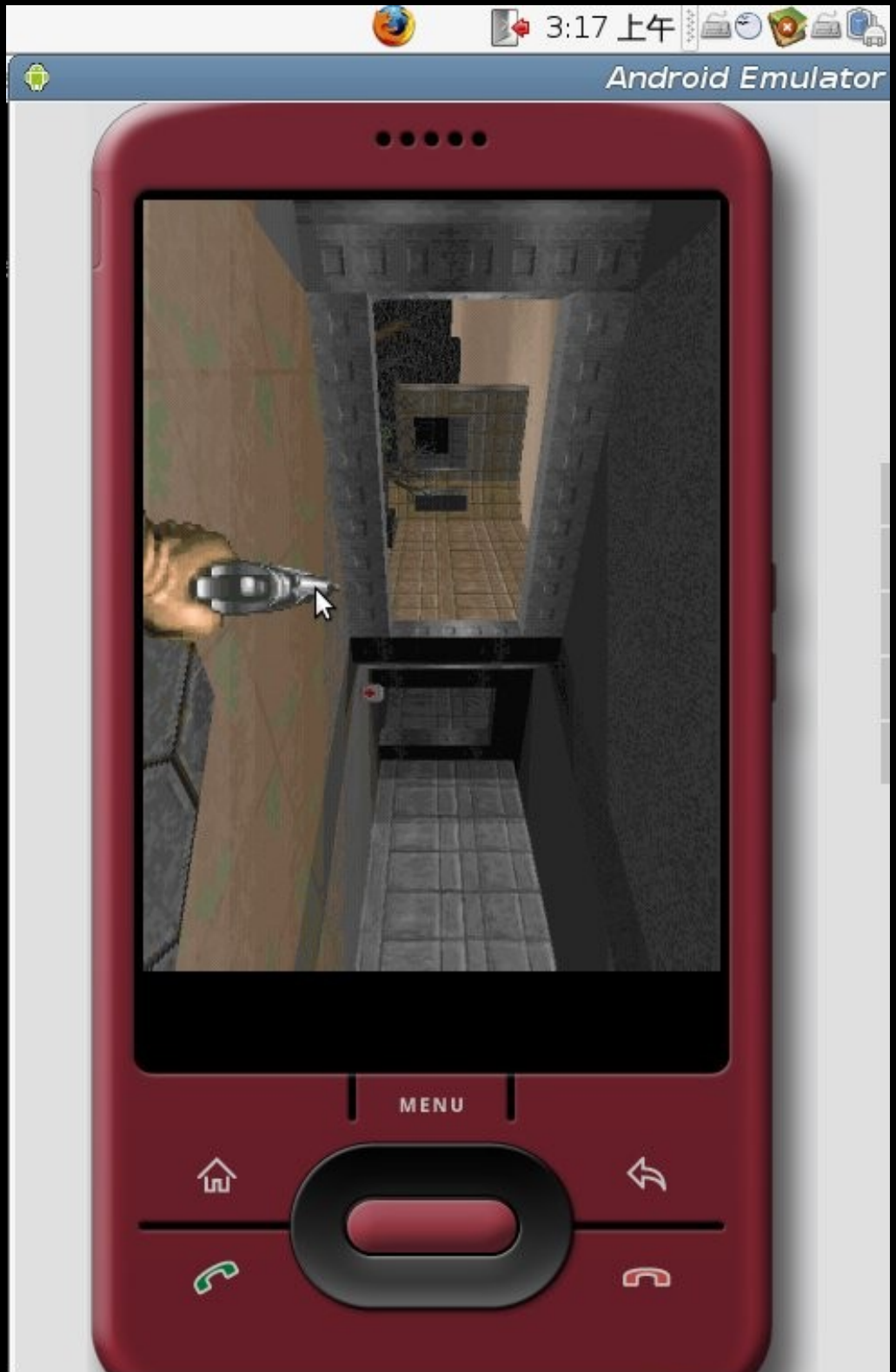
實驗性的 LLVM Backend

LLVM → Android Dalvik bytecode (DEX)

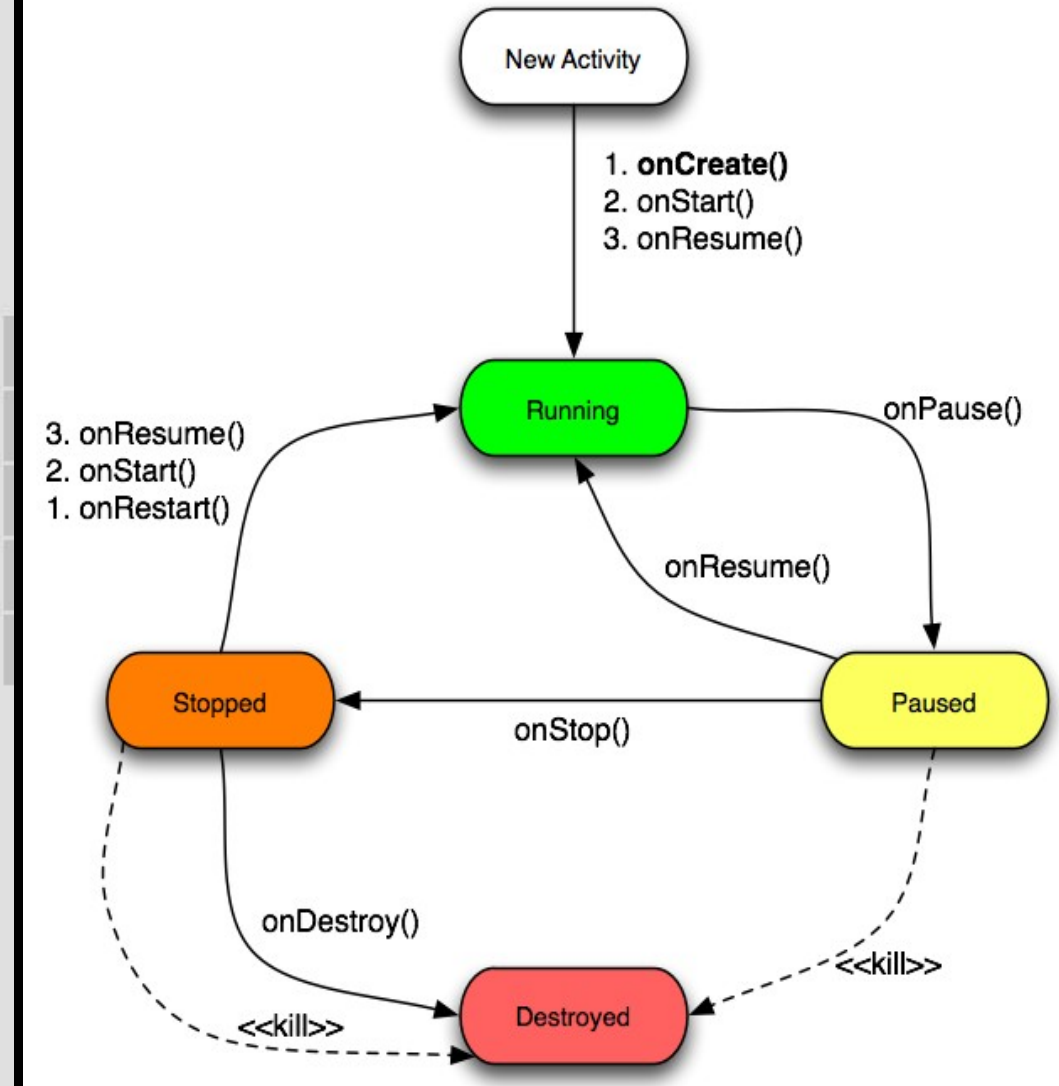
仿效 Adobe Alchemy ，目標平台則是 DalvikVM

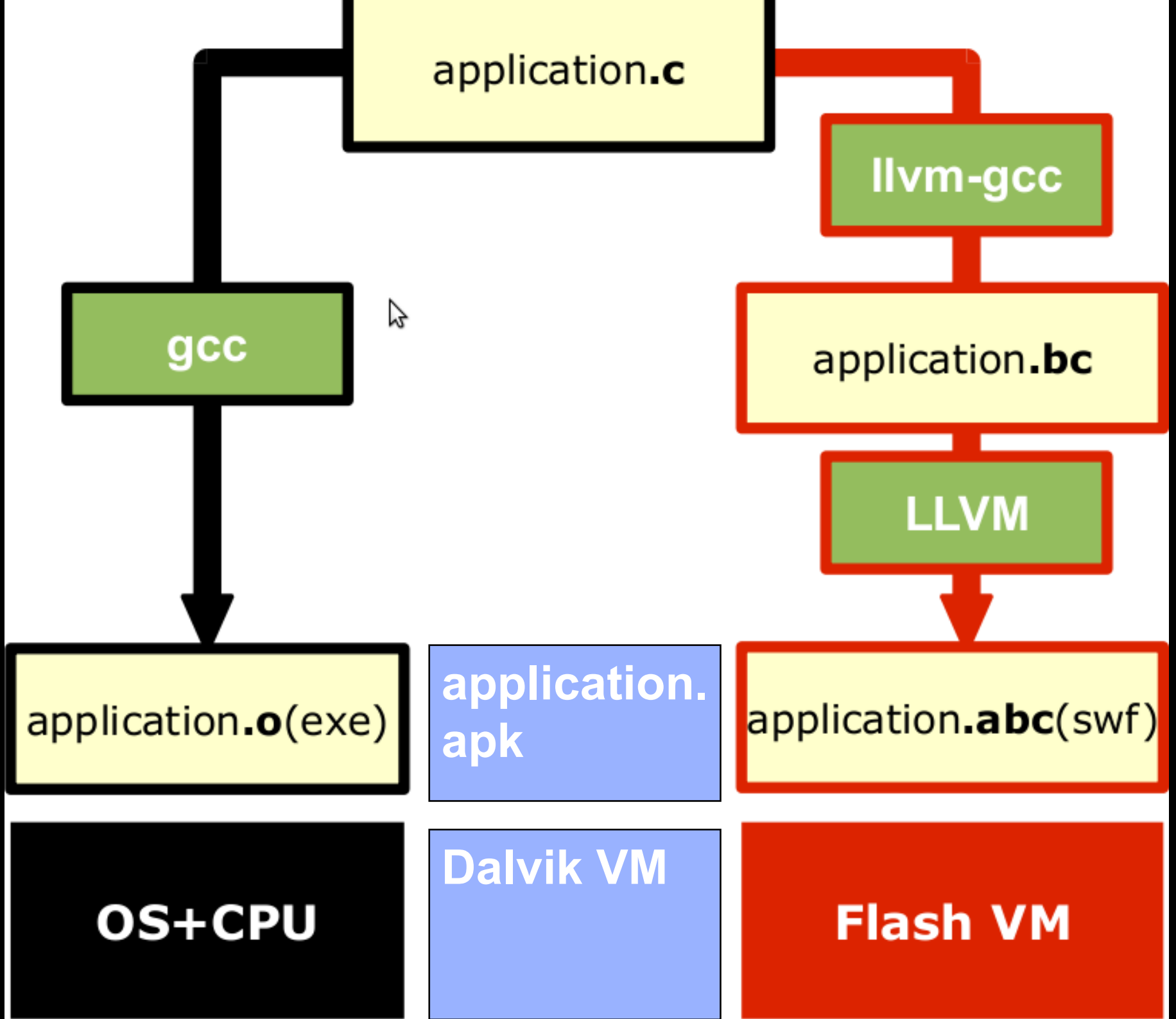
提供 C/C++ 原始碼 → LLVM → DEX bytecode 的執行環境，無 JNI (Java Native Interface) 介入

「我只會寫 Java VM ，不會寫 Java 應用程式」，宅色夫 (2005)



Activity Lifecycle




application.c
gcc
llvm-gcc
application.bc
LLVM
application.o(exe)
application.
apk
application.abc(swf)
OS+CPU
Dalvik VM
Flash VM

技術大融通

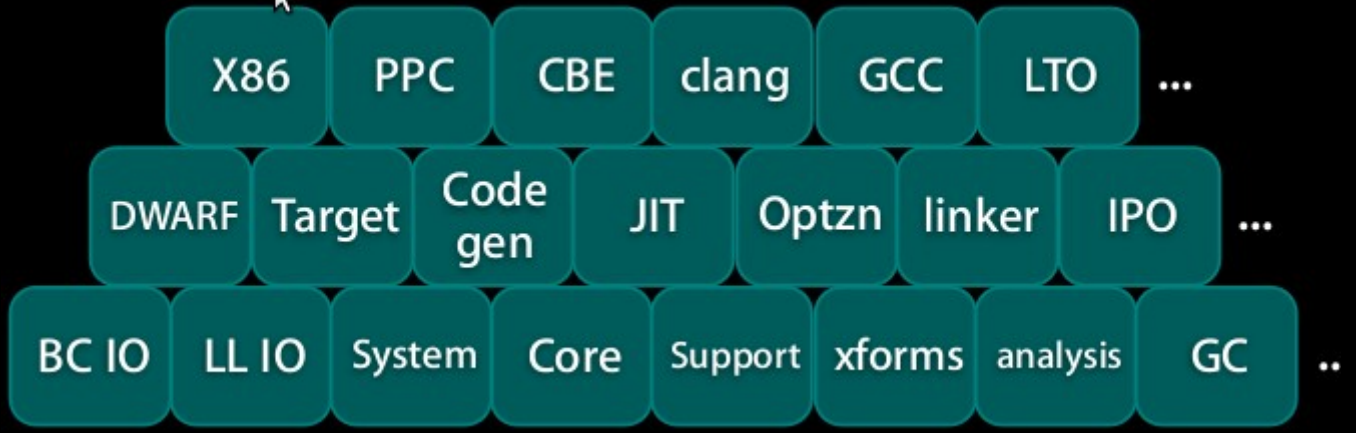
與

未來展望

「你的時代到了！」LLVM 一統天下

LLVM 的時代到了！

clang: 嶄新的 C/C++/Objective-C 語言前端
vmkit: Java/.Net 虛擬機器
llvm + OpenGL
Sun OpenJDK → RedHat Zero/Shark
Trident: VHDL compiler for FPGA
llvmruby, yarv2llvm, RubyComp, MacRuby
PyPy, Google unladen-Swallow
HLVM: Haskell, Ocaml, ...



自由軟體編譯器技術的一統標準已定！

多元的整合，如 clang 可作爲 static compiler，亦可挪用爲 JIT compiler

技術集中，創意多元

「人生兩條路：『生活的幸福』不等於『生命的幸福』，兩樣幸福我們都需要」

王陽明牧師

參考資料

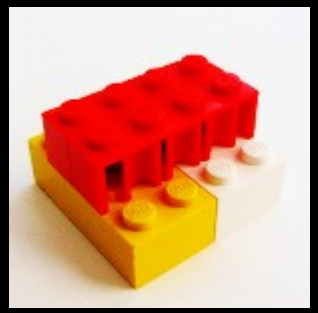
〈*LLVM and Clang:*
Next Generation Compiler Technology〉, *Chris Lattner*
BSDCan 2008

《美麗程式》*(Beautiful Code), O'Reilly*
第八章〈動態產生影像處理程式〉

LLVM – http://www.llvm.org/
Projects built with LLVM – http://www.llvm.org/ProjectsWithLLVM/
Open LLVM Projects – http://www.llvm.org/OpenProjects.html

connect your device to application

UI customizing



Platform Builder

Device Potential

三個願望一次滿足
一系列的自由軟體
四月 27 日開張
(427_{10}=0x1ab)

感謝您！

8xlab