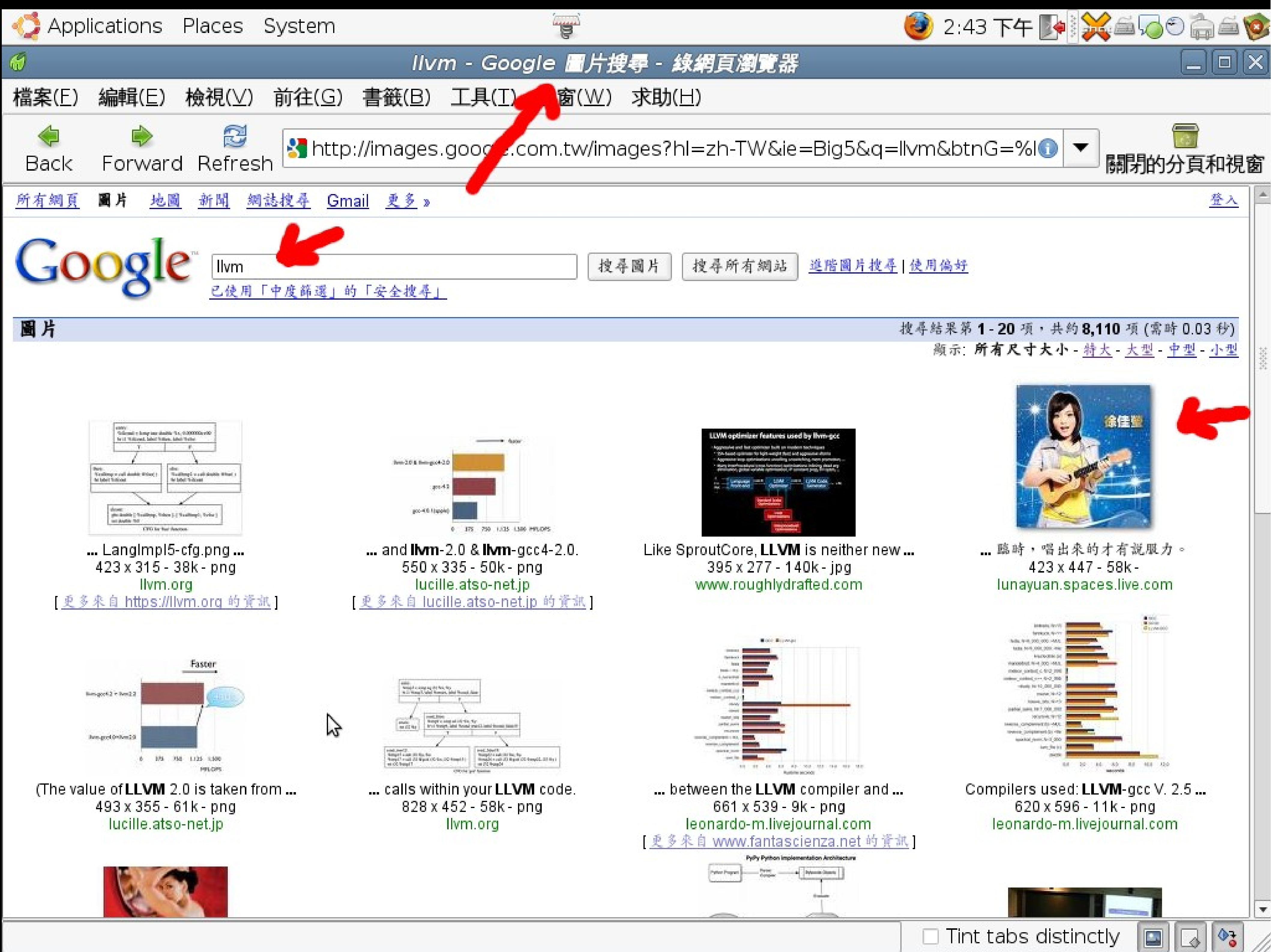
身騎 LLVM ，過三關

淺談編譯器技術的嶄新應用

Jim Huang (黃敬群) “jserv”
website: http://jserv.sayya.org/
blog: http://blog.linux.org.tw/jserv/
May 5, 2009 @ TOSSUG

姊妹議程：
〈窮得只剩下 Compiler〉
OOSDC.tw 2009

http://www.slideshare.net/jserv/what-can-compilers-do-for-us



Applications Places System
2:43 下午
llvm - Google 圖片搜尋 - 綠網頁瀏覽器
檔案(F) 編輯(E) 檢視(V) 前往(G) 書籤(B) 工具(T) 視窗(W) 求助(H)
Back Forward Refresh
關閉的分頁和視窗
Tint tabs distinctly

http://images.google.com.tw/images?hl=zh-TW&ie=Big5&q=llvm&btnG=%I

所有網頁 圖片 地圖 新聞 網誌搜尋 Gmail 更多 » 登入

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

搜尋結果第 **1** - **20** 項，共約 **8,110** 項 (需時 0.03 秒)

顯示: **所有尺寸大小** - 特大 - 大型 - 中型 - 小型

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

... 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]

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

... 臨時，唱出來的才有說服力。
423 x 447 - 58k -
lunayuan.spaces.live.com

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

... calls within your **LLVM** code.
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 optimizer features used by llvm-gcc
Faster
PyPy Python Implementation Architecture
徐佳瑩

公孫龍子：「白馬非馬」
「馬」 指的是馬的形態，「白馬」指的是馬的顏色，
而形態不等於顏色，所以白馬不是馬

LLVM 不是 VM ，也不只是 Compiler

http://p5.p.pixnet.net/albums/userpics/5/8/270458/1208944435.jpg

編譯器技術領導的時代

- 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

- Compiler 大不同：走向 Binary translation, virtualization, JIT/VM 等嶄新的應用模式
- LLVM 簡介
  - 虛擬指令集 /IR
  - 高度整合的編譯器技術元件 (analyzer, code emitter/generator, JIT compiler, GC)
  - 完整的工具集合 (assembler, debugger, linker)
- 技術展示 (from 0xlab)
- 淺談開發趨勢

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

多才多藝的 LLVM

「快快樂樂寫 Compiler」

遜！

「快快樂樂玩 Compiler」

讓 LLVM 拉你一把

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 Core)

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, …

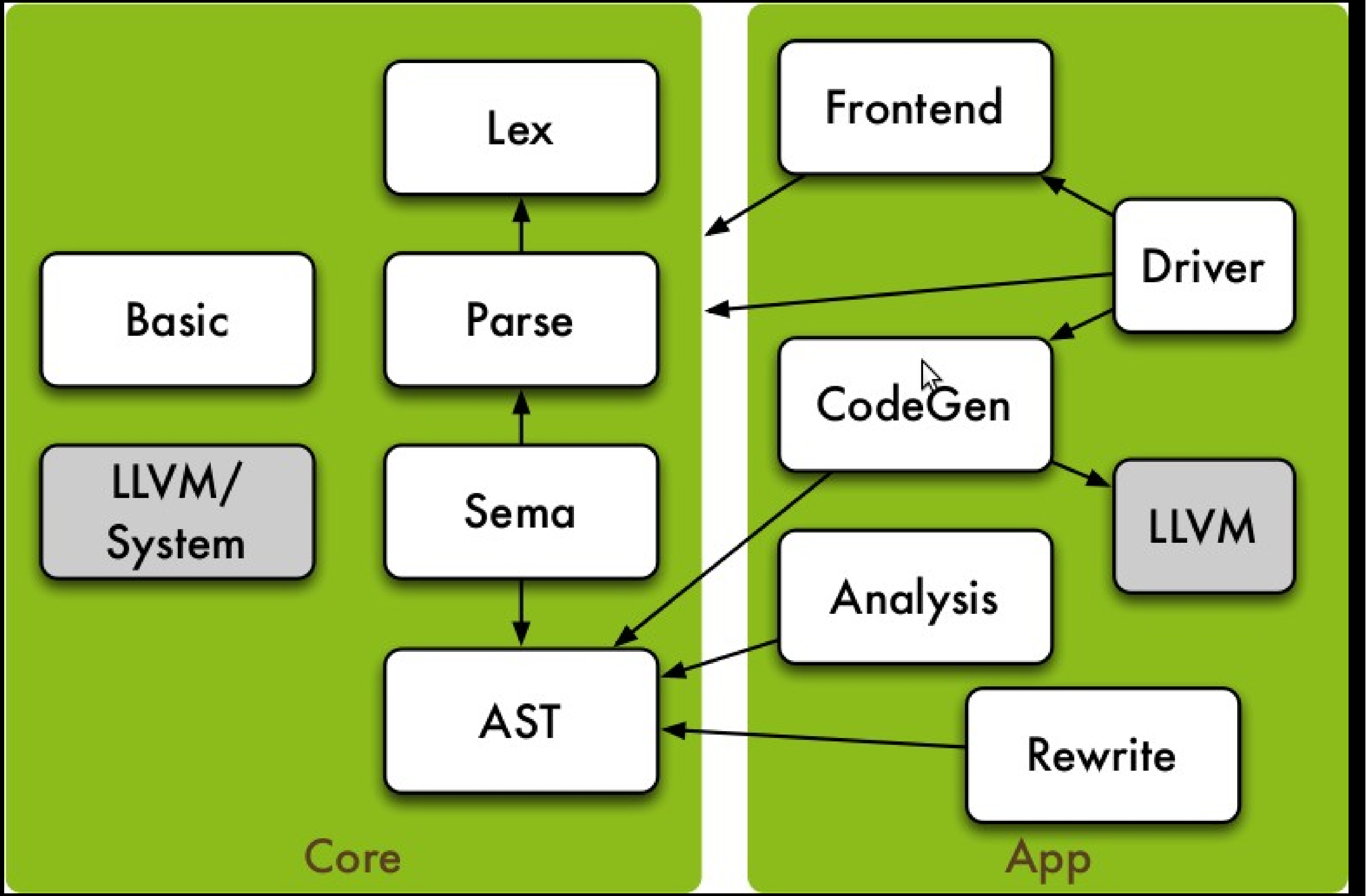
豐富的工具集

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

RISC 式虛擬指令集 (instruction set)

完整的高度整合函式庫與編譯器服務

豐富的工具集



LLVM 的 Low-Level VM
IR 是 Compiler 的心臟

RISC 式虛擬指令集 (instruction set)

- 約 50 個 opcedes
- 豐富的 Intrinsic Function:llvm.*

SSA (Static Single Assignment Form)

- 作爲程式語言優化的基礎

允許整體的優化

- Offline code generation and optimization
  - Install-time target-specific optimization
  - Link-Time interprocedural Optimization (LTO)
    - whole program analysis
- User-based profiling and optimization
  - run-time & idle time

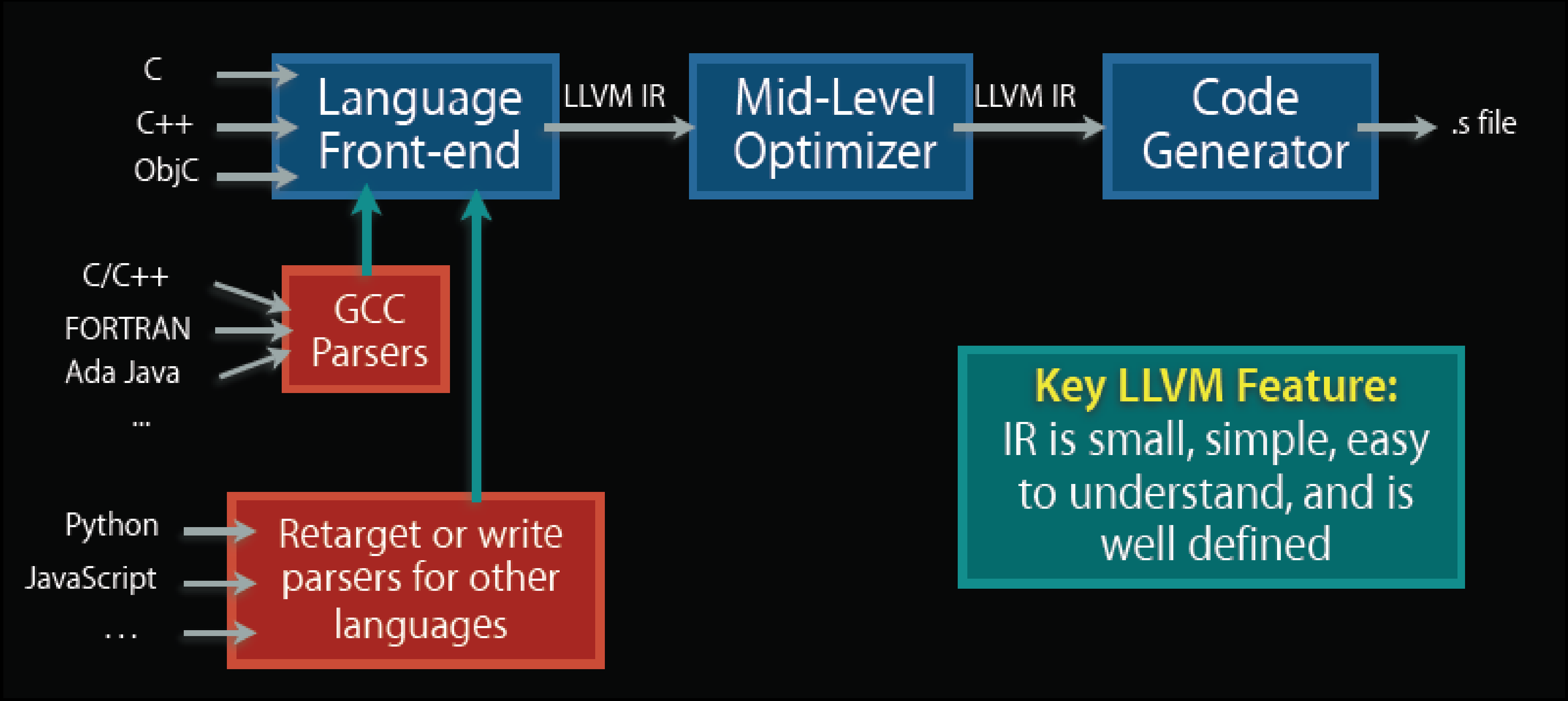
LLVM 的 Low-Level VM

IR 是 Compiler 的心臟

RISC 式虛擬指令集 (instruction set)

SSA (Static Single Assignment Form)

允許整體的優化

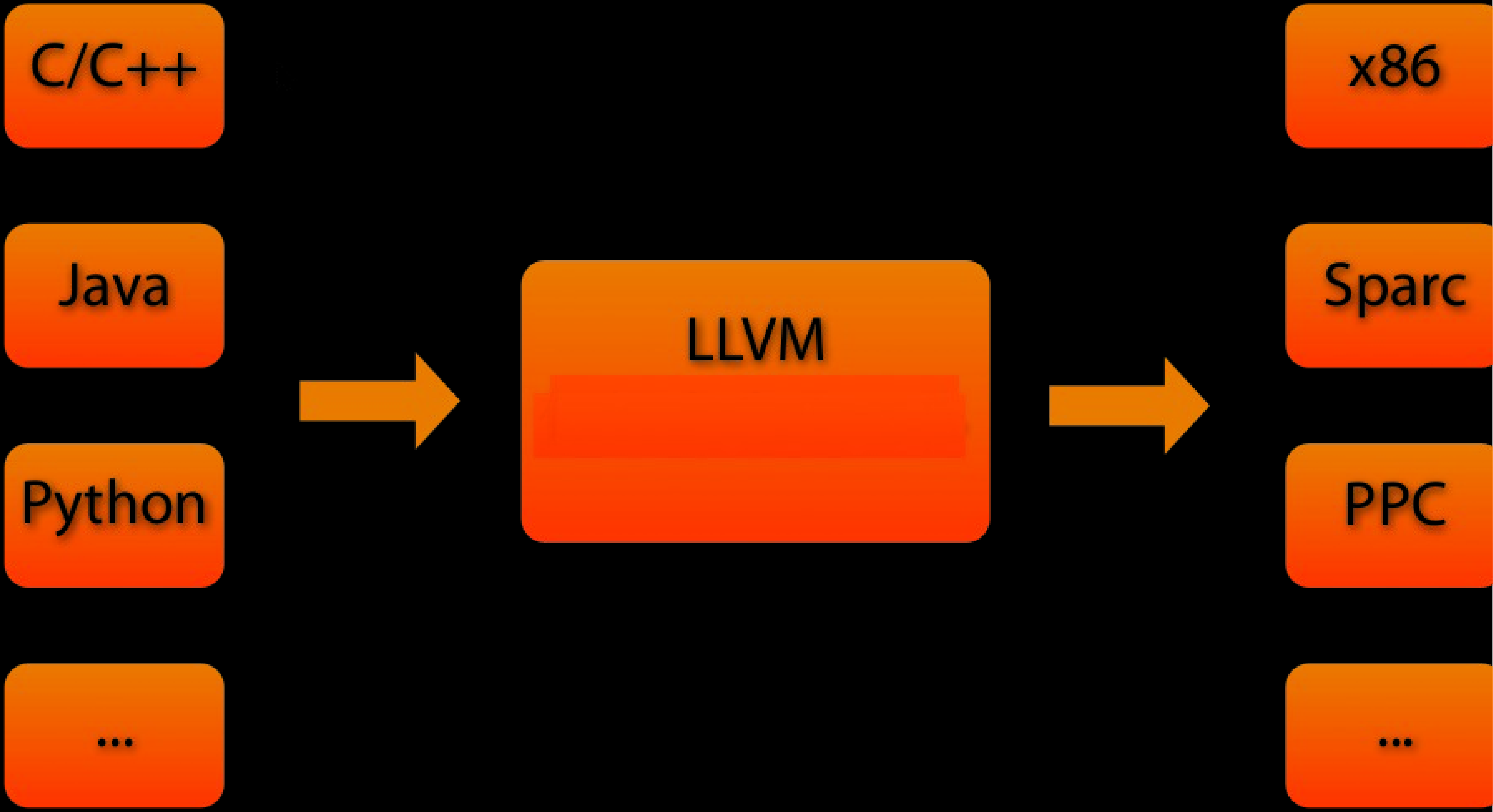


Key LLVM Feature:
IR is small, simple, easy to understand, and is well defined

LLVM:

三頭六臂

Frontend LLVM IR Backend



C/C++

Java

Python

...

LLVM

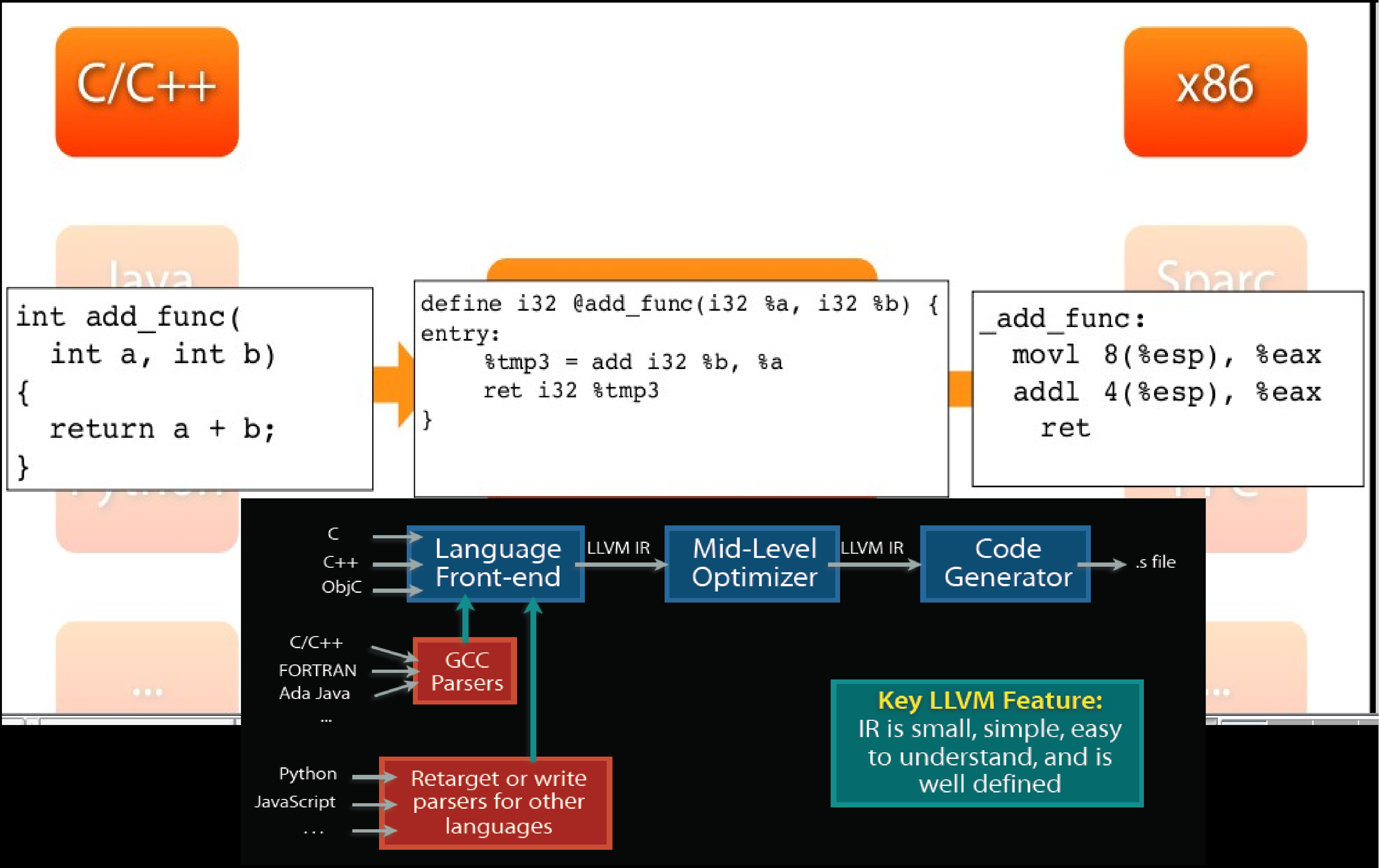
x86

Sparc

PPC

...

Frontend LLVM IR Backend

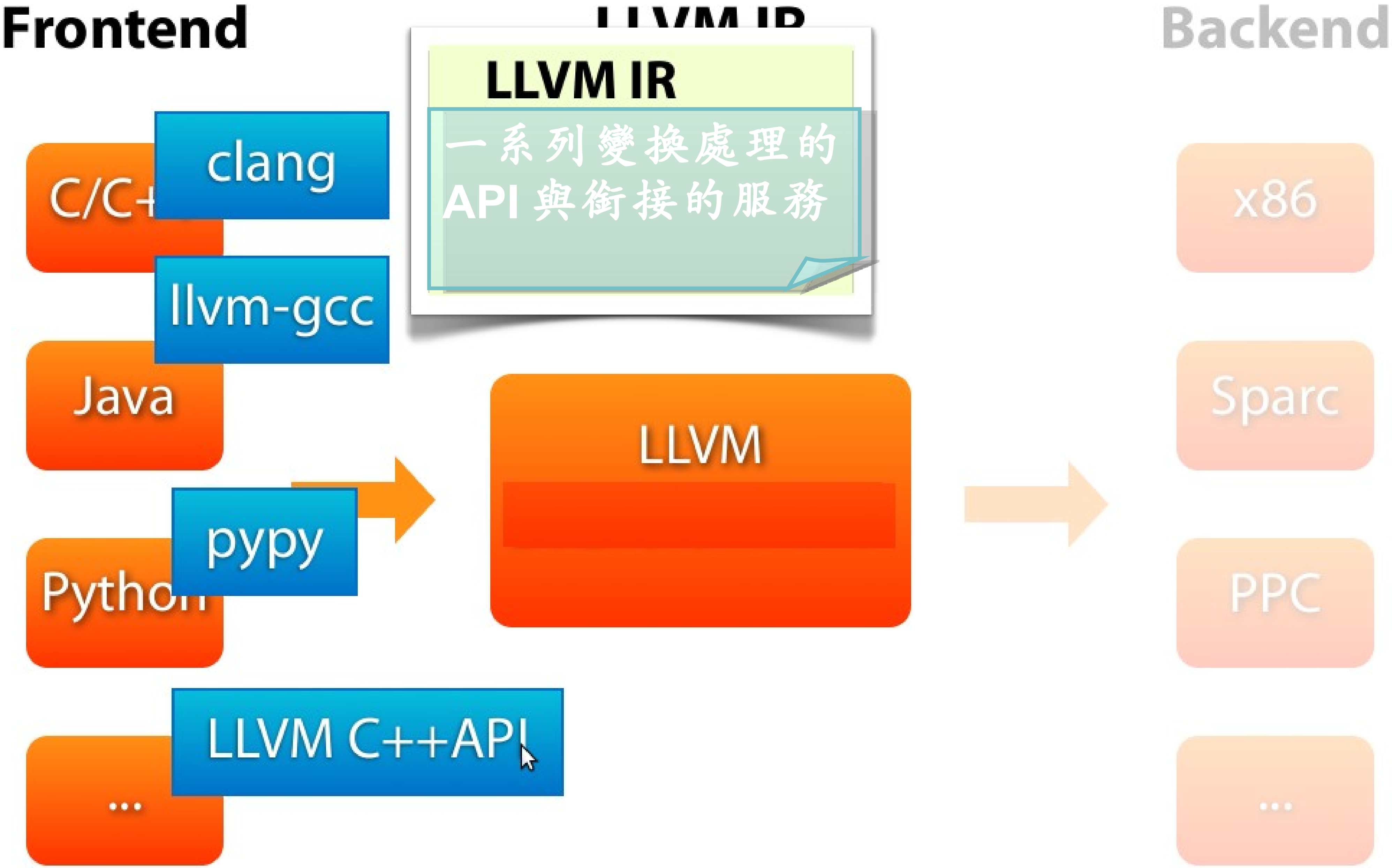


```
int add_func(
  int a, int b)
{
  return a + b;
}
```

```
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
```

Key LLVM Feature:
IR is small, simple, easy to understand, and is well defined



Frontend

LLVM IR

Backend

C/C++

clang

llvm-gcc

Java

Python

pypy

...

LLVM C++API

LLVM

x86

Sparc

PPC

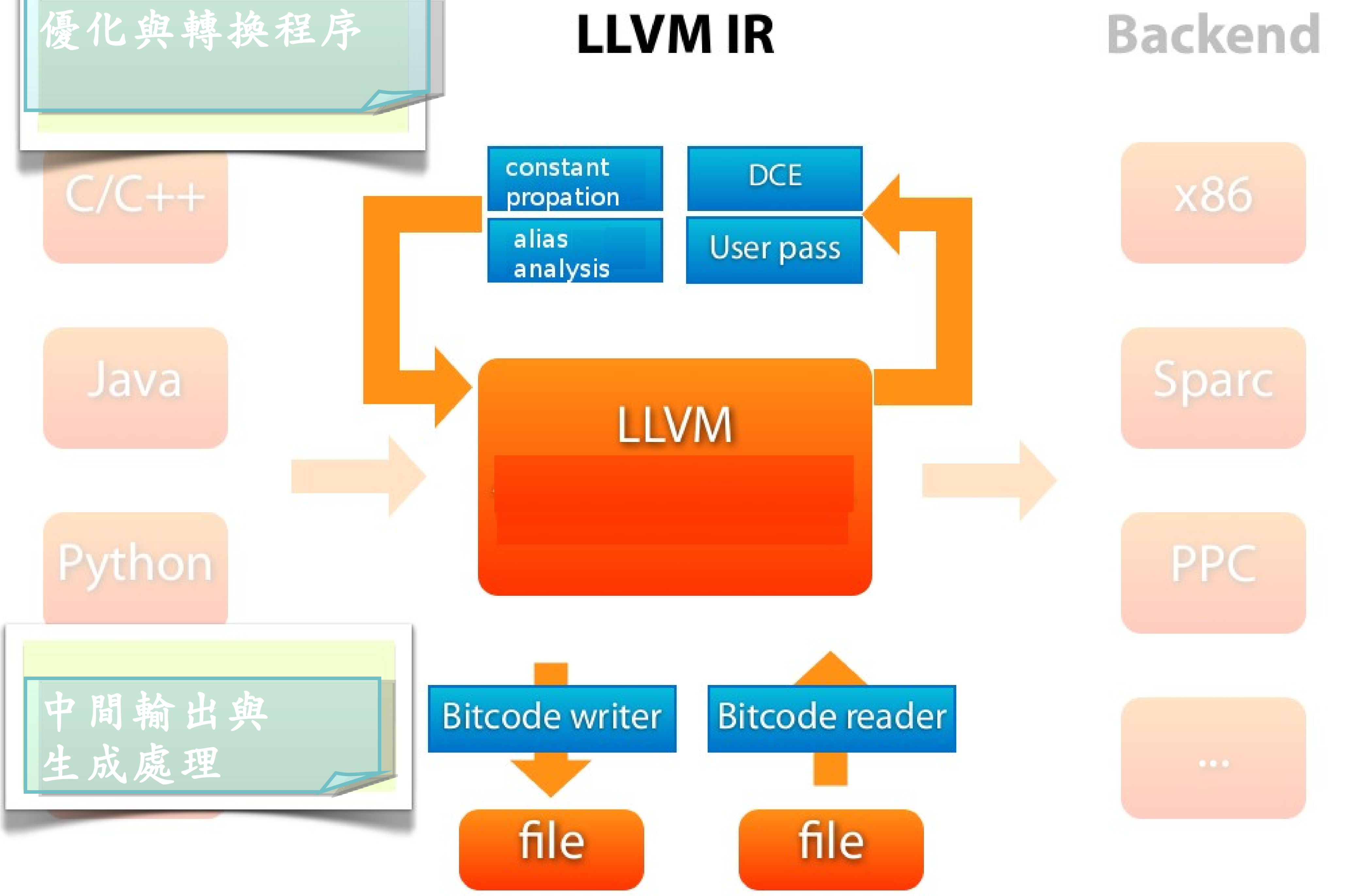
...

LLVM IR

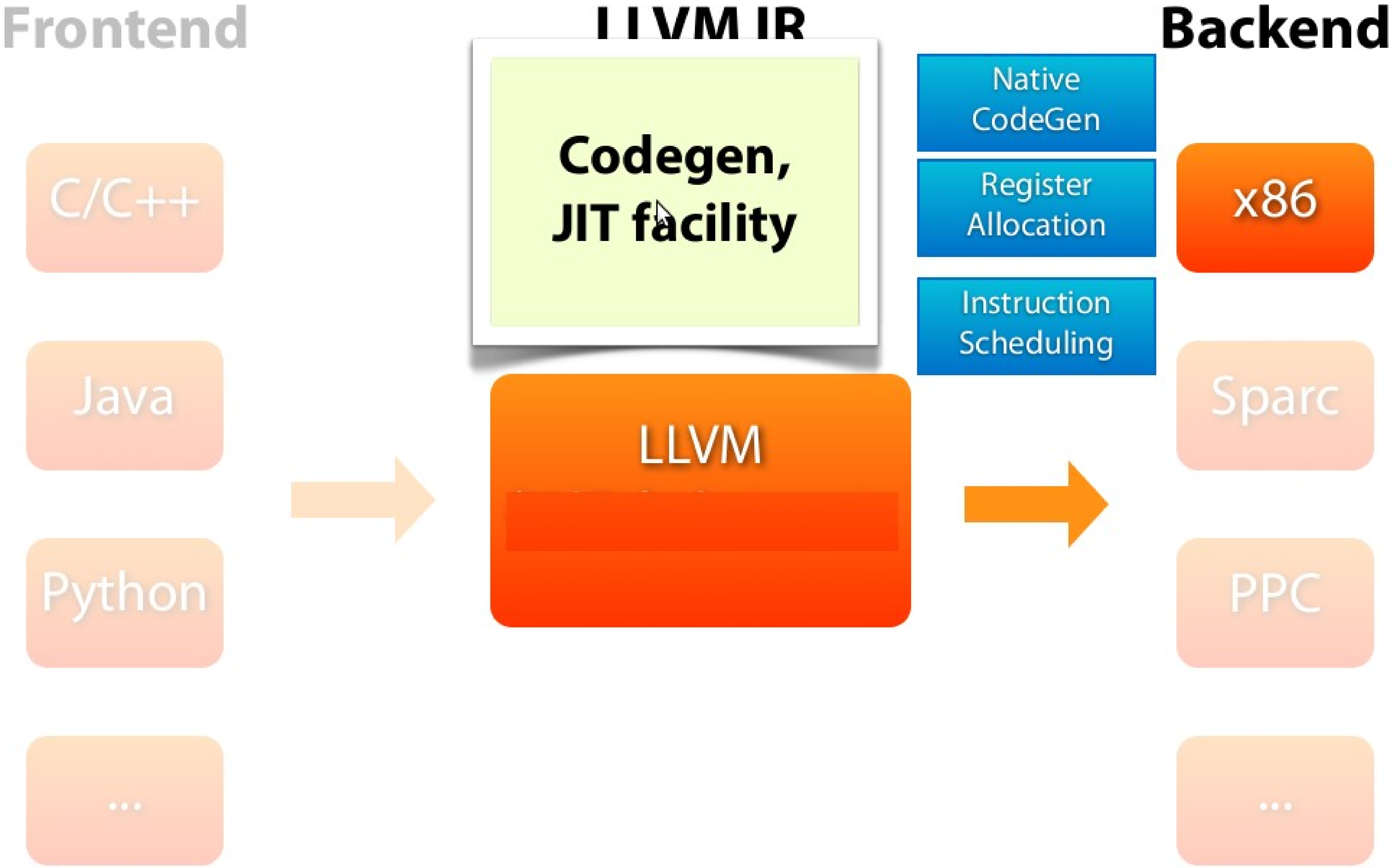
一系列變換處理的
API與銜接的服務

優化與轉換程序

LLVM IR



中間輸出與
生成處理



Frontend

LLVM IR

Backend

C/C++

Java

Python

...

Codegen,
JIT facility

LLVM

Native CodeGen

Register Allocation

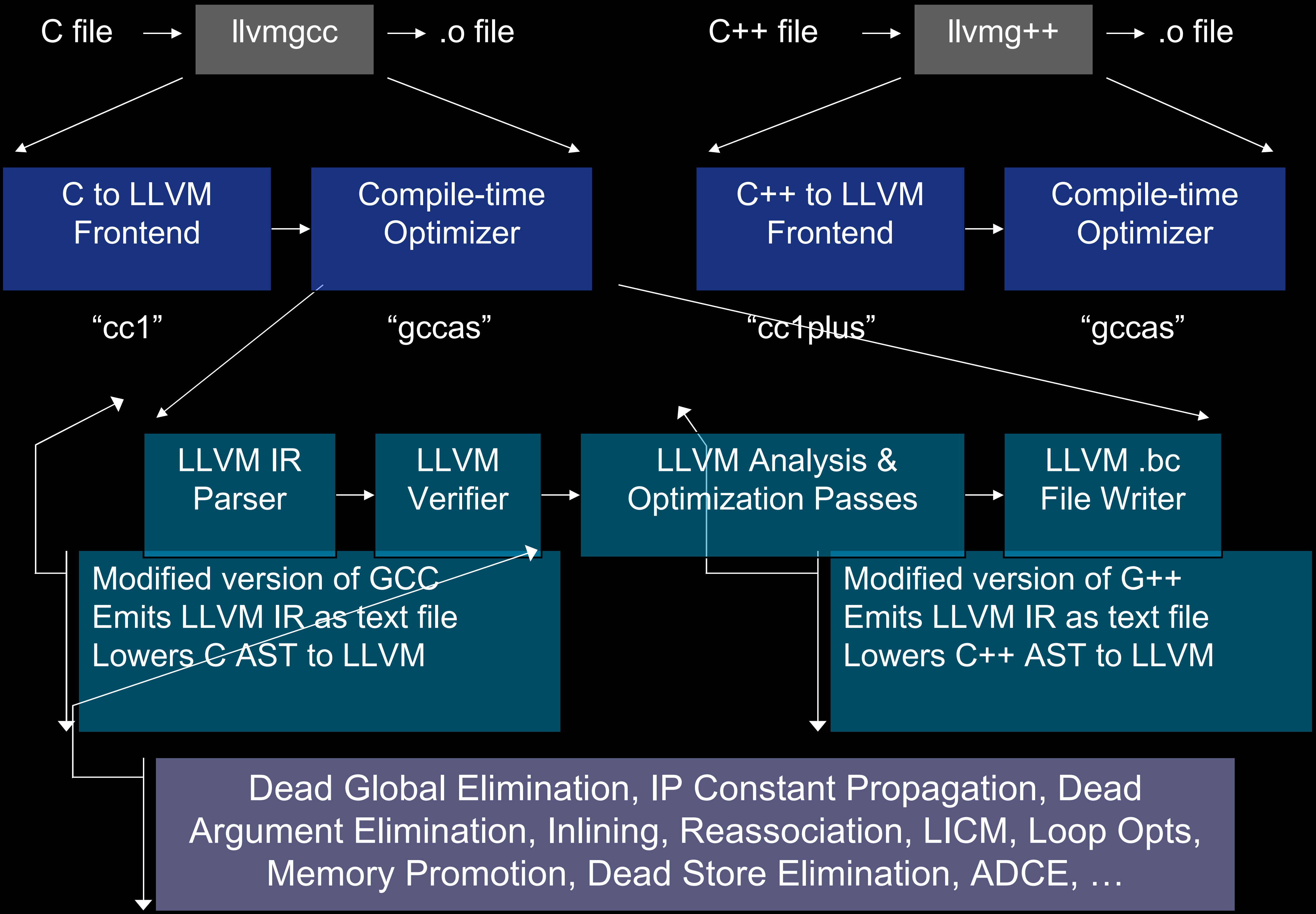
Instruction Scheduling

x86

Sparc

PPC

...



C file → llvmgcc → .o file

C++ file → llvmg++ → .o file

C to LLVM Frontend → Compile-time Optimizer

“cc1”

“gccas”

C++ to LLVM Frontend → Compile-time Optimizer

“cc1plus”

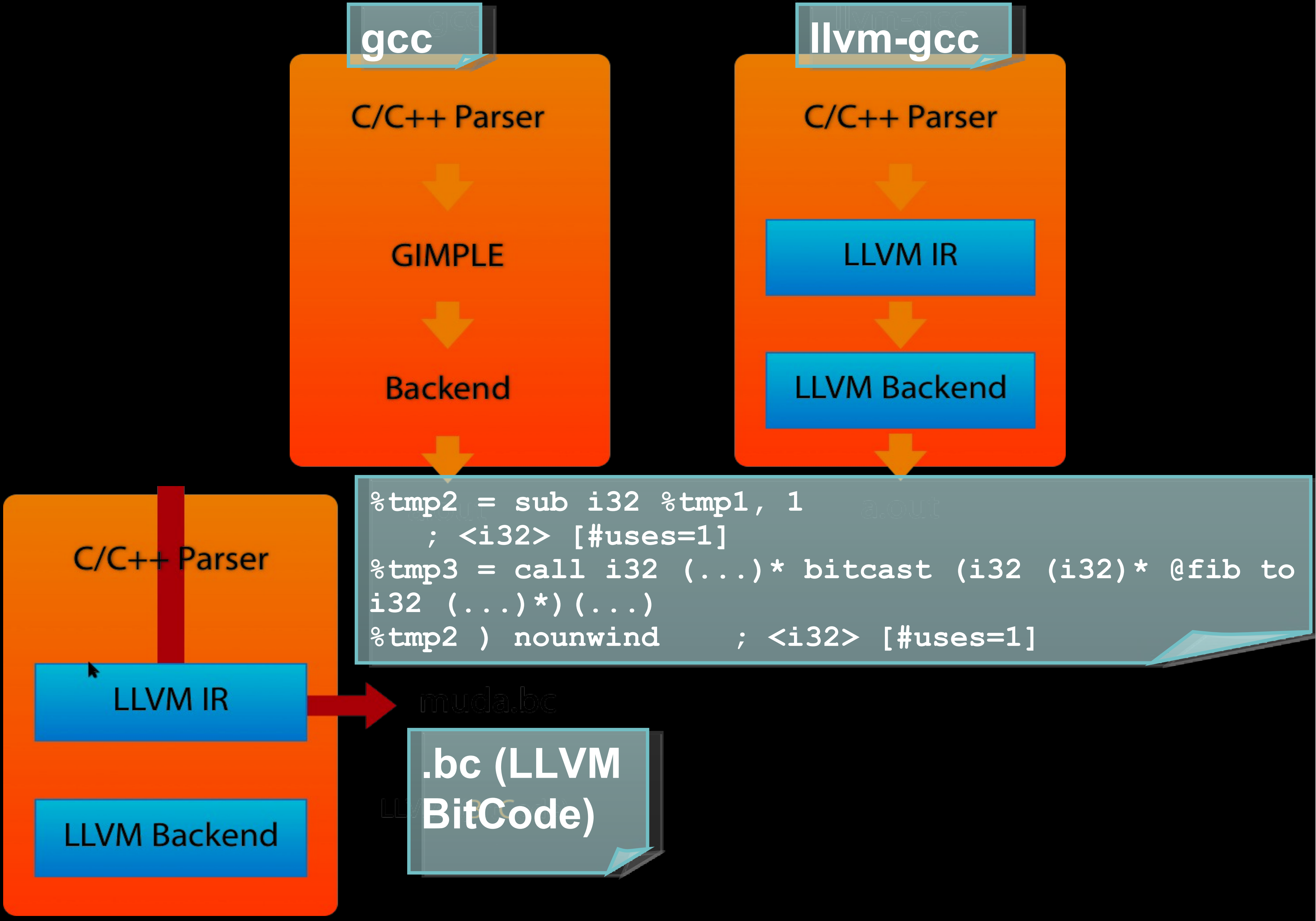
“gccas”

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, …

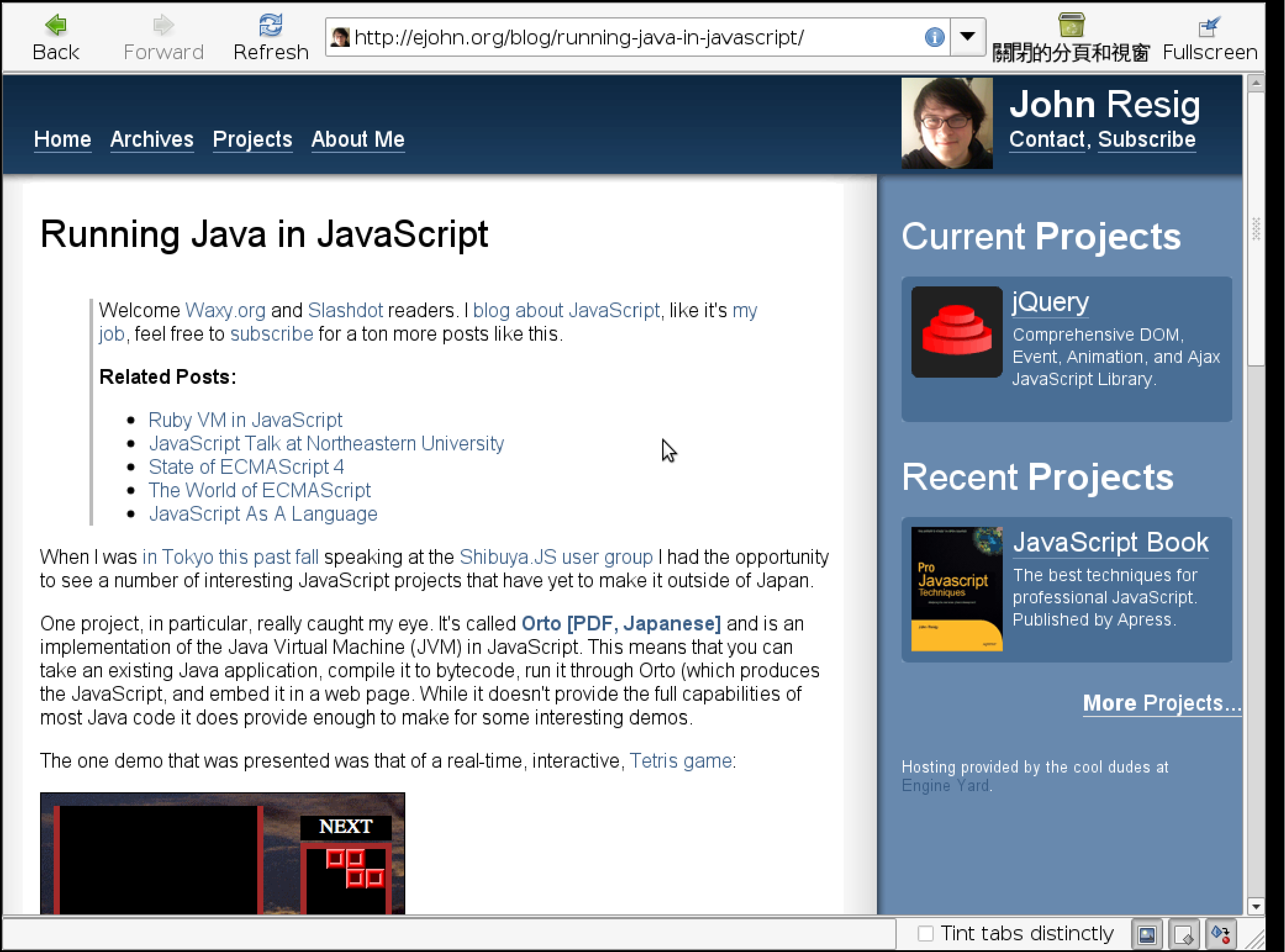


```
%tmp2 = sub i32 %tmp1, 1
    ; <i32> [#uses=1]
%tmp3 = call i32 (...)* bitcast (i32 (i32)* @fib to
i32 (...)*)(...)
%tmp2 ) nounwind    ; <i32> [#uses=1]
```

llvm-gcc -emit-llvm

等等，頭暈了 LLVM 到底能作什麼？

這是一個到處都有虛擬機器 (VM) 的時代



Running Java in JavaScript

> Welcome Waxy.org and Slashdot readers. I blog about JavaScript, like it's my job, feel free to subscribe for a ton more posts like this.
>
> **Related Posts:**
>
> - Ruby VM in JavaScript
> - JavaScript Talk at Northeastern University
> - State of ECMAScript 4
> - The World of ECMAScript
> - JavaScript As A Language

When I was in Tokyo this past fall speaking at the Shibuya.JS user group I had the opportunity to see a number of interesting JavaScript projects that have yet to make it outside of Japan.

One project, in particular, really caught my eye. It's called **Orto [PDF, Japanese]** and is an implementation of the Java Virtual Machine (JVM) in JavaScript. This means that you can take an existing Java application, compile it to bytecode, run it through Orto (which produces the JavaScript, and embed it in a web page. While it doesn't provide the full capabilities of most Java code it does provide enough to make for some interesting demos.

The one demo that was presented was that of a real-time, interactive, Tetris game:

Current Projects

jQuery

Comprehensive DOM, Event, Animation, and Ajax JavaScript Library.

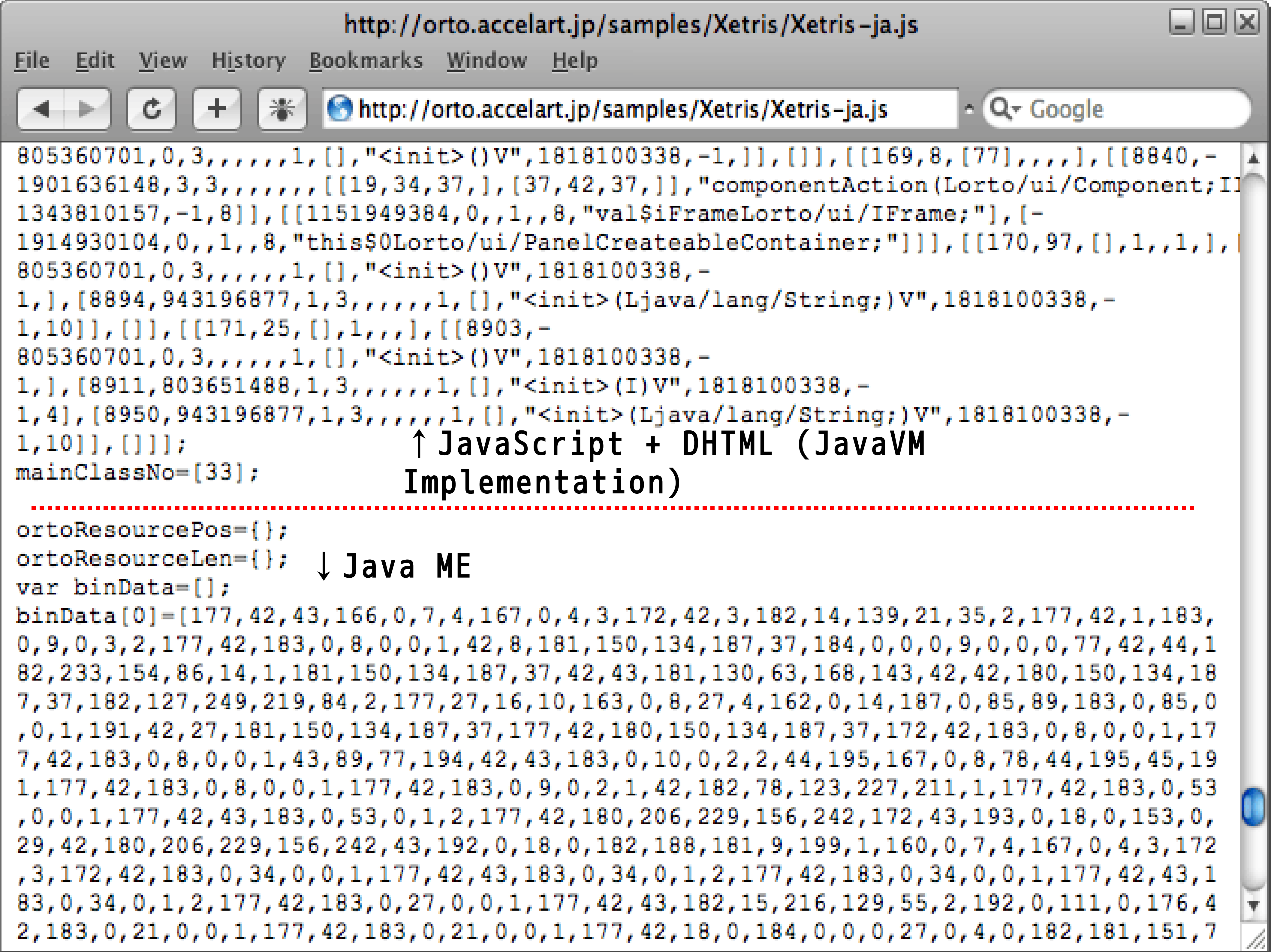
Recent Projects

JavaScript Book

The best techniques for professional JavaScript. Published by Apress.

More Projects...

Hosting provided by the cool dudes at Engine Yard.

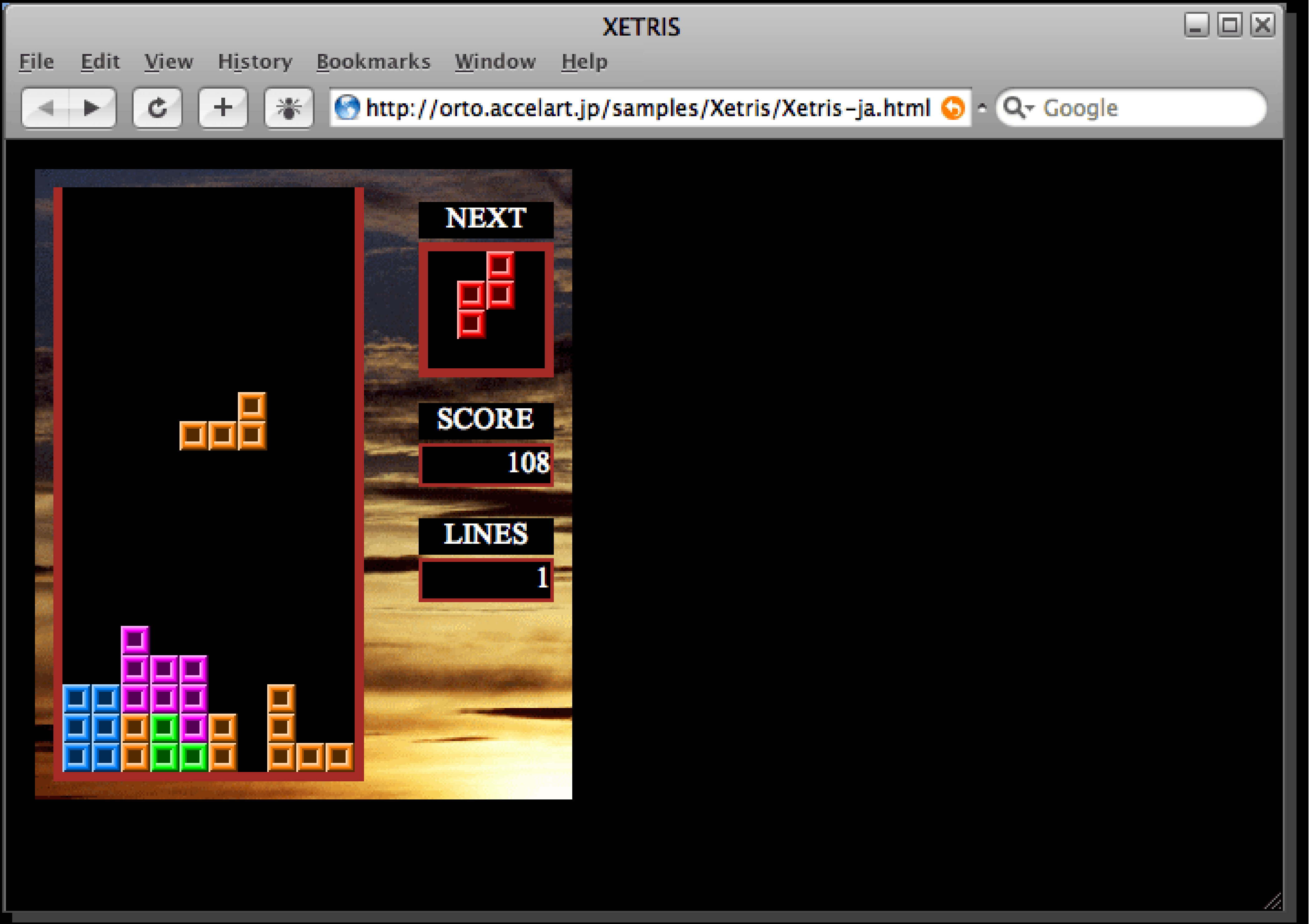

http://orto.accelart.jp/samples/Xetris/Xetris-ja.js
File Edit View History Bookmarks Window Help
http://orto.accelart.jp/samples/Xetris/Xetris-ja.js
Google

```
805360701,0,3,,,,,,1,[],"<init>()V",1818100338,-1,]],[]],[[169,8,[77],,,,],[[8840,-
1901636148,3,3,,,,,,,[[19,34,37,],[37,42,37,]],"componentAction(Lorto/ui/Component;I
1343810157,-1,8]],[[1151949384,0,,1,,8,"val$iFrameLorto/ui/IFrame;"],[-
1914930104,0,,1,,8,"this$0Lorto/ui/PanelCreateableContainer;"]]],[[170,97,[],1,,1,],
805360701,0,3,,,,,,1,[],"<init>()V",1818100338,-
1,],[8894,943196877,1,3,,,,,,1,[],"<init>(Ljava/lang/String;)V",1818100338,-
1,10]],[]],[[171,25,[],1,,,],[[8903,-
805360701,0,3,,,,,,1,[],"<init>()V",1818100338,-
1,],[8911,803651488,1,3,,,,,,1,[],"<init>(I)V",1818100338,-
1,4],[8950,943196877,1,3,,,,,,1,[],"<init>(Ljava/lang/String;)V",1818100338,-
1,10]],[]]];
mainClassNo=[33];
```

↑JavaScript + DHTML (JavaVM Implementation)

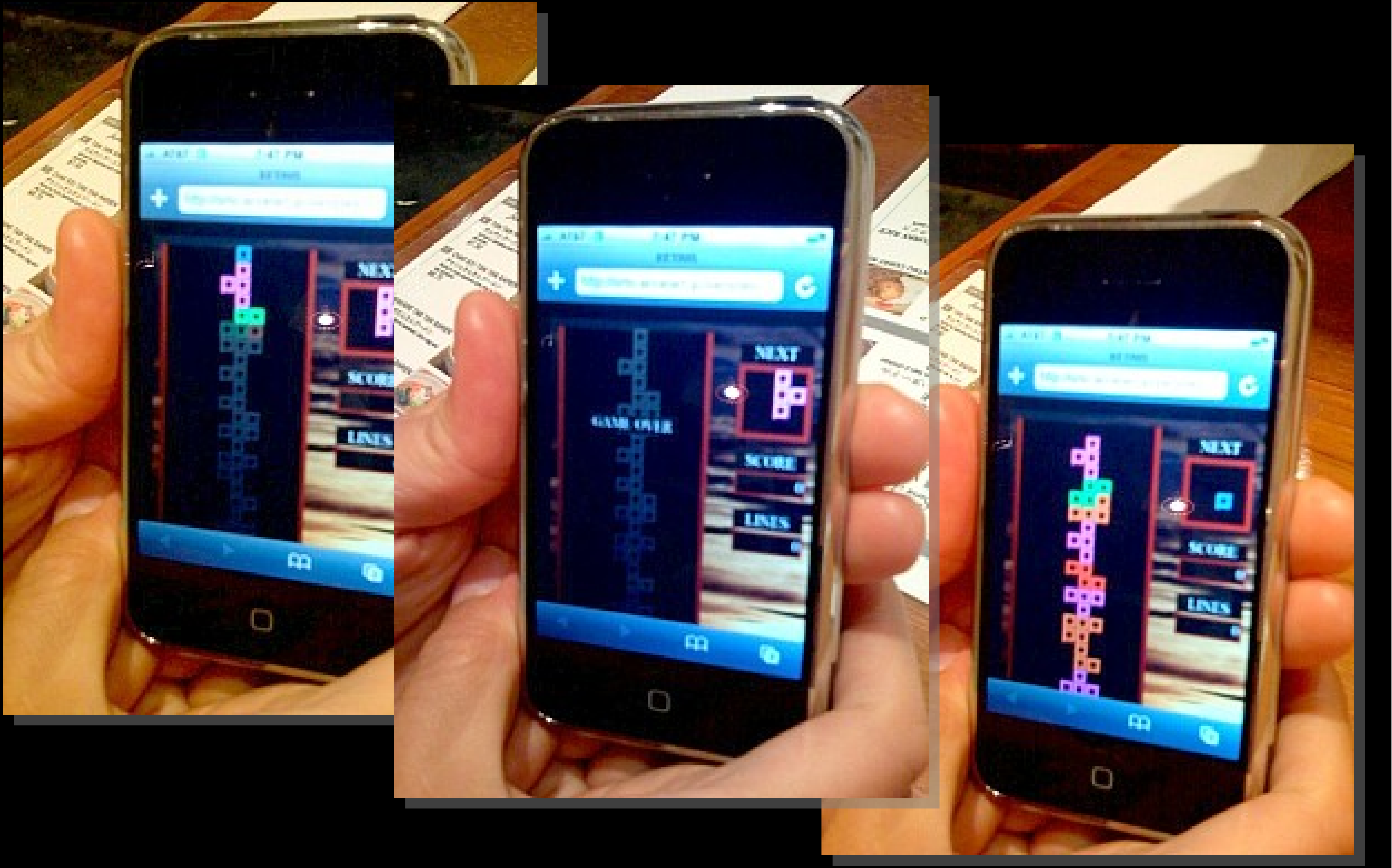
```
ortoResourcePos={};
ortoResourceLen={};
var binData=[];
binData[0]=[177,42,43,166,0,7,4,167,0,4,3,172,42,3,182,14,139,21,35,2,177,42,1,183,
0,9,0,3,2,177,42,183,0,8,0,0,1,42,8,181,150,134,187,37,184,0,0,0,9,0,0,0,77,42,44,1
82,233,154,86,14,1,181,150,134,187,37,42,43,181,130,63,168,143,42,42,180,150,134,18
7,37,182,127,249,219,84,2,177,27,16,10,163,0,8,27,4,162,0,14,187,0,85,89,183,0,85,0
,0,1,191,42,27,181,150,134,187,37,177,42,180,150,134,187,37,172,42,183,0,8,0,0,1,17
7,42,183,0,8,0,0,1,43,89,77,194,42,43,183,0,10,0,2,2,44,195,167,0,8,78,44,195,45,19
1,177,42,183,0,8,0,0,1,177,42,183,0,9,0,2,1,42,182,78,123,227,211,1,177,42,183,0,53
,0,0,1,177,42,43,183,0,53,0,1,2,177,42,180,206,229,156,242,172,43,193,0,18,0,153,0,
29,42,180,206,229,156,242,43,192,0,18,0,182,188,181,9,199,1,160,0,7,4,167,0,4,3,172
,3,172,42,183,0,34,0,0,1,177,42,43,183,0,34,0,1,2,177,42,183,0,34,0,0,1,177,42,43,1
83,0,34,0,1,2,177,42,183,0,27,0,0,1,177,42,43,182,15,216,129,55,2,192,0,111,0,176,4
2,183,0,21,0,0,1,177,42,183,0,21,0,0,1,177,42,18,0,184,0,0,0,27,0,4,0,182,181,151,7
```

↓Java ME



XETRIS
File
Edit
View
History
Bookmarks
Window
Help
http://orto.accelart.jp/samples/Xetris/Xetris-ja.html
Google
NEXT
SCORE
108
LINES
1

It works on iPhone!



But you can't control because iphone has no keyboard;(

（VM之）道在屎溺

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

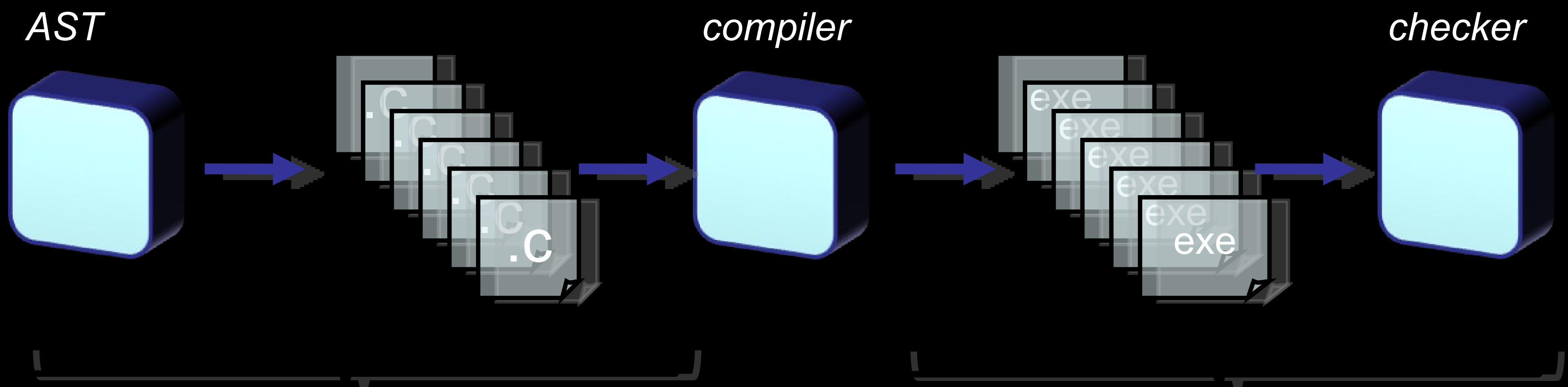
手法：

- →**JIT : Just-In-Time** (Jizz In [My Pants] Time?!) **compiler**
- →**IR to backend translation / code generation**

「你想要的，

LLVM 都給你！」

編譯器流程 ::JIT



程式碼的編譯與生成

執行、評估，與驗證

建立 IR

載入必要的函式庫

連結程式模組

Optimizations + Transforms

codegen

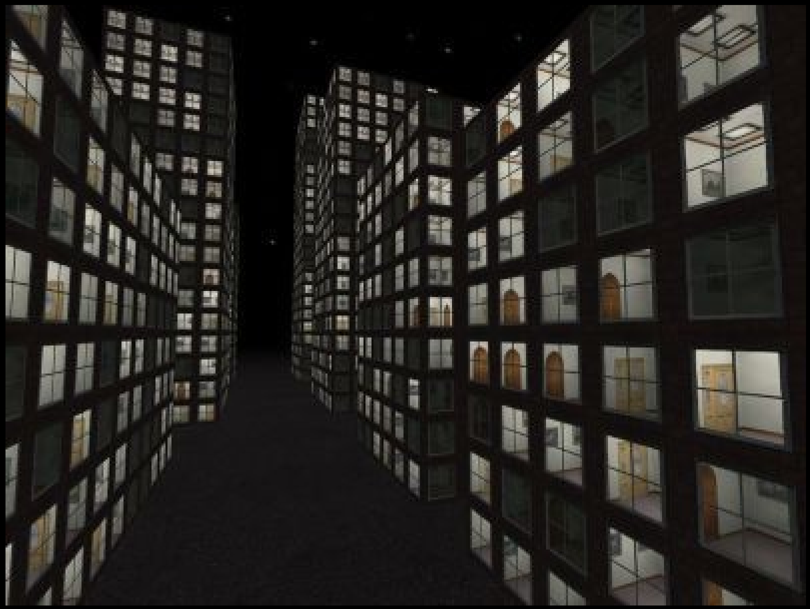
當 LLVM 煞到 OpenGL Shader

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



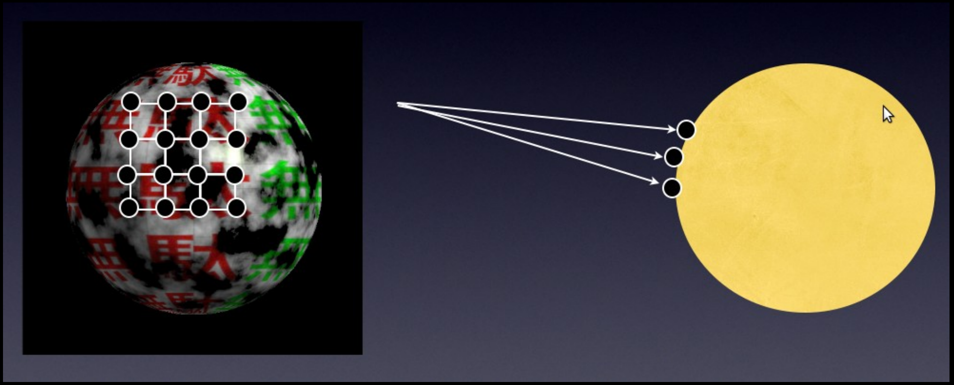
3D/Shader 的考量點

光

反射

移動的演算法

Reyes(scanline,polygon)



Raytracing

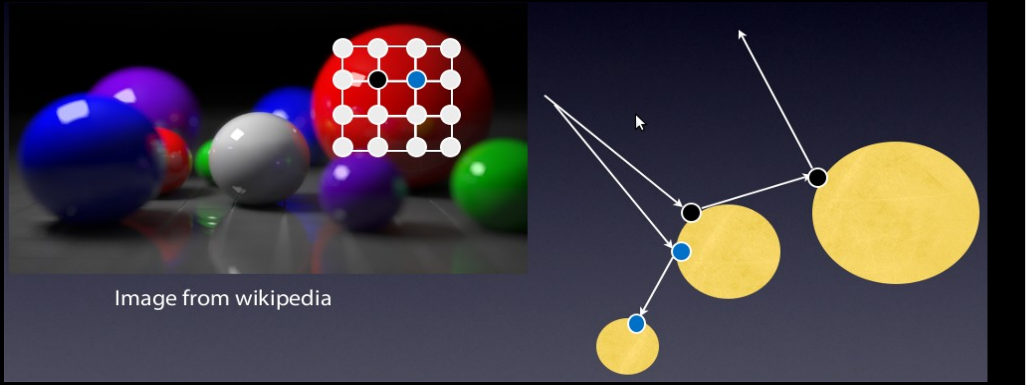
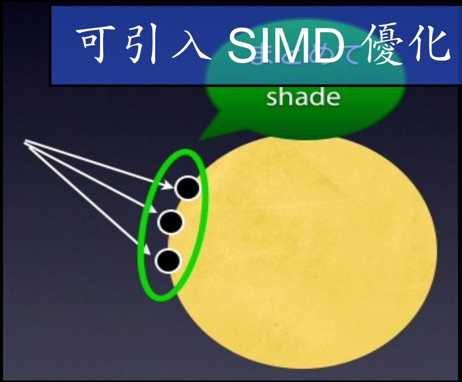
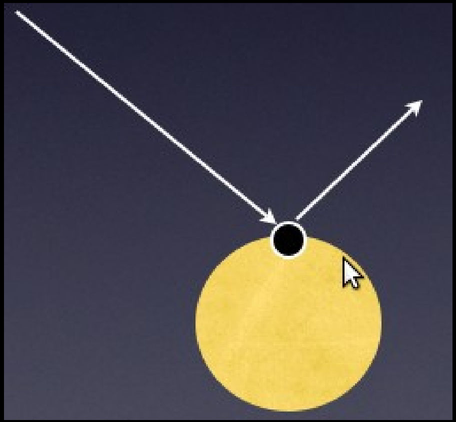


Image from wikipedia



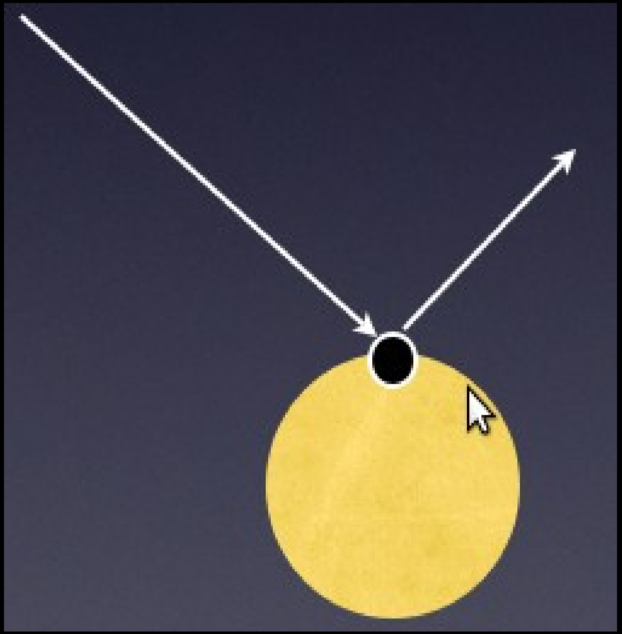
可引入 SIMD 優化



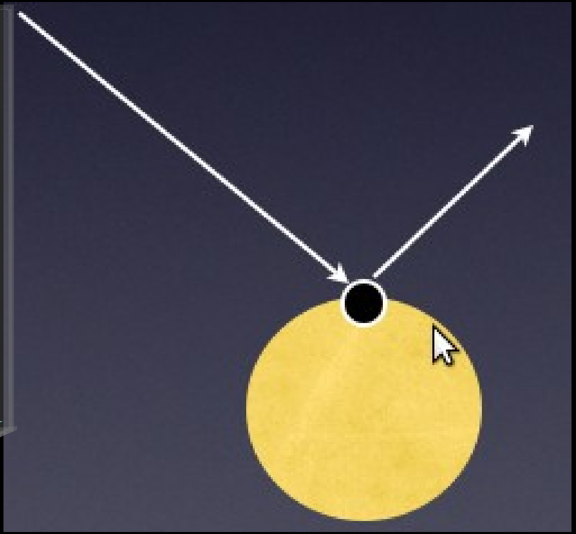
```
surface
matte(float Ka = 1.0; float Kd = 1.1;)
{
  normal Nf = faceforward(
              normalize(N), I);
  Oi = Os;
  Ci = Os * Cs * (Ka * ambient() +
                  Kd * diffuse(Nf));


}
```

```
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));
}
```



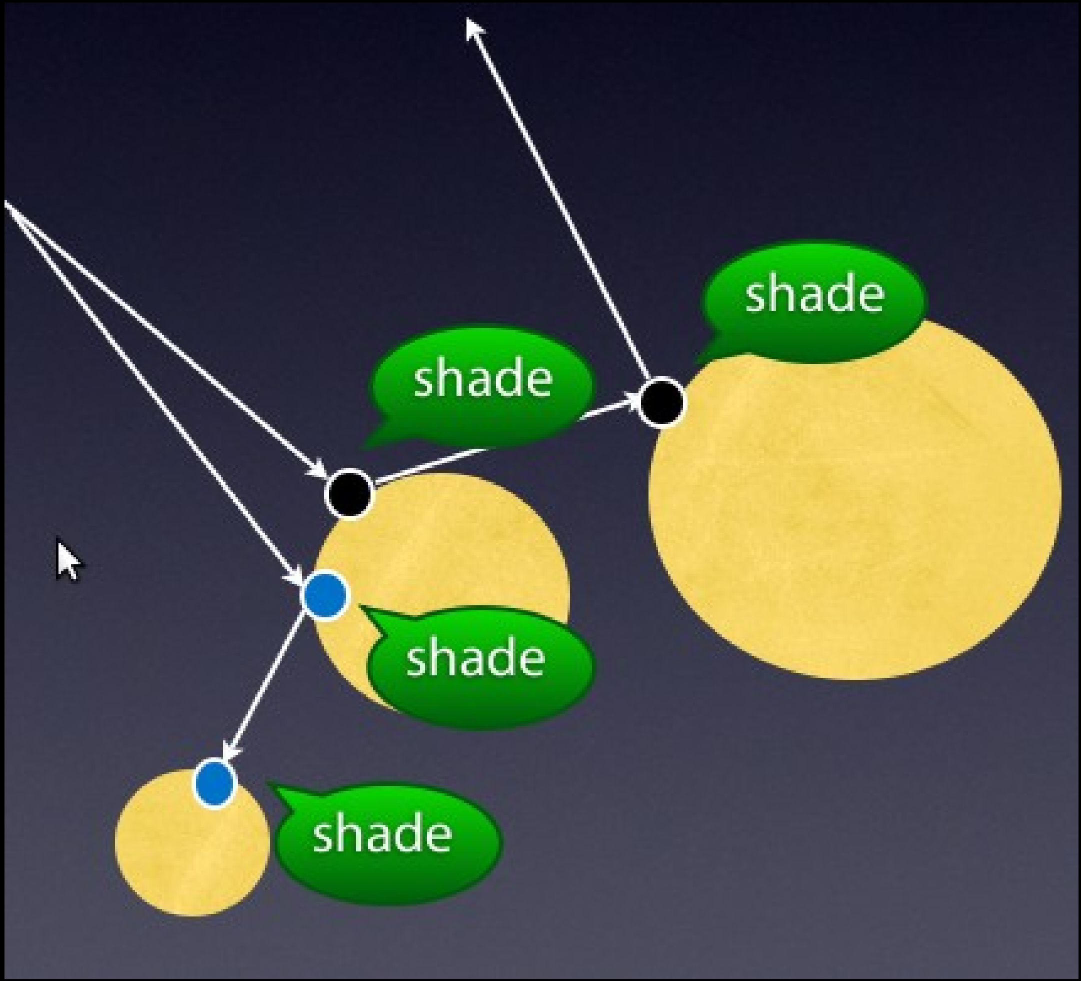
push N
normalize
push I
faceforward
...

Raytracing 的難題

無法善用 SIMD

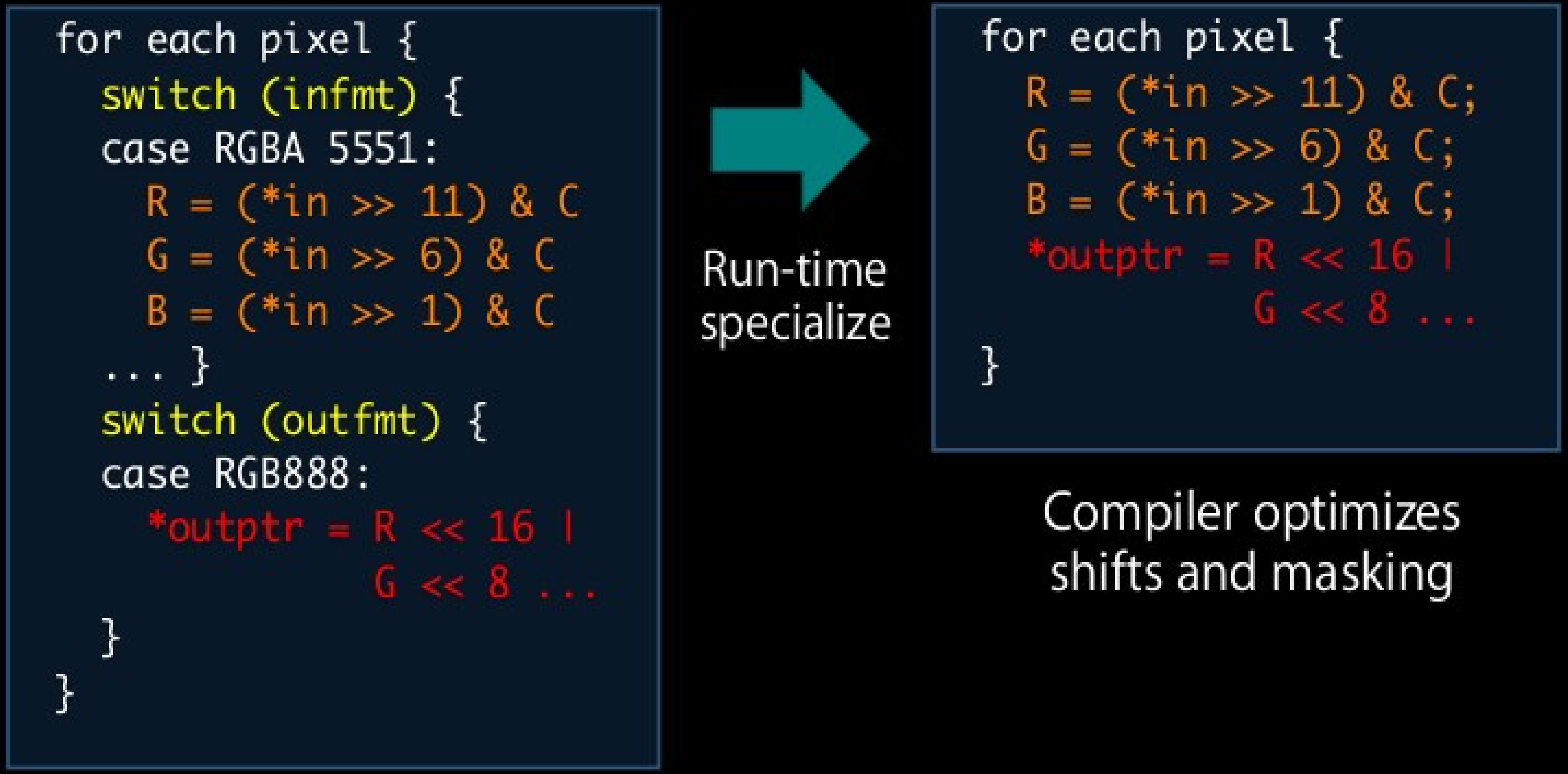
運算相依性高且繁瑣

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



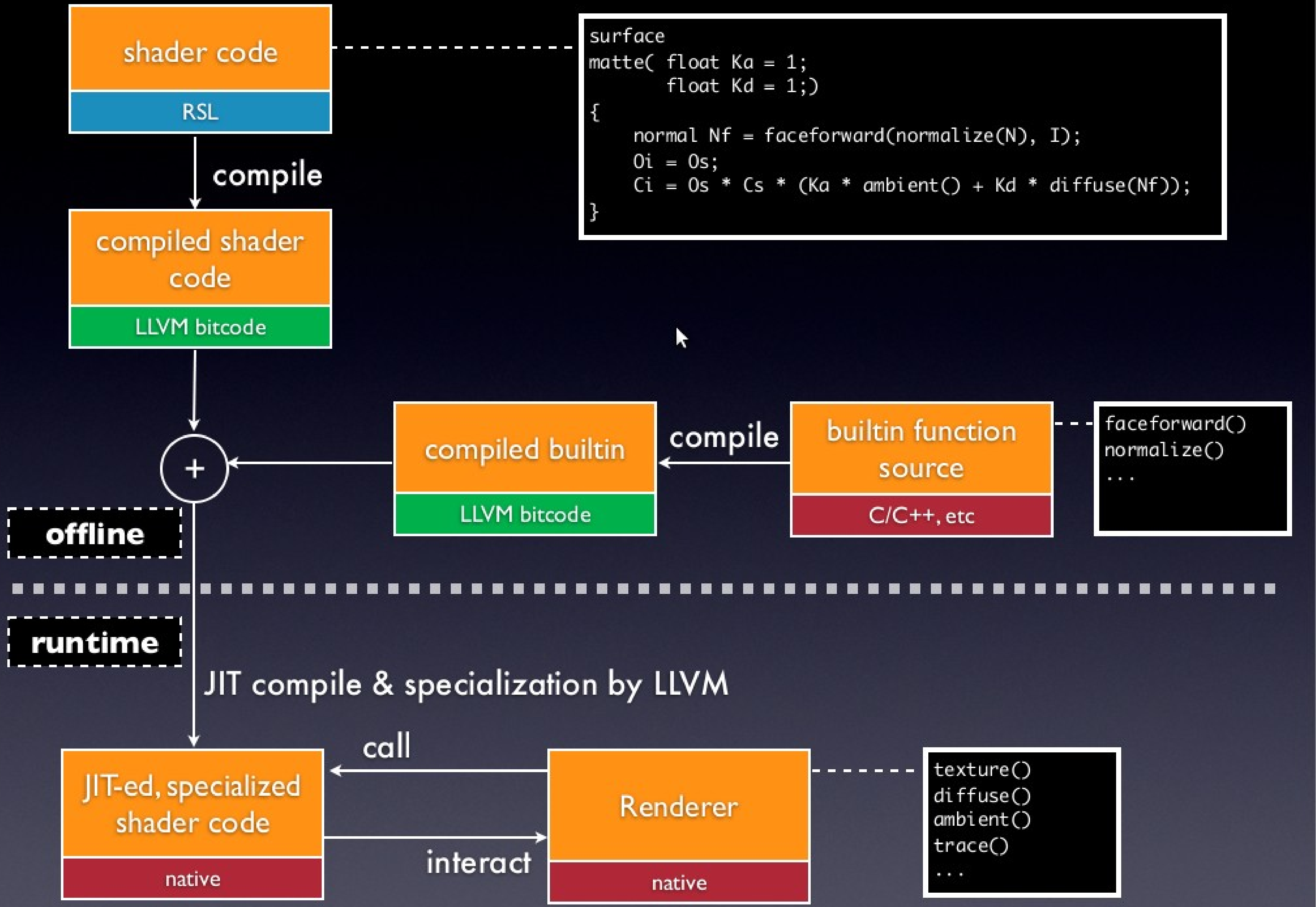
```
for each pixel {
  switch (infmt) {
  case RGBA 5551:
    R = (*in >> 11) & C
    G = (*in >> 6) & C
    B = (*in >> 1) & C
  ... }
  switch (outfmt) {
  case RGB888:
    *outptr = R << 16 |
              G << 8 ...
  }
}
```

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

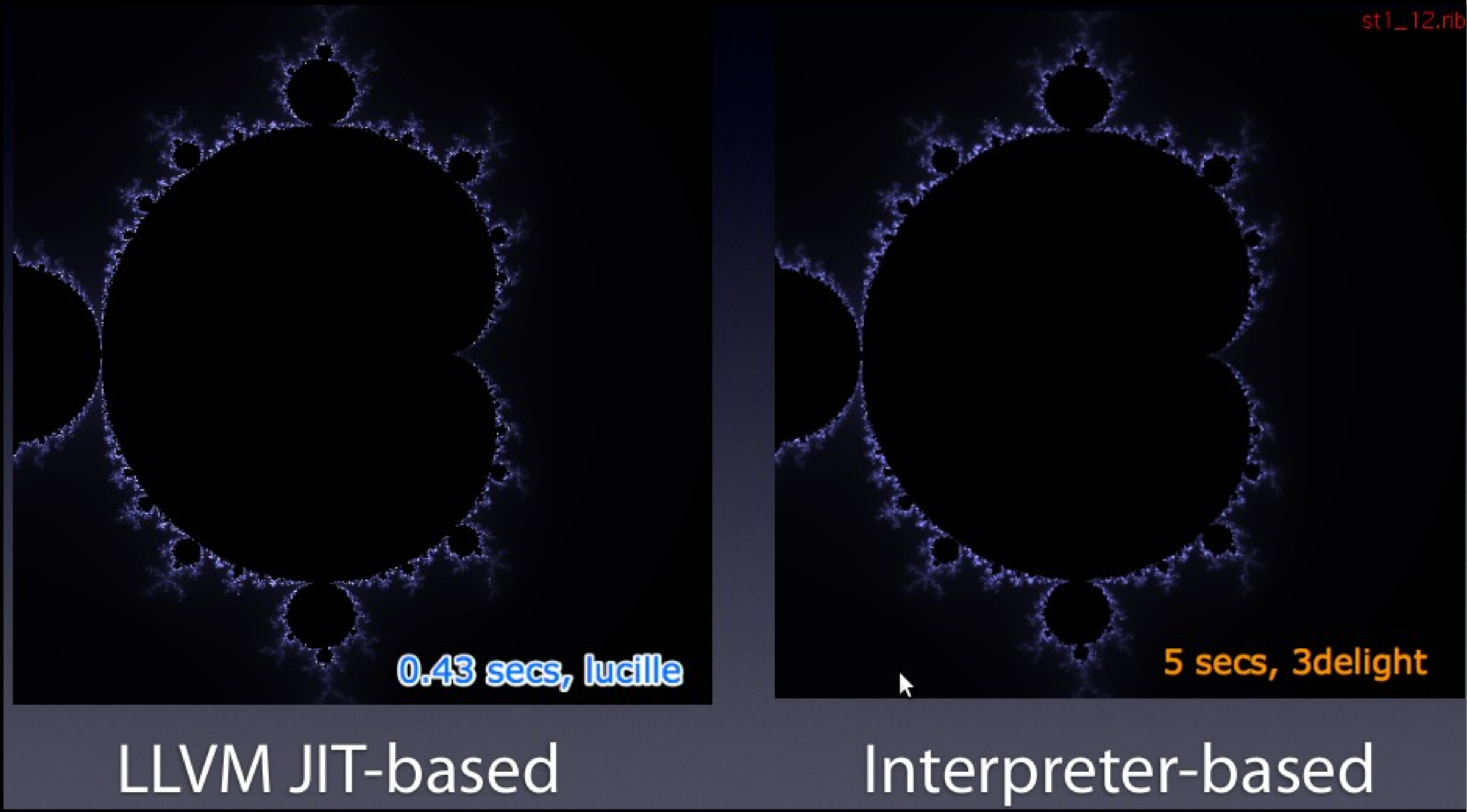
Speedup depends on src/dest format:
– 5.4x speedup on average, 19.3x max speedup: (13.3MB/s to 257.7MB/s)



```
surface
matte( float Ka = 1;
       float Kd = 1;)
{
    normal Nf = faceforward(normalize(N), I);
    Oi = Os;
    Ci = Os * Cs * (Ka * ambient() + Kd * diffuse(Nf));
}
```

```
faceforward()
normalize()
...
```

```
texture()
diffuse()
ambient()
trace()
...
```

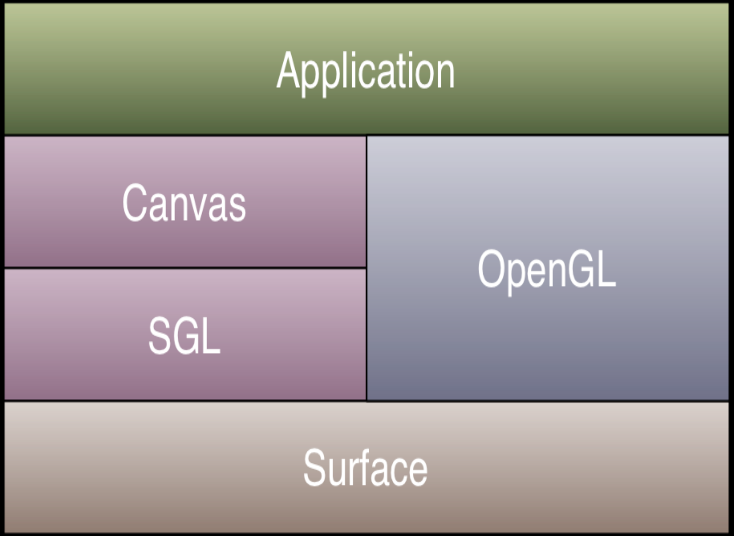


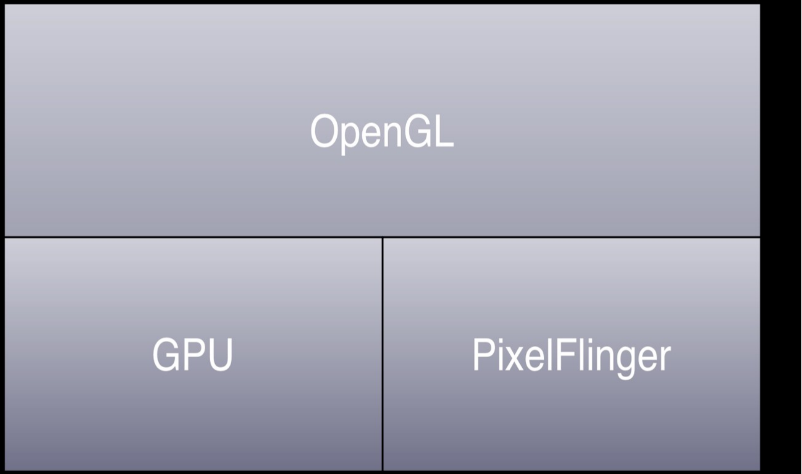
Mandelbort碎形運算透過LLVM JIT後，
提昇效能達到11倍

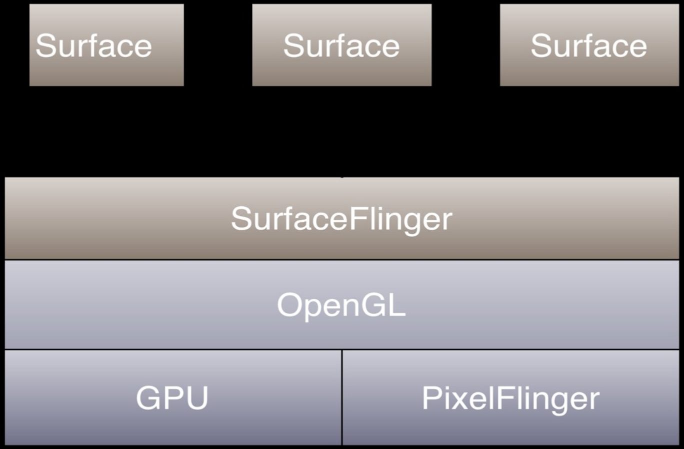
LLVM 與繪圖處理的應用

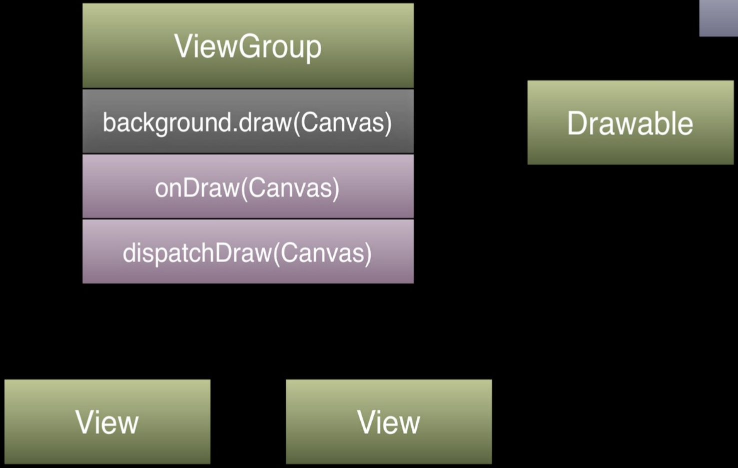
- 手機平台： Android PixelFlinger
- 繪圖軟體： Adobe PixelBender, Shader
- 3D 高品質圖形處理： Gallium3D / OpenGL

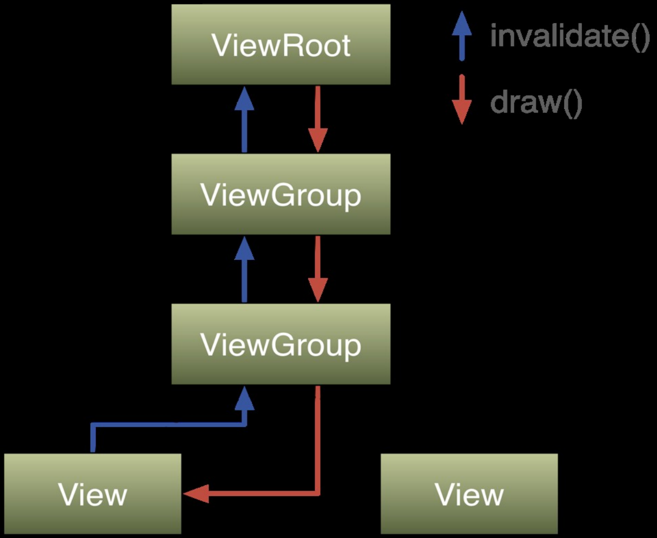
Android OpenGL|ES









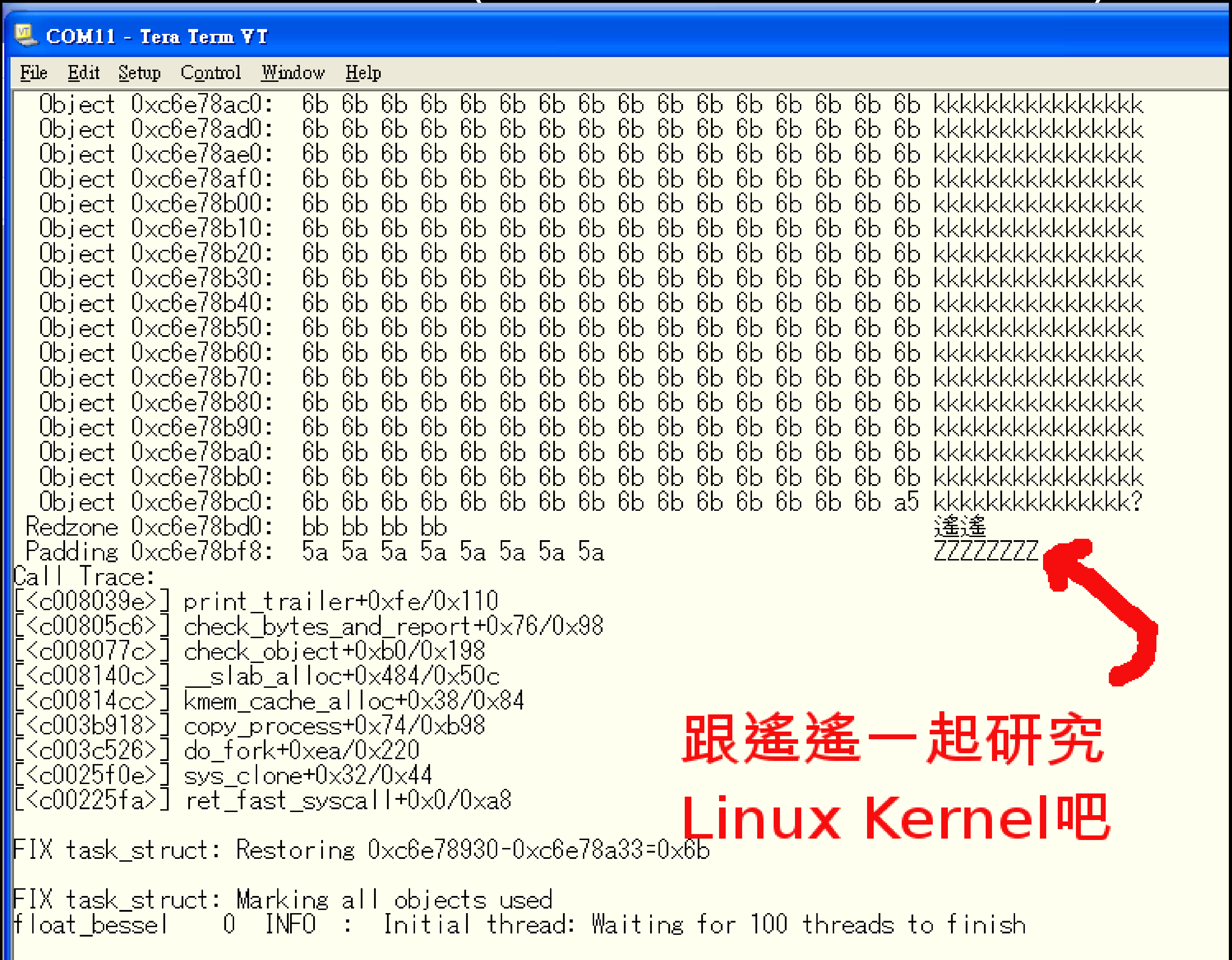


當 LLVM 煞到 Adobe Flash

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)



```
  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

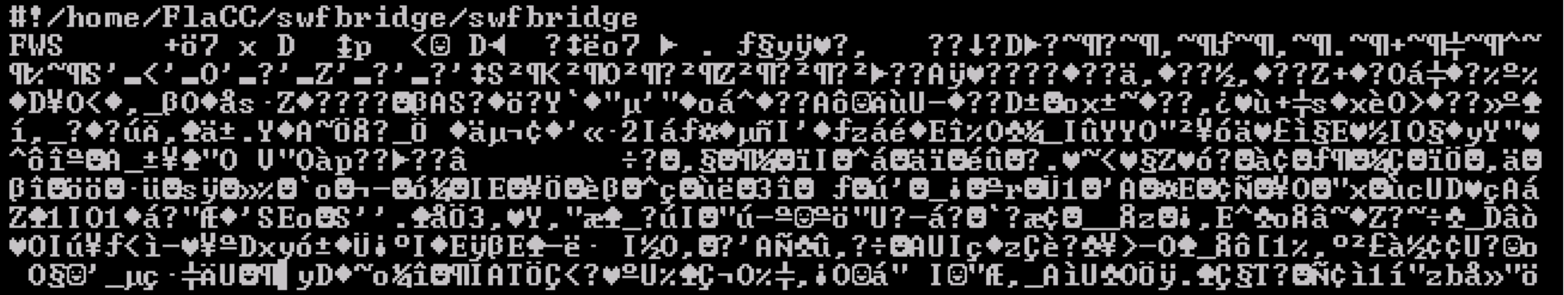
當然啦，慣C的風險也很高

「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



```
#!/home/FlaCC/swfbridge/swfbridge
FWS      +ö7 x D  ↨p  <☺ D◄  ?↨ëo7 ► . ƒ§yÿ♥?,    ??↓?D►?~¶?~¶,~¶ƒ~¶,~¶.~¶+~¶╤~¶^~
¶½~¶S'▬<'▬O'▬?'▬Z'▬?'▬?'↨S²¶K²¶O²¶?²¶Z²¶?²¶?²►??Aÿ♥????♦??ä,♦??½,♦??Z+♦?Oá╤♦?%ª%
♦D¥O<♦,_ßO♦ås·Z♦?????☻ßAS?♦ö?Y`♦"µ'"♦oá^♦??Aô☺ÄùU─♦??D±☻ox±~♦??,¿♥ù+╪s♦xèO>♦??»ª♠
í,_?♦?úÄ,♠ä±.Y♦A~ÖÅ?_Ö ♦äµ¬¢♦'«·2Iáƒ☼♦µñI'♦ƒzáé♦Eî%O♣%_IûYYO"²¥óä♥£ì§E♥½IO§♦yY"♥
^ôîª☻A_±¥♠"O U"Oàp??►??â       ÷?☻,§☻¶½☻ïI☻^á☻äï☻éû☻?.♥~<♥§Z♥ó?☻àç☻ƒ¶☻½Ç☻ïÖ☻,ä☻
ßî☻öö☻·ü☻sÿ☻»½☻`o☻¬─☻ó½☻IE☻¥Ö☻èß☻^ç☻ùë☻3î☻ ƒ☻ú'☻_¡☻ªr☻Ü1☻'A☻☼E☻¢Ñ☻¥O☻"x☻ücUD♥çAá
Z♠1IO1♦á?"Æ♦'SEo☻S''.♠åÖ3,♥Y,"æ♠_?úI☻"ú─ª☺ªö"U?─á?☻`?æ¢☻__Åz☻¡,E^♣oÅâ~♦Z?~÷♣_Dâò
♥OIú¥ƒ<ì─♥¥ªDxyó±♦Ü¡ºI♦EÿßE♠─ë· I½O,☻?'AÑ♣û,?÷☻AUIç♦zÇè?♣¥>─O♠_Åô[1%,º²£à½¢¢U?☺o
 O§☺'_µç·╪äU☻¶▌yD♦~o½î☻¶ÏATÖÇ<?♥ªU%♠Ç¬O%╪,¡O☺á" I☺"Æ,_AìU♣OÖÿ.♠Ç§T?☻Ñ¢ìIí"zbå»"ö
```

背景知識

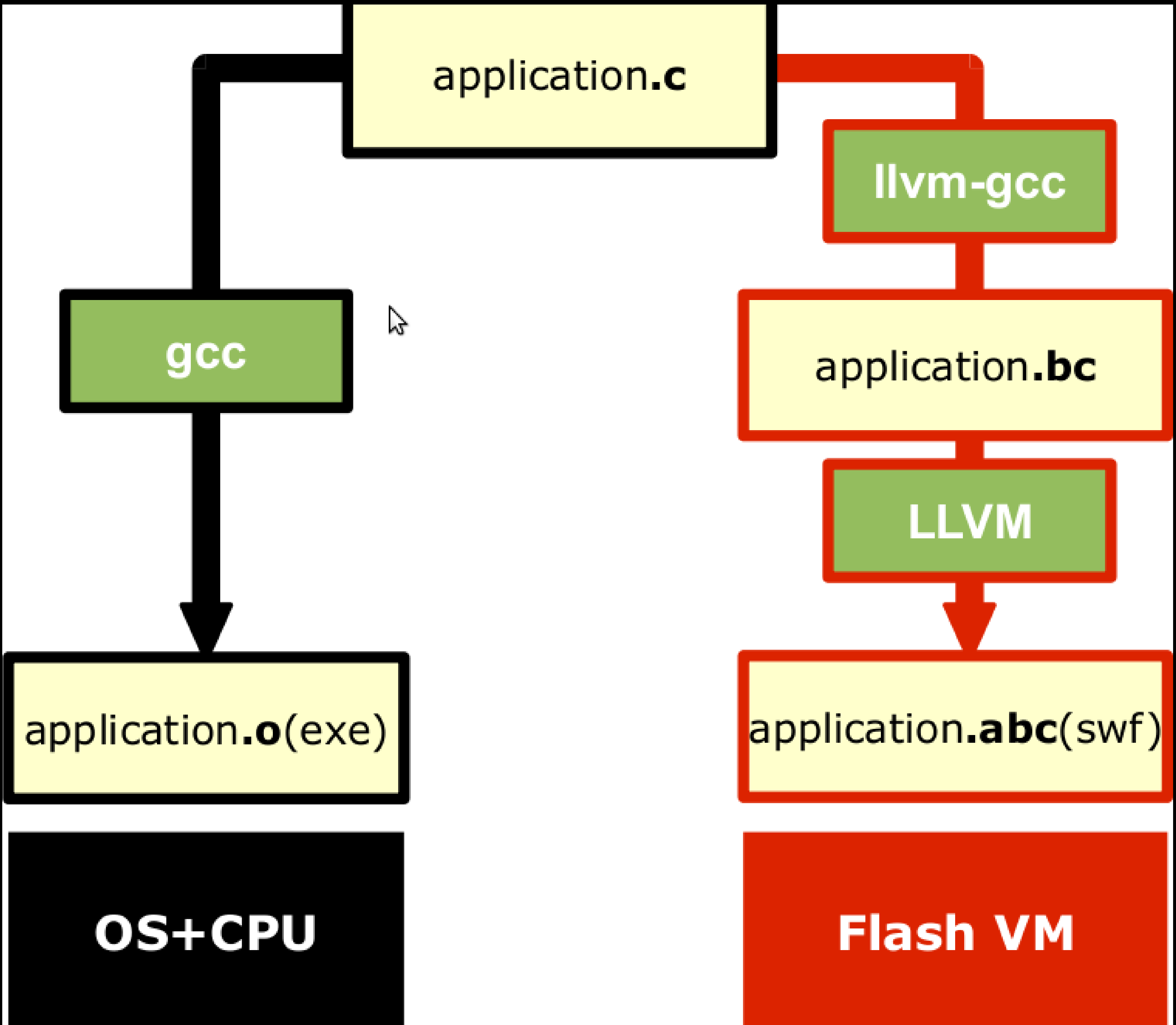
- from **Adobe, Mozilla, and Tamarin**

  http://hecker.org/mozilla/adobe-mozilla-and-tamarin

- Flash player 9 在 AVM2 內建 JIT-based VM
  - Tamarin 由 Adobe 捐贈予 Mozilla
  - Flash player 9 = AVM2 + graphics, musics, videos, networking, etc components
- ActionScript 3.0 (Flash 5 對 ActionScript 的稱呼)
  - JavaScript 的姊妹規格
  - 參考 ECMAScript 4 (ECMA-262)
  - Free Flex 2. SDK (foo.as -> foo.swf)
  - Flash player 9 內部具備 ABC 執行引擎

Tamarin

- ECMAScript version 4 (ES4) 的高速實做
- Tamarin VM, Mozilla, JavaScript, SpiderMonkey, Adobe Flash Player, AVM
- AVMPlus
- AVMPlus (VM library), MMgc (garbage collection library), avmplus
- Avmplus, ABC, JIT
- Adobe Flex 2 SDK includes ASC (ActionScript Compiler)



application**.c**

gcc

application**.o**(exe)

OS+CPU

llvm-gcc

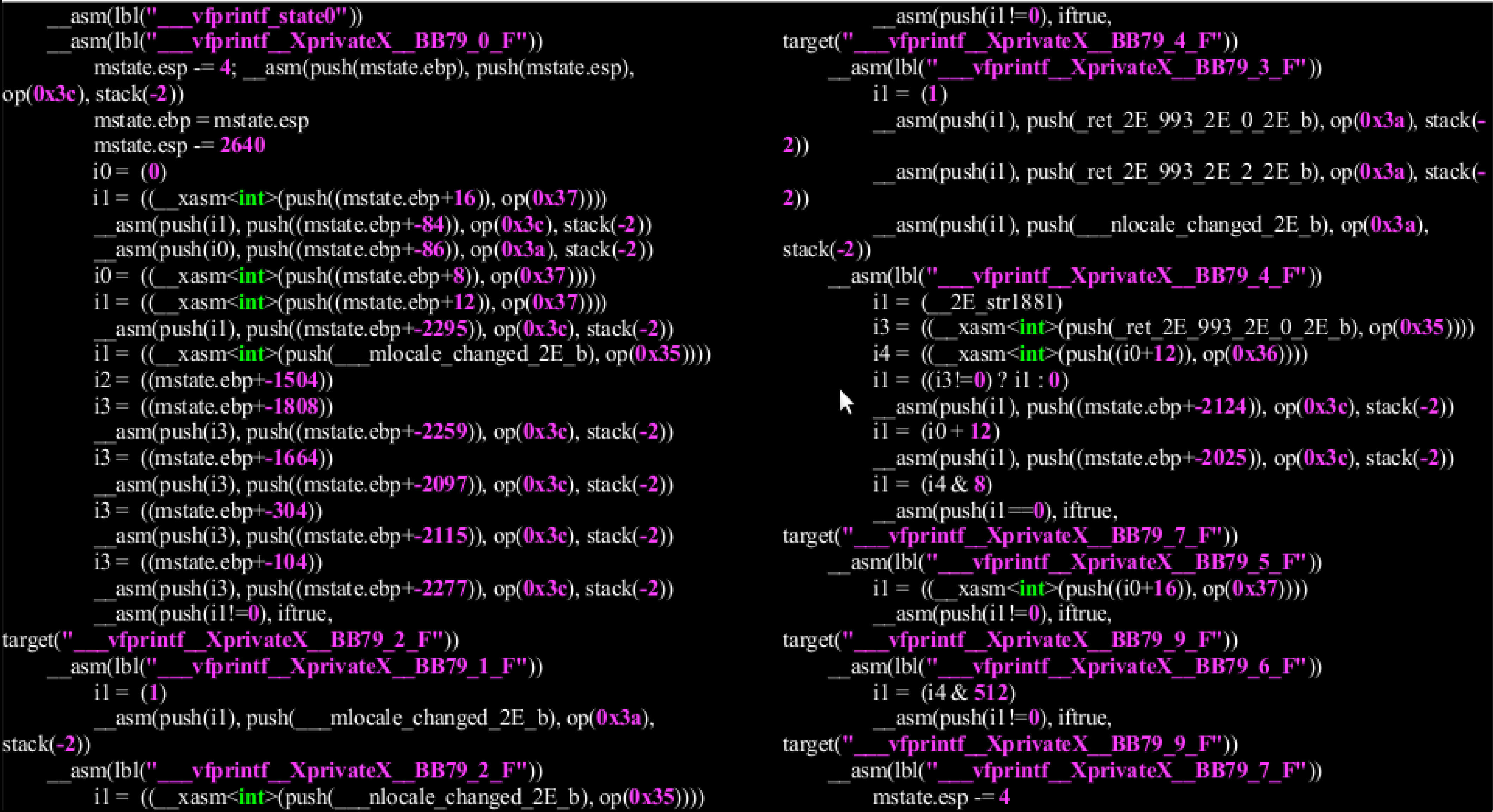
application**.bc**

LLVM

application**.abc**(swf)

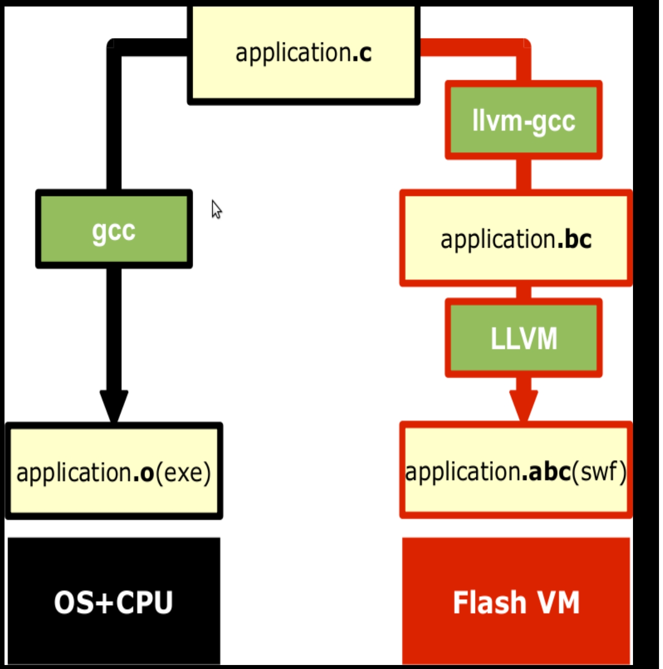
Flash VM

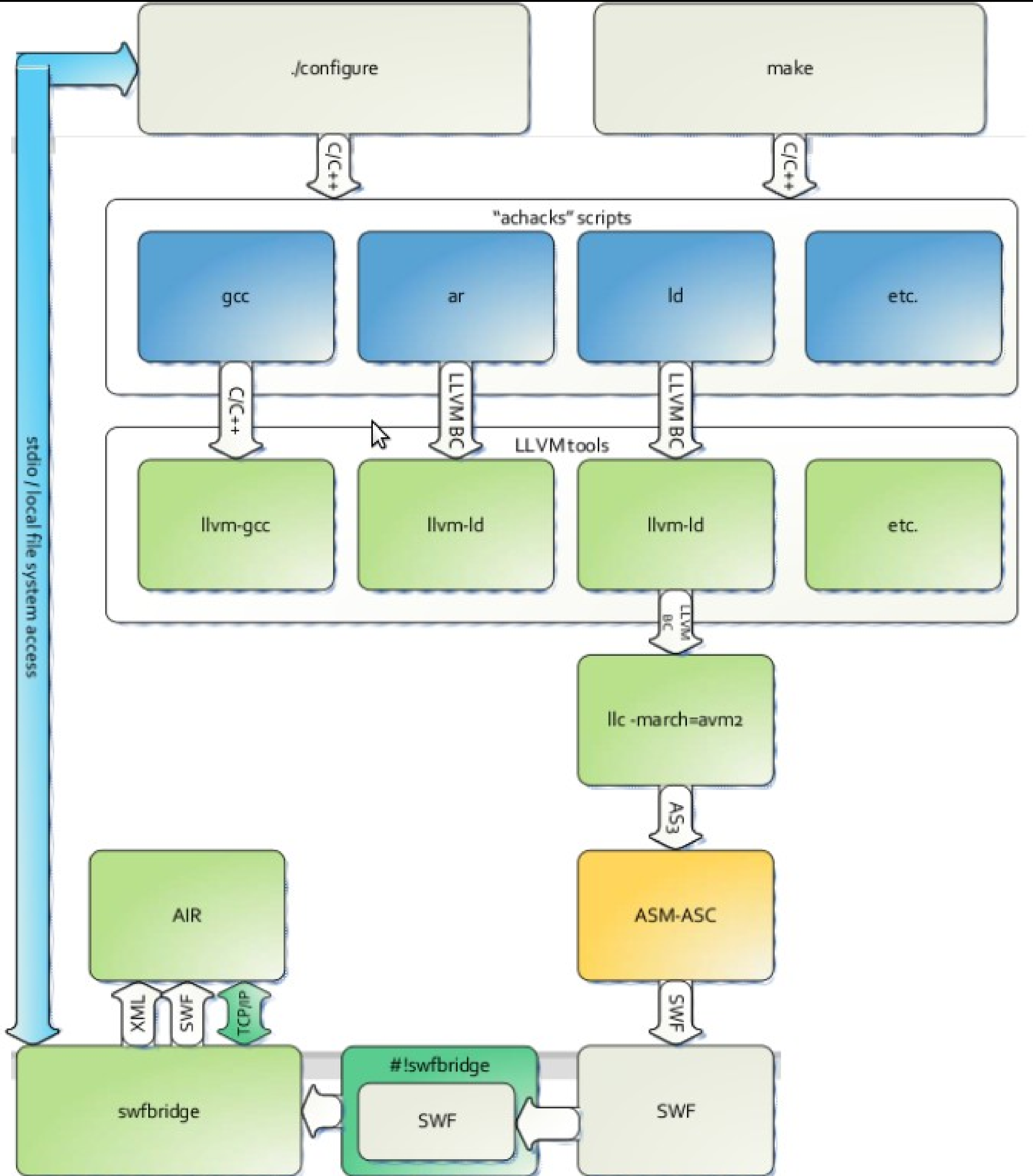
Example generated AS3



```
    __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
LLVMtools
llvm-gcc
llvm-ld
llvm-ld
etc.
LLVM BC
llc -march=avm2
AS3
ASM-ASC
SWF
AIR
XML
SWF
TCP/IP
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

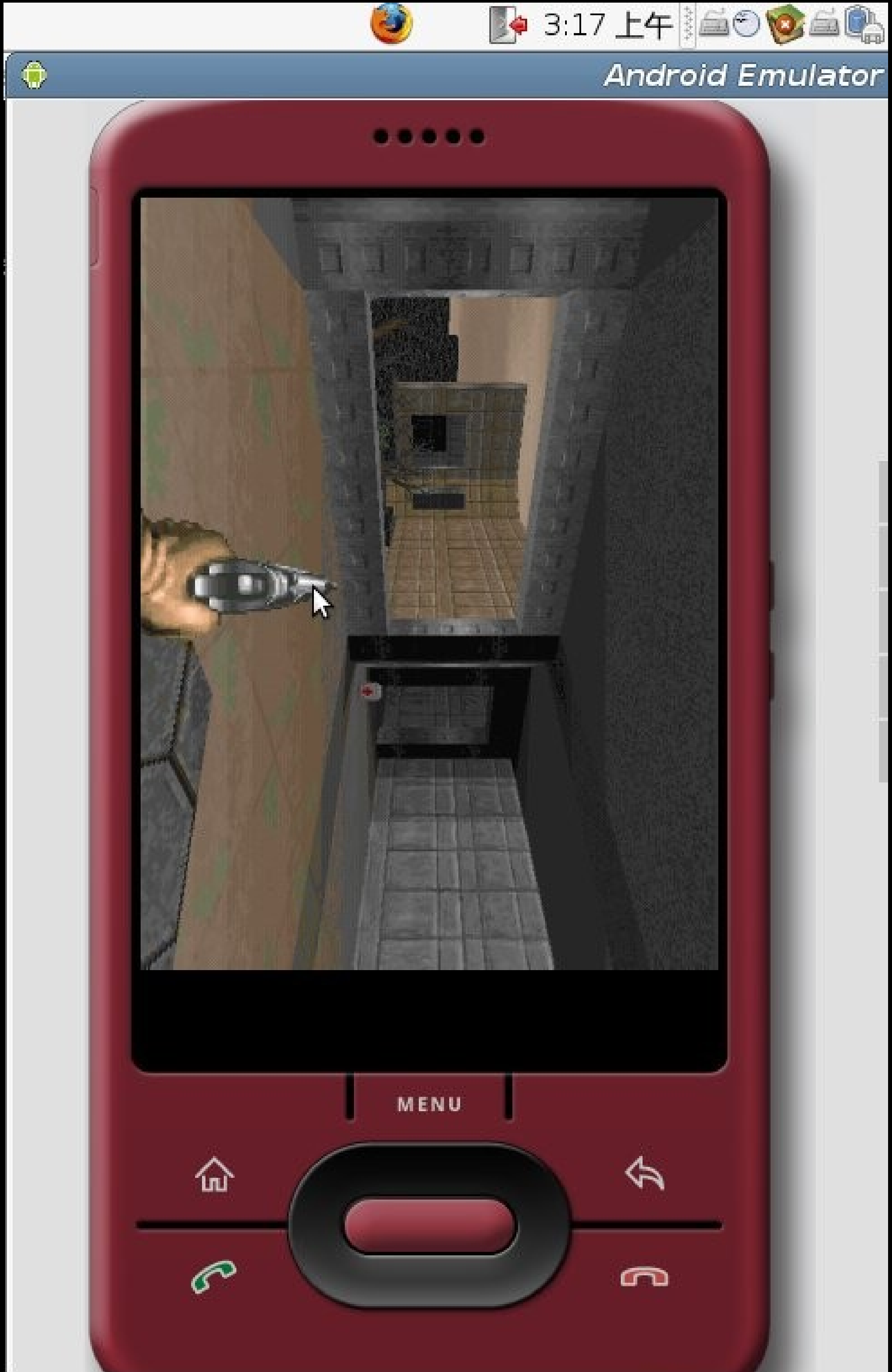
實驗性的 LLVM Backend

LLVM → Android Dalvik bytecode (DEX)

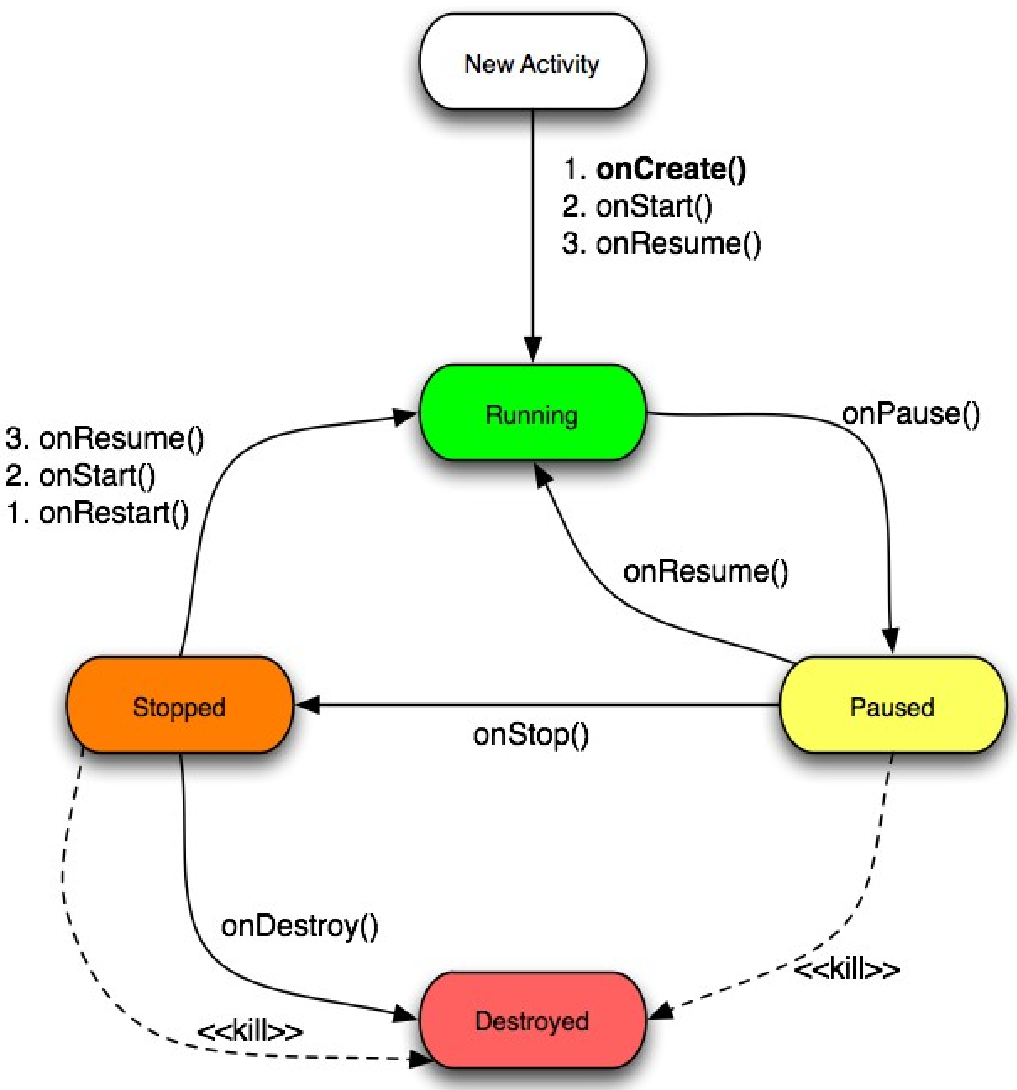
仿效 Adobe Alchemy ，目標平台則是 DalvikVM

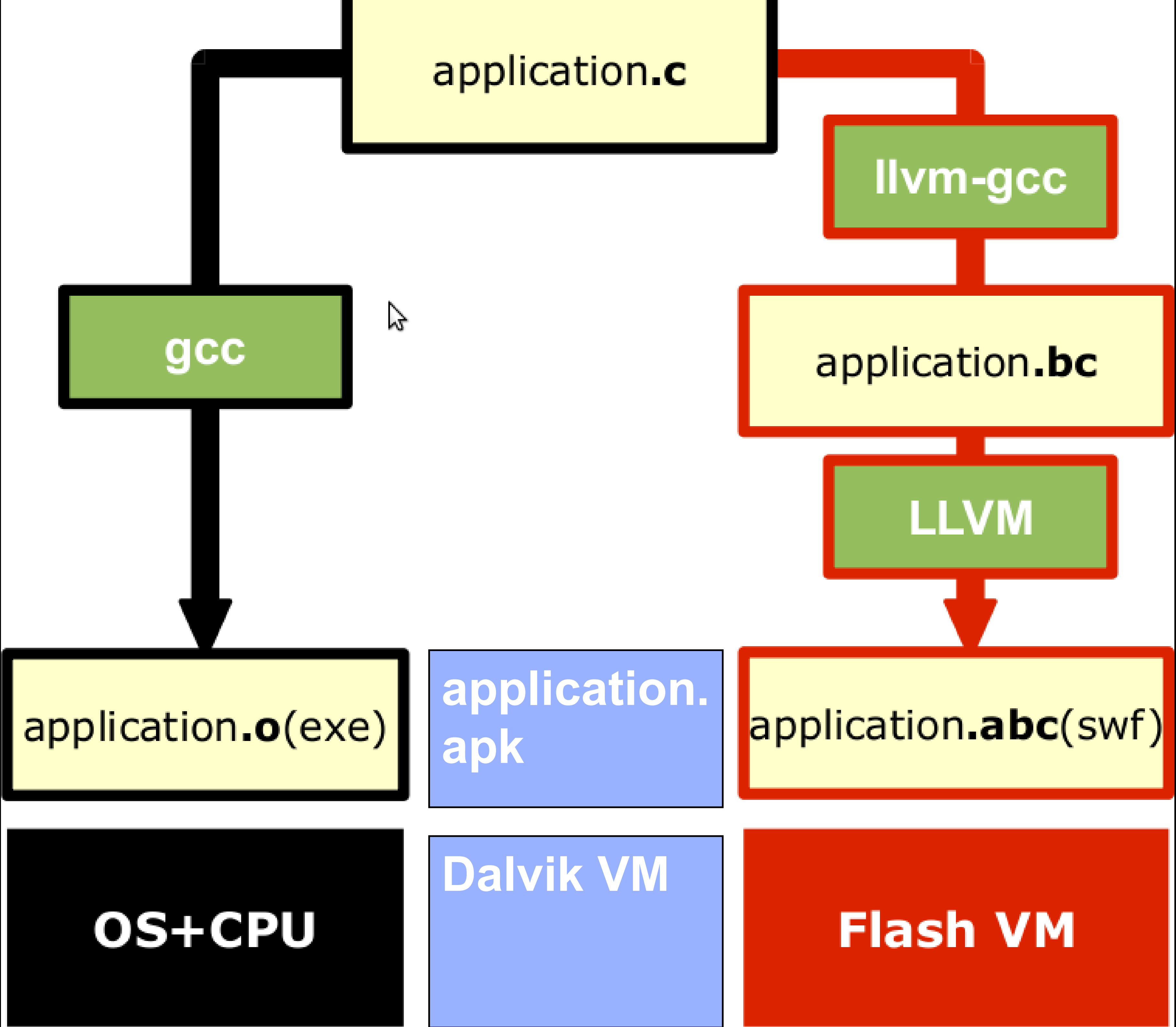
提供 C/C++ 原始碼 → LLVM → DEX bytecode 的執行環境，無 JNI (Java Native Interface) 介入

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



Activity Lifecycle





application.c

gcc

application.o(exe)

OS+CPU

application.apk

Dalvik VM

llvm-gcc

application.bc

LLVM

application.abc(swf)

Flash VM

技術大融通

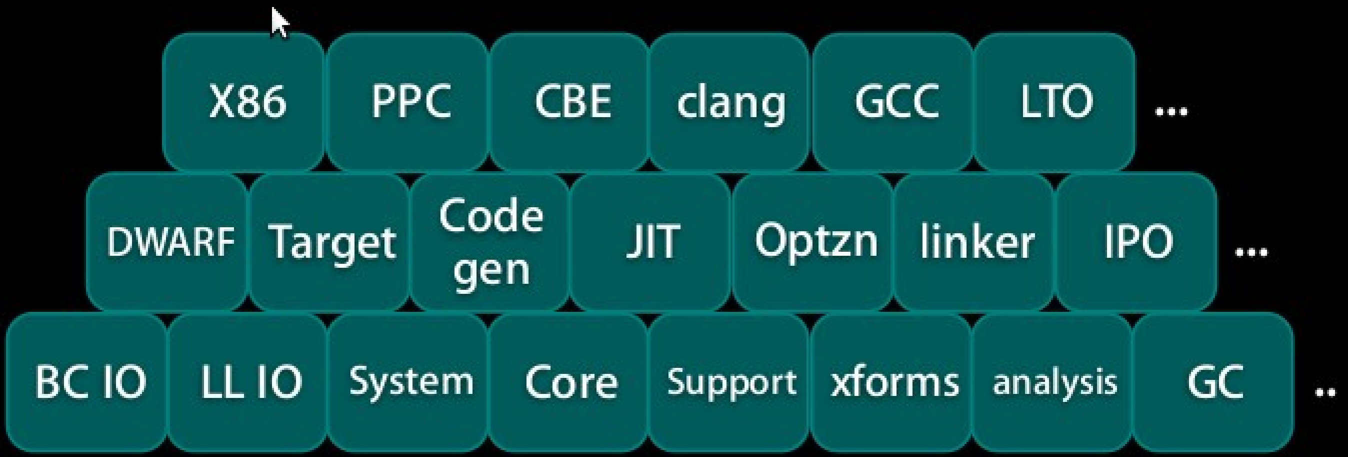
與

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

參考資料

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

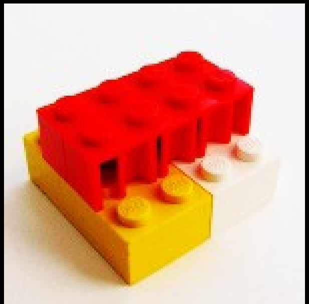
《美麗程式》(Beautiful Code), O'Reilly
第八章〈動態產生影像處理程式〉

LLVM – http://www.llvm.org/

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

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

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)
http://0xlab.org

感謝您！