



# Optimización de Aplicaciones Android\* para x86 Intel Architecture

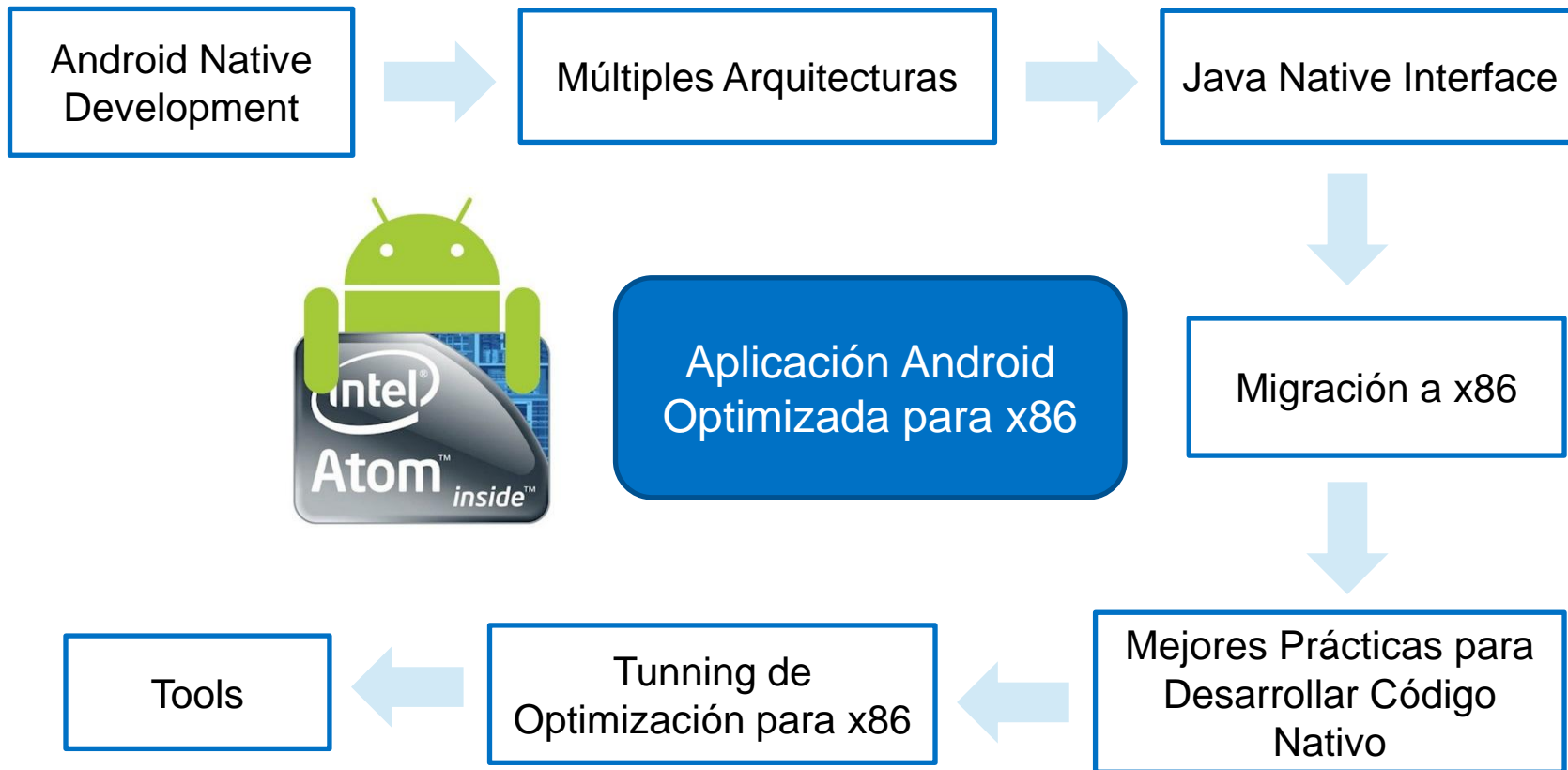
Albertina Durante

@albertinad16

ASDC - Argentina Software Design Center



# Roadmap



# Dispositivos Intel Atom

Smartphones 2012

Intel Atom Z2460  
1.6GHz Hyper-Threading



Orange San Diego



ZTE Grand X In



Motorola RAZR i



Lava Xolo  
X900



Megafon Mint



Lenovo K800

# Dispositivos Intel Atom

## Smartphones 2013

Intel Atom Z2420  
1.2 GHz



Safaricom Yolo



Lenovo K900



ZTE Grand  
X2 In

Intel Atom Z2580  
2 GHz dual-core



ASUS Fonepad  
Note FHD-6"



ZTE Geek 5"

# Dispositivos Intel Atom

## Tablets 2013

Intel Atom Z2460  
1.6 GHz dual-core



ASUS Fonepad 7

Intel Atom Z2560  
2.0 GHz dual-core



ASUS MeMO Pad FHD  
10



Samsung Galaxy  
Tab 3 10.1

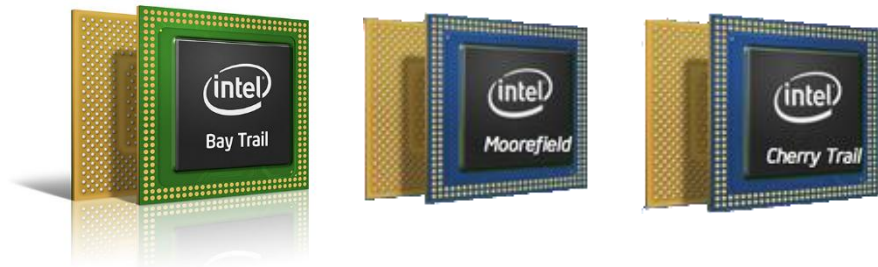


Dell Venue 7/8

# Dispositivos Intel Atom

## Nueva Generación

- Tri-gate microarchitecture 22nm
- Arquitectura Silvermont
- Intel Atom Processor Z3000 Series



Bay Trail

Tablets

Merrifield

Smartphones

Moorefield

- 14 nm microarchitecture

Arquitectura 64-bits  
Android 64 bits support

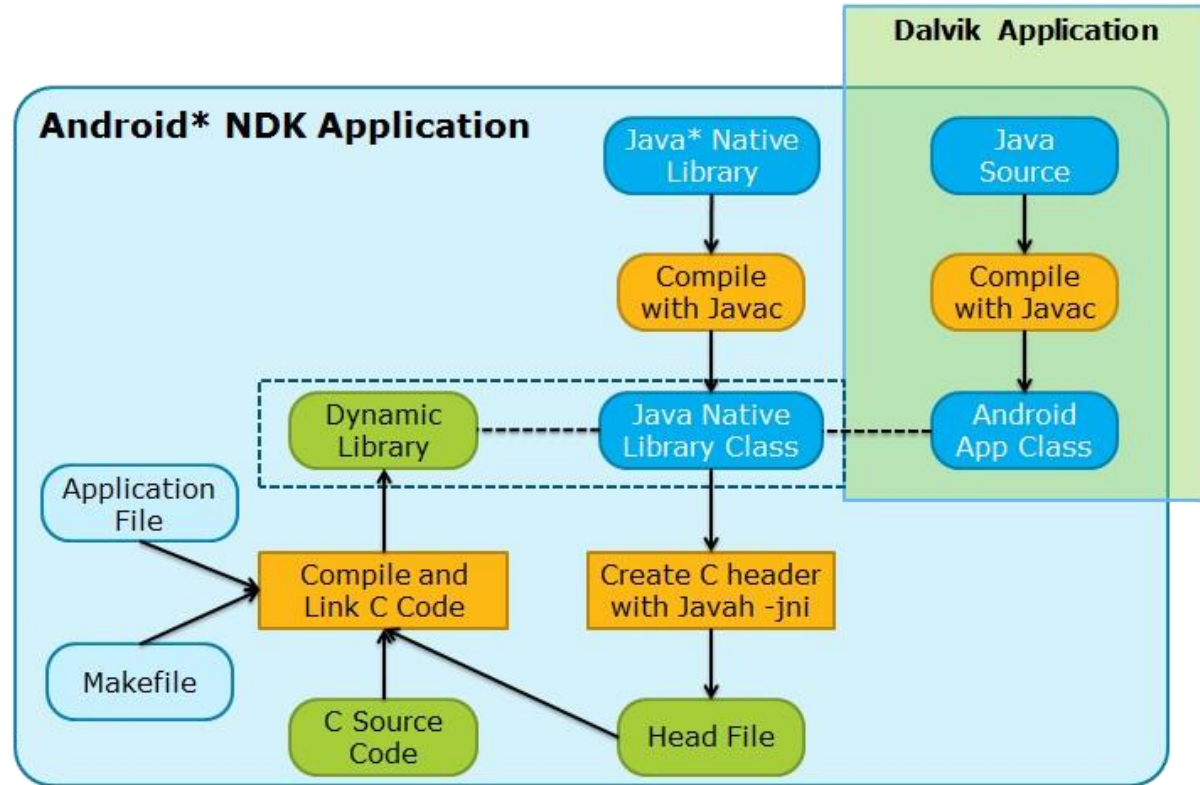
# Android Applications: APK

## Aplicaciones Dalvik

- Código Java
- Recursos: xml, imágenes
- Android SDK

## Aplicaciones NDK

- Código Java y Resources
- Código C/C++
- Código assembly
- Librerías dinámicas .so



# Aplicación NDK (Native Development Kit)

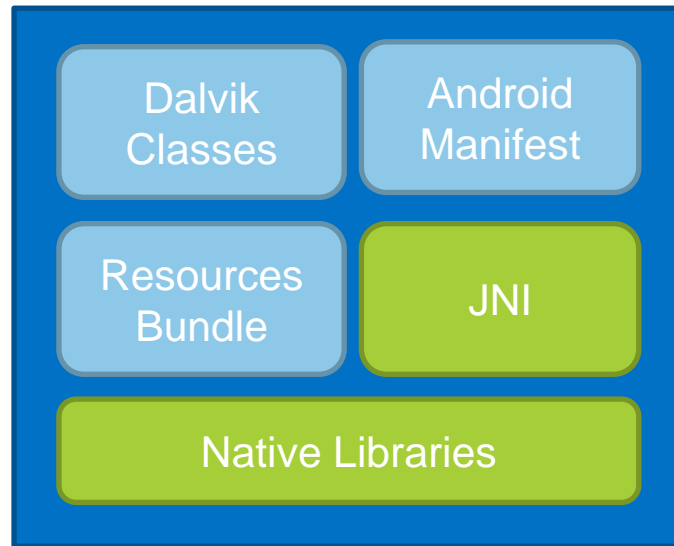
Aplicación Android empaquetada en un APK que utiliza librerías nativas

- Llamadas a funciones nativas de la aplicaciones
- Frameworks y librerías estaticas/dinámicas nativas

Librerías nativas se compilan en binarios “.so” en  
libs/CPU\_ABI

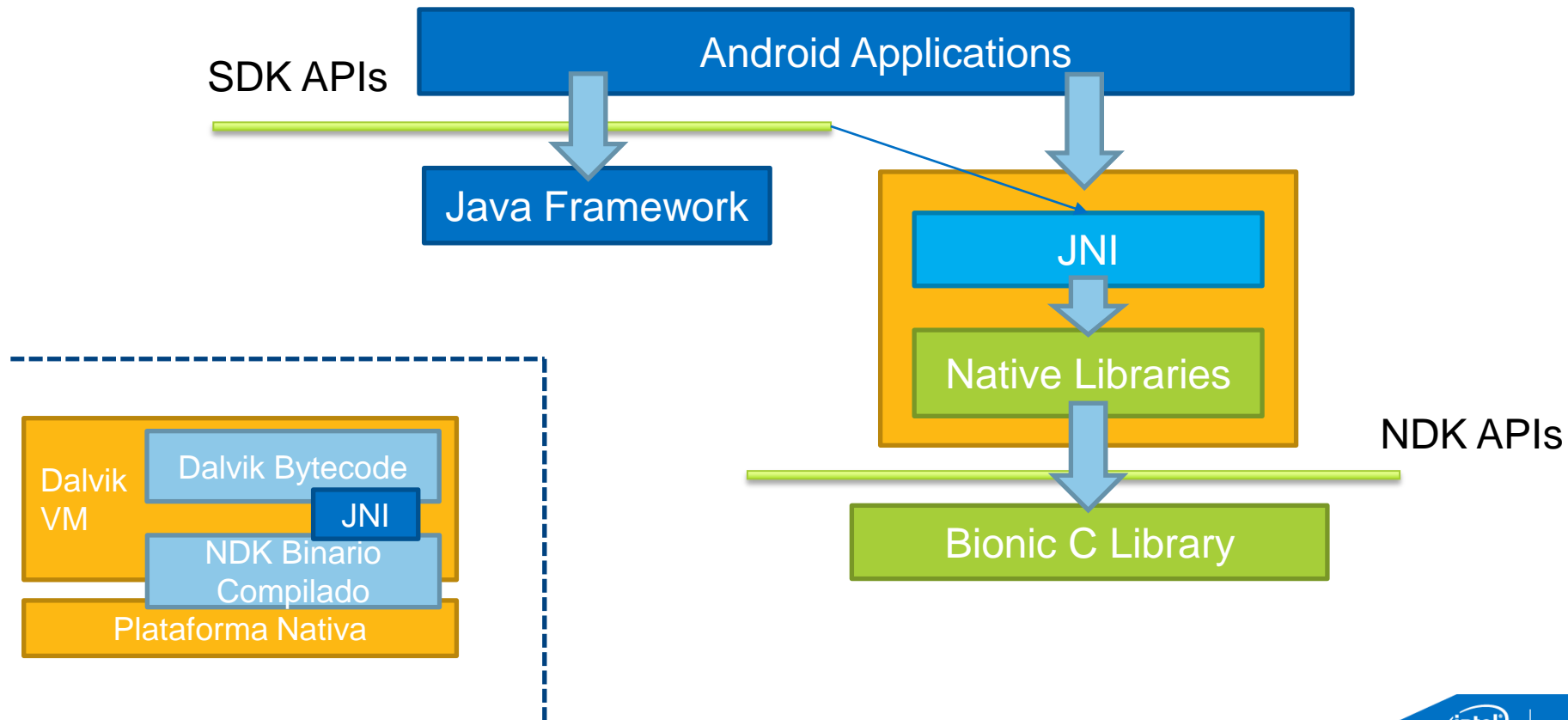
No existe una aplicación 100% nativa (C/C++ y assembly)

Utilidades: `native_app_glue`





# Desarrollo de Aplicaciones NDK



# NDK

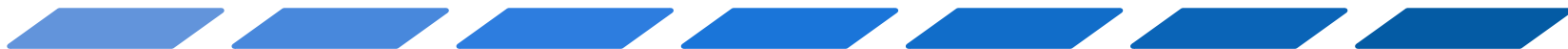
Toolset y build scripts que permiten implementar partes de una aplicación en código nativo como C/C++



Compilar código C/C++ a librerías y ejecutables nativos: específicos a la plataforma

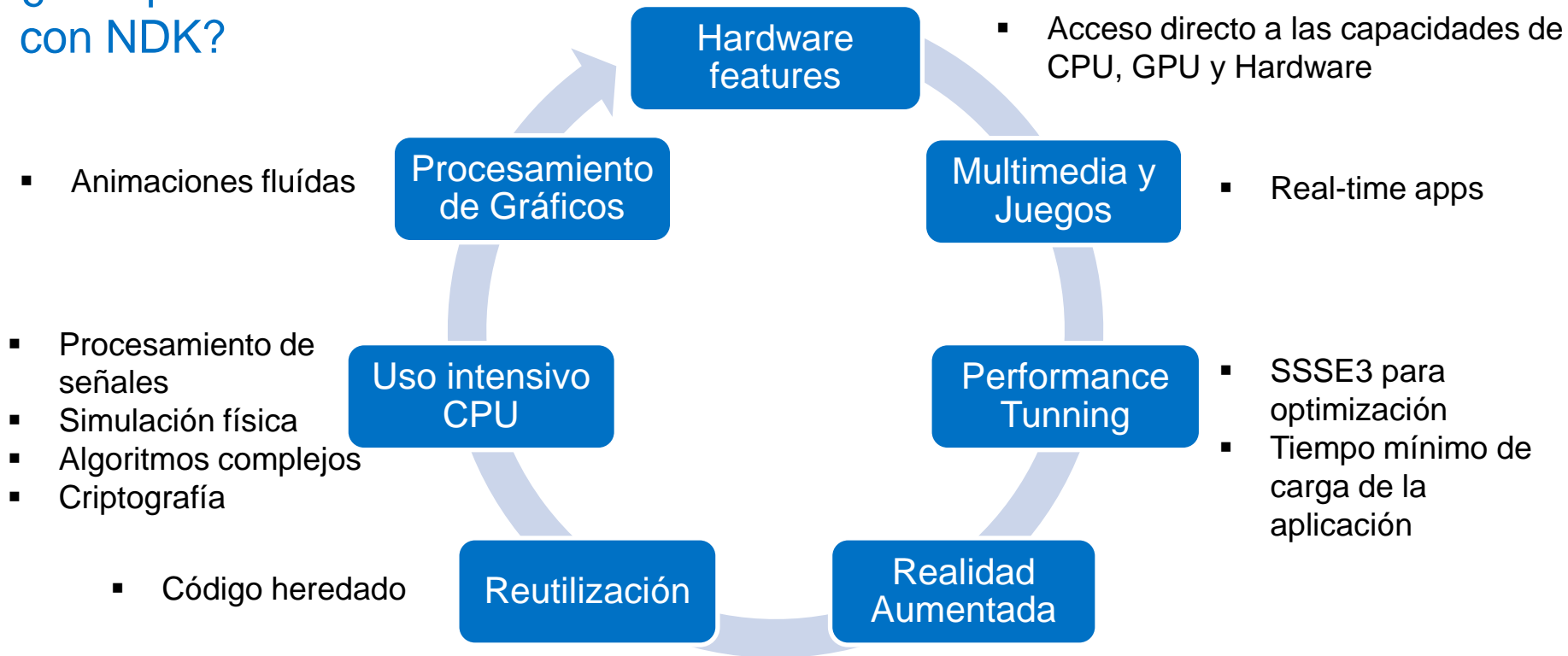


Se debe compilar para cada plataforma que a soportar: CPU\_ABI



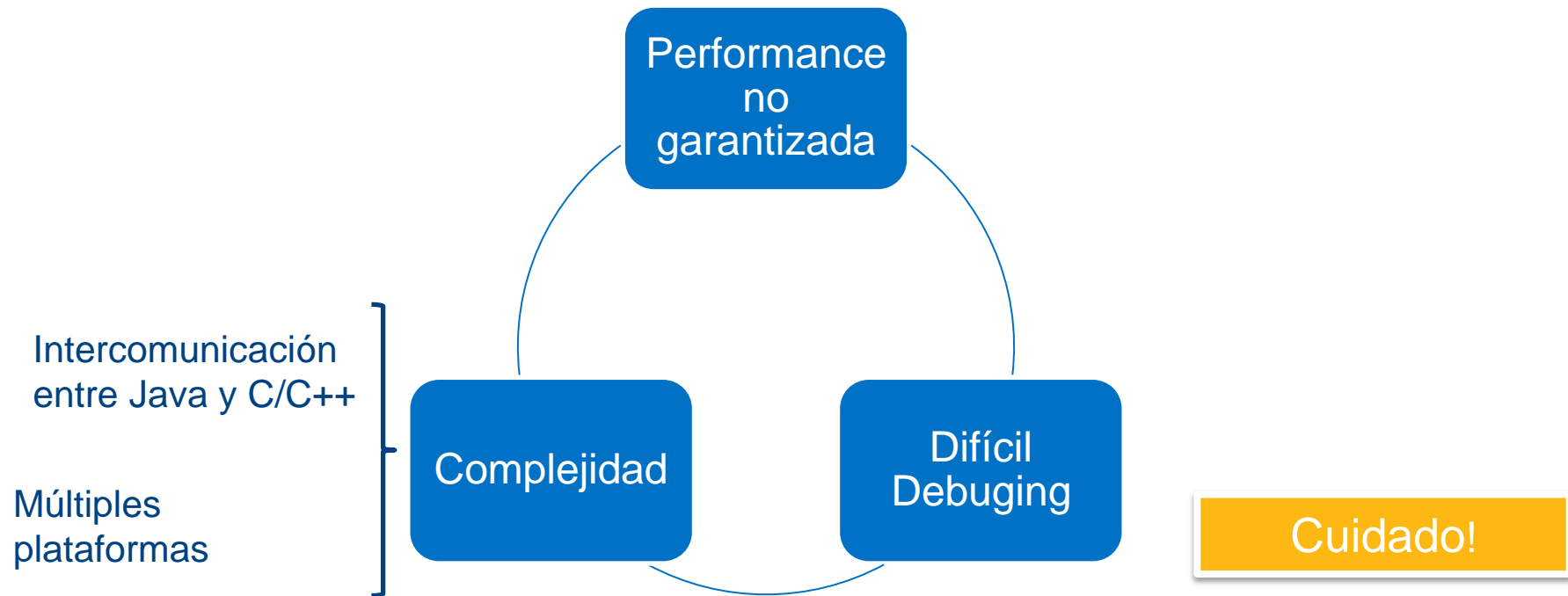
# NDK

## ¿Por qué desarrollar con NDK?

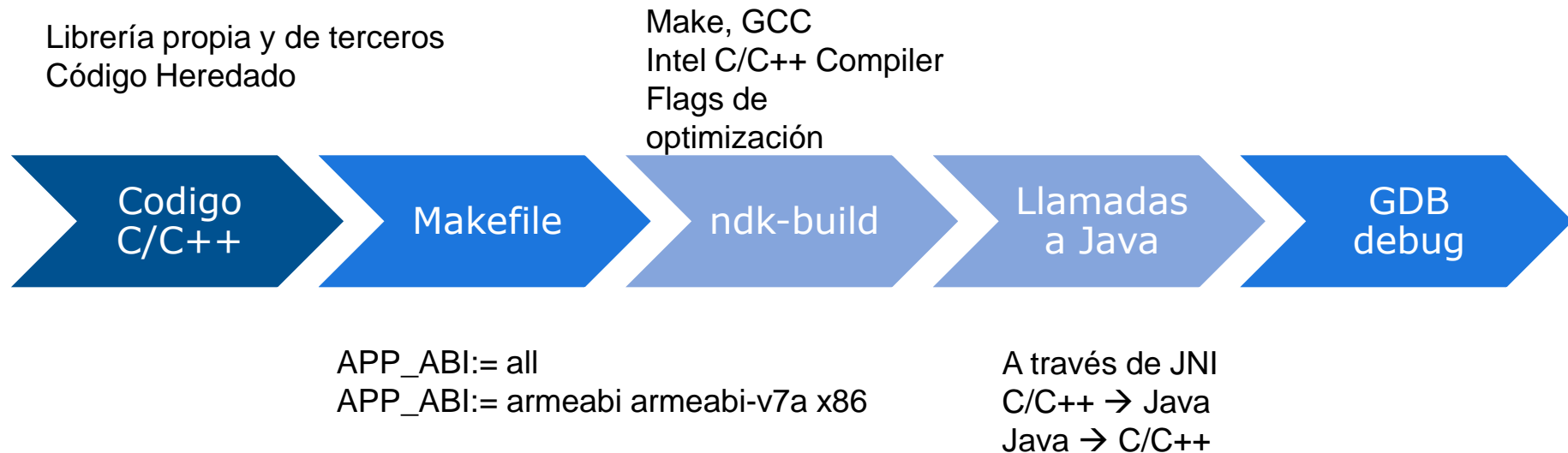


# NDK

¿Por qué no utilizar código nativo?



# Desarrollo de Aplicaciones NDK



# Java Native Interface

Framework que permite a código Java corriendo en una instancia de JVM ser llamado y llamar a aplicaciones y librerías nativas en lenguajes como C/C++

Permite a funciones nativas manipular objetos Java: código nativo accede a las features de Java VM a través de funciones JNI

Funciones JNI disponibles a través de interface pointers: puntero a puntero

# Funciones Nativas en Java

Funciones nativas se declaran con el keyword “native”

```
package com.intel.applatina.lib;  
public class NativeMessage {  
    public native String getMessage();  
}
```

Cargar la librería antes de utilizar

- System.loadLibrary
- System.load(<full\_path>)

```
static {  
    System.loadLibrary("nativemessages");  
}
```

Al compilar con el build script, se genera la librería dinámica compartida “libnativemessages.so”

¿Cómo asociar código Java y código Nativo? javah y  
JNI\_OnLoad

# Javah

- Herramienta que ayuda a generar los headers JNI a partir de una clase Java.
- Glue code que permite la interacción entre código Java y C/C++

El siguiente  
comando,

```
javah -classpath bin/classes/ -d jni/ com.intel.applatina.lib.NativeMessage
```

Genera el stub JNI con la siguiente definición para getMessage()

```
JNIEXPORT jstring JNICALL  
Java_com_intel_applatina_lib_NativeMessage_getMessage (JNIEnv *, jobject);
```



# JNI\_OnLoad

Método ejecutado cuando el ClassLoader instancia la clase java: Main entry point

- Enfoque recomendado para cargar libs nativas
- Evita errores cuando se hace refactoring de código
- Agregar y remover funcionalidad con mayor simplicidad y control
- Cachear referencias a objetos Java
  - FindClass en el contexto del ClassLoader
  - Crear variables y objetos, settear estado, no crear sesiones con servicios o base de datos

# Manejo de Excepciones

La ejecución de funciones Java pueden lanzar Exceptions

```
jthrowable javaException = (*env)->ExceptionOccurred(env);  
  
if (javaException !=NULL)  
{  
    (*env)->ExceptionClear(env);  
    // TODO: manejar la excepción  
}  
  
(*env)->DeleteLocalRef(env, javaException );
```

# Lanzar excepciones Java

Cuando la llamada a la función nativa retorna a Java se lanza la Exception

```
jclass clazz = (*env)->FindClass(env, "java/lang/Exception");  
  
if (clazz != NULL)  
{  
    (*env)->ThrowNew(env, clazz, "Java Exception from Native");  
}
```

# NativeActivity

En conjunto con la API nativa, permite desarrollar una aplicación Android escribiendo 100% código nativo

Librería estática **native\_app\_glue.h** para crear y administrar Activities nativas

```
<application android:label="@string/app_name" android:hasCode="false">
```

Indicar que no  
contiene código  
Java

```
<activity android:name="android.app.NativeActivity"  
  android:label="@string/app_name"  
  android:configChanges="orientation|keyboardHidden">
```

Agregar NativeActivity

```
<meta-data android:name="android.app.lib_name"  
  android:value="native-activity" />
```

Definir nombre de la librería  
nativa

```
<intent-filter>  
  <action android:name="android.intent.action.MAIN" />  
  <category android:name="android.intent.category.LAUNCHER" />  
</intent-filter>
```

```
</activity>  
</application>
```

```
#include <android_native_app_glue.h>
```

```
/**
```

```
 * This is the main entry point of a native application that is using  
 * android_native_app_glue. It runs in its own thread, with its own  
 * event loop for receiving input events and doing other things.  
 */
```

```
void android_main(struct android_app* state) { ... }
```

# Distribucion de APKs: Fat Binaries

Recomendado



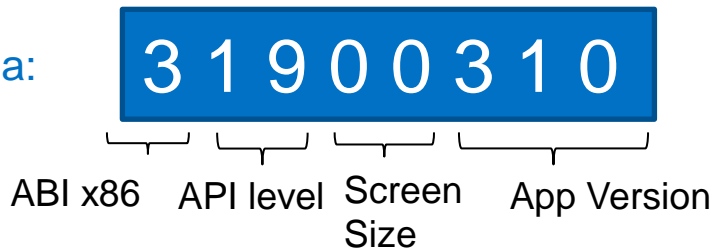
Filtrado durante el proceso de instalación

# Distribucion de APKs: Multiples APK

Un APK por Aquitectura → Mecanismo de filtrado para el dispositivo: AndroidManifest.xml  
android:versionCode



Convención recomendada:



# Migración hacia Android x86

# Android Apps: De ARM hacia Intel Atom



¿Cómo correr Apps en Android x86?



¿Toda aplicación es compatible?



¿Se debe migrar?



¿Es necesario optimizar?



# Soporte de NDK Apps en Intel Platforms

Gran parte de las Apps funcionan sin recompilar



Android NDK provee toolchain para x86 desde 2011

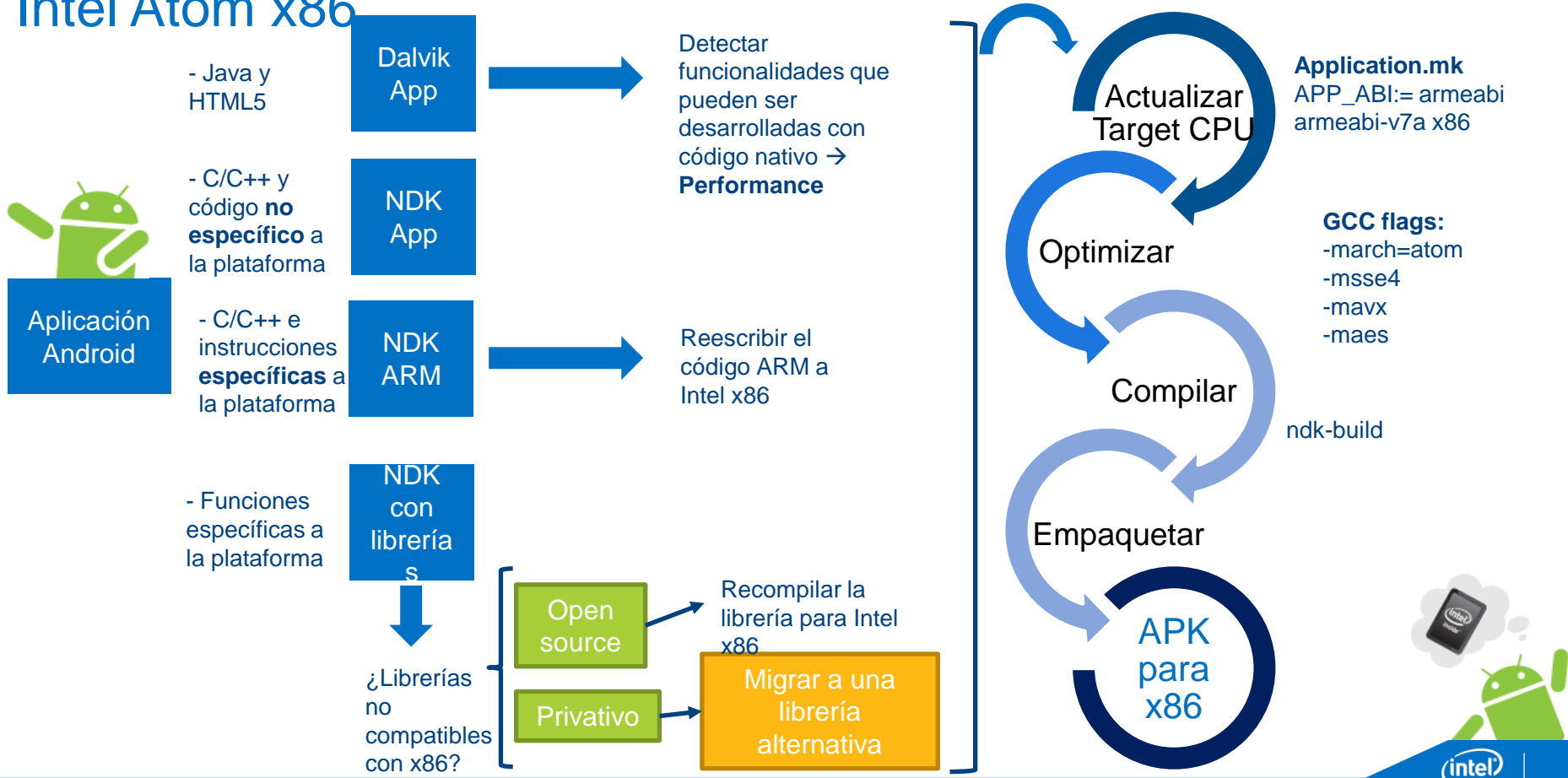


Recompilar para x86 mejora la performance



Código dependiente de la plataforma necesita reescribirse

# Cheet Sheet: Porting y Optimización de Android Apps para Intel Atom x86



# Intel Binary Translator

Como correr Apps nativas en Intel Atom sin esfuerzo?

Intel Binary Translator: Librería que traduce código nativo ARM a código nativo x86 en runtime

## RECOMENDADO:

Re-compilar con ABI x86 para generar librería dinámica para Intel Atom



- **APP\_ABI := armeabi armeabi-v7a x86** (o “all”) en **Android.mk**
- **ndk-build**

Ejecutar aplicaciones Android con librerías nativas para la arquitectura ARM en la plataforma de Intel Atom con performance acceptable



# Games Engines y Frameworks

# Games Engines

Soporte para x86

## Cocos2DX



- 2D Engine Cross-platform
- Open Source
- C/C++, JavaScript, Lua

## LibGDX

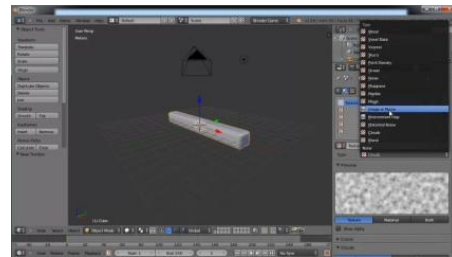
- 2D/3D Engine Cross platform
- Open Source
- Basado en C++ y Java
- Box2d physics



libGDX

# Project Anarchy

- Engine para juegos mobile, para las plataformas iOS, Android, Android x86 y Tizen, que incluye Havok Vision Engine, Physics, Animation Studio y AI
- Arquitectura C/C++ extensible basada en plugins
- Optimización para rendering mobile
- Lua scripting
- Sistema flexible de manejo de assets
- Tools de debugging



# Games Engines

Sin soporte para x86

## Corona SDK

- No posee soporte para x86 en la versión Free, pero las aplicaciones funcionan en la plataforma
- **Versión paga permite soporte para x86**

## Unity 3D

- No poseen soporte oficial y es privativo
- Para soportar x86 y optimizar, migrar engine a otras alternativas



# Frameworks

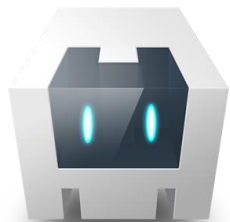
Soporte para x86

## Appcelerator

Entorno de desarrollo extensible para crear aplicaciones cross-platform con código base en HTML5 + JavaScript

- Appcelerator Platform: enterprise platform suite: APIs, Analytics, Build, Deploy
- Titanium:
  - open source framework cross-platform HTML5 + JS code base
  - Módulos extensibles: incluir librería nativa optimizada para x86

## Cordova



- Plataforma para desarrollar apps cross-platform con HTML5 + JS
- Accede a features nativas de la plataforma
- Basado en plugins: extensible

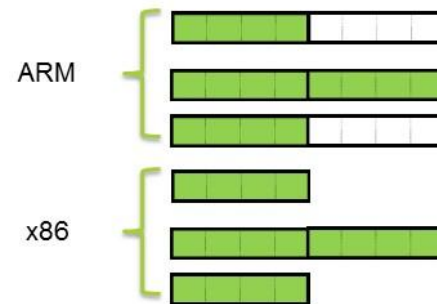


# Mejores Prácticas para Desarrollar Código Nativo

## Alineación de Memoria

Por default

```
struct TestStruct {  
    int var1;  
    long long var2;  
    int var3;  
};
```



Solución de Layout de Memoria ARM  $\leftrightarrow$  Intel Atom

- Agregar “malign-double” flag al compilador GCC
- Declarar atributo con `__attribute__((aligned(8)))`

```
struct TestStruct {  
    int var1;  
    long long var2 __attribute__((aligned(8)));  
    int var3;  
};
```



# Mejores Prácticas para Desarrollar Código Nativo

## Porting de instrucciones ARM NEON a instrucciones Intel SSE

### Intel SSE

- Streaming SIMD Extension, equivalente de Intel Architecture a ARM NEON
  - SS3, SSE2, SSE3, SSSE3 (Supplemental Streaming SIMD Extension 3)
- La mayoría de las funciones de NEON tienen una equivalencia 1:1 con Intel SSE
- Implementar código SIMD
  - Funciones intrínsecas C/C++ language-level: intercambiables con ARM NEON
- NEON provee librerías C nativas, deben reescribirse para ser compatibles con x86
- Intel provee un header C++ con mapping de funciones entre NEON y SSE para los desarrolladores
- *NEONtoSSE.h*

# Optimización para Android x86

# Optimización de la Performance

Velocidad de  
Ejecución

Tamaño del  
Código

Consumo de  
Energía



Enfoques de optimización

Automático por el Compilador

Asistencia por Tools de Desarrollo

Manual por el Developer

# Vectorización

## Loop-unrolling y generación avanzada de instrucciones SIMD

- Tarea manual del developer: no es escalable e implica costo de adaptación para cada arquitectura
- Auto-vectorización realizada por el compilador

```
APP_CFLAGS := -O3 -xSSSE3_ATOM -vec-report3
```

## Pragmas

Assumptions Conservativas con `__restrict__` es costoso en términos de mantenimiento

Intel SIMD pragma → simplificar vectorización

Instrucciones SIMD desactivado  
en Kernel Mode: `-mno-sse`

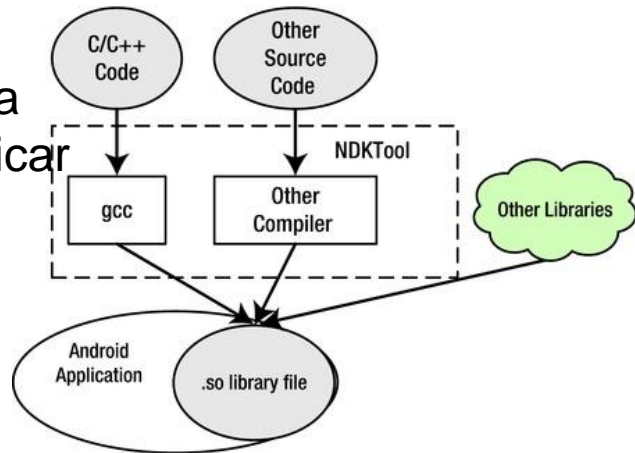
# Compiladores para NDK

## GNU GCC

- Cross-compiling: compatible con compilación nativa
- Switch de código compilado nativamente sin modificar código cross compilado

## Intel C/C++ Compiler

- Utiliza features de la plataforma x86
- Código optimizado resulta mucho mejor



Flags de optimización: independientes de la plataforma y asociadas a la plataforma

# GCC Compiler: Flags de Optimización

Opciones de compilación independientes a la plataforma

-O0, -O ó -O1, -O2, -O3

Opción m\* de GCC es para la familia de procesadores Intel i386 y x86-64

## Código para tipo específico de CPU

- -march=cpu-type
- -mtune=cpu-type

## Vectorización automática

- -msse, -msse2, -msse3, -mssse3, -msse4.1, -msse4.2, -msse4
- -mmmx
- -mno-sse, -mno-sse2
- -mno-mmx

## Código generado para arqu 32/64

- -m32-m64

# Targets de Compilación

## Proceso de Build de NDK:

- Se evalúa el make file Android.mk para cada arquitectura
- *TARGET\_ARCH\_ABI: arquitectura actual*

```
ifeq ($(TARGET_ARCH_ABI),x86)
LOCAL_CFLAGS      := -mtune=atom -mssse3
endif

ifeq ($(TARGET_ARCH_ABI),armeabi-v7a)
LOCAL_CFLAGS := -march=armv7-a
Endif
```

## TARGET\_ARCH\_ABI

- x86
- armeabi
- armeabi-v7a
- mips



# Recomendación para GCC Compiler

## Nivel de Optimizacion:

- `-O2` o superior, `-Ofast` for peak

## Arquitectura:

- `-march=atom -mtune=atom -mssse3` para Atom
- `-march=slm -mtune=slm -msse4.2` para Silvermont

`-march=atom` activa `-mmovbe` no soportado en todas las plataformas x86.

Para evitar agregar `-mno-movbe`.

## Math:

- `-ffast-math` – Más rápido pero menos preciso
- `-mfpmath=sse` – Usar SSE para cálculos FP en lugar de i387

## Mayor Performance

- `-flto`
- `-funroll-loops`

# Recomendación para Intel C/C++ Compiler

## Nivel de Optimizacion

- `-O2`, `-fast` for peak (implica-static)

## Arquitectura

- `-xATOM_SSSE3` para Atom
- `-xATOM_SSE4.2` para Silvermont

## Math:

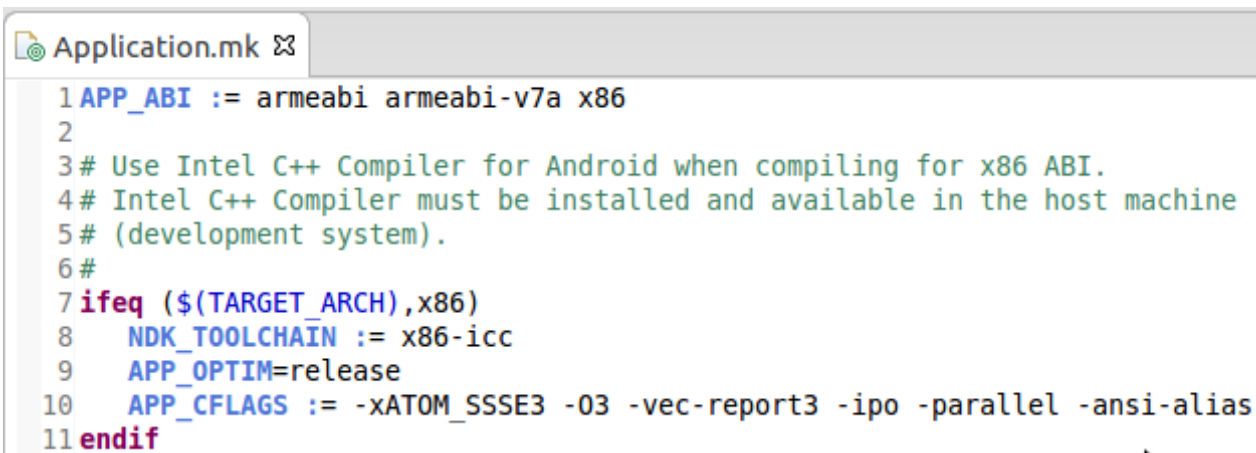
- `-no-prec-div` – Más rápido pero menos preciso
- `-mfpmath=sse` – Usar SSE para cálculos FP en lugar de i387

## Mayor Performance

- `-O3`
- `-ansi-alias`
- `-ipo`
- `-auto-p32`
- `-parallel`

# Intel C++ Compiler también genera binarios para ARM?

No, solamente genera binarios para x86



```
Application.mk
1 APP_ABI := armeabi armeabi-v7a x86
2
3 # Use Intel C++ Compiler for Android when compiling for x86 ABI.
4 # Intel C++ Compiler must be installed and available in the host machine
5 # (development system).
6 #
7 ifeq ($(TARGET_ARCH),x86)
8     NDK_TOOLCHAIN := x86-icc
9     APP_OPTIM=release
10    APP_CFLAGS := -xATOM_SSSE3 -O3 -vec-report3 -ipo -parallel -ansi-alias
11 endif
```

# Testing en Múltiples Dispositivos

# Testing en Múltiples Dispositivos

## Emuladores

- Hardware del device Host
- Bundle con SDK, Intel y Third-party
- Emulación: no todas las features soportadas

Android SDK  
Intel HAXM  
Third-party: Genymotion

## Cloud-based Testing

- Dispositivos Reales en el Cloud
- Tests corren en todos los dispositivos
- No permite teste de Usabilidad



AppThwack

## Dispositivos Reales

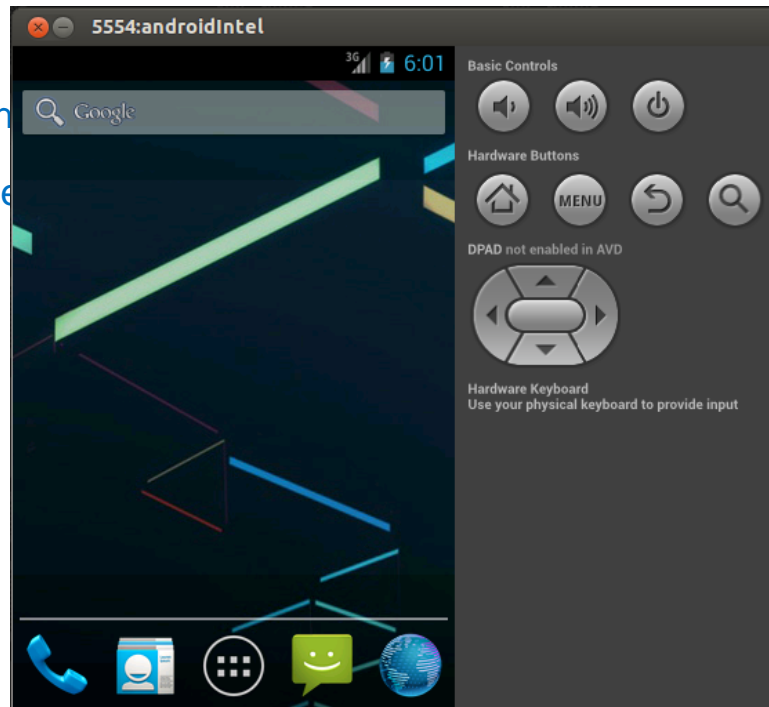
- Tests de Usabilidad y UX
- Idealmente
- Costoso

Nexus, Samsung,  
Motorola, LG,  
Lenovo, y más

# Intel® Hardware Accelerated Execution Manager

Motor de virtualización asistido por hardware (Hypervisor) basado en la tecnología Intel Virtualization Technology (Intel® VT) para acelerar la emulación de apps Android en una máquina host

- Integrado a Android SDK
- Usuarios de linux descargan el zip de Intel Developer Zone
- Máquina de desarrollo con procesador Intel con soporte de
  - Intel VT-x habilitado
  - Intel Disable Execute Bit habilitado
- Soporta la emulación de Dalvik y NDK apps

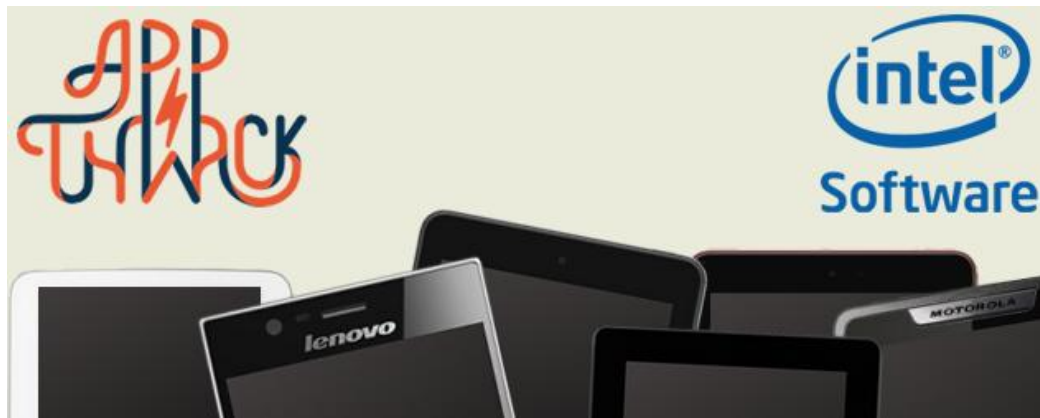


# AppThwack

## Testing de aplicaciones Android en dispositivos Intel Atom en el Cloud

Dispositivos incluidos:

- Asus MeMO Pad FHD 10
- Dell Venue 7
- Dell Venue 8
- Lenovo IdeaPhone K900
- Motorola Droid RAZR i
- Samsung Galaxy Tab 3 10

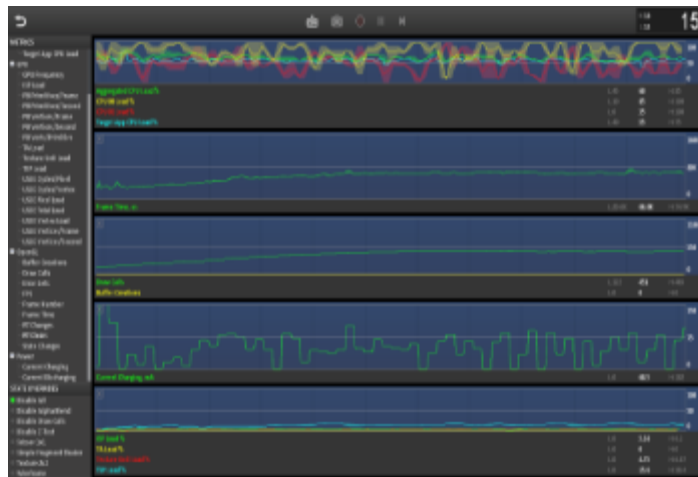


# Tools de Intel para Android x86



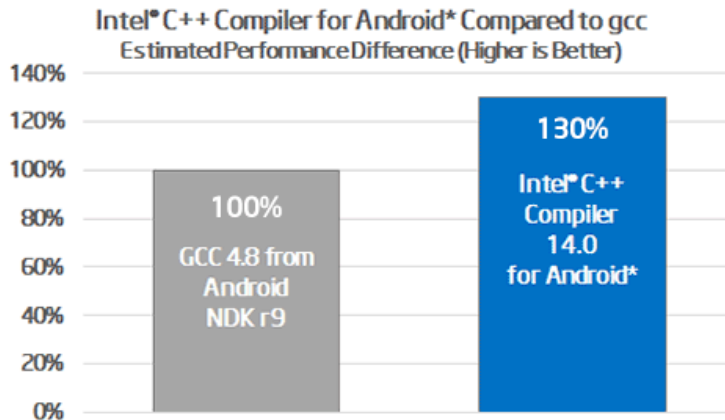
# Intel® Graphics Performance Analyzer Tool

- Análisis de performance en tiempo real a nivel de sistema para dispositivos basados en Android x86
- Permite al desarrollador realizar experimentos y aislar problemas de performance de CPU y GPU
- Métricas de CPU, GPU, API, memoria, red, alimentación, etc



# Intel® C++ Compiler for Android

- Basado en Intel® C/C++ Compiler XE 14.0 for Linux
- Compatible con GNU\* C++ en Android NDK para soporte de múltiples arquitecturas
- Integrado a Android NDK como toolchain adicional
- Soporte de optimización para Intel® Atom™
- Aumento de performance sólo con recompilar
- Disponible para Windows\*, OS X\* y Linux\*
- Soporte de command-line para Eclipse\*
- 79.95\$



Aumento de 30% de la performance en aplicaciones de cómputo de integers intensivo



# Intel® Threading Building Blocks

Librería de C++ Templates para ejecutar eficientemente código paralelizable y obtener la mejor performance en arquitecturas multicores

# Intel® Integrated Performance Primitives

Intel® IPP es una librería extensiva de funciones (building blocks) optimizadas para el procesamiento de multimedia y datos

# Intel® Beacon Mountain

Framework de desarrollo de aplicaciones Android nativas para dispositivos basados en las plataformas x86 y ARM

## Intel® Integrated Native Developer Experience

Suite de desarrollo nativo cross-platform (Intel Architecture y ARM) con una performance nativa, battery-life y acceso a las capacidades de la plataforma



- Herramientas nativas para C/C++ y Java
- Herramientas integradas a IDEs populares
- Ejemplos para Android y Microsoft Windows

Intel®  
INDE

Solución Cross-platform para desarrollar aplicaciones web e híbridas una sola vez y distribuirlas en múltiples app stores y form factors

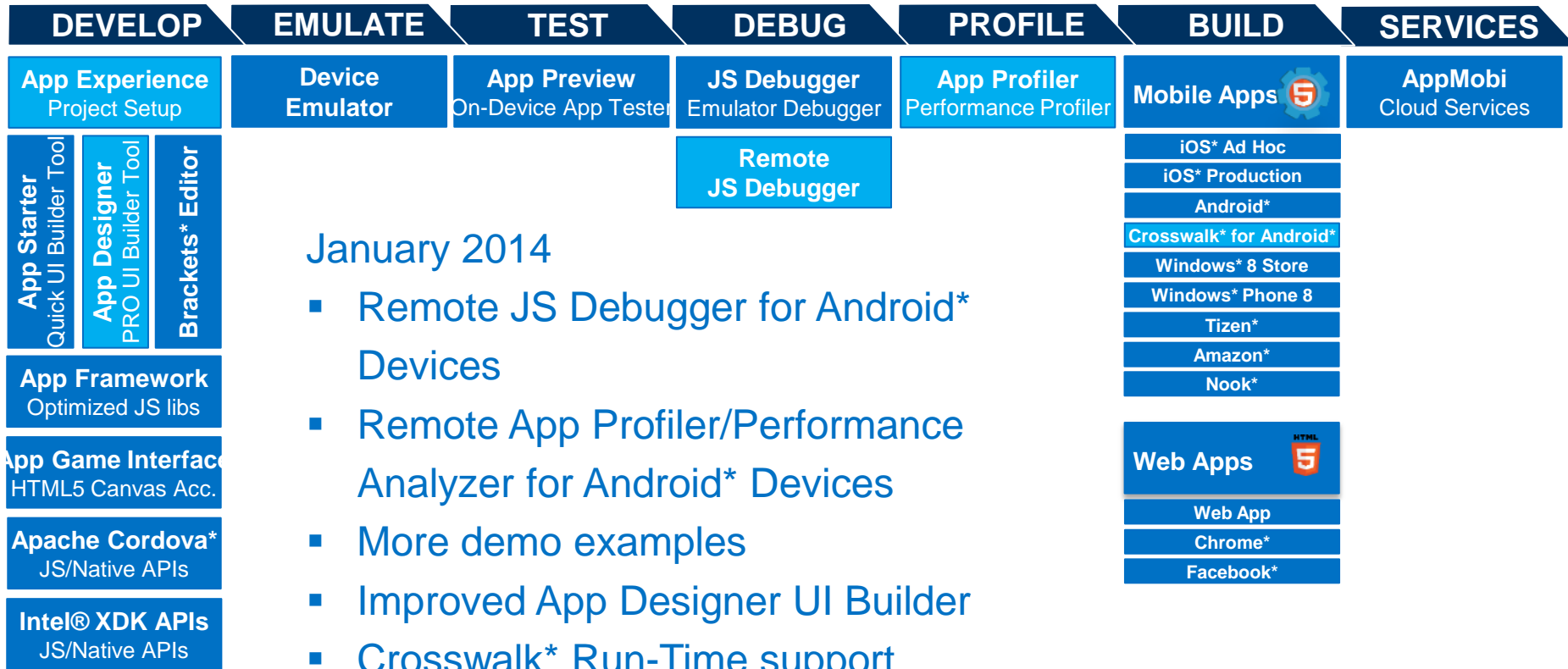


- **Crosswalk\* web runtime para Android\***
  - Extender capacidades de apps híbridas
- Cloud-based building system
- Apps para múltiples app stores
- Publicar a múltiples app stores y a form factors de forma sencilla y rápida

- Brackets Editor
- Apache Cordova
- AppFramework
- Apache Ripple Emulator



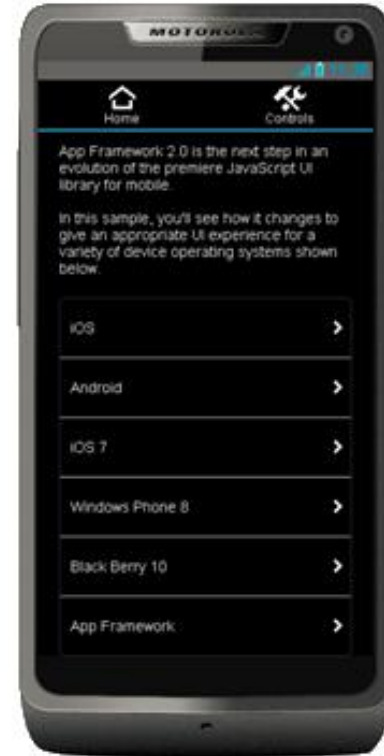
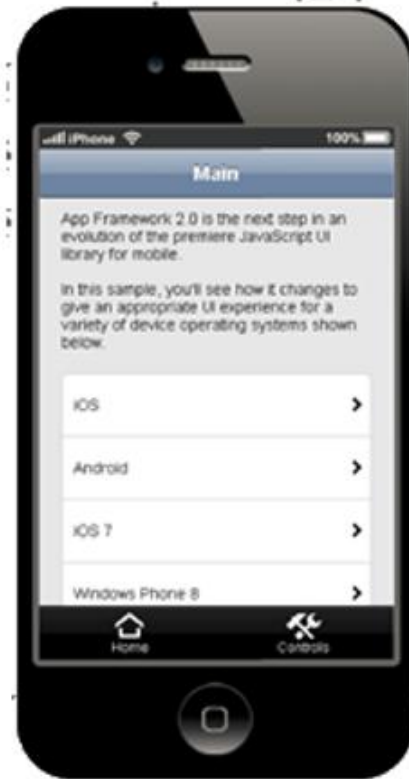
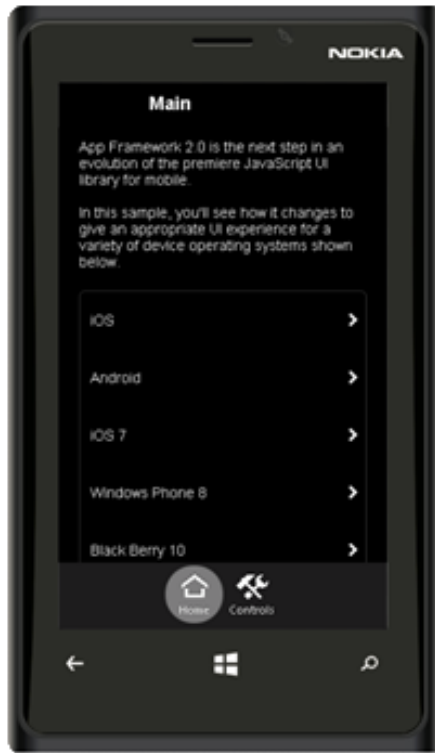
# Intel® XDK Architecture



New feature added

Windows\*, OS X\*, and Linux\* host OS support

# Build your HTML5 app with a native look and feel using App Framework UI/UX libraries



# Build

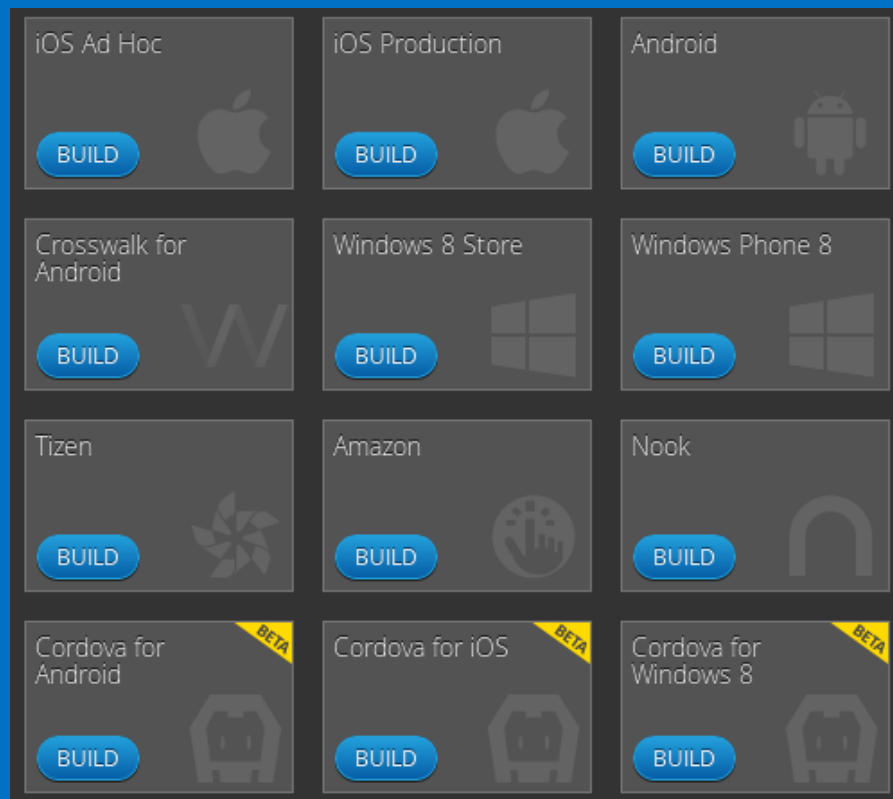
## Mobile App

Native app package ready for submission to the Store

HTML5 Hybrid containers:

- Standard Intel XDK
- Apache Cordova
- Crosswalk for Android
- Crosswalk for Tizen

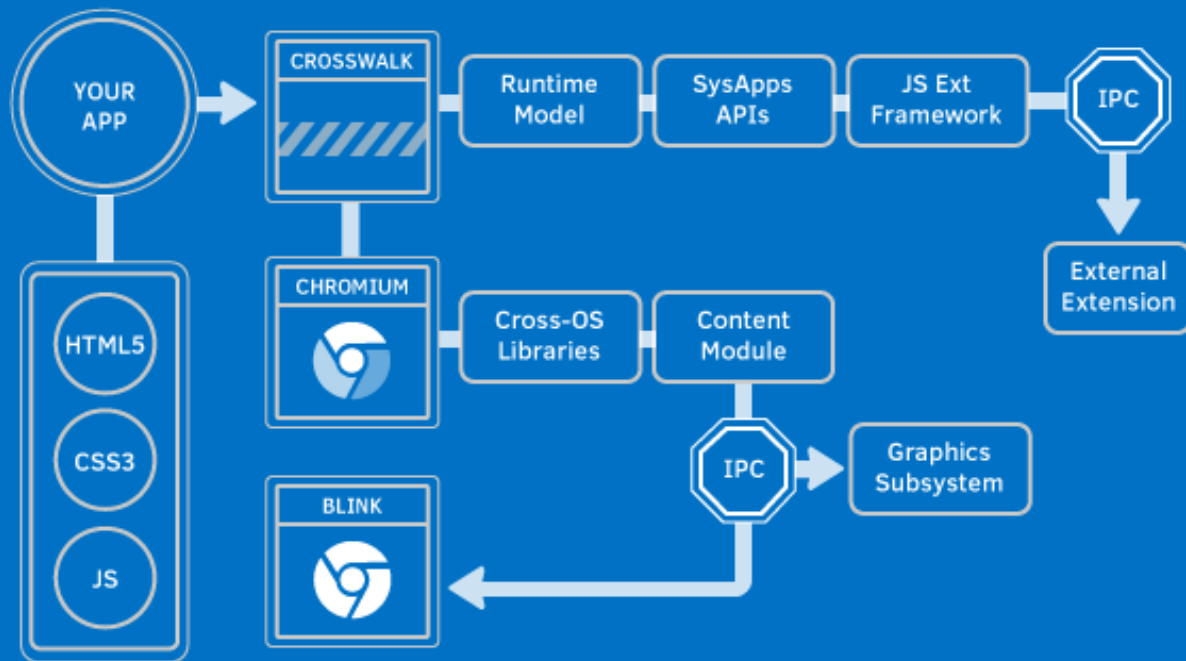
Package building process on the Cloud





# Crosswalk

## Crosswalk



Web runtime for ambitious HTML5 applications.

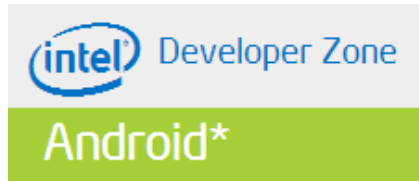
Extends the features of a modern browser with deep device integration

API for adding native extensions

# Documentación y Links

Presentación Completa:

<https://software.intel.com/es-es/articles/optimizaci-n-de-aplicaciones-android-para-arquitectura-x86>



<http://software.intel.com/es-es/android>

<http://software.intel.com/es-es/articles/android-application-development-and-optimization-on-the-intel-atom-platform>

<http://software.intel.com/en-us/blogs/2012/12/12/from-arm-neon-to-intel-mmxsse-automatic-porting-solution-tips-and-tricks>

<http://software.intel.com/en-us/android/articles/ndk-android-application-porting-methodologies>

# Documentación y Links

**eBook:** Android on x86 An Introduction to Optimizing for Intel® Architecture

<http://software.intel.com/en-us/blogs/2014/03/19/free-ebook-download-from-apress-android-on-x86-an-introduction-to-optimizing-for>

<http://gcc.gnu.org/onlinedocs/gcc/i386-and-x86-64-Options.html>

<http://software.intel.com/en-us/html5/home>

**Procesadores Intel con soporte VT-x**

<http://ark.intel.com/es/Products/VirtualizationTechnology>

# Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Copyright © , Intel Corporation. All rights reserved. Intel, the Intel logo, Xeon, Xeon Phi, Core, VTune, and Cilk are trademarks of Intel Corporation in the U.S. and other countries.

## Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804



