Androidを
ICEで追いかけてみた(1)

京都マイクロコンピュータ
小林哲之

自己紹介

- ずっと組み込みシステムのエンジニア
- 京都マイクロコンピュータとしては新入社員
- 前職では組み込み向けのJavaVM
- そのまた前職ではリアルタイムOSなどをやってました。
- CE Linux Forumのテクニカルジャンボリーで時々話をしています。

KMC

ICEとは

- In-Circuit Emulator
- 元々はCPUをソケットから抜いて、そこにCPUと同等の動作をするデバッガを差した。
- 現在はJTAG インタフェースを使用してCPU内部のデバッグ支援機構を利用する。
- CPUの実行制御、メモリやレジスタの読み書き、実行履歴の取得などができる。

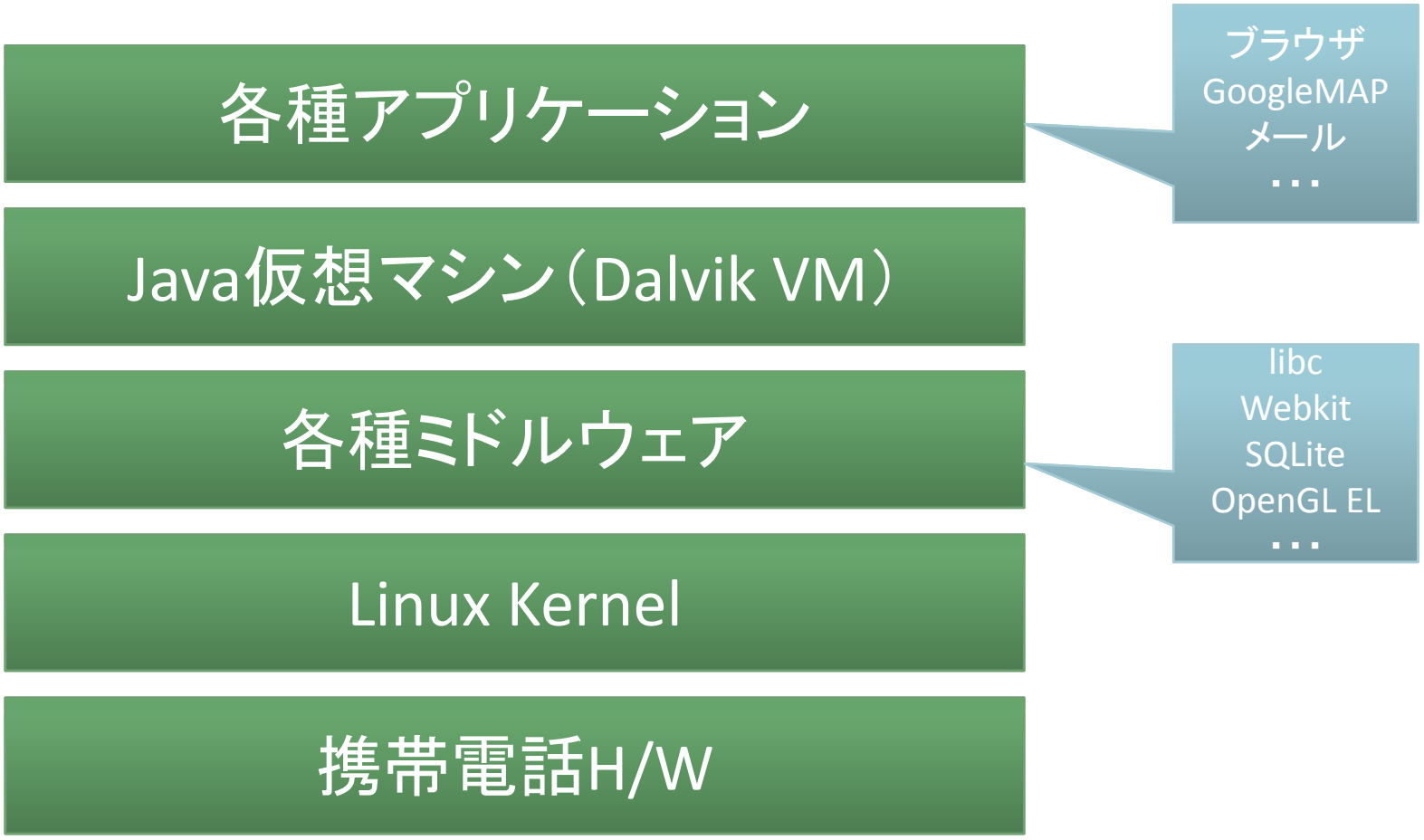
Android

- Googleが開発した携帯電話用ソフトウェアプラットフォーム
  - 無償提供
  - オープンソース（カーネル以外はApache License）
  - OSからブラウザなど上位アプリケーションまでをパッケージング

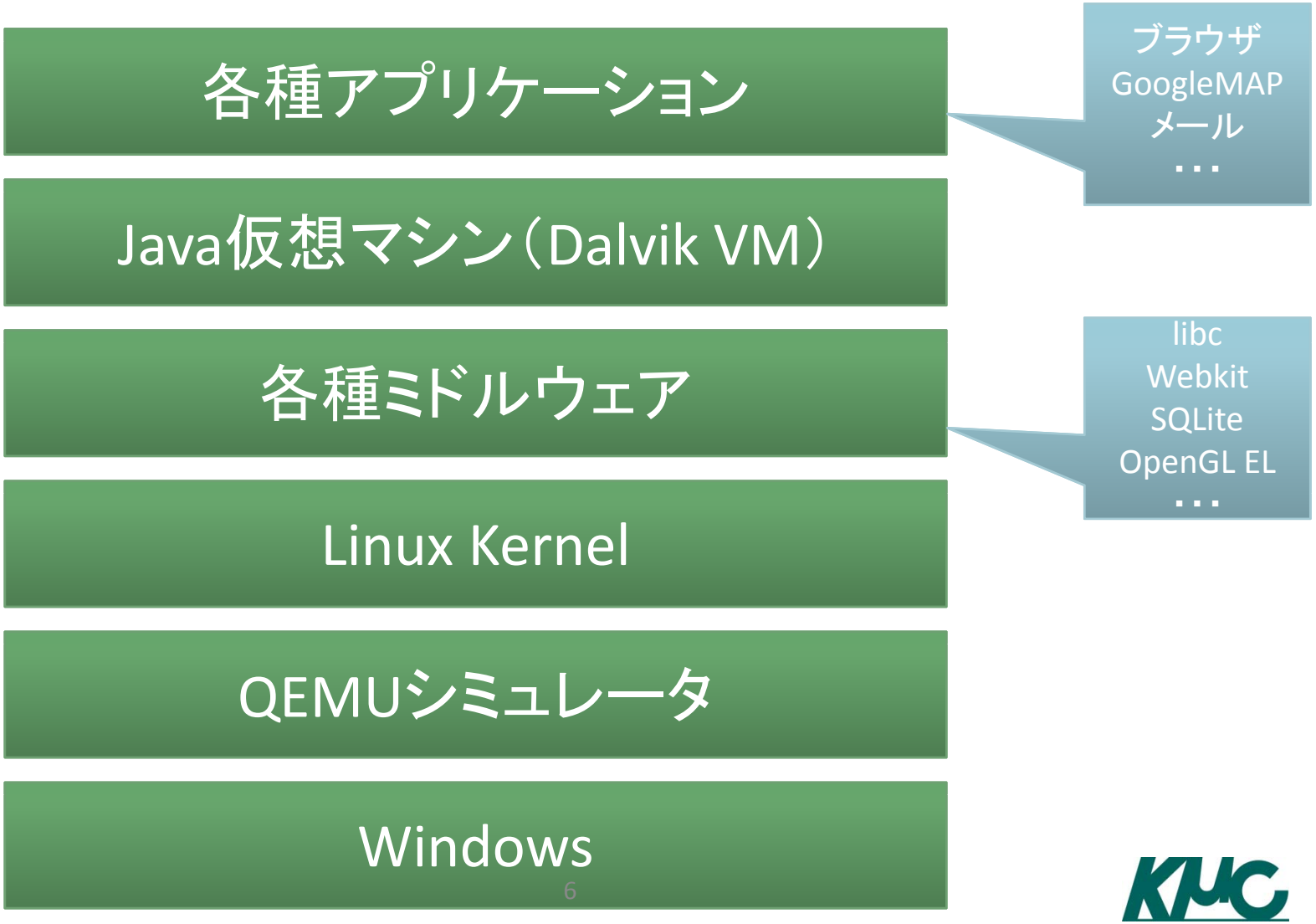
- 2008年末にも搭載したセットが出てくると言われている。そのあとにソースが提供されるらしい。

http://code.google.com/android/
http://www.openhandsetalliance.com/

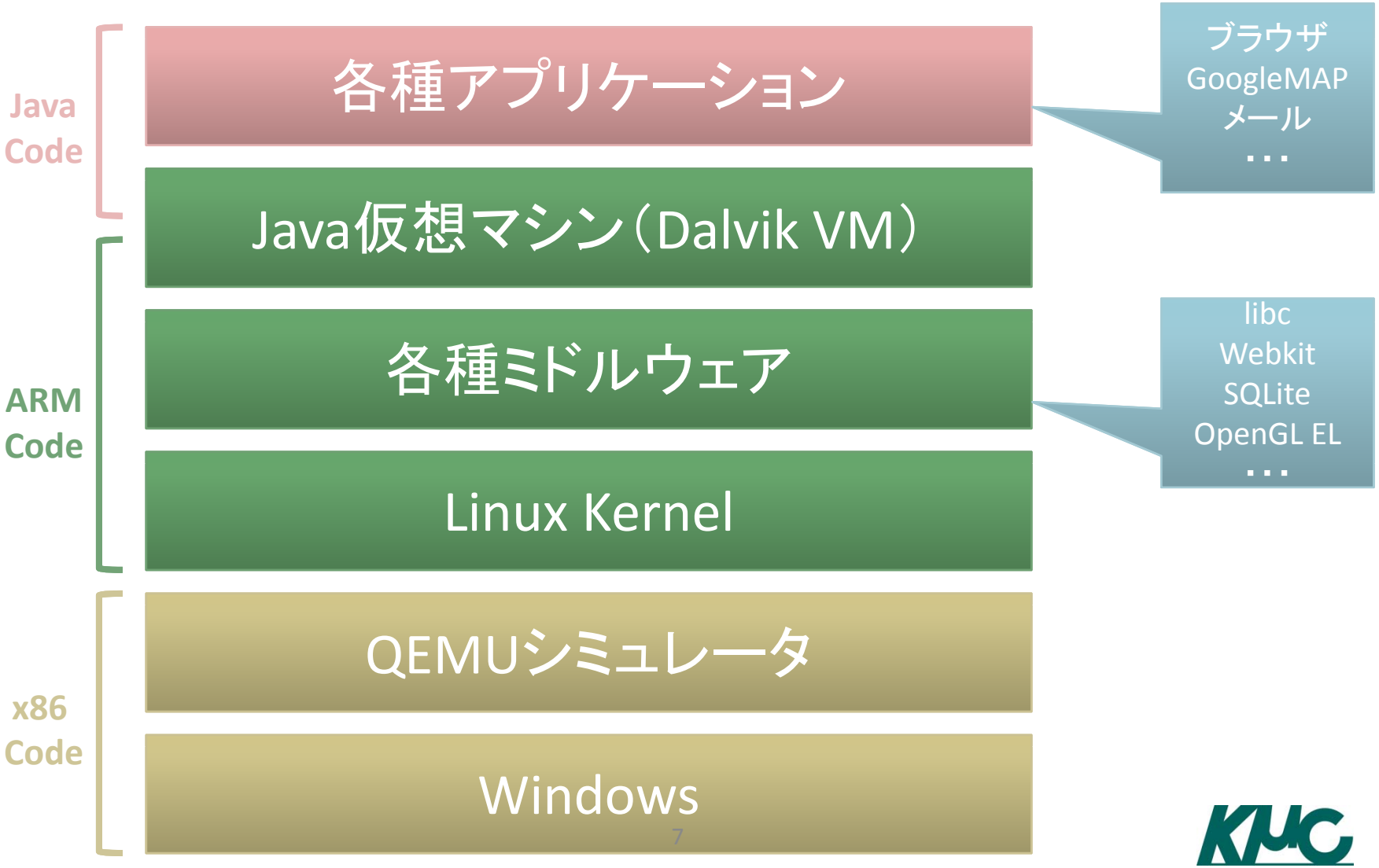
プラットフォームの構造



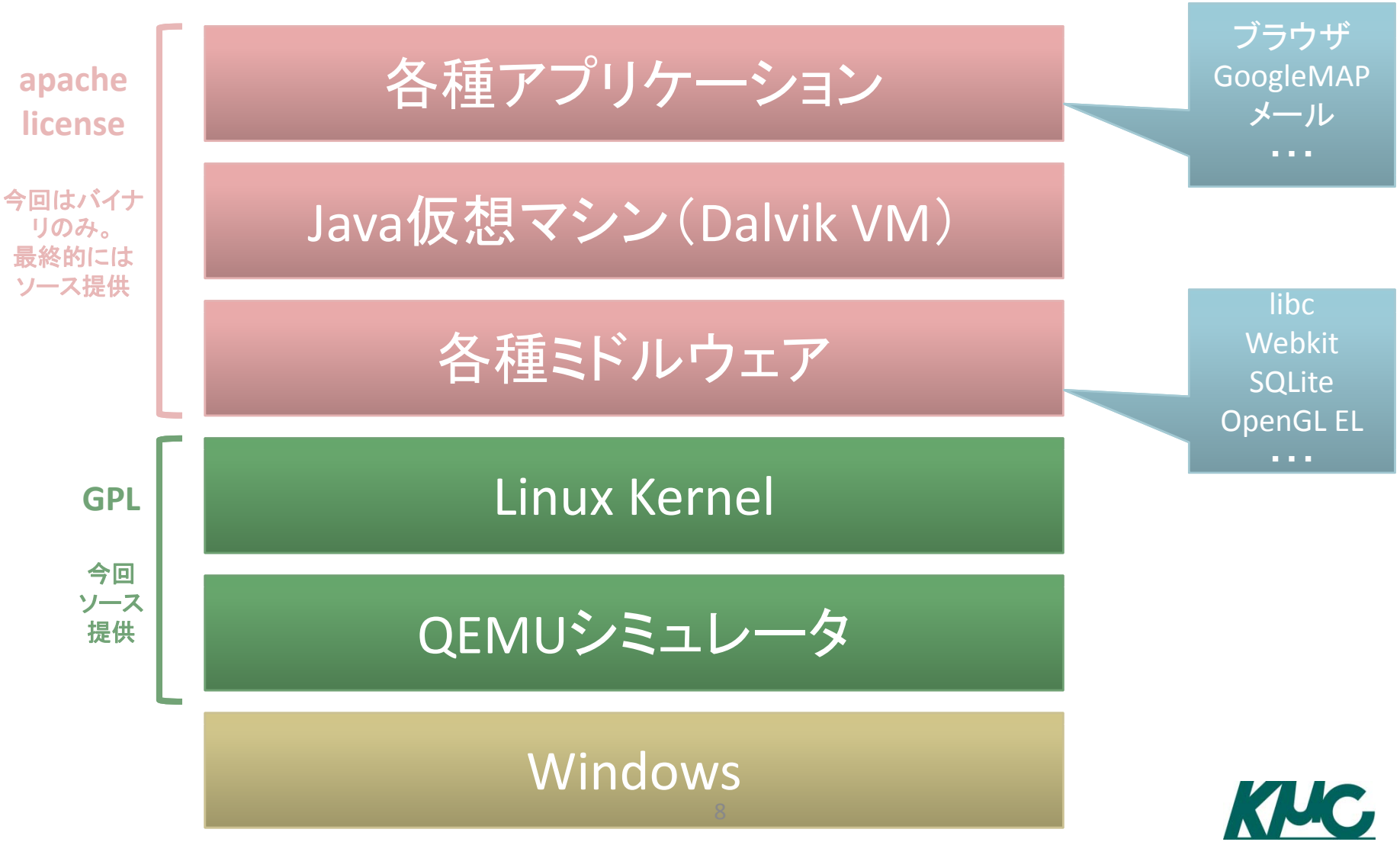
入手できるSDKの構造



入手できるSDKの構造



入手できるSDKの構造



実際のCPUボードに移植

実際のCPUボードに移植

- QEMUより上位のARMバイナリを弊社のARM11のボードに載せてみました。
- 他の方がやられている方法と基本的に同じ。

KZM-ARM11-01

- ARM11搭載評価ボード
  - 豊富なペリフェラル
  - Linux2.6実装済み

- CPU freescale i.MX31
  - ARM1136JF-S/532MHz
  - L2キャッシュ256KByte
- メモリ
  - DDR SDRAM 128MByte
  - NOR FLASH 64MByte
  - NAND FLASH 128MByte
- 周辺機器
  - QVGAタッチパネル付きLCD
  - 2.5” IDE-HDD
  - SD/MMCメモリカードスロット
  - USB2.0 OTG
  - 100Mbps Ethernet
  - AC97 Audio
  - 26万画素CCDカメラ
  - PCMCIAカードスロット
  - 2ch RS232C

KZM-ARM11-01にAndroidを実装

- linuxカーネルのみをAndroid用に変更
  - linux-2.6.22 + KZM-ARM11-01がベース
  - Binderデバイスを追加
    - Android用に公開されたlinux-2.6.23からコピー
  - Android用ドライバの追加
    - Android用に公開されたlinux-2.6.23/drivers/androidをコピー
    - コピーして組み込んだだけで、動作チェックはしていない
  - FBドライバを4行コメントアウト
    - 描画更新の頻度を上げる
  - カーネルコンフィグレーションの変更
    - thumb, USBキーボードをサポート
  - 最初に起動するプロセス名をハードコード
- ルートファイルシステム
  - Android SDKから抽出
    - スタティックリンクしたbusyboxを用意して、tarで固めて取り出し
  - NFSルートでのマウント
- その他
  - ネットワーク関係（IPアドレスやDNSサーバなど）の設定

- 以上の作業で、KZM-ARM11-01でAndroidが動作し、ブラウザやGoogleMapが動作しました


LAP-LINK ADAPTOR
Western Digital
WD1200VE
MAC: 00019b0400bd
S/N :E0500189-04
12:00 AM
Browser
KZM-ARM11-01
Kyoto Microcomputer Co.,Ltd.
RUN
POWER
ACTIVE
BUSY
PARTNER-Jet
KMC

12:00 AM
牛込
神楽坂
九段下
御茶ノ水
秋葉原
神保町
小川町
岩本町
市ケ谷
千代田
区役所
神田
小伝馬町
竹橋
半蔵門
皇居
大手町
人形
麹町
三越前
二重橋前
東京
日本橋
永田町
桜田門
日比谷
京橋
赤坂見附
国会
霞ケ関
銀座
宝町

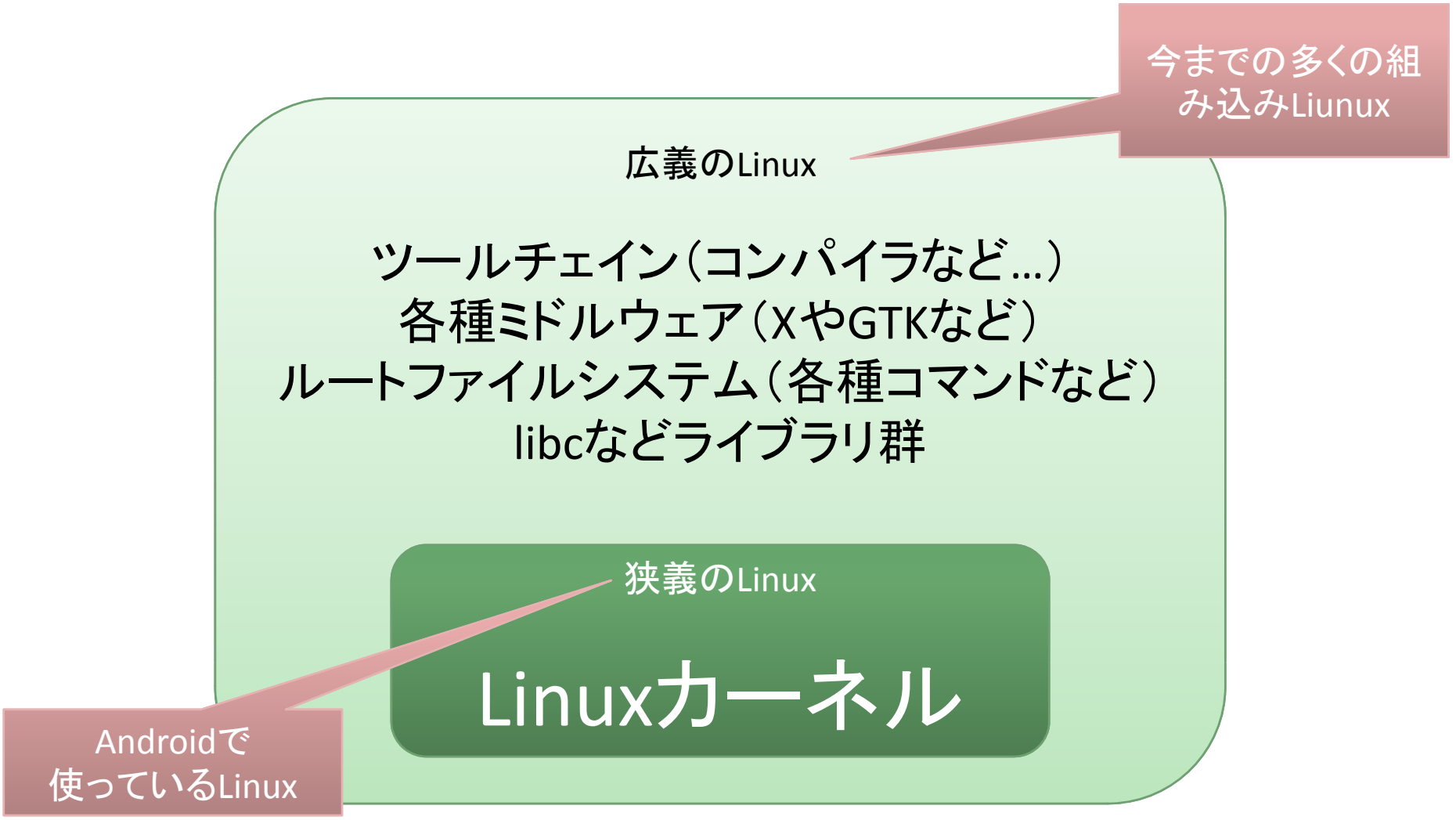

12:00 AM
ICE、デバッガのパイオニア-京都マイク

INSIDE ANDROID

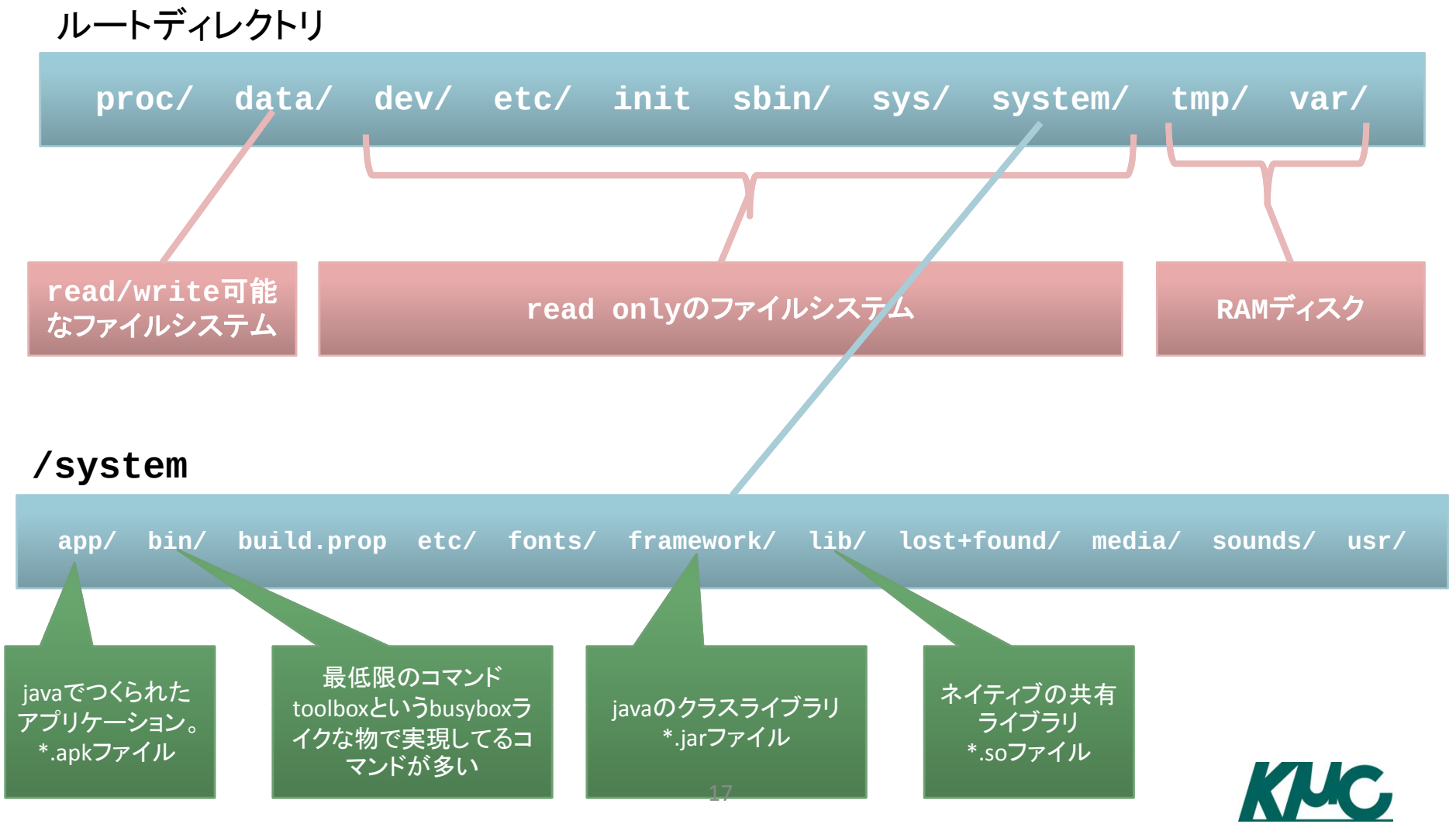
AndroidのLinux環境

- 本当にカーネルしか使っていない！！
- 通常のLinux的な（UNIXライク）な初期化も無い
  - init.rcなどのスクリプトでの初期化が少ない
  - /initで小さな初期化をしたら、殆どすぐアプリケション実行環境へ
  - ネットワーク関係など、簡単な初期化のみ
- ルートファイルシステムも特徴的
- libcもLinuxでよく使われるglibcやuClibcでなく、BSDのlibcを利用

Androidはカーネルのみ利用



ファイルシステム


ルートディレクトリ
proc/ data/ dev/ etc/ init sbin/ sys/ system/ tmp/ var/
read/write可能なファイルシステム
read onlyのファイルシステム
RAMディスク
/system
app/ bin/ build.prop etc/ fonts/ framework/ lib/ lost+found/ media/ sounds/ usr/
javaでつくられたアプリケーション。*.apkファイル
最低限のコマンドtoolboxというbusyboxライクな物で実現してるコマンドが多い
javaのクラスライブラリ*.jarファイル
ネイティブの共有ライブラリ*.soファイル

動いているプロセス

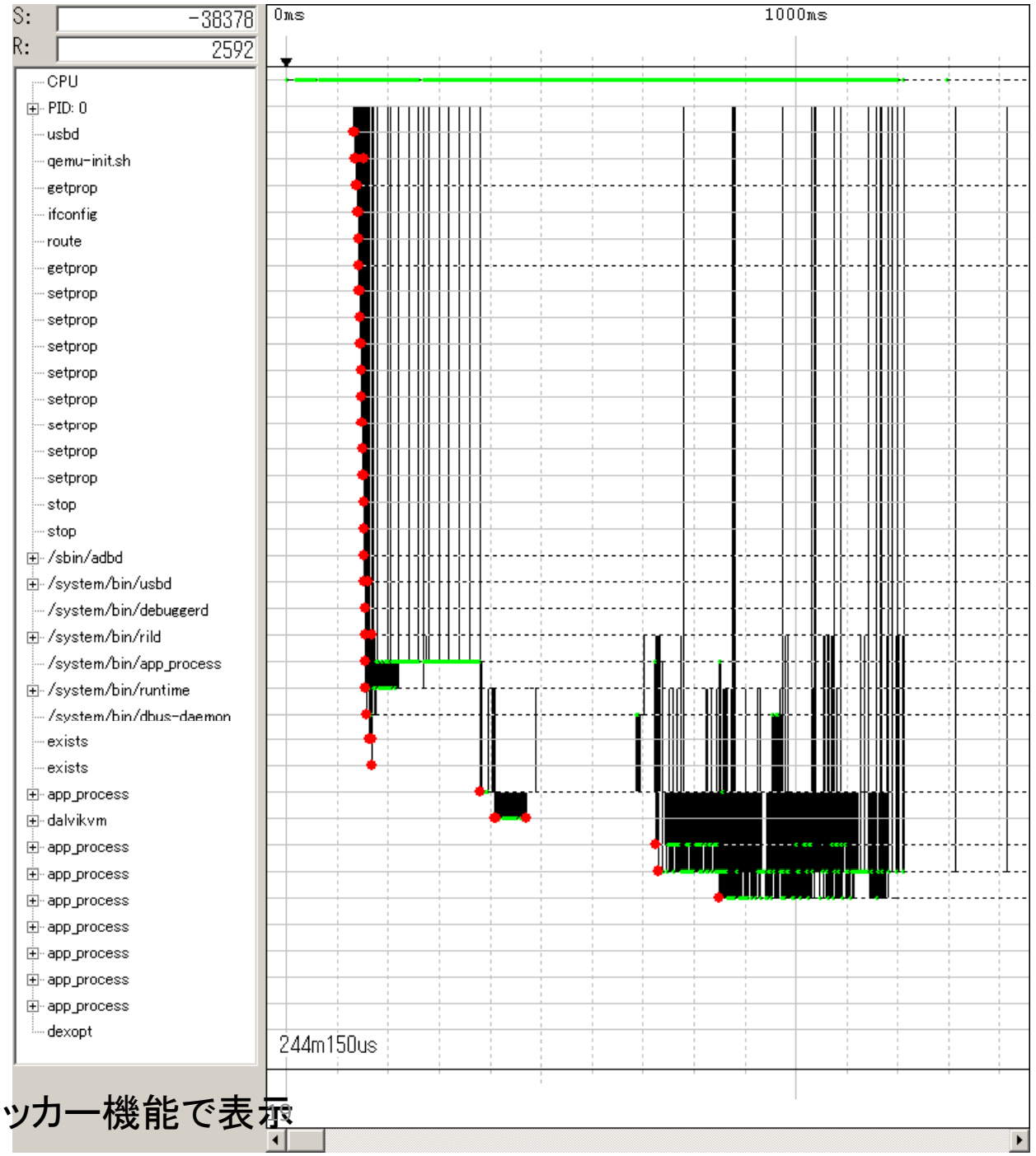
適当に色々動かした状態でPARTNER-デバッガのTOPコマンド（Linux対応機能）にて表示

- 1(0x1) /init
- 446(0x1be) /sbin/adbd
- 447(0x1bf) /system/bin/usbd
- 448(0x1c0) /system/bin/debuggerd
- 449(0x1c1) /system/bin/rild
- 450(0x1c2) /system/bin/app_process
- 451(0x1c3) /system/bin/runtime
- 452(0x1c4) /system/bin/dbus-daemon
- 461(0x1cd) /system/bin/app_process
- 496(0x1f0) /system/bin/app_process
- 500(0x1f4) /system/bin/app_process
- 508(0x1fc) /system/bin/app_process
- 534(0x216) /system/bin/app_process
- 556(0x22c) /system/bin/app_process
- 580(0x244) /system/bin/app_process
- 589(0x24d) /system/bin/app_process

アプリケーションを起動しても、起動するのはこのapp_processだけ。このapp_processがDalvikVMを使って、Javaアプリを動かすようだ

動いたプロセスの様子

非常にシンプルな事が分かる。
シェルスクリプトを使った初期化でみられる、shやlnや[やsedなど各種unixコマンドが全く見あたらない。
普通のコマンドは、ifconfigとrouteコマンドぐらいか。



PARTNERデバッガのイベントトラッカー機能で表示

ユーザー空間のメモリマップ

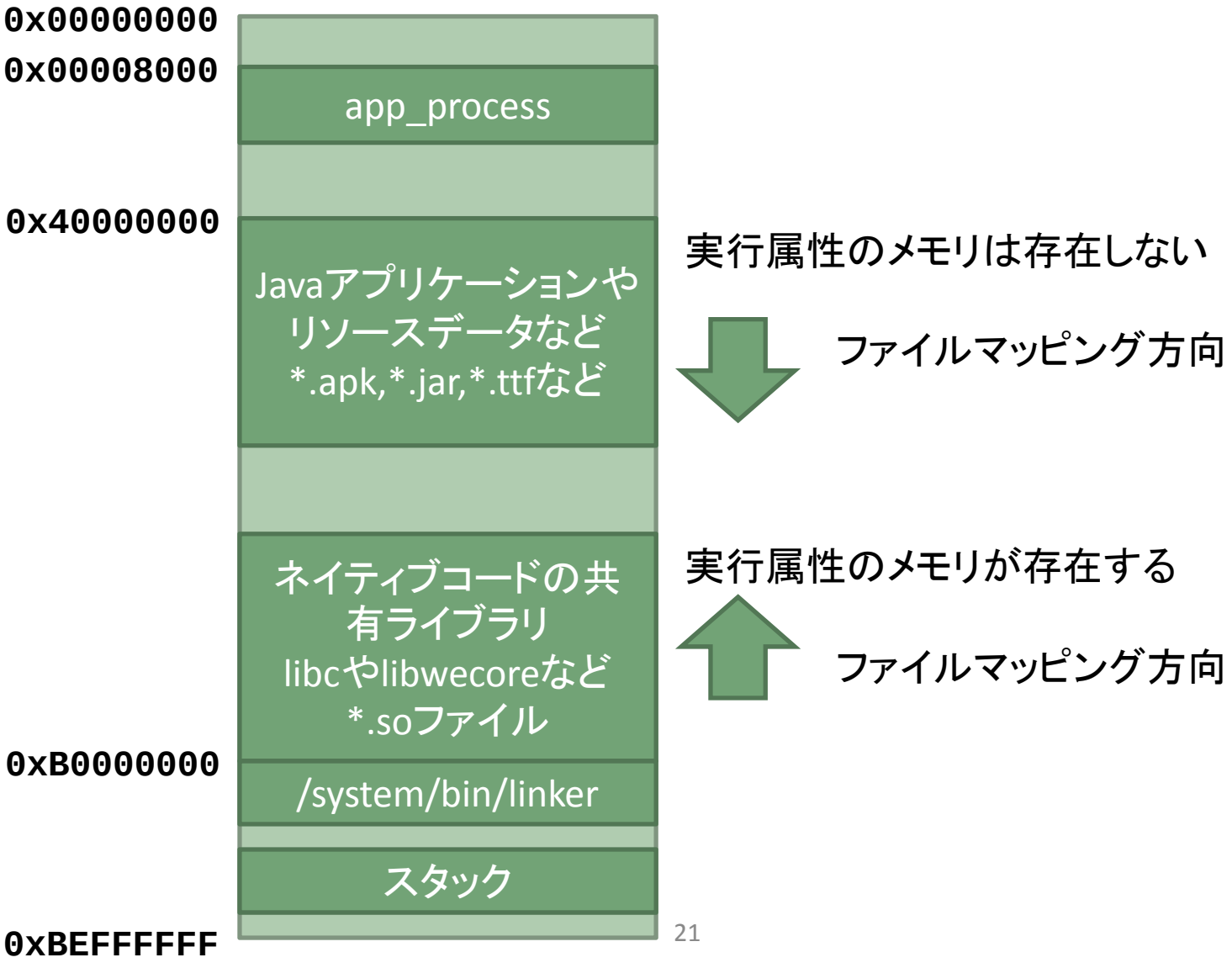
GoogleMapアプリケーションと思われる
プロセスのマッピング状況
一部の初期化アプリケーションを除き、
殆どが同じような感じ

```
>maps 523
00008000-0000a000 rwxp /system/bin/app_process

40000000-40008000 r--s /system_properties
40008000-40139000 rw-p /tmp/dalvik-heap-450
4140a000-41655000 r--s /system/framework/core.jar
41d40000-41d4b000 r--s /system/framework/ext.jar
41d6d000-41f73000 r--s /system/framework/framework.jar
4255e000-42561000 r--s /system/framework/framework-tests.jar
42568000-42642000 r--s /system/framework/framework-res.apk
42664000-42e64000 r--p /dev/binder
42e66000-42e6d000 r--s /system/app/ContactsProvider.apk
42e7d000-42e7e000 r--s /tmp/android-shared_heap.461.0x118cf8
42e7e000-42e85000 r--s /system/app/ContactsProvider.apk
42e87000-42e89000 r--s /system/app/GoogleAppsProvider.apk
42eba000-42ee4000 r--s /system/app/Maps.apk
42f3d000-42f42000 r--s /system/app/MediaProvider.apk
42f52000-42f56000 r--s /system/app/ImProvider.apk
42f60000-43060000 rw-s /tmp/android-shared_heap.508.0x16d680
43064000-4306f000 r--s /system/app/GoogleApps.apk
43147000-43176000 r--s /system/app/XmppService.apk
43176000-43178000 r--s /system/app/XmppService.apk
```

```
9d800000-9d80a000 r-xp /system/lib/libdrm1.so
9dc00000-9dc09000 rwxp /system/lib/libaes.so
a9c00000-a9c06000 r-xp /system/lib/libhardware.so
a9d00000-a9d54000 r-xp /system/lib/libutils.so
a9f00000-a9f02000 rwxp /system/lib/libpim.so
aa000000-aa2fd000 r-xp /system/lib/libwebcore.so
aa800000-aa81e000 rwxp /system/lib/libexpat.so
aac69000-aac6a000 rwxp /system/lib/libsqlite.so
ab20b000-ab20c000 rwxp /system/lib/libmedia.so
ab303000-ab304000 rwxp /system/lib/libmedia_jni.so
ab617000-ab618000 rwxp /system/lib/libsonivox.so
abc95000-abc96000 rwxp /system/lib/libpv.so
ac0d2000-ac0d4000 rwxp /system/lib/libsgl.so
ac419000-ac41a000 rwxp /system/lib/libui.so
acd15000-acd16000 rwxp /system/lib/libopengles_cm.so
ace00000-ace0b000 rwxp /system/lib/libcorecg.so
acf1a000-acf1b000 rwxp /system/lib/libpixelflinger.so
ad064000-ad066000 rwxp /system/lib/libdvm.so
ad227000-ad22a000 rwxp /system/lib/libnativehelper.so
ad349000-ad34d000 rwxp /system/lib/libandroid_runtime.so
ad4ad000-ad4ae000 rwxp /system/lib/libicui18n.so
ad5bd000-ad5bf000 rwxp /system/lib/libicuuc.so
ad600000-ad734000 rwxp /system/lib/libicudata.so
ae84b000-ae84c000 rwxp /system/lib/libdbus.so
af5c3000-af5d1000 rwxp /system/lib/libcrypto.so
af723000-af726000 rwxp /system/lib/libssl.so
af90d000-af90e000 rwxp /system/lib/libz.so
afb0d000-afb0e000 rwxp /system/lib/libcutils.so
afc1f000-afc21000 rwxp /system/lib/libm.so
afd00000-afd01000 rwxp /system/lib/libstdc++.so
afe36000-afe39000 rwxp /system/lib/libc.so
b0000000-b0014000 rwxp /system/bin/linker
bedf5000-bee0a000 rwxp
```

ユーザー空間のメモリマップ



カーネル

- カーネルも大きな変更は必要なさそう
  - linux-2.6.23を採用
  - KMCではlinux-2.6.22でも動作した
    - linux-2.6.16では動作しませんでした
  - 大きな改造は必要ないが、以下の変更が必要
    - Binderを組み込む必要あり
      - open binder project, プロセス間通信に使われる
      - カーネルソースに追加する必要あり
    - thumb,EABIを有効にする必要あり
      - カーネルコンフィグレーションで対応
    - ARM v5（926)以上が確実と思われる
    - goldfish H/W依存のソースを追加
      - ボードによっては必要ありません
    - 表示関係はFBをダブルバッファにする必要あり
      - ドライバによっては改造が必要

Dalvik VM

- ソース未提供。
- Javaのバイトコード(8bit単位の命令長、スタックアーキテクチャ) を独自バイトコード (16bit単位の命令長、レジスタアーキテクチャ) に変換して実行しているという話。

Dalvik VMのインタープリタ

- インタープリタはジャンプテーブルを多用する。
- ICEで実行履歴を見ると、それらしい箇所を発見 (次のスライド)
- JITのように実行時に命令コードを生成して実行している形跡はない。このバージョンではインタプリタのみで動作しているようだ。

インタープリタの断片？

```
116 000m014 IE/A  AD019ECC LDR     r0,[r13,#7C]
            IE/A  AD019ED0 MOV     r14,r11,lsr #0C
            IE/A  AD019ED4 MOV     r5,r11,lsr #8
            IE/A  AD019ED8 LDR     r12,[r0,r14,lsl #2]
            IE/A  AD019EDC LDR     r8,[r13,#18]
            IE/A  AD019EE0 LDRH    r11,[r9,#2]!
            IE/A  AD019EE4 LDR     r14,[r13,#10]
            IE/A  AD019EE8 AND     r2,r5,#0F
            IE/A  AD019EEC LDR     r4,[r0,r2,lsl #2]
            IE/A  AD019EF0 ADD     r1,r8,r14
            IE/A  AD019EF4 AND     r8,r11,#0FF/r
            IE/A  AD019EF8 LDR     r5,[r1,r8,lsl #2]
            IE/A  AD019EFC ADD     r3,r4,r12
            IE/A  AD019F00 STR     r3,[r0,r2,lsl #2]
            IE/A  AD019F04 MOV     r15,r5
```

テーブルジャンプ

Dalvik VMのインタープリタ

- 実行アドレスから、これはlibdvm.soのライブラリの中とわかる。
- さらに調べるとこのコードは“dvmInterpretStd” の関数の中。

→間違いなくこれがインタープリタの本体

命令コードのデコード

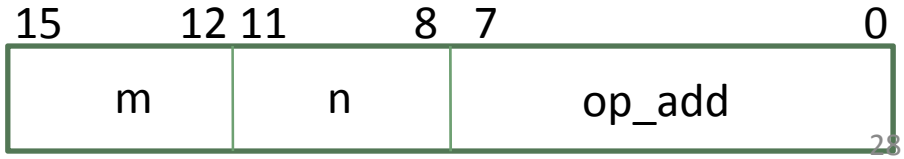
```
     116 000m014 IE/A   AD019ECC LDR     r0,[r13,#7C]
                 IE/A   AD019ED0 MOV     r14,r11,lsr #0C
                 IE/A   AD019ED4 MOV     r5,r11,lsr #8
                 IE/A   AD019ED8 LDR     r12,[r0,r14,lsl #2]
                 IE/A   AD019EDC LDR     r8,[r13,#18]
                 IE/A   AD019EE0 LDRH    r11,[r9,#2]!
                 IE/A   AD019EE4 LDR     r14,[r13,#10]
                 IE/A   AD019EE8 AND     r2,r5,#0F
                 IE/A   AD019EEC LDR     r4,[r0,r2,lsl #2]
                 IE/A   AD019EF0 ADD     r1,r8,r14
                 IE/A   AD019EF4 AND     r8,r11,#0FF/r
                 IE/A   AD019EF8 LDR     r5,[r1,r8,lsl #2]
                 IE/A   AD019EFC ADD     r3,r4,r12
                 IE/A   AD019F00 STR     r3,[r0,r2,lsl #2]
                 IE/A   AD019F04 MOV     r15,r5
```

16bit符号なしロード
＋オートインクリメント
r11 = *(uint16*)(r9++)

r9がVMのプログラムカウンタ
r13がVMのフレームポインタと推測できる。

命令コードの実行

```
     116 000m014 IE/A  AD019ECC LDR     r0,[r13,#7C]
                 IE/A  AD019ED0 MOV     r14,r11,lsr #0C
                 IE/A  AD019ED4 MOV     r5,r11,lsr #8
                 IE/A  AD019ED8 LDR     r12,[r0,r14,lsl #2]
                 IE/A  AD019EDC LDR     r8,[r13,#18]
                 IE/A  AD019EE0 LDRH    r11,[r9,#2]!
                 IE/A  AD019EE4 LDR     r14,[r13,#10]
                 IE/A  AD019EE8 AND     r2,r5,#0F
                 IE/A  AD019EEC LDR     r4,[r0,r2,lsl #2]
                 IE/A  AD019EF0 ADD     r1,r8,r14
                 IE/A  AD019EF4 AND     r8,r11,#0FF/r
                 IE/A  AD019EF8 LDR     r5,[r1,r8,lsl #2]
                 IE/A  AD019EFC ADD     r3,r4,r12
                 IE/A  AD019F00 STR     r3,[r0,r2,lsl #2]
                 IE/A  AD019F04 MOV     r15,r5
```



Reg[n] += Reg[m]

Dalvik VMのインタープリタ

- よく見ると、まだチューニングの余地がありそう。
  - ジャンプテーブルの先頭アドレスをレジスタに保持していれば、4命令節約できそう。
  - おそらくこれは手でアセンブラで書いたのではなくC言語でかかれているのでは？
    - 他のCPUへの移植が容易

KMC

チューニングの余地？

```
116 000m014 IE/A  AD019ECC LDR     r0,[r13,#7C]
            IE/A  AD019ED0 MOV     r14,r11,lsr #0C
            IE/A  AD019ED4 MOV     r5,r11,lsr #8
            IE/A  AD019ED8 LDR     r12,[r0,r14,lsl #2]
            IE/A  AD019EDC LDR     r8,[r13,#18]
            IE/A  AD019EE0 LDRH    r11,[r9,#2]!
            IE/A  AD019EE4 LDR     r14,[r13,#10]
            IE/A  AD019EE8 AND     r2,r5,#0F
            IE/A  AD019EEC LDR     r4,[r0,r2,lsl #2]
            IE/A  AD019EF0 ADD     r1,r8,r14
            IE/A  AD019EF4 AND     r8,r11,#0FF/r
            IE/A  AD019EF8 LDR     r5,[r1,r8,lsl #2]
            IE/A  AD019EFC ADD     r3,r4,r12
            IE/A  AD019F00 STR     r3,[r0,r2,lsl #2]
            IE/A  AD019F04 MO      r15,r5
```

レジスタに保持しておけばいいのに？？

まとめ

- Androidは組み込みLinuxのシステムとしても異色。
  - Linuxはカーネルのみ
  - ルートファイルシステム、ダイナミックリンクの仕組みは独自。おそらくBSDライセンスのもの。
  - サーバ向けシステムの流用でなく、小さな組み込みシステム向けに考えぬかれている
- ソースが公開されればもっといろいろなことがわかるし、勉強になるテクニックが満載！！
- To be continued..

KMC

Thank you

Kyoto Microcomputer Co., Ltd.

http://www.kmckk.co.jp/