Porter AAC Decoder en version X86 sur Android

Par Intel

Date de publication : 18 décembre 2013

N'hésitez pas à **Commentez** !

I - Introduction....3
II - Comme d'habitude, ajoutez x86 à la liste des ABI supportées....3
III - Corriger la liste des sources dans le Makefile d'opencore-mp3dec....3
IV - Remerciements....4

Copyright ® 2013 Intel. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

I - Introduction

« Porter » est un bien grand mot pour le cas d'AAC Decoder, car, comme souvent, il suffit de corriger quelques fichiers Makefiles.

libaacdecoder supporte le décodage AAC ainsi que MP3 au travers des décodeurs OpenCore. La bibliothèque utilise elle-même **opencore-aacdec** qui est directement compilable pour x86.

Le package distribué sur le site du projet AAC Decoder, en version 0.6.1, n'inclut que les binaires pour ARMv5 et ARMv7. Si vous essayez de recompiler la bibliothèque vous-même à partir des sources, vous pourriez même croire que le x86 n'est pas supporté… mais c'est faux !

Si vous êtes pressé, voici le package (version 0.6.1) avec la version x86 de libaacdecoder.so ajoutée : **aacdecoder-android-libs-0.6.1.zip**

Quant aux détails techniques, voici les deux modifications nécessaires :

II - Comme d'habitude, ajoutez x86 à la liste des ABI supportées

Ouvrez ./decoder/jni/Application.mk et ajoutez x86 à la liste des ABI ou mettez directement « all » :

```
1. APP_ABI := all
```

Vous pouvez lancer ndk-build à nouveau. Le script essaiera alors de générer les versions x86 de la bibliothèque, mais échouera en renvoyant de nombreuses erreurs telles que celles-ci :

```
1. ...
2.      [exec] ./codecs_v2/audio/mp3/dec/src/asm/pvmp3_dct_16_gcc.s:225: Error: no
 such instruction: `smull r9,r7,r11,r7'
3.      [exec] ./codecs_v2/audio/mp3/dec/src/asm/pvmp3_dct_16_gcc.s:226: Error: too many memory
 references for `add'
4.      [exec] ./codecs_v2/audio/mp3/dec/src/asm/pvmp3_dct_16_gcc.s:227: Error: too many memory
 references for `sub'
5. ...
```

Car le compilateur essaie d'intégrer du code assembleur ARM ! Il nous faut donc corriger ça :

III - Corriger la liste des sources dans le Makefile d'opencore-mp3dec

Ouvrez ./decoder/jni/opencore-mp3dec/Android.mk.

Vous verrez des fichiers asm spécifiques à l'architecture ARM qui sont inclus :

```
1. LOCAL_SRC_FILES := \
2. ...
3.      src/asm/pvmp3_dct_16_gcc.s \
4.      src/asm/pvmp3_dct_9_gcc.s \
5.      src/asm/pvmp3_mdct_18_gcc.s \
6.      src/asm/pvmp3_polyphase_filter_window_gcc.s
```

Bien sûr, ceux-ci ne sont pas utilisables pour une compilation pour l'architecture x86… mais dans src/ vous avez déjà des équivalents C++ qui peuvent l'être ! Donc vous avez simplement besoin de les inclure dans le cas d'une compilation x86 :

```
1. ifeq ($(TARGET_ARCH),arm)
2.  LOCAL_SRC_FILES += \
```


Copyright ® 2013 Intel. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

```
3.      src/asm/pvmp3_dct_16_gcc.s \
4.      src/asm/pvmp3_dct_9_gcc.s \
5.      src/asm/pvmp3_mdct_18_gcc.s \
6.      src/asm/pvmp3_polyphase_filter_window_gcc.s
7. else
8.  LOCAL_SRC_FILES += \
9.      src/pvmp3_dct_16.cpp \
10.      src/pvmp3_dct_9.cpp \
11.      src/pvmp3_mdct_18.cpp \
12.      src/pvmp3_polyphase_filter_window.cpp
13. endif
```

Dans ce fichier aussi, vous trouverez une macro ARM toujours définie :

```
1. LOCAL_CFLAGS := -DPV_ARM_GCC_V4 $(PV_CFLAGS)
```

C'est mieux de la définir seulement dans le cadre d'une compilation pour l'architecture ARM :

```
1. LOCAL_CFLAGS :=  $(PV_CFLAGS)
2. ifeq ($(TARGET_ARCH),arm)
3.   LOCAL_CFLAGS += -DPV_ARM_GCC_V4
4. endif
```

Une fois ces changements faits, vous pourrez lancer ndk-build à nouveau et obtenir libaacdecoder.so pour toutes les architectures supportées par le NDK, y compris x86.

Voici le package correspondant à la dernière mise à jour, avec les bibliothèques présentes pour toutes les plateformes supportées : **aacdecoder-android-libs-0.6.1.zip**.

IV - Remerciements

L'équipe de rédaction Developpez.com tient à remercier **Phanloga** pour la relecture orthographique de cet article et **infosam76** pour la gabarisation.

Merci à **Intel Developer Zone** et à Xavier Hallade pour la rédaction de ce tutoriel.

Retrouvez **le Blog de Xavier Hallade**, Software Engineer chez Intel.

Pour toute question, merci de contacter Slim Soussi, EMEA Program Manager chez Intel : slim(pt)soussi(at)intel(pt)com

N'hésitez pas à commenter cet article ! **Commentez**

Copyright ® 2013 Intel. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.