Investigation report on 64bit support and some of new features in AOSP

Hidenori Yamaji, Sony Mobile
@hidenorly

Disclaimer

- This report is by my personal investigation regarding Android Open Source Project.
- This doesn’t represent my employer’s view in anyway :-)
- Please let me know if you find wrong, changed, etc.

64bit support will come! (SoC)

- Qualcomm (Cortex-A53)
  - http://www.qualcomm.com/media/releases/2013/12/09/qualcomm-technologies-introduces-snapdragon-410-chipset-integrated-4g-lte
  - http://www.anandtech.com/show/7784/snapdragon-610-615-qualcomm-continues-down-its-64bit-warpath-with-48core-cortex-a53-designs
- Intel (ATOM)
  - http://newsroom.intel.com/community/intel_newsroom/blog/2014/02/24/intel-gaining-in-mobile-and-accelerating-internet-of-things
  - http://pc.watch.impress.co.jp/docs/news/event/20140225_636752.html
- MediaTek (MT6752 : Cortex-A53 @ 2GHz x 8 cores)
  - http://www.mediatek.com/en/news-events/mediatek-news/mediatek-launches-mt6752-a-64-bit-octa-core-lte-soc-latest-lte-product-to-enable-super-mid-market/
  - http://www.mediatek.com/en/mwc2014/mediatek-launches-mt6732-a-64-bit-lte-soc-to-spur-the-new-super-mid-market/
- nVIDIA (Denver x 2 cores)
  - http://blogs.nvidia.com/blog/2014/01/05/live-las-vegas/

64bit support will come! (AOSP)

- We can observe 64bit support on master branch of AOSP (Android Open Source Project).

- But please note that this is a snap shot report of what AOSP is working to support 64bit! as my personal view.

Executive Summary

- Both 32bit and 64bit executions on 64bit device
  - x86_64, arm64(ARMv8), mips64
- Using ART for 64bit environment
  - NOT Dalvik
- Some of daemon seems to be running as 32bit.
  - But seems to try to support 64bit.
- No impact on Java application (of course)

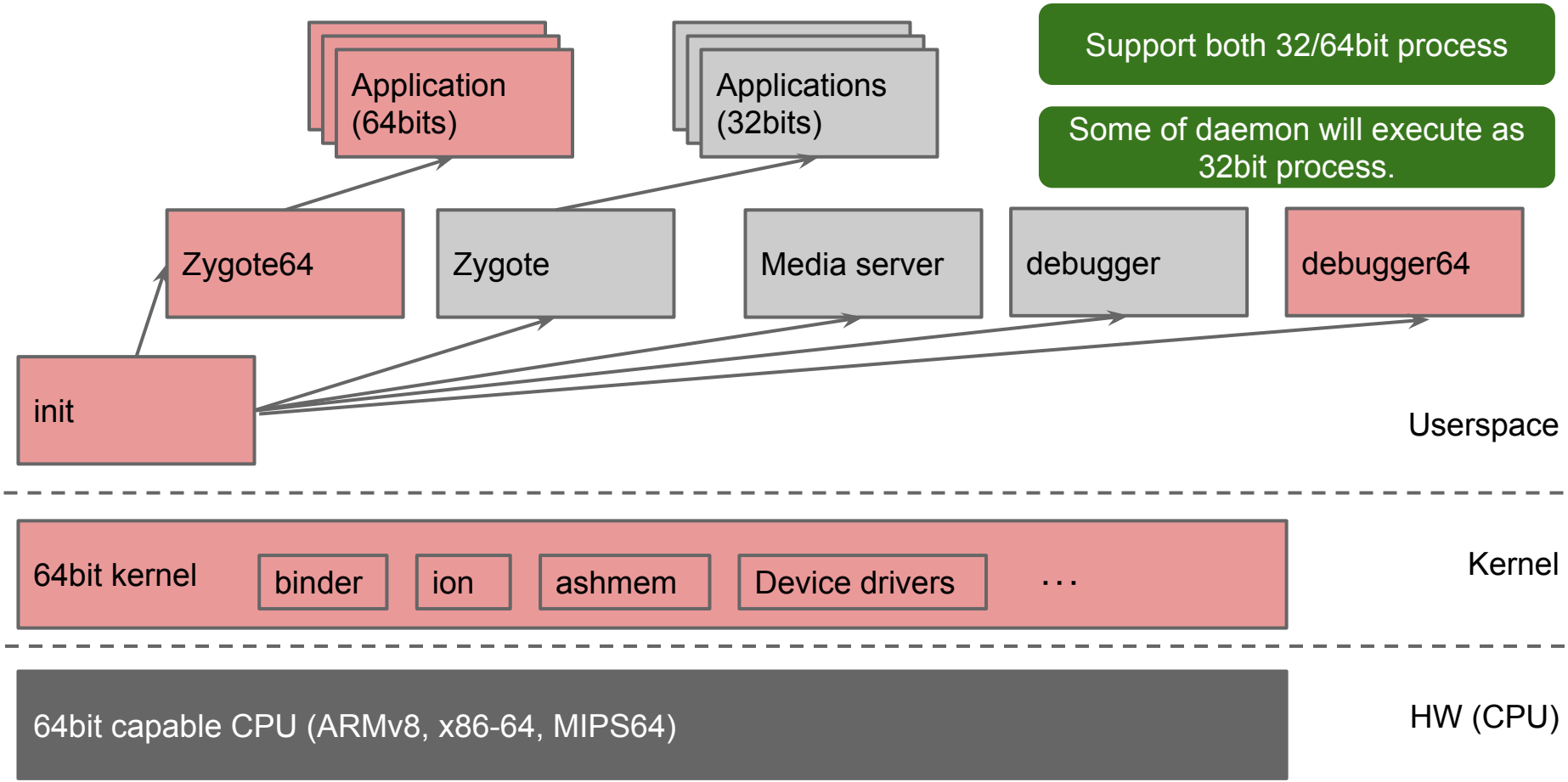
AOSP is working to support 64bit! (1/2)

- build
- art
  - no support on dakvik
- system
  - debuggered
  - cutils/atomic
  - libpixelflinger
  - toolbox
  - backtrace
- bionic
- ndk
- sdk
  - opengl on emu
- external (since before)
  - chromium_org/../v8
  - clang, llvm
  - valgrind
  - linux-tools-perf
  - libvpx

AOSP is working to support 64bit! (2/2)

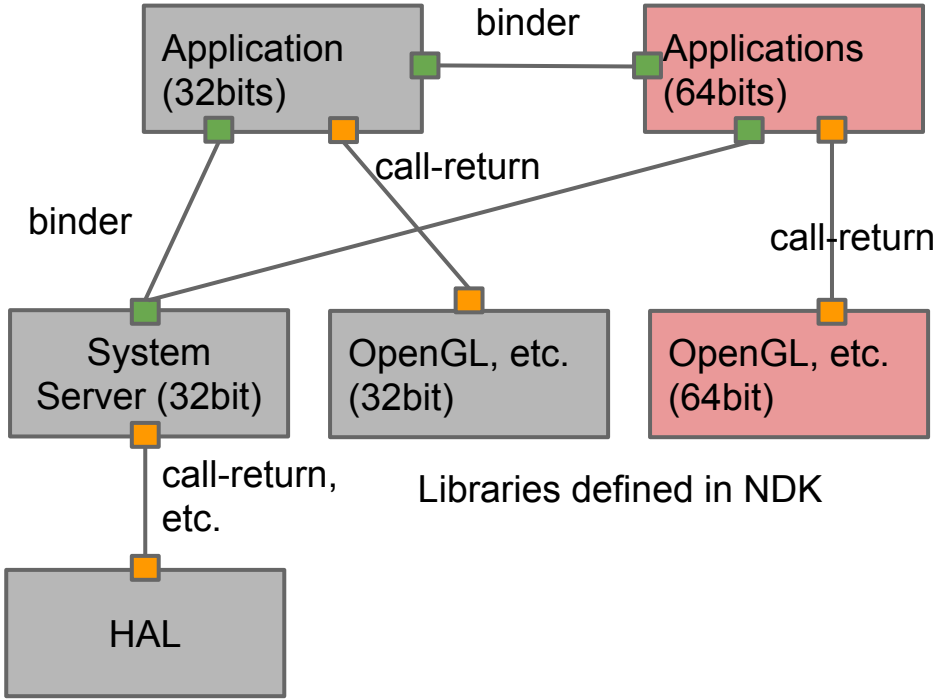
- frameworks/*
  - base/
  - av/
    - services/
      - audioflinger
  - native/
    - binder
    - opengl
  - compile/*, rs/* (renderscipt)
- hardware/
  - libhardware/
    - hardware.c
- libcore/
  - luni/
    - java_math_NativeBN.cpp

Architecture of 64bit support



C&C view: Runtime architecture

This diagram shows in the case of system/core/rootdir/init.zygote32_64.rc



System server executes on 32bit or 64bit (single instance in system)

And system server executes as 64bit in case of system/core/rootdir/init.zygote64.rc

hardware/libhardware/hardware.c has support to choose /system/lib or /system/lib64

Seems to support only LP64

	ILP32	LP64
short	16	16
int	32	32
long	32	64
long long	64	64
pointer	32	64

In “frameworks/base/cmds/app_process/app_main.cpp”

```
#if defined(__LP64__)
static const char ABI_LIST_PROPERTY[] = "ro.product.cpu.abilist64";
static const char ZYGOTE_NICE_NAME[] = "zygote64";
#else
static const char ABI_LIST_PROPERTY[] = "ro.product.cpu.abilist32";
static const char ZYGOTE_NICE_INAME[] = "zygote";
#endif
```

Most of common 64bit support is regarding long & pointer (reinterpret_cast and format string in LOG*, etc.)

Supported instruction architecture

	TARGET_ARCH	TARGET_ARCH_VARIANT	TARGET_CPU_ABI, (TARGET_CPU_ABI2)
64bit	x86_64	x86_64	x86_64, (x86)
	arm64	armv8-a	arm64-v8a, (armeabi-v7a)
	mips64	mips64r2	mips64, (mips)
32bit	x86	x86	x86
	arm	armv7-a, armv5te	armeabi-v7a / armeabi
	mips	mips32r2-fp	mips

TARGET_CPU_ABI_LIST := $(TARGET_CPU_ABI) $(TARGET_CPU_ABI2) $(TARGET_2ND_CPU_ABI) $(TARGET_2ND_CPU_ABI2)
TARGET_CPU_ABI_LIST_64_BIT := $(TARGET_CPU_ABI) $(TARGET_CPU_ABI2)
TARGET_CPU_ABI_LIST_32_BIT := $(TARGET_CPU_ABI) $(TARGET_CPU_ABI2)

Build -> Runtime : system property

- ro.product.cpu.abi=$TARGET_CPU_ABI
- ro.product.cpu.abi2=$TARGET_CPU_ABI2
- ro.product.cpu.abilist=$TARGET_CPU_ABI_LIST
- ro.product.cpu.abilist32=$TARGET_CPU_ABI_LIST_32_BIT
- ro.product.cpu.abilist64=$TARGET_CPU_ABI_LIST_64_BIT

Build: build target

In build/envsetup.sh

add_lunch_combo aosp_arm-eng

add_lunch_combo aosp_arm64-eng

add_lunch_combo aosp_mips-eng

add_lunch_combo aosp_mips64-eng

add_lunch_combo aosp_x86-eng

add_lunch_combo aosp_x86_64-eng

add_lunch_combo vbox_x86-eng

Build: 64bit indicator

- TARGET_IS_64_BIT
  - build/core/combo/TARGET_linux-arm64.mk(46):
    TARGET_IS_64_BIT := true
  - build/core/combo/TARGET_linux-mips64.mk(46):
    TARGET_IS_64_BIT := true
  - build/core/combo/TARGET_linux-x86_64.mk(34):
    TARGET_IS_64_BIT := true

Build: Blacklist for 64bit

- Blacklist specifies build target with 32bit
  - After fixing issue on 64bit, it seems to be removed from the list.
  - At build/core/main.mk
    - L95
      include $(BUILD_SYSTEM)/64_bit_blacklist.mk

Build: Options to specify 32bit only

- LOCAL_32_BIT_ONLY == true
  - Build 32bit binary only which is specified by LOCAL_MODULE in Android.mk

32bit only process (not on 64bit as of now)

frameworks/av/cmds/screenrecord/Android.mk:43:LOCAL_32_BIT_ONLY := true

frameworks/av/cmds/stagefright/Android.mk:26:LOCAL_32_BIT_ONLY := true

frameworks/av/drm/drmserver/Android.mk:42:LOCAL_32_BIT_ONLY := true

frameworks/av/media/libmediaplayerservice/Android.mk:57:LOCAL_32_BIT_ONLY := true

frameworks/av/media/mediaserver/Android.mk:38:LOCAL_32_BIT_ONLY := true

frameworks/av/services/medialog/Android.mk:11:LOCAL_32_BIT_ONLY := true

frameworks/av/services/audioflinger/Android.mk:52:LOCAL_32_BIT_ONLY := true

frameworks/base/cmds/bootanimation/Android.mk:28:LOCAL_32_BIT_ONLY := true

frameworks/base/packages/services/PacProcessor/jni/Android.mk:38:LOCAL_32_BIT_ONLY := true

frameworks/native/services/surfaceflinger/Android.mk:124:LOCAL_32_BIT_ONLY := true

frameworks/rs/driver/runtime/Android.mk:105:LOCAL_32_BIT_ONLY := true

Build: Indicator to build 32/64

- LOCAL_MULTILIB
  - ”32”: Build 32bit library ONLY
  - “both”: build both 64bit & 32bit libraries.
  - This is set by TARGET_SUPPORTS_64_BIT_APPS , TARGET_SUPPORTS_32_BIT_APPS and TARGET_IS_64BIT

Build: Detail of LOCAL_MULTILIB

```
# We don't automatically set up rules to build packages for both
# TARGET_ARCH and TARGET_2ND_ARCH.
# By default, an package is built for TARGET_ARCH.
# To build it for TARGET_2ND_ARCH in a 64bit product, use "LOCAL_MULTILIB := 32".
..snip..
ifeq ($(TARGET_SUPPORTS_32_BIT_APPS)|$(TARGET_SUPPORTS_64_BIT_APPS),
true|true)
  # packages default to building for either architecture,
  # the preferred if its supported, otherwise the non-preferred.
else ifeq ($(TARGET_SUPPORTS_64_BIT_APPS),true)
  # only 64-bit apps supported
  ifeq ($(filter $(my_module_multilib),64 both first),$(my_module_multilib))
    # if my_module_multilib was 64, both, first, or unset, build for 64-bit
    my_module_multilib := 64
  else
    # otherwise don't build this app
    my_module_multilib := none
  endif
else
  # only 32-bit apps supported
  ifeq ($(filter $(my_module_multilib),32 both),$(my_module_multilib))
    # if my_module_multilib was 32, both, or unset, build for 32-bit
    my_module_multilib := 32
  else ifeq ($(my_module_multilib),first)
    ifndef TARGET_IS_64_BIT
      # if my_module_multilib was first and this is a 32-bit build, build for
      # 32-bit
      my_module_multilib := 32
    else
      # if my_module_multilib was first and this is a 64-bit build, don't build
      # this app
      my_module_multilib := none
    endif
  else
    # my_module_mulitlib was 64 or none, don't build this app
    my_module_multilib := none
  endif
endif
```

Deploy: /system/lib64 and /system/lib

In “build/core/envsetup.mk”

```
ifneq ($(filter %64,$(TARGET_ARCH)),)
# /system/lib always contains 32-bit libraries,
# and /system/lib64 (if present) always contains 64-bit libraries.
TARGET_OUT_SHARED_LIBRARIES := $(TARGET_OUT)/lib64
else
TARGET_OUT_SHARED_LIBRARIES := $(TARGET_OUT)/lib
endif
..snip..
# Out for TARGET_2ND_ARCH
TARGET_2ND_ARCH_VAR_PREFIX := 2ND_
TARGET_2ND_ARCH_MODULE_SUFFIX := _32
..snip..
$(TARGET_2ND_ARCH_VAR_PREFIX)TARGET_OUT_SHARED_LIBRARIES := $(TARGET_OUT)/lib
..snip..
```

Deploy : /system/lib{64}/apkname

Support per-package lib dirs for bundled apps

Bundled apps can now use /system/lib/apkname or /system/lib64/apkname in addition to the (globally shared) /system/lib and /system/lib64 directories. Note that when an app is updated post hoc the update APK will look to its normal library install directory in /data/data/[packagename]/lib, so such updates must include *all* needed libraries -- the private /system/lib/apkname dir will not be in the path following such an update.

"apkname" here is the base name of the physical APK that holds the package's code. For example, if a 32-bit package is resident on disk as /system/priv-app/SettingsProvider.apk then its app-specific lib directory will be /system/lib/SettingsProvider

Deploy : system image size will be increased

- Increase system image size of generic_x86_64 to 650MB.
- Increase system image size of generic_x86_64 to 750MB.
- Increase system image size to 650M for generic_arm64

Runtime : package manager

Package manager changes for dual zygote stack.

- Pass down the app's instruction set to dexopt so that it can compile the dex file for the right architecture.
- Also pass down the app's instruction set to rmdex, movedex and getSize so that they can construct the cache file location properly.
- Temporarily compile "system" jars such as am,wm etc. for both architectures. A follow up change will ensure that they're compiled only for one architecture (the same arch. as the system server).
- Java "shared" libraries are now compiled for the right architecture when an app requires them.
- Improve the app native library ABI detection to account for system apps installed in /system/lib{64}/<packagename> and also handle sdcard and forward locked apps correctly.

Runtime : Adjust instruction sets for shared UID apps

Adjust instruction sets for shared UID apps.

Since shared UID apps are run in the same process, we'll need to make sure they're compiled for the same instruction set.

This change implements the recompilation of apps that don't have any ABI constraints.

Apps that *do* have ABI constraints are harder to deal with, since we'll need to rescan them to figure out the full list of ABIs they support and then re-extract the native libraries from these apps once we find an ABI we can use throughout.

Allow 32bit or 64/32bit (64bit only might not be allowed)

In build/core/config.mk

```
# If for a 64bit build we have a 2nd architecture but the zygote isn't 64bit,
# assume DEX2OAT should DEXPREOPT for the 2nd architecture.
ifdef TARGET_2ND_ARCH
  ifeq (true,$(TARGET_IS_64_BIT))
        ifeq ($(filter ro.zygote=zygote64,$(PRODUCT_DEFAULT_PROPERTY_OVERRIDES)),)
        DEX2OAT_TARGET_ARCH := $(TARGET_2ND_ARCH)
        DEX2OAT_TARGET_CPU_VARIANT := $(TARGET_2ND_CPU_VARIANT)
        endif
  endif
endif
```

init: zygote64 kicks system server

In system/core/rootdir/init.zygote64.rc

service zygote /system/bin/app_process64 -Xzygote /system/bin --zygote --start-system-server

class main

socket zygote stream 660 root system

onrestart write /sys/android_power/request_state wake

onrestart write /sys/power/state on

onrestart restart media

onrestart restart netd

Only 64bit Zygote doesn't have compatibity for existing binary. Therefore this seems to be debugging purpose for 64bit.

System server executes on Zygote32 or Zygote64

In system/core/rootdir/init.zygote32_64.rc (not used by build/*)

```
service zygote /system/bin/app_process -Xzygote /system/bin --zygote --start-system-server --socket-name=zygote
    class main
    socket zygote stream 660 root system
    onrestart write /sys/android_power/request_state wake
    onrestart write /sys/power/state on
    onrestart restart media
    onrestart restart netd

service zygote_secondary /system/bin/app_process64 -Xzygote /system/bin --zygote --socket-name=zygote_secondary
    class main
    socket zygote_secondary stream 660 root system
    onrestart restart zygote
```

Currently system server is in 32bit
but we can see 64bit porting efforts in HAL.
Therefore it will be switched to 64bit primary?

Synchronization between Zygote 32 and 64

- Wait for secondary zygote before bringing up the system_server.
  - The zygote that's responsible for starting up the system server now checks if there's another zygote on the system, and waits for it to start up. Also, a few minor clean ups :
    - Address a long standing TODO about zygote retries.
    - Have functions throw IOException where appropriate and wrap them in ZygoteStartFailedEx with a filled in cause.
- Have "stop" stop the secondary zygote as well.
  - Would've been nice if we could use the sys property observer to start and stop all services in a service class but service classes do not appear to be fully supported.

debuggerd64

In system/core/rootdir/init.rc

..snip..

service debuggerd /system/bin/debuggerd

..snip..

service debuggerd64 /system/bin/debuggerd64

..snip

In system/core/debuggerd/Android.mk

LOCAL_MODULE := debuggerd

LOCAL_MODULE_STEM_32 := debuggerd

LOCAL_MODULE_STEM_64 := debuggerd64

LOCAL_MULTILIB := both

..snip..

LOCAL_MODULE_TAGS := optional

LOCAL_MODULE := crasher

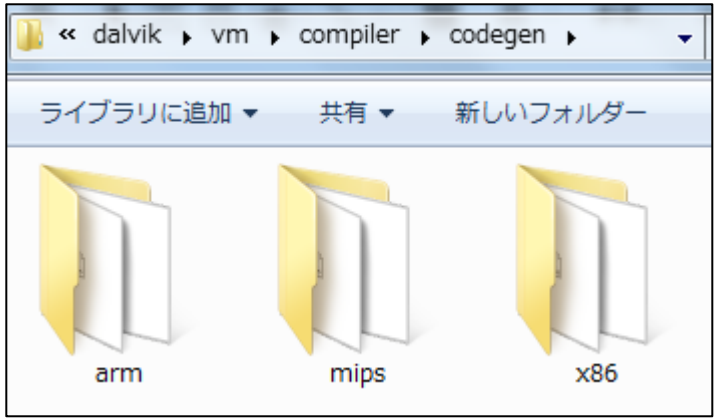
LOCAL_MODULE_STEM_32 := crasher

LOCAL_MODULE_STEM_64 := crasher64

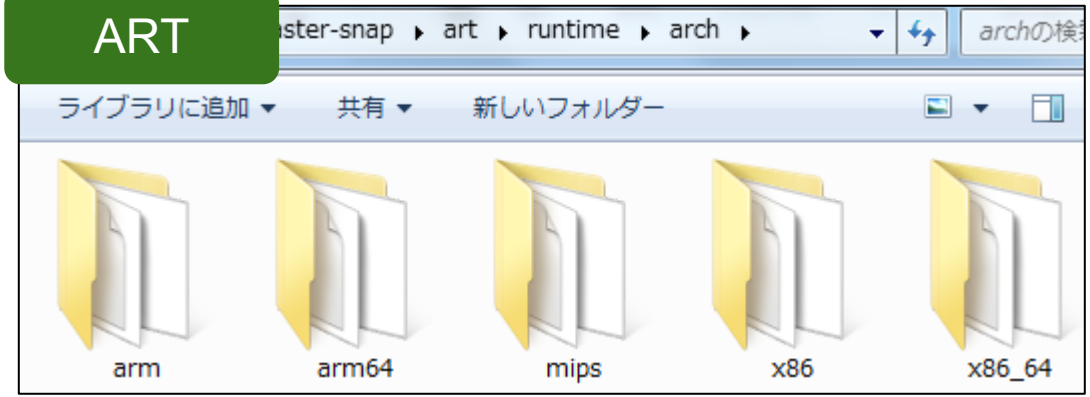
LOCAL_MULTILIB := both

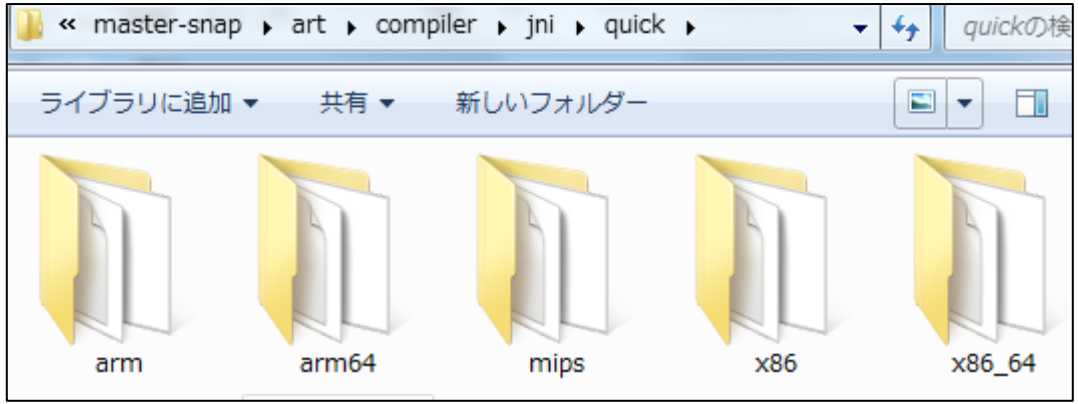
ART!!! (No 64bit support on Dalvik)

Dalvik



ART





According to my experiment to enable ART on Nexus5, 2.3 times of total size of dex/odex is additionally required.

Binder (64bit support)

- TARGET_USES_64_BIT_BINDER
  - Not used… in frameworks/native/libs/binder
- __LP64__ for casting pointer to int

```
void BBinder::attachObject( const void* objectID, void* object, void* cleanupCookie,  object_cleanup_func func){
    Extras* e = mExtras;

    if (!e) {
        e = new Extras;
#ifdef __LP64__
        if (android_atomic_release_cas64(0, reinterpret_cast<int64_t>(e),
                reinterpret_cast<volatile int64_t*>(&mExtras)) != 0) {
#else
        if (android_atomic_cmpxchg(0, reinterpret_cast<int32_t>(e),
                reinterpret_cast<volatile int32_t*>(&mExtras)) != 0) {
#endif
```

OpenGL (64bit support)

In “frameworks/native/opengl/libs/EGL/Loader.cpp”

```
void* Loader::open(egl_connection_t* cnx)
..snip..
#if defined(__LP64__)
    cnx->libEgl   = load_wrapper("/system/lib64/libEGL.so");
    cnx->libGles2 = load_wrapper("/system/lib64/libGLESv2.so");
    cnx->libGles1 = load_wrapper("/system/lib64/libGLESv1_CM.so");
#else
    cnx->libEgl   = load_wrapper("/system/lib/libEGL.so");
    cnx->libGles2 = load_wrapper("/system/lib/libGLESv2.so");
    cnx->libGles1 = load_wrapper("/system/lib/libGLESv1_CM.so");
#endif
```

(cont.)

```
void *Loader::load_driver(const char* kind, egl_connection_t* cnx,
uint32_t mask)
{
..snip..
        pattern.appendFormat("lib%s", kind);
        const char* const searchPaths[] = {
#if defined(__LP64__)
            "/vendor/lib64/egl",
            "/system/lib64/egl"
#else
            "/vendor/lib/egl",
            "/system/lib/egl"
#endif
```

RenderScript is only 32bit but still try to add 64bit

In master/frameworks/rs/driver/runtime/Android.mk

..snip..

Build the ARM version of the library

..snip..

FIXME for 64-bit

LOCAL_32_BIT_ONLY := true

..snip.

Build the x86 version of the library

..snip..

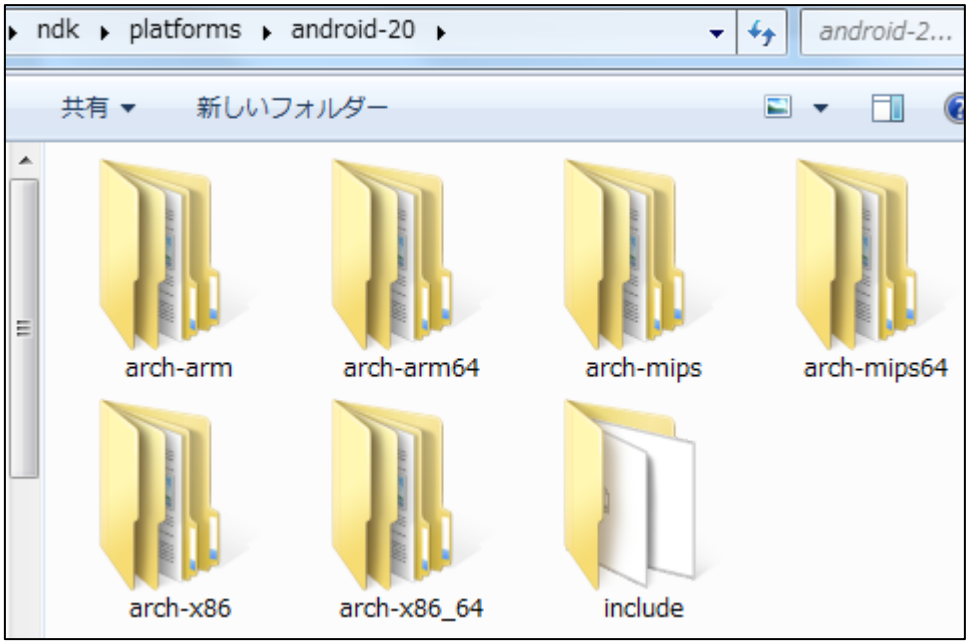
FIXME for 64-bit

LOCAL_32_BIT_ONLY := true

..snip..

- Scan for renderscript files before deciding ABIs.
- Merge "Add aarch64 relocations"
  - commit f981663b7e4568dea18ea1e9988dd0ee0e48a24e
    - Date: Thu May 22 00:08:35 2014 +0000
  - commit 723ba16bac04e65c147742fa08ae2b87da3c0fd5
    - Date: Mon May 19 15:29:32 2014 -0700
    - This adds missing aarch64 relocations for the RS linker.

NDK will support 64bit



android-3.10 branch on kernel/common

3.14 is under experiment phase.

Binder (64bit support)

- android-3.10
  - android: binder: fix binder interface for 64bit compat layer
  - android: binder: fix ABI for 64bit Android
  - android: binder: Support concurrent 32 bit and 64 bit processes.

  ...
- master (userspace)
  - Binder: Use 64 bit pointers in 32 processes if selected by the target
  - ServiceManager: Use 32/64 bit types from new binder header
  - jni: binder 64-bit compile issues

  ...

ARM 64bit support (ARM64 / aarch64)

- fiq_debugger: move into drivers/staging/android/fiq_debugger/
- fiq_debugger: add ARM64 support
- ARM64: copy CONFIG_CMDLINE_EXTEND from ARM
- arm64: cmpxchg: update macros to prevent warnings
- arm64: mm: permit use of tagged pointers at EL0
- Staging: android: binder: Support concurrent 32 bit and 64 bit processes.
- Fix aarch64 build issue with ION
- arm64: ptrace: avoid using HW_BREAKPOINT_EMPTY for disabled events
- arm64: ptrace: fix compat registes get/set to be endian clean
- arm64: debug: consolidate software breakpoint handlers

- ARM64: add option to build Image.gz/dtb combo

```
config BUILD_ARM64_APPENDED_DTB_IMAGE
        bool "Build a concatenated Image.gz/dtb by default"
        depends on OF
        help
          Enabling this option will cause a concatenated Image.gz and list of
          DTBs to be built by default (instead of a standalone Image.gz.)
          The image will built in arch/arm64/boot/Image.gz-dtb

config BUILD_ARM64_APPENDED_DTB_IMAGE_NAMES
        string "Default dtb names"
        depends on BUILD_ARM64_APPENDED_DTB_IMAGE
        help
          Space separated list of names of dtbs to append when
          building a concatenated Image.gz-dtb.
```

TODO: Investigate Linaro :-)

qemu (64bit support)

- kernel
  - https://android-review.googlesource.com/#/q/topic:emu64
- qemu
  - https://android-review.googlesource.com/#/q/project:platform/external/qemu
    - idea133?
  - x86_64 emulator related
    - android: avd: add -x86_64 to kernel filename on x86_64
    - Update 64bit tests to also use EMULATOR_BUILD_64BITS

Appendix : Network

multi network!?

- net: ipv6: autoconf routes into per-device tables
  - ..snip..This causes problems for connection managers that want to support multiple simultaneous network connections and want control over which one is used by default (e.g., wifi and wired).
- netd_client (It has moved to the internal tree!?)
  - Introduce netd_client, a dynamic library that talks to netd.
  - New network selection APIs. (but Abandoned)
    - https://android.googlesource.com/platform/system/core/+/654a41b4fbff21b8d7dfc72a08de05014670fdac/include/netd_client/NetdClient.h

    bool setNetworkForSocket(unsigned netId, int socketFd);
    bool setNetworkForProcess(unsigned netId);
    bool setNetworkForResolv(unsigned netId);
- netd: Replace iface with opaque netid in resolver.

fwmark is actively working in kernel/common, android-3.10

- net: support marking accepting TCP sockets
- net: Use fwmark reflection in PMTU discovery.
- net: add a sysctl to reflect the fwmark on replies
- net: support marking accepting TCP sockets
- net: Use fwmark reflection in PMTU discovery.
- net: add a sysctl to reflect the fwmark on replies
- Introduce fwmarkd: a service to set the fwmark of sockets.
- Introduce fwmarkd: a service to set the fwmark of sockets.
- Set kernel proc files for fwmark reflection and table numbers for RAs.
- net: add a sysctl to reflect the fwmark on replies
- net: support marking accepting TCP sockets

Appendix : ADF : Atomic Display Framework

from lwn:

> ADF is an experimental display framework that I designed after experimenting with a KMS-based hardware composer for Android. ADF started as an proof-of-concept implemented from scratch
>
> ..snip..
>
> ADF represents display devices as collections of overlay engines and interfaces. Overlay engines (struct adf_overlay_engine) scan out images and interfaces (struct adf_interface) display those images. Overlay engines and interfaces can be connected in any n-to-n configuration that the hardware supports.
>
> Clients issue atomic updates to the screen by passing in a list of buffers (struct adf_buffer) consisting of dma-buf handles, sync fences, and basic metadata like format and size. If this involves composing multiple buffers, clients include a block of custom data describing the actual composition (scaling, z-order, blending, etc.) in a driver-specific format.
>
> Drivers provide hooks to validate these custom data blocks and commit the new configuration to hardware. ADF handles importing the dma-bufs and fences, waiting on incoming sync fences before committing, advancing the display's sync timeline, and releasing dma-bufs once they're removed from the screen.
>
> ADF represents pixel formats using DRM-style fourccs, and automatically sanity-checks buffer sizes when using one of the formats listed in drm_fourcc.h. Drivers can support custom fourccs if they provide hooks to validate buffers that use them.
>
> ADF also provides driver hooks for modesetting, managing and reporting hardware events like vsync, and changing DPMS state. These are documented in struct adf_{obj,overlay_engine,interface,device}_ops, and are similar to the equivalent DRM ops.

Appendix : ADF : Recent related commits in AOSP

- Kernel
  - video: adf: export the adf_attachment_allow symbol to modules.
  - video: adf: replace fbdev helper's open flag with refcount
  - video: adf: ensure consistent alignment on userspace facing structs
  - video: adf: adf_memblock_export symbol should be exported
  - video: adf: note adf_format_validate_yuv's origin
  - video: adf: add buffer padding quirk
  - video: adf: use rb_erase in adf_obj_destroy.
  - video: adf: memblock: map buffer for dma
  - video: adf: fbdev: add stubs for kernels without ADF_FBDEV
  - Add policies for Atomic Display Framework
- system/core
  - add libadf / add libadfhwc
  - rootdir: add ueventd.rc rule for adf subsystem

Appendix : selinux

- selinux: Report permissive mode in avc: denied messages.
  - This is related to selinux’s audit (access vector cache)
- Related new in AOSP
  - auditd is added in logd
    - https://android-review.googlesource.com/#/q/auditd+logd

Appendix : 64bit VM

http://www.oracle.com/technetwork/java/hotspotfaq-138619.html#64bit_description

Appendix

- Silvermont
  - http://www.4gamer.net/games/047/G004743/20130913026/
  - https://android-review.googlesource.com/92607
    - This is used for Baytrail targets.
- ARMv8 Architecture
  - http://www.arm.com/files/downloads/ARMv8_Architecture.pdf