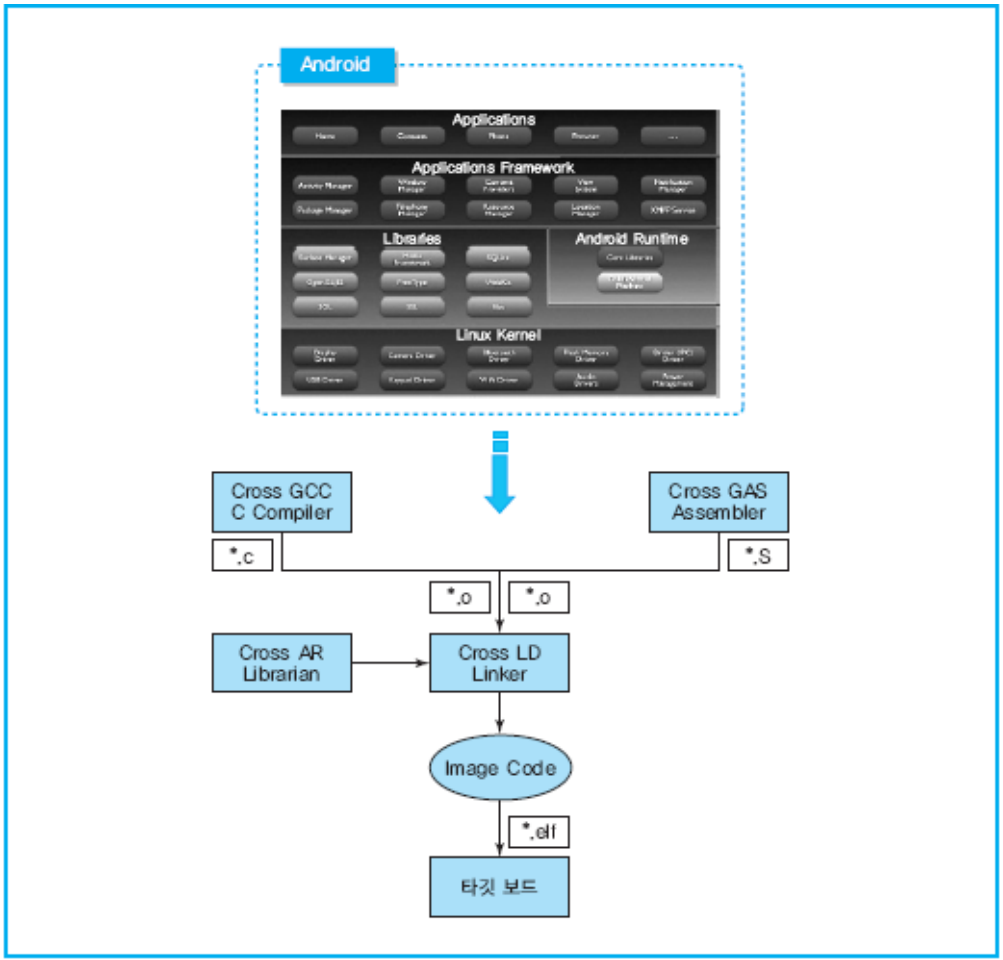
13장 실제 타깃 제작하기

시작하면서

- **목차**
  - 13.1 타깃 개발 개요
  - 13.2 안드로이드 모바일 하드웨어 플랫폼
  - 13.3 실제 타깃 개발 환경 구축
  - 13.4 실제 타깃 커널 제작
  - 13.5 타깃으로 이미지 다운로드
  - 13.6 파일시스템 구성
  - 13.7 부팅 및 실행

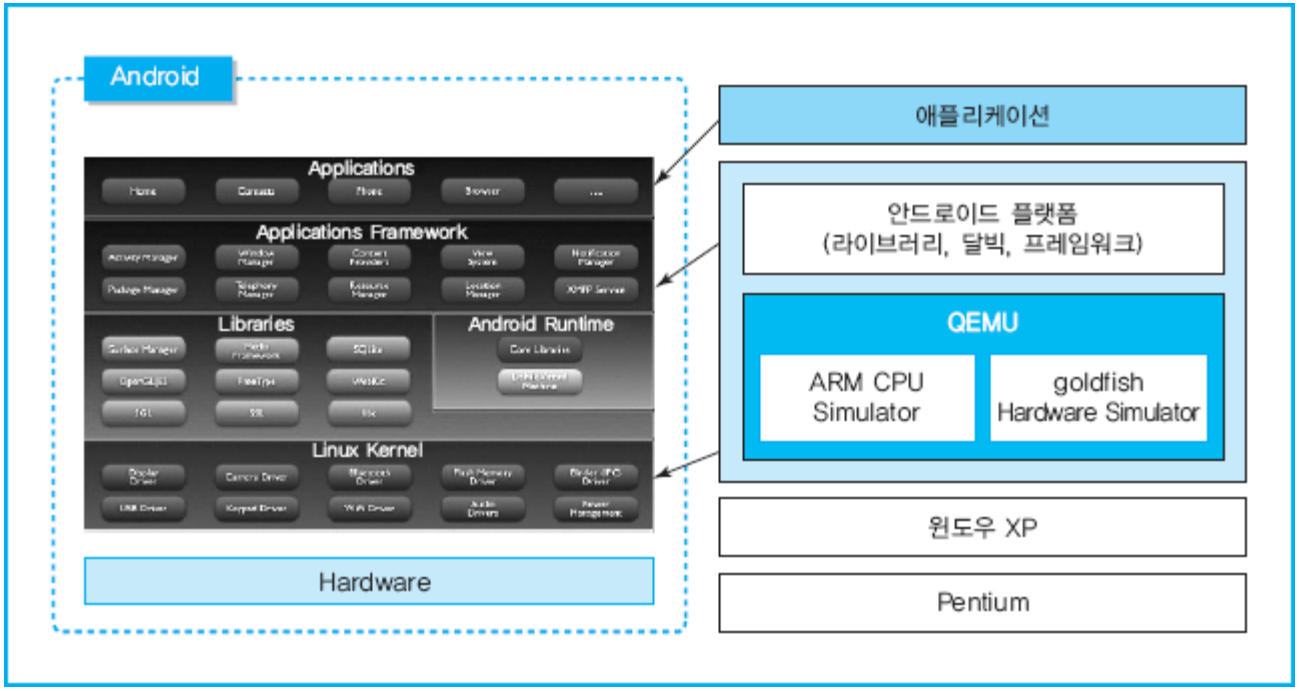
13.1 타깃 개발 개요

- **porting**
  - 어떻게 안드로이드 소프트웨어 플랫폼을 빌드하여 포팅하는가?
- 계층구조의 플랫폼을 빌딩하는 방법?
- **교차 컴파일**
- **링크**
- **부팅**
- **실제 ARM에서 실행**



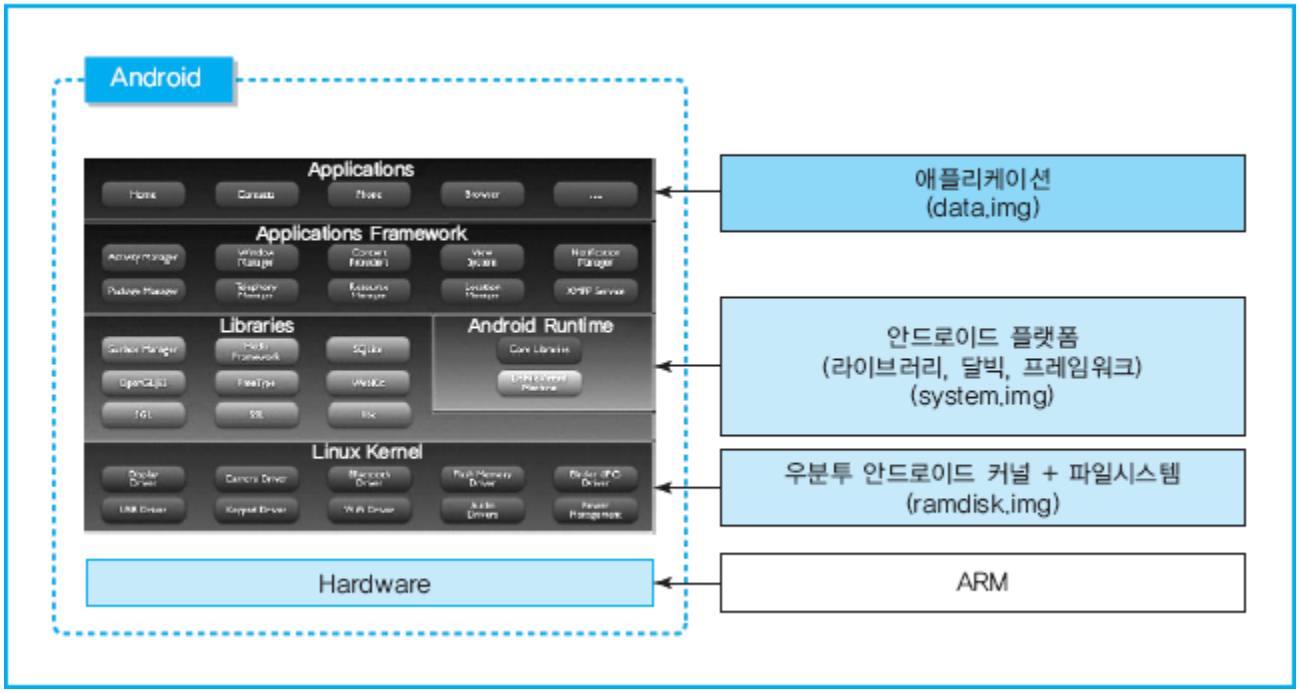
[그림 13-1] 안드로이드 플랫폼 포팅 개요

13.1 타깃 개발 개요



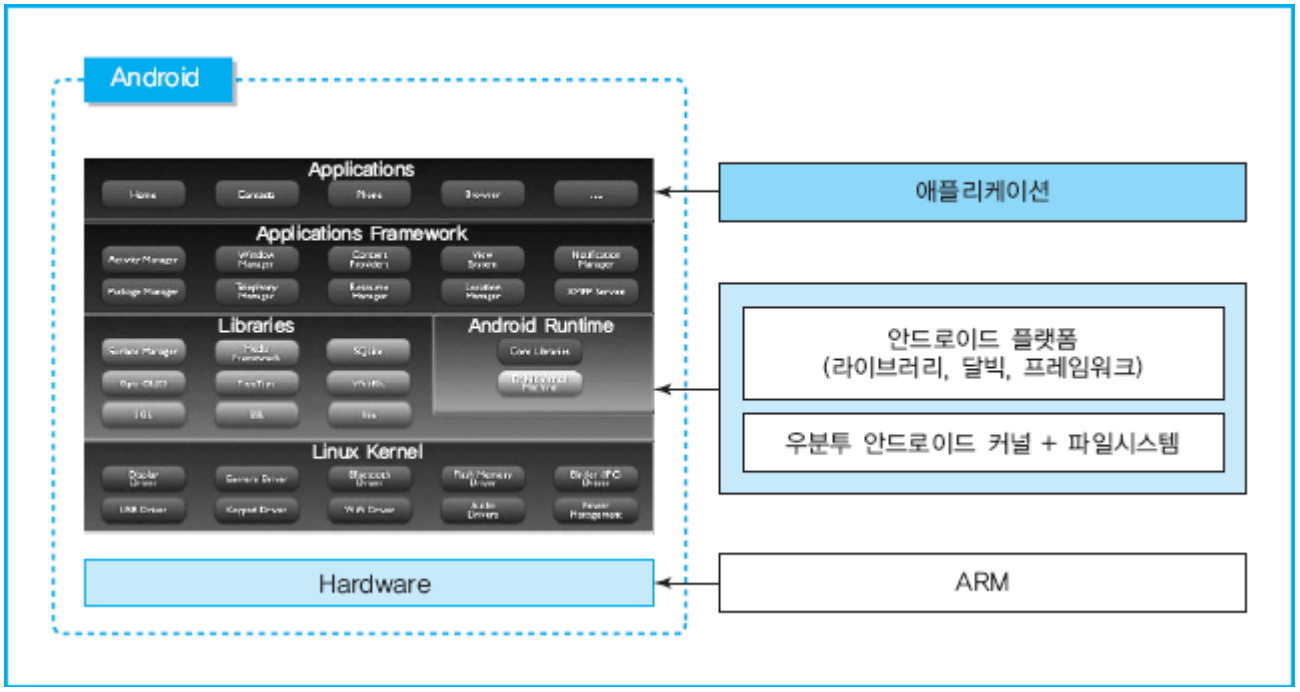
[그림 13-2] 에뮬레이터 기반 에뮬레이터 플랫폼 구성

13.1 타깃 개발 개요



[그림 13-3] 일반 타깃 보드(하이버스 X-Hyper320TKU) 포팅 예

13.1 타깃 개발 개요



[그림 13-4] 모바일 전용 타깃 보드(하이버스 H-AndroSV210) 포팅 예

13.2 모바일 하드웨어 플랫폼

- **플랫폼 구성 및 특징**
- 안드로이드 구현 ARM 필요 사양

• ARM926 또는 그 이상(ARM11 , ARM MPC, ARM Cortex)을 포함한 SOC
주의 : ARMv4 기반인 ARM920T는 동작하지 않음
• Thumb, MMU, EABI 를 지원
• Display/Frame buffer (Frame buffer는 double buffer/page flipping을 지원)
• Keyboard
• USB(optional)
• RTC(optional)
• Serial console
• Storage (64MB 이상) e.g. NFS / USB stick / NAND / NOR / MMC /SD card
• 충분한 main memory (SDRAM, 32 MB 이상)

13.2 모바일 하드웨어 플랫폼

◉ 안드로이드 구현 ARM SoC 종류

●H-AndroSV210 : 강력한 성능을 자랑하는 Cortex-A8 기반 1GHz의 Samsung 최신 프로세서 S5PV210을 탑재한 보드
●X-Hyper320(PXA 320)
●OMAP1(ARM v5 ARM926)
●OMAP2(ARM v6 ARM11)
●Sharp Zaurus SL-C760(PXA255)
●Sharp Zaurus SL-C3000(PXA270)
●A&W6410(삼성 S3C6410)

13.2 모바일 하드웨어 플랫폼

〈표 13-1〉 H-AndroSV210 사양

품명	내용
CPU	Processor(SamSung Cortex-A8 S5PV210)
DDR SDRAM	DDR2 SDRAM 768Mbyte
NAND Flash	NAND Flash 256Mbyte
전원	리튬-이온 배터리 사용(USB 충전)
Wi-Fi, Bluetooth	MBH7BWZ04(SDIO) - IEEE 802.11 b/g
USB	USB 2.0 Host 1 port, USB 2.0 OTG 1port
Audio	ALC5622(I2S) - mic / speaker AMP and jack
Camera	CMOS 1.5M Pixel
SD	SDIO 1Slot
Touch Screen	저항막 방식
Display	4.8 inch 800 x 480 TFT LCD/HDMI OUTPUT
GPS	SiRF
Sensor	3 Axis acceleration, Digital Compass
LED	Power:1EA, Charging:1EA, App:3EA
KEY	7EA
Debug Board	Debug UART, Comm UART, JTAG I/F, GPIO, PI, I2C, 어댑터

[그림 13-5] 타깃 장치

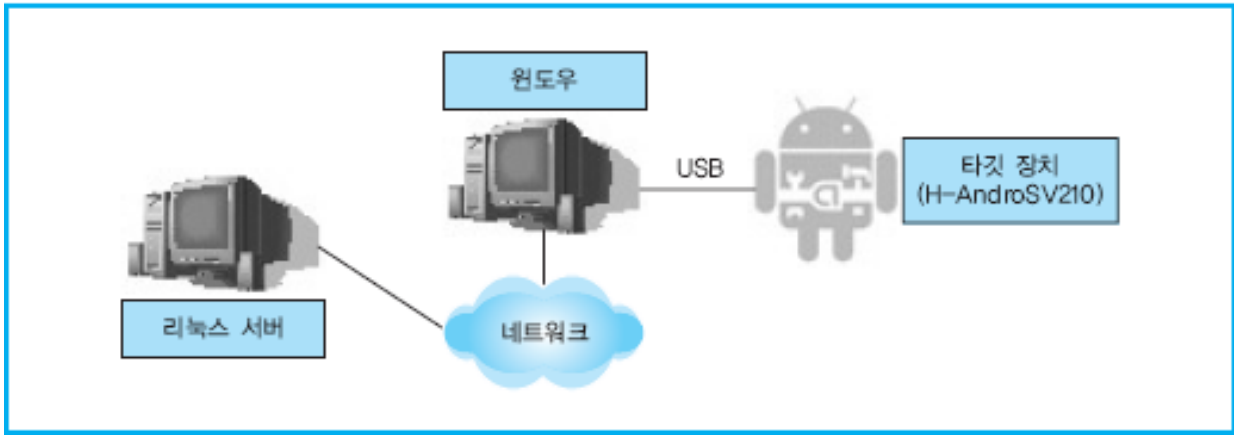
13.2 모바일 하드웨어 플랫폼

- **추상계층**
  - 하드웨어 제어 및 관리

[그림 13-6] 하드웨어 추상계층

13.3 실제 타깃 개발 환경 구축

- **실제 타깃 개발 지원 환경**
  - 리눅스 서버 설치
  - 통신 환경



[그림 13-7] 실제 타깃 개발 지원 환경

13.3 실제 타깃 개발 환경 구축

- **교차 개발 환경 설계**
  - 교차 컴파일러
  - 교차 어셈블러
  - 링커
  - 라이브러리
  - 디버깅 툴

13.3 실제 타깃 개발 환경 구축

- **사용자 인터페이스**
  - 리눅스 쉘 프로그램 선정
  - Bash 환경
  - Busybox
- **파일시스템**
  - Ext, jffs, yaffs
- **다운로드 방법**
  - JTAG, BDM
  - RS-232
  - USB – H-AndroSV210
  - Tftp(이더넷)
- **부팅**
  - Elf

●데이터는 RAM, 코드는 ROM에서 실행

●ROM에서 RAM으로 복사한 후 실행

●데스크톱에서 타깃의 RAM으로 직접 다운로드하여 실행

13.3 실제 타깃 개발 환경 구축

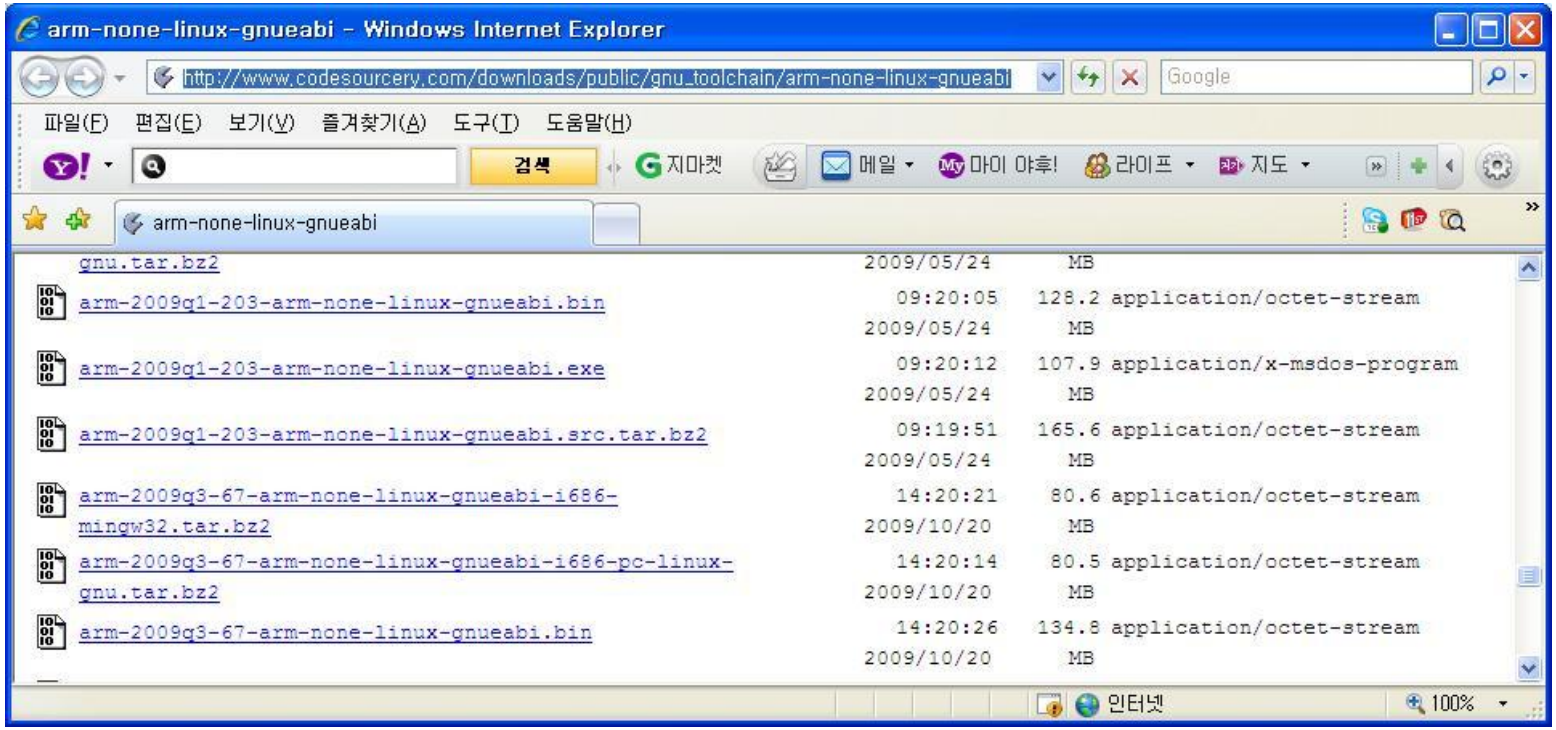
- **개발 지원 도구 구축**
  - 툴체인 설치
  - 컴파일러
  - 어셈블러
  - 로더
  - Native compiler
  - cross compiler
  - ARM ABI -> ARM EABI

- **툴체인 준비 방법**
  - Codesourcery 툴체인 소스 다운로드
  - 기존 빌드된 툴체인 사용

13.3 실제 타깃 개발 환경 구축

- **[실습 13-1] 툴체인 설치**

(1)http://www.codesourcery.com/downloads/public/gnu_toolchain/armnone-linux- gnueabi 사이트를 방문하면 [그림 13-8]처럼 다양한 플랫폼에 대한 툴체인이 있다.

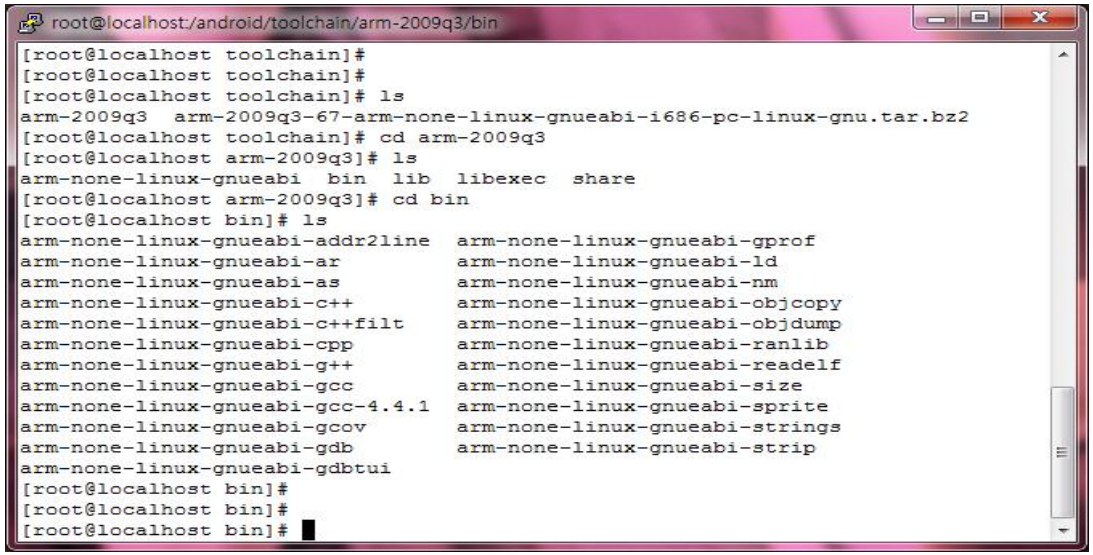


[그림 13-8] 툴체인 선택 및 다운로드

13.3 실제 타깃 개발 환경 구축

- **[실습 13-] 툴체인 설치**

(2) arm-2009q3-67-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2
툴체인 파일을 /toolchain 디렉터리에 다운로드받는다.
(3) bzip2-d arm-2009q3-67-arm-none-linux-gnueabi-i686-pc-linuxgnu.tar.bz2
(4) tar xvf arm-2009q3-67-arm-none-linux-gnueabi-i686-pc-linuxgnu.tar
bzip2와 tar를 사용하여 /toolchain 디렉터리에 저장된 툴체인 파일에 대하여 압축을 해제한다. 그러면 실제 타깃에서 사용할 커널 및 파일시스템을 만들고 포팅을 하기 위한 각종의 교차 컴파일 도구가 /toolchain 디렉터리에 생성된다. [그림 13-9]는 설치된 툴체인의 종류를 보여준다.



[그림 13-7] 설치된 툴체인 종류

13.3 실제 타깃 개발 환경 구축

- **busybox 설치**
  - 기존 임베디드 리눅스 busybox와의 차이점 : static 옵션 빌드
  - 기존 busybox(1.8MB) – 안드로이드 busybox (150KB)

- **지원도구 설치**
  - **Flex, bison, gperf, libsdl-dev, libesd0-dev, libwxgtk2.6-dev(optional),**
    build-essential, zip, curl
  - **Valgrind**
  - **libreadline**
  - **Repo**

13.4 실제 타깃 커널 제작

- **소스 : 3GB, 128,000 파일로 구성**
  - 우분투 리눅스 소스
  - 안드로이드 2.2 플래폼 소스 (2.1 GB)
  - U-Boot 소스
  - 문서
- **빌드**
  - Goldfish, QEMU 비활성화

13.4 실제 타깃 커널 제작

- 안드로이드 커널 빌드

[그림 13-10] 커널 제작 및 부팅 절차

13.4 실제 타깃 커널 제작

◉ 안드로이드 커널 소스 준비

```
# mkdir ~/bin // bin 디렉토리를 생성
# curl http://android.git.kernel.org/repo > ~/bin/repo //repo 스크립트 다운로드
# chmod 755 ~/bin/repo // repo를 실행가능하게 설정
# cp ~/bin/repo /bin
# repo init -u git://android.git.kernel.org/platform/manifest.git // repo init 를 사용하여 repo의 최신 버전을 다운로드 받는다.
# repo sync
```

```
$ vi ~/.bashrc                    // vi로 ~/.bashrc에 다음과 같이 환경 변수 추가
export PATH=/home/<your_home>/bin:$PATH:
```

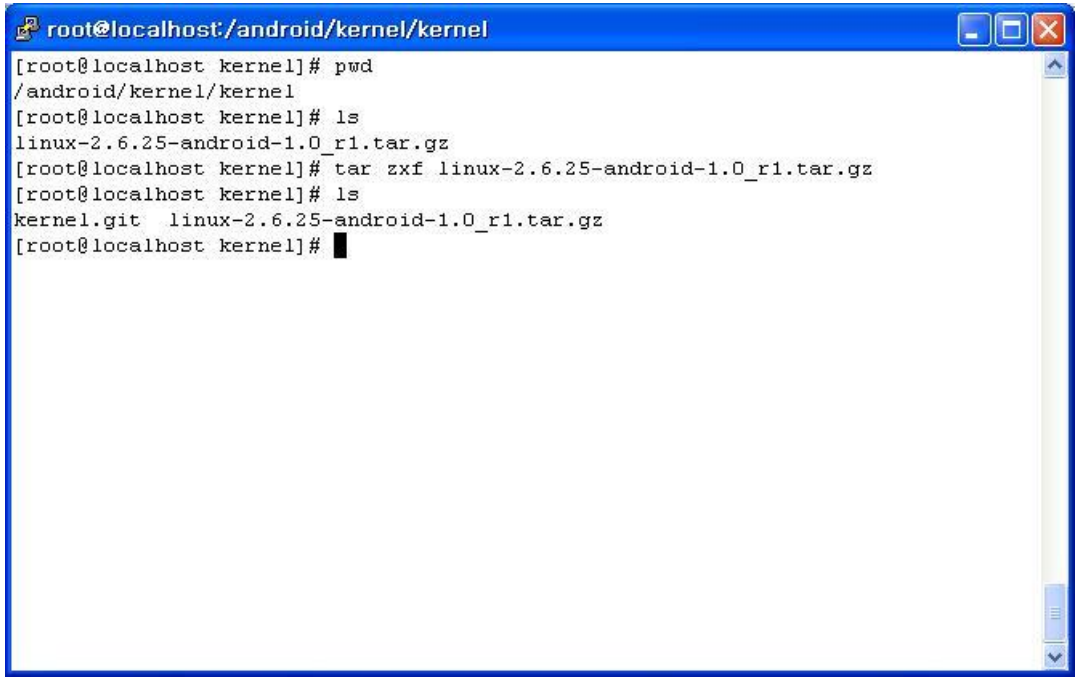
13.4 실제 타깃 커널 제작

- **[실습 13-2] 커널 소스 다운로드하기**

  (1) 안드로이드 홈페이지에서 linux-2.6.25-android-1.0_r1을 리눅스 서버에 /kernel 디렉터리를 만들어 다운로드한다.

  (2) tar zxf linux-2.6.25-android-1.0_r1.tar.gz

  /kernel 디렉터리에 압축 해제하면 [그림 13-11]처럼 kernel.git 디렉터리가 생성된다.



[그림 13-13] 커널소스 압축해제

13.4 실제 타깃 커널 제작

- **[실습 13-3] 커널 소스 패치하기**

  (1) gzip linux-2.6.25-android-1.0_r1-xhyper320tku_patch_v1.gz | patch -p1

  패치 파일이 압축 파일 형태로 제공되므로 gzip을 사용하여 패치 파일의 압축을 해제하고 소스에 패치한다. 패치는'patch'명령어와 옵션을 사용하여 이루어진다. 압축 해제는 /kernel.git 디렉터리에 한다.

  (2) /kernel.git 디렉터리로 이동하여 패치 결과를 확인한다.

  [그림 13-12]와 같은 소스 폴더가 생성된다. 일반 리눅스 소스 폴더에서 보던 것과 유사하다는 것을 알 수 있다.

[그림 13-14] 리눅스 커널 소스 구조

13.4 실제 타깃 커널 제작

◉ 안드로이드 커널 설정

일반적인 안드로이드 kernel 2.6.25 설정 내용

- Select Target Board
- CPU Implementations
- General Setup
- System Type
- Kernel features : Use the ARM EABI to compile the kernel
  - Power management options
  - Memory Technology Device (MTD) support
  - NAND device support
  - Misc devices
  - Support for frame buffer devices : frame buffer-android
  - Console display driver support : frame buffer console support
  - Bootup logo : standard 224-color linux logo
  - Touchscreen
  - Android (RAM 버퍼 console, Android power driver, Binder IPC Driver)
  - File System

13.4 실제 타깃 커널 제작

- IPC 설정
- Low memory killer , logger, 전력관리, yaffs2 설정

h_androsv210_config 파일 주요 내용

```
## Automatically generated make config: don't edit
# Linux kernel version: 2.6.32.9
# Thu Sep 30 22:06:51 2010
CONFIG_HAVE_PWM=y
CONFIG_GENERIC_GPIO=y
CONFIG_DEFCONFIG_LIST="/lib/modules/$UNAME_RELEASE/.config"
## General setup
CONFIG_BLK_DEV_INITRD=y   // 초기 램디스크(initramfs) 사용 설정
## System Type
CONFIG_ARCH_S5PV210=y
CONFIG_PLAT_SAMSUNG=y
## Boot options
CONFIG_S3C_LOWLEVEL_UART_PORT=2
## Power management
```

- 컴파일 및 빌드

13.4 실제 타깃 커널 제작

- **[실습 13-4] 커널 빌드하기**

(1) 터미널 창을 열고 root 계정으로 로그인한다.

(2) CD-ROM에 H-AndroSV210의 BSP CD를 삽입한다.

(3) 우분투 9.10을 기준으로 CD는 자동으로 마운트되는데 /cdrom이란 폴더에 자동 마운트된다.

(4) mkdir /mnt/cdrom/KERNEL 명령으로 /mnt/cdrom/KERNEL이란 폴더를 생성한다.

(5)" cp -r /cdrom /mnt/cdrom/KERNEL"명령으로 CD-ROM의 압축 커널 파일 android_kernel_2.6.32_sv210.tgz 내용을 /mnt/cdrom/KERNEL에 복사한다.

(6)" ls /mnt/cdrom/KERNEL"으로 복사한 내용을 확인한다.

(7) 압축 커널 파일 android_kernel_2.6.32_sv210.tgz을 해제한다.

[그림 13-13]처럼 CD 이미지가 있는 폴더로 이동하여 tar 명령을 이용해서 압축을 해제한다.

[그림 13-13] H-AndroSV210의 커널 파일 준비

13.4 실제 타깃 커널 제작

- **[실습 13-4] 커널 빌드하기**

  (8) cd android_kernel_2.6.32_sv210; ls

  압축을 해제한 커널의 디렉터리로 이동하여 ls 명령으로 압축 해제된 파일 구성 내용을 [그림 13-14]처럼 확인해 본다.

[그림 13-14] 커널 파일 구성 확인

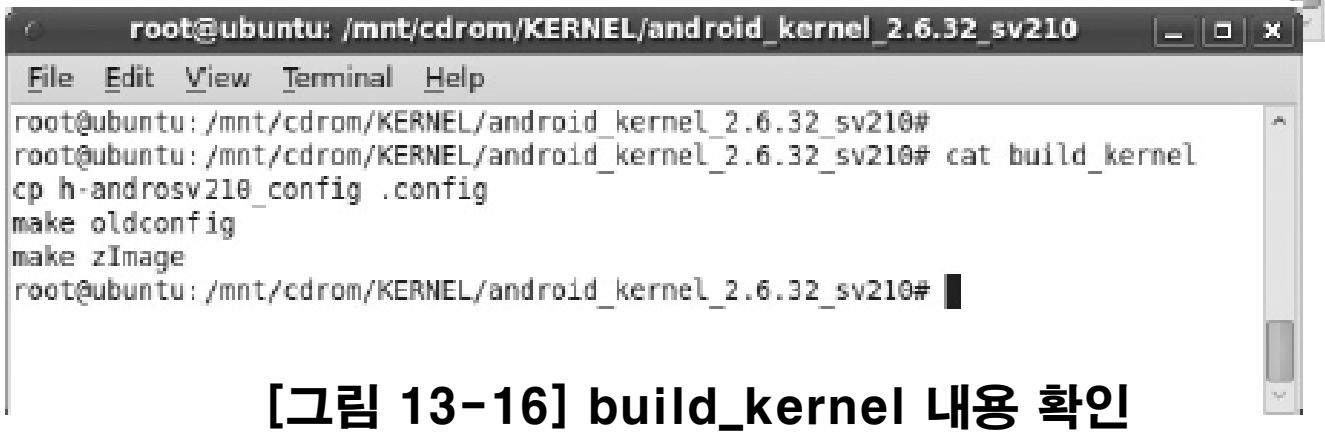
13.4 실제 타깃 커널 제작

- **[실습 13-4] 커널 빌드하기**

  (9) ./build_kernel

  build_kernel이라는 스크립트를 이미 정의해두어 사용자가 일일이 커널 설정을 하는 것을 간소화하고 쉽도록 구성해놓았다. "./build_kernel"을 실행하여 [그림 13-15]처럼 커널 빌드를 한다.

[그림 13-15] 커널 빌드 작업



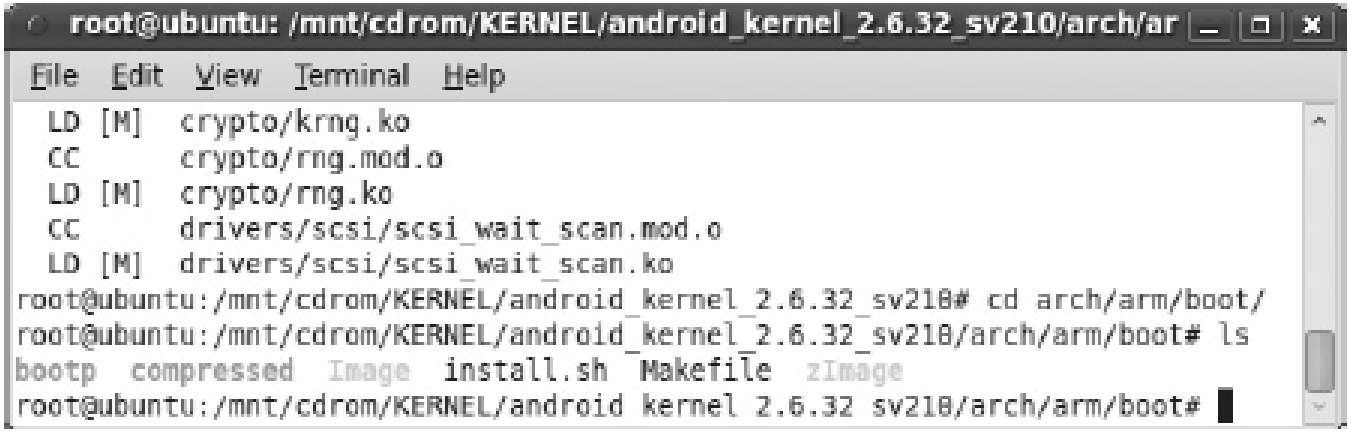
[그림 13-16] build_kernel 내용 확인

13.4 실제 타깃 커널 제작

- **[실습 13-4] 커널 빌드하기**

  (10) cd arch/arm/boot; ls

  컴파일이 완료되면 [그림 13-17]과 같이 프롬프트가 출력이 되며, arch/arm/boot/zImage라는 파일로 컴파일된 3.2MB 크기의 zImage 이미지가 생성된 것을 볼 수 있다.

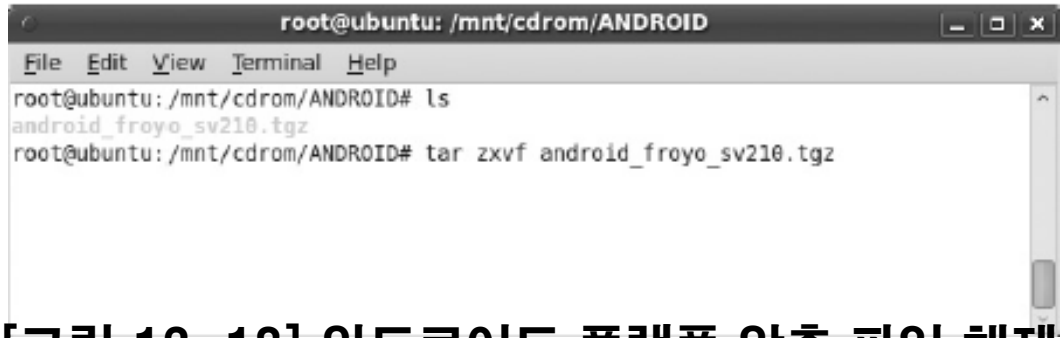


[그림 13-17] 커널 이미지 생성 확인

13.4 실제 타깃 커널 제작

[실습 13-5] 안드로이드 플랫폼 이미지 빌드하기

(1) CD-ROM에 H-AndroSV210의 BSP CD를 삽입한다.
(2) 우분투 9.10을 기준으로 CD는 자동으로 마운트되는데 /cdrom이란 폴더에 자동 마운트된다.
(3) mkdir /mnt/cdrom/ANDROID 명령으로 /mnt/cdrom/ANDROID란 폴더를생성한다.
(4)“ cp -r /cdrom /mnt/cdrom/ANDROID”명령으로 CD-ROM의 압축 플랫폼 파일 android_froyo_sv210.tgz 내용을 /mnt/cdrom/ANDROID에 복사한다.
(5)“ ls /mnt/cdrom/ANDROID”으로 복사한 내용을 확인한다.
(6) tar zxvf android_froyo_sv210.tgz
압축 플랫폼 파일 android_froyo_sv210.tgz을 [그림 13-18]처럼 해제한다. 파일 용량이 크기 때문에 압축 해제 과정은 PC에 따라 수 분이 걸릴 수 있다.



[그림 13-18] 안드로이드 플랫폼 압축 파일 해제

13.4 실제 타깃 커널 제작

- **[실습 13-5] 안드로이드 플랫폼 이미지 빌드하기**

(7) cd android_froyo_sv210; ls

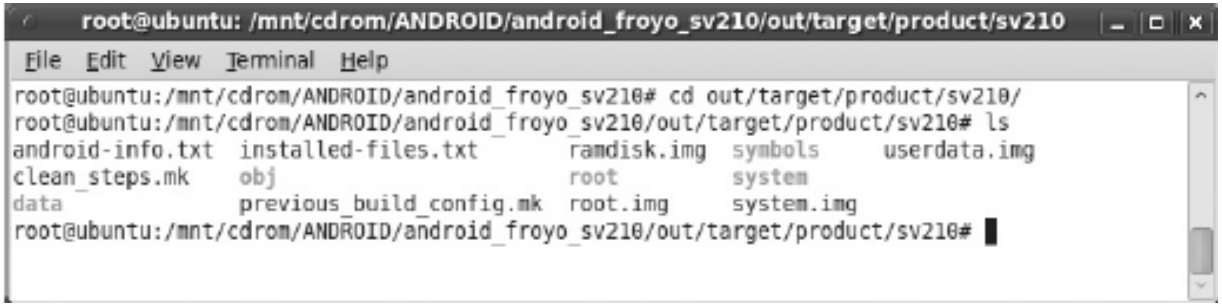
압축 해제를 완료했으면 압축이 해제된 android_froyo_sv210 폴더로 이동한다. 그리고 ls 명령어로 폴더 내용을 확인한다.

(8) ./build_android.sh

build_android.sh 파일은 system.img와 root.img 같은 안드로이드 플랫폼 이미지를 생성하기 위해 이미 작성해놓은 스크립트 파일이다.

(9) cd out/target/product/sv210; ls

안드로이드 이미지의 생성은 PC 사양에 따라 수 시간이 소요된다. 안드로이드 이미지의 빌드가 완료되면 out/target/product/sv210 이하에 이미지가 생성된 것을 [그림 13-19]처럼 볼 수 있다. 기본 파일 시스템 이미지 root.img와 플랫폼 이미지 파일 system.img가 생성된다.



[그림 13-19] 안드로이드 플랫폼 이미지 생성

13.5 타깃으로 이미지 다운로드

- zImage, root.img, system.img 생성된 이미지 다운로드

13.5 타깃으로 이미지 다운로드

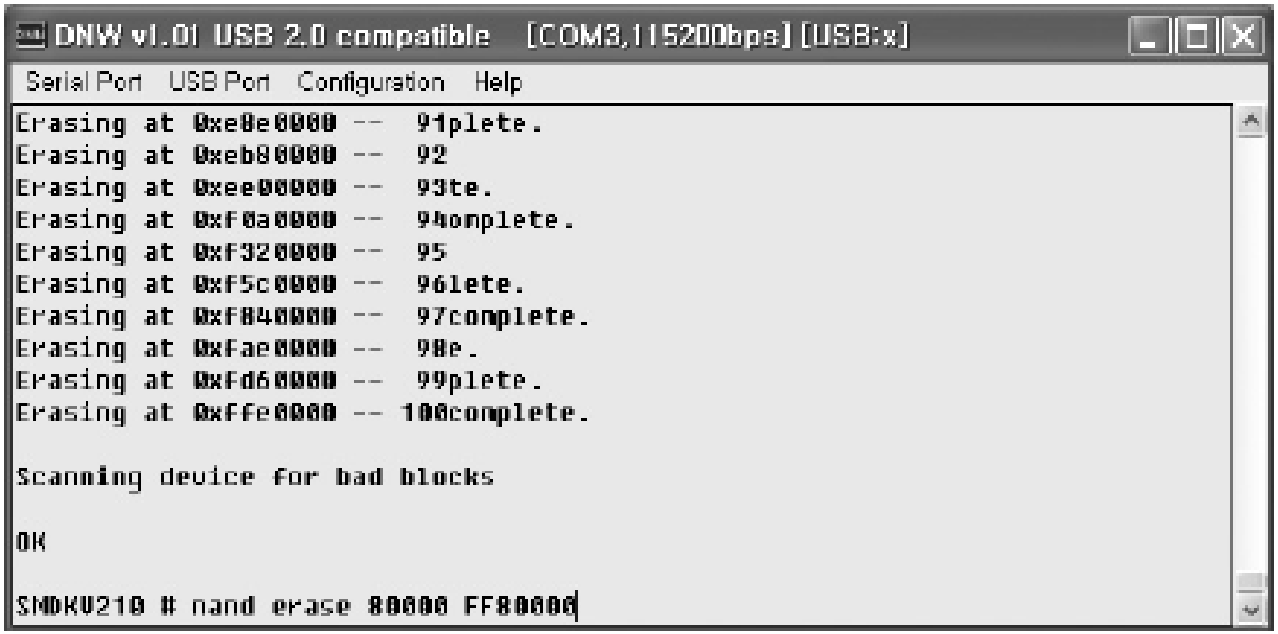
- **[실습 13-6] 안드로이드 이미지 다운로드**

  (1) nand scrub

  다운로드하기에 앞서 이미 Nand Flash에 올라가 있던 내용을 삭제한다.

  (2) nand erase 80000 FF8000

  nand scrub 작업이 모두 진행되었으면 [그림 13-20]처럼 NAND 플래시의 내용을 모두 지운다. 메모리 영역 중 80000~FF8000의 내용을 삭제한다.



[그림 13-20] NAND 메모리 지우기

13.5 타깃으로 이미지 다운로드

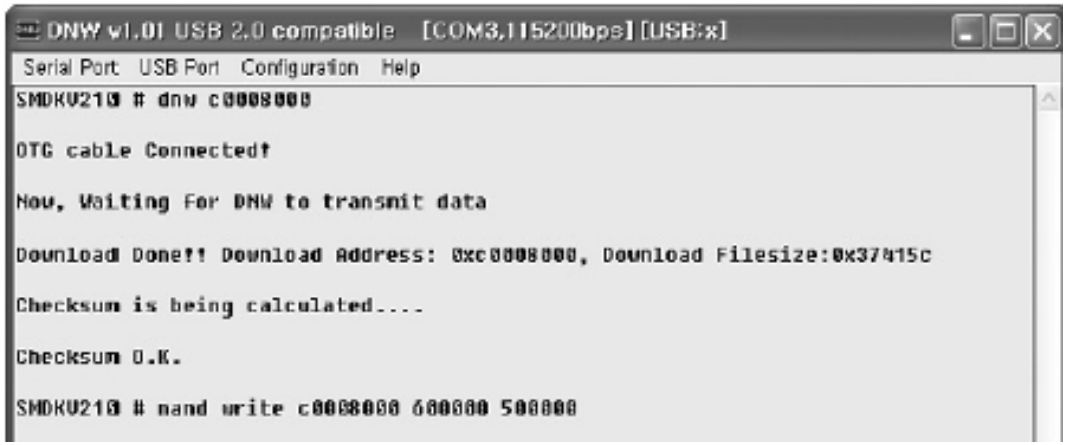
- **[실습 13-6] 안드로이드 이미지 다운로드**

  (3) dnw c0008000

  Nand Flash의 내용을 삭제하였으면 커널의 이미지를 다운로드하기 위하여 USB Port의 통신 연결을 설정한다. 그리고 zImage 파일이 있는 곳의 경로를 지정한다. 파일이 선택되면 다운로드가 진행된다. 결과적으로 zImage라는 커널 이미지가 SDRAM에 다운로드되었다. SDRAM의 경우 휘발성 메모리이기 때문에 SDRAM에 다운로드된 zImage를 Nand Flash로 Write해주어야 한다.

  (4) nand write c0008000 600000 500000

  Nand Flash에 zImage를 [그림 13-21]처럼 쓰기를 한다.



[그림 13-21] zImage 다운로드하기

13.5 타깃으로 이미지 다운로드

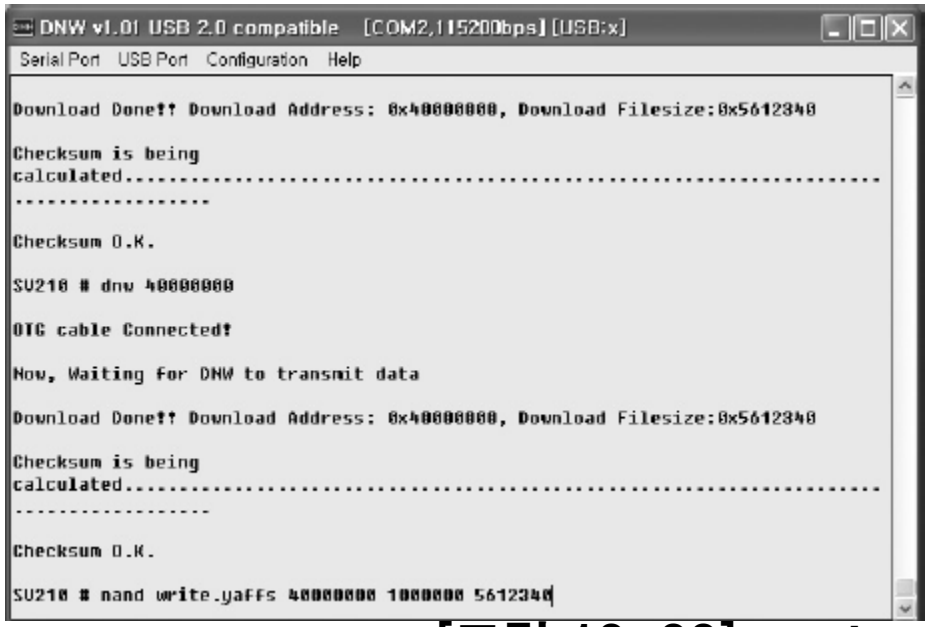
- **[실습 13-6] 안드로이드 이미지 다운로드**

  (5) dnw 40000000

  다음은 앞에서 생성한 system.img를 H-AndroSV210에 다운로드한다. 안드로이드는 root와 system 이미지로 나누어져 있기 때문에 2개의 파일시스템 이미지를 업로드하여야 한다. (3)의 과정을 반복하는데, 차이점은 다운로드 주소가 다르다는 것이다.

  (6) nand write.yaffs 40000000 1000000 [byte size of system.img]

  Nand Flash에 system.img를 [그림 13-22]처럼 쓰기를 한다.



(7) root.img 이미지를 다운로드하여 플래시에 저장한다.

[그림 13-22] system.img 다운로드하기

13.6 파일시스템 구성

- 임시 루트 파일시스템
- Init 프로세스 실행시 루트 파일시스템 마운팅
- Initrd, initramfs
- H-hyper320TKU : initrd (block device, 드라이버)
- H-AndroSV210 : initramfs (non-block device, 캐시 메모리 사용)
- Yaffs : root.img, system.img 생성

13.6 파일시스템 구성

build_android.sh 파일 내용

```
#!/bin/bash
SEC_PRODUCT=sv210
CLEAN_BUILD='false'
#CLEAN_BUILD='true'
ROOT_DIR=$(pwd)
BUILD_OUT_DIR="$ROOT_DIR/out/target/product/$SEC_PRODUCT"
CPU_JOB_NUM=$(grep processor /proc/cpuinfo | awk '{field=$NF};END{print field+1}')
function check_exit()
{   if [ $? != 0 ]
    then
        exit $?
    fi
}function build_android()
{   echo
    echo '[[[[[[[ Build android platform ]]]]]]]'
    echo
        START_TIME='date +%s'
.....
}function make_output()
{   cd $ROOT_DIR
    echo
    echo '[[[[[[[ BusyBox, Bash Copy ]]]]]]]'
    echo
    cp -a $ROOT_DIR/vendor/rootfs_base/* $BUILD_OUT_DIR/system
    $ROOT_DIR/vendor/mkyaffs2image $BUILD_OUT_DIR/system $BUILD_OUT_ DIR/system.img
    $ROOT_DIR/vendor/mkyaffs2image $BUILD_OUT_DIR/root $BUILD_OUT_ DIR/root.img
    sync
}echo
echo '               Build android for 'HyBus_$SEC_PRODUCT''
echo
## Clean output folder
echo ok success !!!
exit 0
```

13.7 부팅 및 실행

- **[실습 13-7] 안드로이드 부팅**

  (1) H-AndroSV210을 재부팅한다.

  (2) 잠시 후 H-AndroSV210이 정상적으로 부팅되는 것을 그림처럼 볼 수 있다.

[그림 13-23] 안드로이드 부팅

◉끝