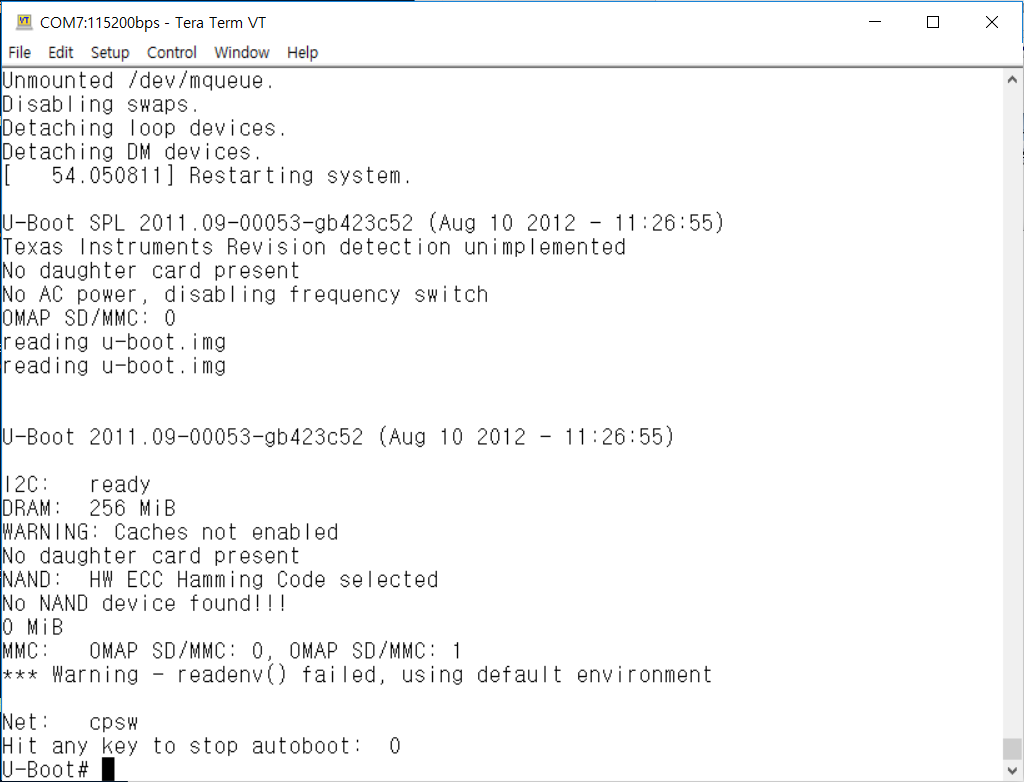
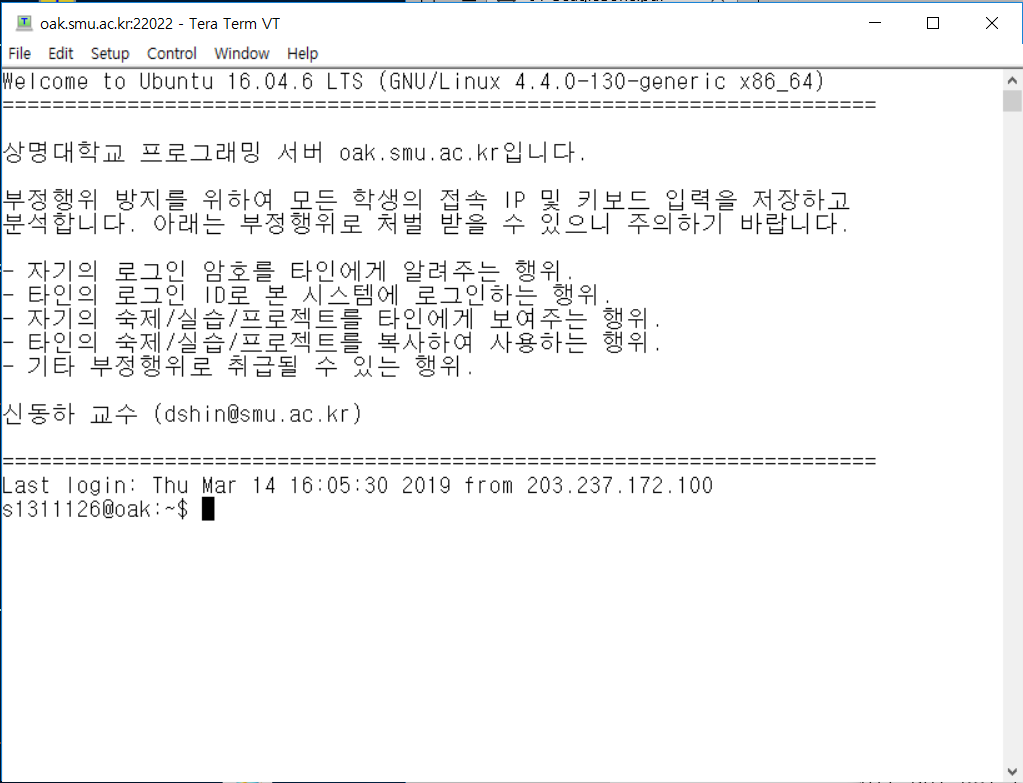
1.2



2.2



3.1 명령어 “make” 수행 시 화면에 나타나는 내용을 쓰고 그 의미를 설명하라.

-> make 명령어는 Makefile이 존재하는 디렉토리에서 make 명령어만 사용하면, Makefile에 적혀있는 대로 컴파일러에 명령하여 SHELL 명령어를 수행합니다.

Arm-eabi-gcc // –g –mcpu=cortex-a8 –mfpu=neon // -c –o start.o start.S

-> 컴파일러 이름(CC) // 컴파일시 추가옵션(ASFLAGS) // 컴파일옵션

-> arm-eabi-gcc 컴파일러를 사용하여 해당 옵션을 적용하고 start.o파일을 start.S파일로 컴파일합니다. 메인함수가 없어도 오류가 나도록 하지 않기 위해 –c 옵션을 추가한 것으로 보입니다.

Arm-eabi-gcc // –g –Wall –marm –mcpu=cortex-a8 –mfpu=neon –mfloat-abi=softfp –mabi=aapcs-linux –l . –l../libuart // -c –o beaglebone.o beaglebone.c

-> 컴파일러 이름(CC) // 컴파일 옵션 (CFLAGS) // object file을 c파일로 컴파일

-> arm-eabi-gcc 컴파일러를 이용하고, 매개변수에 여러 정보를 컴파일 옵션으로 주고, beaglebone.o 파일을 beaglebone.c 파일로 컴파일합니다.

Arm-eabi-ld-T link.lds start.o beaglebone.o –luart –lgcc –L ../libuart –L /home/bin/gcc-linaro-4.9-2015.02-3-x86\_64\_arm-eabi/bin/../lib/gcc/arm-eabi/4.9.3 –o beaglebone.elf

->해당 명령어의 recipe는 $(LD) –T $(LINKSCRIPT) start.o beaglebone.o $(LIBRARIES) $(LDFLAGS) –o beaglebone.elf 로, 각각의 매개변수들은 Makefile안에 명시된 대로 입력되어 명령어가 실행됩니다. 결과적으로는 beaglebone.elf파일을 만들게 되는데, elf파일은 executeable,linkable format으로 실행파일, 목적파일, 공유 라이브러리, 코어 덤프를 위한 표준 파일 형식입니다.

Arm-eabi-objcopy -O binary beaglebone.elf beaglebone.bin

->recipe: $(OBJCOPY) –O binary beaglebone.elf beaglebone.bin

-> arm-eabi-objcopy 유틸리티를 이용하여 beaglebone.elf파일을 beaglebone.bin 바이너리 파일로 복사합니다.

Arm-eabi-objdump –DSx beaglebone.elf > beaglebone.dis

-> recipe: $(OBJDUMP) –DSx beaglebone.elf > beaglebone.dis

-> beaglebone.elf 파일의 모든 구간 디스어셈블(D), 동적 재배열(S), 이용 가능한 모든 헤더 정보(x)를 beaglebone.dis 파일에 기록합니다

3.2 명령어 “make” 수행 후 어떤 파일이 생성되었는지 파일 이름을 쓰고 그 파일의 의미를 설명하라.

beaglebone.dis : 위에서 확인한 바에 의하면, 해당 파일은 컴파일 중에 일어난 프로그램의 정보를 담은 파일입니다. 직접 확인해 본 결과 디스어셈블 정보가 어셈블러 언어로 적혀있었고, 프로그램의 모든 사용 가능한 헤더와 메모리 이동 기록이 적혀 있었습니다.

beaglebone.o : 소스파일에 대한 오브젝트 파일로, 마지막 파일(.bin)을 완성하기 전에 오브젝트 파일을 만들어둡니다.

start.o : 어셈블러로 이루어진 start.S파일을 컴파일하는 과정에서 생긴 object파일로, link.lds파일에서 \_start를 사용하는 데에 관련이 있을 것으로 보입니다. Main이 없어도 Makefile을 이용한 컴파일에 오류가 생기지 않도록 하기 위한 것으로 보입니다.

Beaglebone.elf : Makefile내의 첫번째로 적힌 Target으로, object파일을 바이너리파일로 만들기 위해 사용될 elf파일입니다.

4.1 이 실습을 수행하기 위하여 어떤 작업 혹은 명령어 등을 수행하였는지 순서대로 설명하라.

개발 서버에서 Kermit –s beaglebone.bin 명령어를 수행하여, 해당 beaglebone.bin 파일을

s(send)하도록 하고, TeraTerm 프로그램의 목록에서 File – change directory 항목을 선택하여

저의 로컬컴퓨터(본인의 노트북)의 바탕화면 으로 설정하고, File – Transfer – Kermit – receive

선택을 통하여 바탕화면 beaglebone.bin파일을 옮겼습니다.

비글본 보드를 노트북과 연결한 상태에서 TeraTerm serial connetion을 이용하여 비글본 개발환

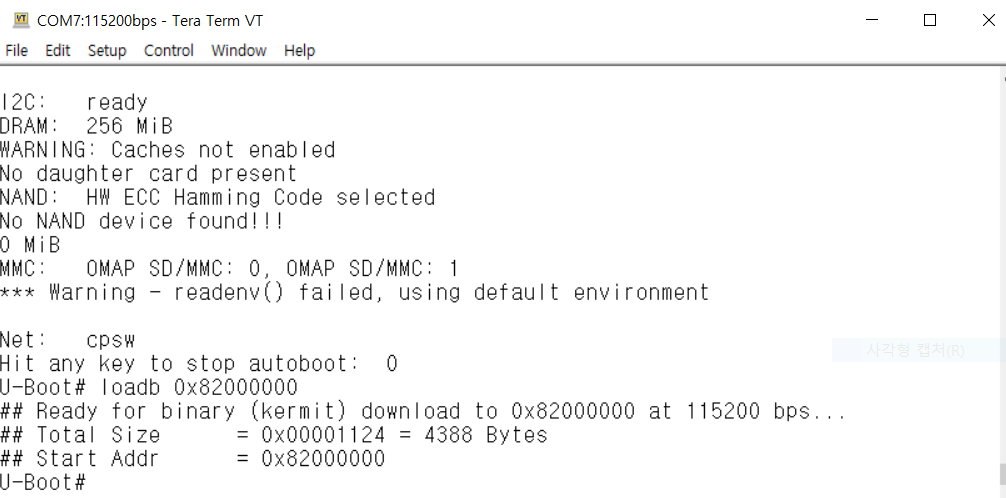
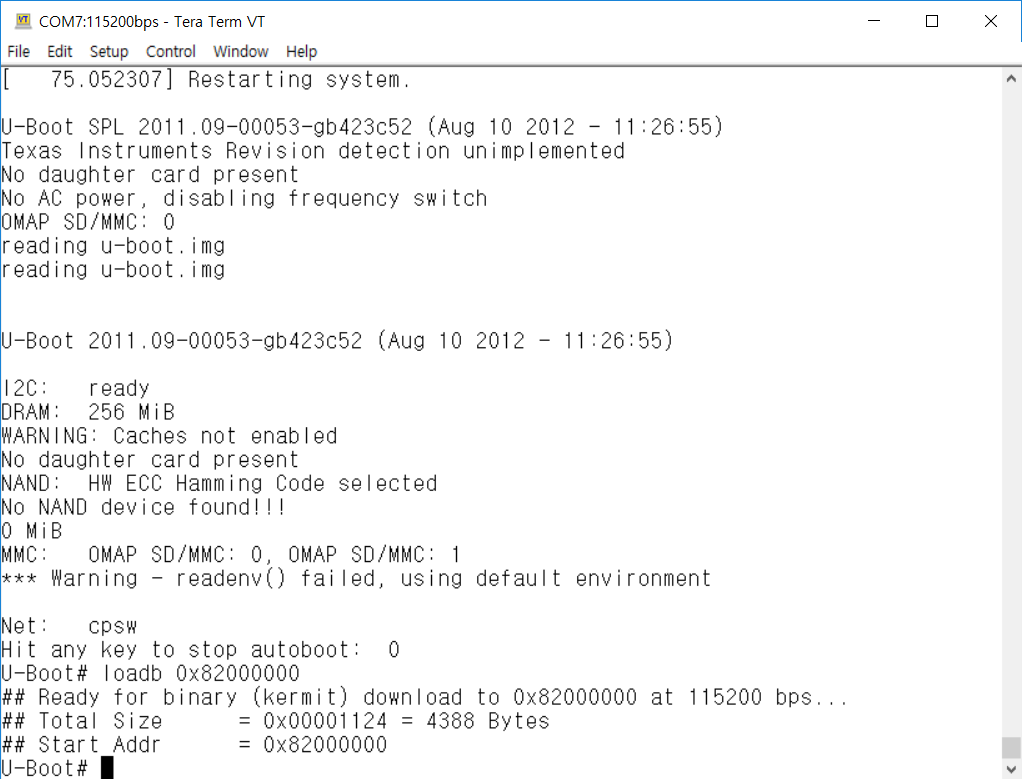
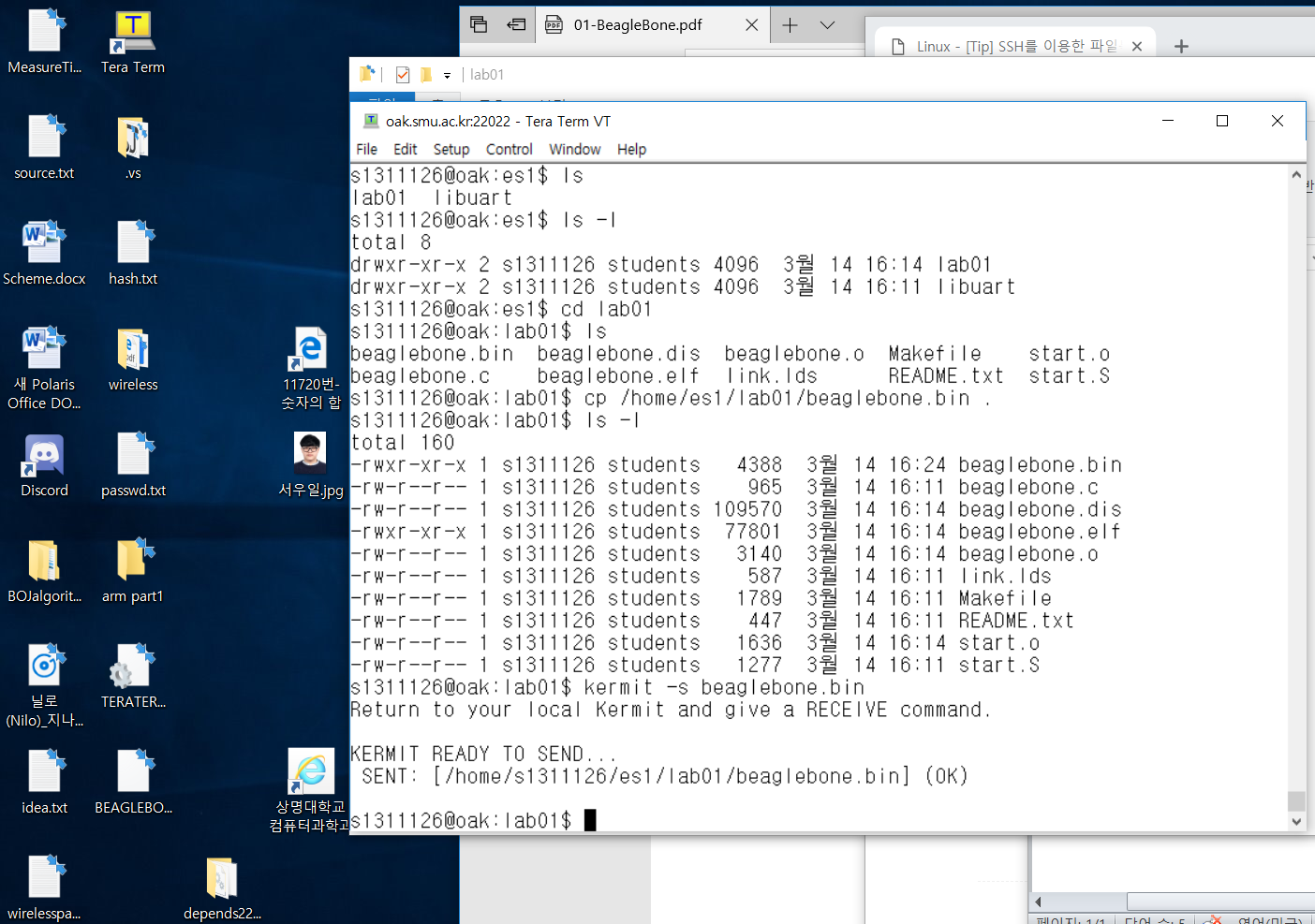
경내에서 loadb 0x82000000 명령어를 수행하여 메모리주소공간 0x82000000부터 파일을 저장

하도록 하고, File – Transfer – Kermit – Send를 이용하여 전에 받았던 beaglebone.bin파일을 선

택하여 비글본의 메모리에 저장합니다.

이후에 go 0x82000000 명령을 이용하여 0x82000000에 있는 app을 실행하게 만듭니다.

4.2 이 실습 과정에 나타나는 주요 화면을 출력하여 제출하라.



5.1 이 실습을 수행하기 위하여 어떤 U-boot 명령어를 수행하였는지 적고 그 의미를 설명하라.

이전에 보드 내의 0x82000000 번지의 메모리 공간에 데이터를 넣어놓았고, 이 번지부터 프로그램을 수행하기 위해 go (메모리 번지) 명령어를 사용하였습니다.

5.2 이 실습 수행 후 나타나는 화면을 출력하여 제출하라.

