Intro. to System Programming, Fall 2018

Homework 1: xv6 부팅하기

제출 마감 : 2018.9.13 11:00am (이메일로 제출한다.)

제출처: 손인엽 조교

제출물: 보고서파일을 이메일로 제출

Boot xv6

xv6 소스 다운로드:

```
$ git clone git://github.com/mit-pdos/xv6-public.git
Cloning into 'xv6-public'...
...
$
```

xv6 빌드:

```
$ cd xv6-public
$ make
...
gcc -O -nostdinc -I. -c bootmain.c
gcc -nostdinc -I. -c bootasm.S
ld -m    elf_i386 -N -e start -Ttext 0x7C00 -o bootblock.o
bootasm.o bootmain.o
objdump -S bootblock.o > bootblock.asm
objcopy -S -O binary -j .text bootblock.o bootblock
...
$
```

Linux 환경을 사용하지 않고 다른 환경 (e.g Mac OS)를 사용한다면, **Website** 를 참고하여 xv6를 설치

Finding and breaking at an address

아래의 명령어를 사용하여, _start (kernel entry point) 주소를 탐색한다.

```
$ nm kernel | grep _start
8010a48c D _binary_entryother_start
8010a460 D _binary_initcode_start
```

```
0010000c T _start
```

예시의 경우, 주소값은 0x0010000c

아래의 명령어를 사용하여, QEMU GDB를 실행시키고 상단에서 찾은 _start의 주소를 Break pointer로 설정한다.

```
$ make qemu-gdb
...
<leave "make qemu-gdb" running, and in a new terminal,
navigate to the same
directory and run the following.>

$ gdb
GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/
licenses/gpl.html>
This is free software: you are free to change and
redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type
"show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources
online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to
"word".
+ target remote localhost:26000
The target architecture is assumed to be i8086
[f000:fff0]    0xffff0:     ljmp   $0xf000,$0xe05b
0x0000fff0 in ?? ()
+ symbol-file kernel
(gdb) br * 0x0010000c
Breakpoint 1 at 0x10000c
(gdb) c
Continuing.
The target architecture is assumed to be i386
=> 0x10000c:     mov    %cr4,%eax
```

```
Breakpoint 1, 0x0010000c in ?? ()
(gdb)
```

출력화면은 gdb version에 따라서 예시와 다를 수 있으나 gdb는 'mov' 명령어에서 멈춘다.

gdb가 auto-loading을 사용할 수 없다고 나타날 수 있다. 이 경우 화면에 출력되는 auto-loading 명령어들을 수행을 통해 해결 할 수 있다.

Exercise: What is on the stack?

상단에 설정한 breakpoint 지점에서 register 들의 값과 stack 내용들을 확인한다.

```
(gdb) info reg
...
(gdb) x/24x $esp
...
(gdb)
```

stack에서 0이 아닌 값들이 무엇을 의미하는 지 3~5단어로 기술하라. 출력문의 어느 부분이 실제 stack인가?

xv6 소스코드 파일들 중에서 bootblock.asm를 찾을 수 있다. 이는 complier / assembler가 bootasm.S, bootmain.c를 통해 만든 파일이다. 특정한 assembly 명령어의 의미를 알아 보기 위해, Reference page의 x86 assembly language 필드를 참고할 수 있다.

과제의 목표는 xv6 kernel에 진입 직전 위에서 본 stack의 내용을 이해하고 설명하는 것이다. 이를 이해하기 위해, 초기 부팅 동안 stack이 어디서 어떻게 setup되는지 관찰하고, kernel 이 수행되기 전까지 stack의 변화를 tracking 해야 한다.

아래 내용은 과제수행에 도움을 줄 수 있는 몇 가지 질의가 존재한다.

- qemu와 gdb를 다시 시작하고 boot block의 시작점인 0x7c00에 break-point를 설정한다. 이 후 gdb 화면에 si 명령어를 입력하여 한 단계 씩 수행한다. bootasm.S의 어느 부분에서 stack pointer가 초기화 되는가? (Stack pointer register(%esp) 로 값을 넣는 명령어를 찾을 때까지 한 단계씩 수행하라)
- Bootmain의 코드를 한 단계씩 수행한다; 현재 stack에는 무엇이 있는가?
- Bootmain의 첫 번째 assembly 명령어들이 stack에서 하는 일은 무엇인가? bootblock.asm의 bootmain을 분석하라.

- gdb를 (필요하다면 breakpoints를 사용) 통해 추적하고 eip를 0x10000c로 변경하는 코드를 탐색한다. 그 코드가 stack에서 하는 일은 무엇인가? (힌트: 이 호출이 boot sequence를 완료하기 위해 무엇을 시도하는지 생각해보라. 그리고 bootmain.c에서 boot sequence를 수행하기 위한 코드와 bootblock.asm에서 boot관련 코드의 위치를 확인하는 과정은 적절한 breakpoints를 설정하는데 도움을 줄 수 있다.)

과제: gdb에서 x/24x $esp 명령어 통해 출력된 결과와 출력된 스택 영역 데이터의 의미를 기술하여 제출

제출 양식: **hw1_자신의 학번**(text editor 종류는 관계없음)