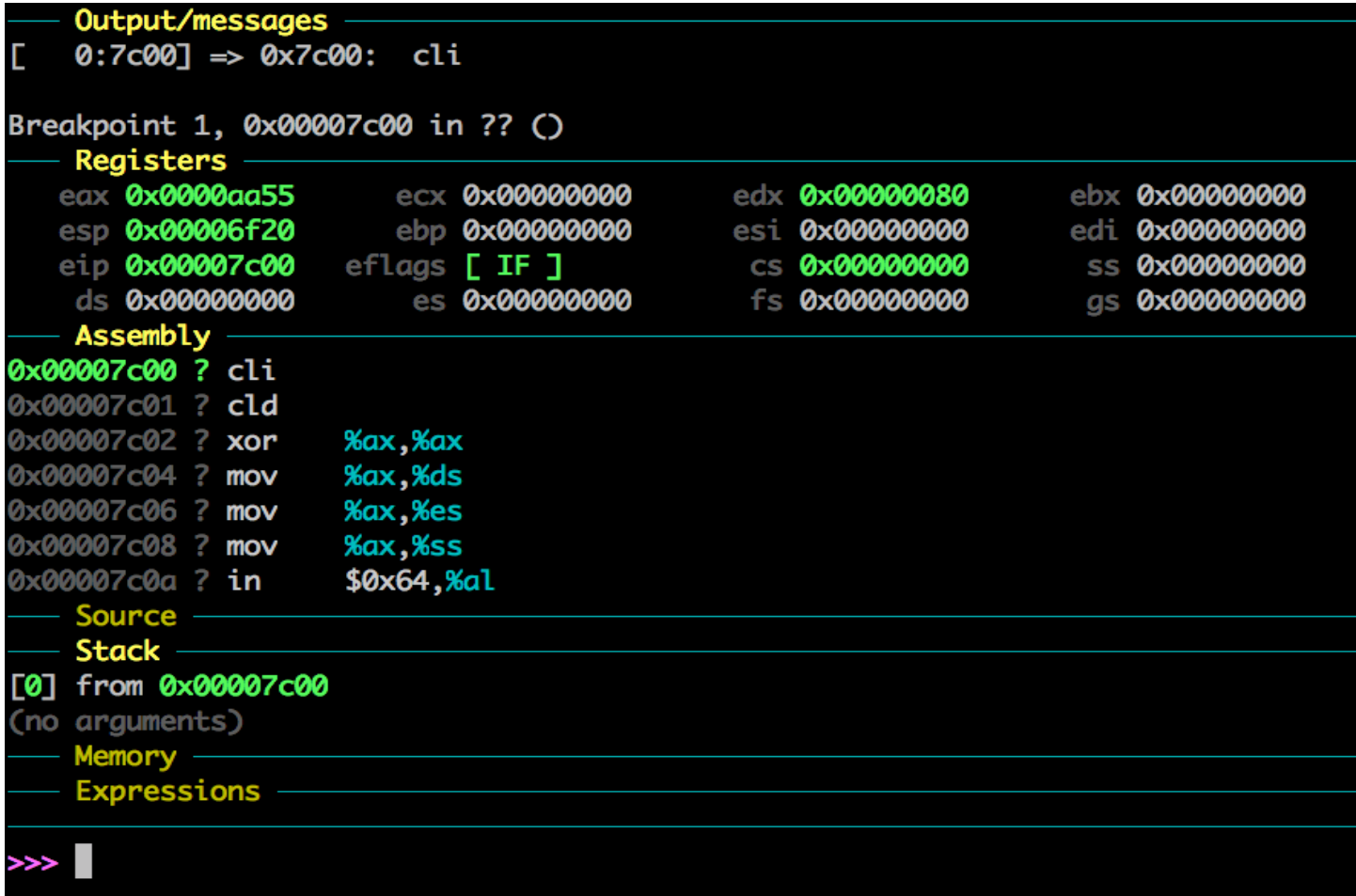
CS 444/544 OS II
Lab Tutorial #2 (part 1)

Booting and GDB Practice

Acknowledgement: Slides drawn heavily from Yeongjin Jiang

Lab Setup Check



Contents

- Following the Booting Sequence
- Exercise 3-6
- Coding Style

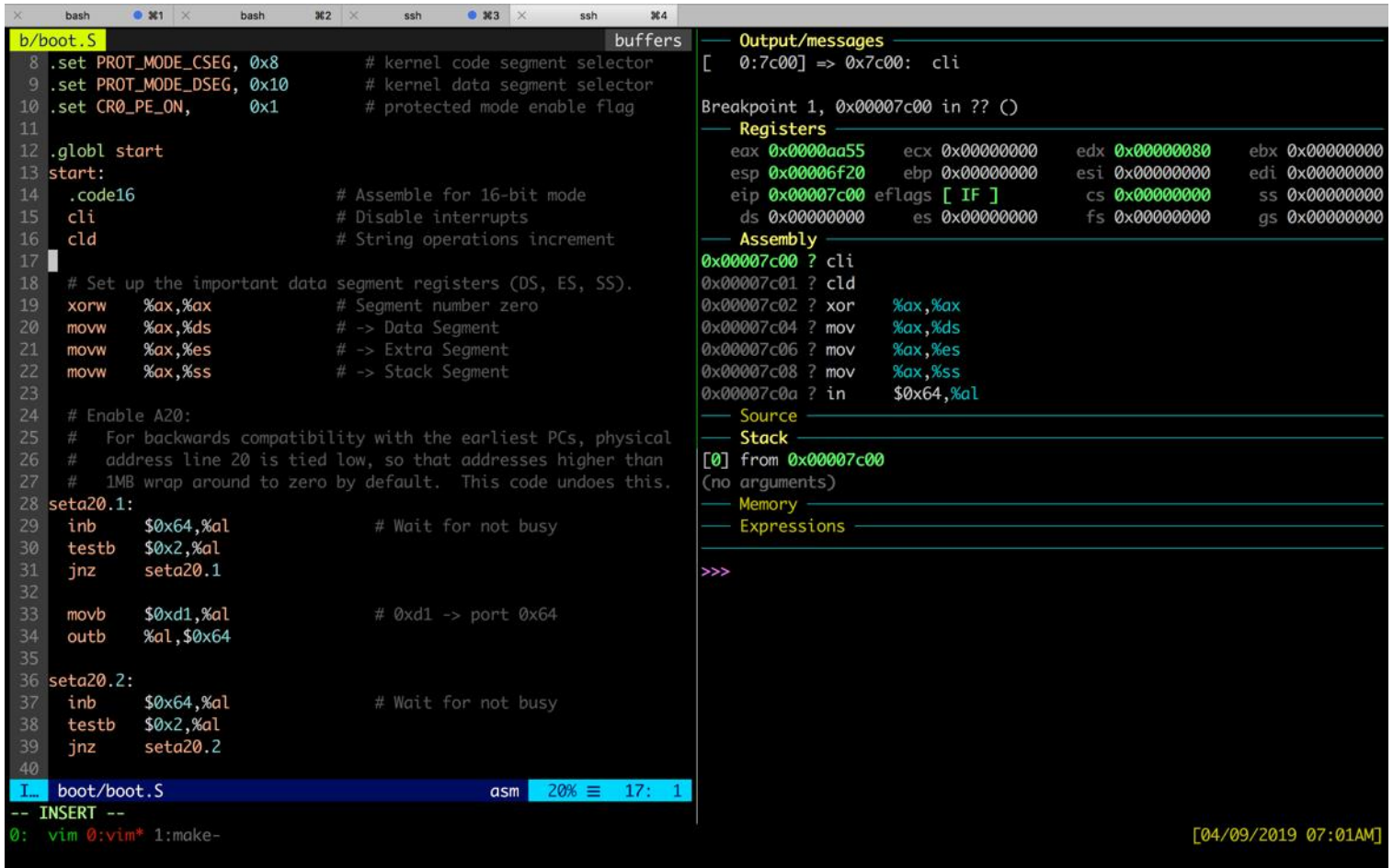
Exercise 3

Set a breakpoint at address 0x7c00, which is where the boot sector will be loaded. Continue execution until that breakpoint. Trace through the code in `boot/boot.S`, using the source code and the disassembly file `obj/boot/boot.asm` to keep track of where you are. Also use the *x/i* command in GDB to disassemble sequences of instructions in the boot loader, and compare the original boot loader source code with both the disassembly in `obj/boot/boot.asm` and GDB.

Exercise 3: How?

- Use tmux, open boot/boot.S and gdb at the same time..

Use ni,si, breakpoint



Exercise 3: Enabling Protected Mode

PROT_MODE_CSEG 0x8

Bootmain() is in boot/main.c

```
  lgdt    gdtdesc
  movl    %cr0, %eax
  orl     $CR0_PE_ON, %eax
  movl    %eax, %cr0

  # Jump to next instruction, but in 32-bit code segment.
  # Switches processor into 32-bit mode.
  ljmp    $PROT_MODE_CSEG, $protcseg

  .code32                     # Assemble for 32-bit mode
protcseg:
  # Set up the protected-mode data segment registers
  movw    $PROT_MODE_DSEG, %ax    # Our data segment selector
  movw    %ax, %ds                # -> DS: Data Segment
  movw    %ax, %es                # -> ES: Extra Segment
  movw    %ax, %fs                # -> FS
  movw    %ax, %gs                # -> GS
  movw    %ax, %ss                # -> SS: Stack Segment

  # Set up the stack pointer and call into C.
  movl    $start, %esp
  call bootmain
```

Exercise 3: bootmain

```
0x00007c45 ? call    0x7d0a
0x00007c4a ? jmp     0x7c4a
0x00007c4c ? add     %al,(%eax)
0x00007c4e ? add     %al,(%eax)
0x00007c50 ? add     %al,(%eax)
0x00007c52 ? add     %al,(%eax)
0x00007c54 ? (bad)
```

```
# Set up the stack pointer and call into C.
movl    $start, %esp
call bootmain
```

In boot/main.c

```
void
bootmain(void)
{
    struct Proghdr *ph, *eph;

    // read 1st page off disk
    readseg((uint32_t) ELFHDR, SECTSIZE*8, 0);

    // is this a valid ELF?
    if (ELFHDR->e_magic != ELF_MAGIC)
        goto bad;

    // load each program segment (ignores ph flags)
    ph = (struct Proghdr *) ((uint8_t *) ELFHDR + ELFHDR->e_phoff);
    eph = ph + ELFHDR->e_phnum;
    for (; ph < eph; ph++)
        // p_pa is the load address of this segment (as well
        // as the physical address)
        readseg(ph->p_pa, ph->p_memsz, ph->p_offset);

    // call the entry point from the ELF header
    // note: does not return!
    ((void (*)(void)) (ELFHDR->e_entry))();
```

Exercise 4-6

- Ex 4: Understand why pointer.c works like that and read about ELF header and the ELF file..
  - https://en.wikipedia.org/wiki/Executable_and_Linkable_Format

- Ex 5: Use si, ni to follow the instructions after changing 0x7c00 to others, e.g., 0x6b00 or something else..

```
$(OBJDIR)/boot/boot: $(BOOT_OBJS)
    @echo + ld boot/boot
    $(V)$(LD) $(LDFLAGS) -N -e start -Ttext 0x7C00 -o $@.out $^
    $(V)$(OBJDUMP) -S $@.out >$@.asm
    $(V)$(OBJCOPY) -S -O binary -j .text $@.out $@
    $(V)perl boot/sign.pl $(OBJDIR)/boot/boot
```

- Ex 6: practice gdb commands

GDB Command for Reading Memory

- x/100wx [address or register]
  - Examine
  - 100 values
  - sized as word (w, 4 bytes)
    - b – byte
    - g – 8 bytes
  - In hexadecimal (x)
    - d - decimal

```
>>> x/100wx 0x7c00
0x7c00: 0xc031fcfa      0xc08ed88e      0x64e4d08e      0xfa7502a8
0x7c10: 0x64e6d1b0      0x02a864e4      0xdfb0fa75      0x010f60e6
0x7c20: 0x0f7c6416      0x8366c020      0x220f01c8      0x7c32eac0
0x7c30: 0xb8660008      0xd88e0010      0xe08ec08e      0xd08ee88e
0x7c40: 0x007c00bc      0x00c0e800      0xfeeb0000      0x00000000
0x7c50: 0x00000000      0x0000ffff      0x00cf9a00      0x0000ffff
0x7c60: 0x00cf9300      0x7c4c0017      0xba550000      0x000001f7
0x7c70: 0x83ece589      0x403cc0e0      0xc35df875      0x57e58955
0x7c80: 0x0c5d8b53      0xfffe1e8       0x01f2baff      0x01b00000
0x7c90: 0xc3b60fee      0x0feef3b2      0xf4b2c7b6      0xb2d889ee
0x7ca0: 0x10e8c1f5      0xeec0b60f      0xb218ebc1      0x83d888f6
0x7cb0: 0xb0eee0c8      0xeef7b220      0xffffade8      0x087d8bff
0x7cc0: 0x000080b9      0x01f0ba00      0xf2fc0000      0x5d5f5b6d
0x7cd0: 0xe58955c3      0x0c7d8b57      0x10758b56      0x085d8b53
0x7ce0: 0x0109eec1      0xe38146df      0xfffffe00      0x1273fb39
0x7cf0: 0x81534656      0x000200c3      0xff7ee800      0x5a58ffff
0x7d00: 0x658deaeb      0x5f5e5bf4      0x8955c35d      0x6a5356e5
0x7d10: 0x10006800      0x00680000      0xe8000100      0xfffffffb1
0x7d20: 0x810cc483      0x0100003d      0x4c457f00      0xa1387546
0x7d30: 0x0001001c      0x0000988d      0xb70f0001      0x01002c05
0x7d40: 0x05e0c100      0x3903348d      0xff1673f3      0xc3830473
0x7d50: 0xf473ff20      0xe8ec73ff      0xffffff75      0xeb0cc483
0x7d60: 0x1815ffe6      0xba000100      0x00008a00      0xff8a00b8
0x7d70: 0xb8ef66ff      0xffff8e00      0xfeebef66      0x00000000
0x7d80: 0x00000000      0x00000000      0x00000000      0x00000000
```

Coding Convention (CODING)

- No space after a function name in a call
  - cprintf(“asdf”) GOOD
  - cprintf (“asdf”) NO
- One space after if/for/while/switch
  - if (a == 1) { GOOD
  - if(a==1) { NO
- function_and_variable_names_look_like_this
  - NoCamelCase
- Macros are ALL UPPERCASE
  - e.g., SEG()

Coding Convention (CODING)

- Pointer types includes a space before *
  - (uint32_t *) GOOD
  - (uint32_t*) NO
- Use ‘//’ for your comment
  - All imported comments are /**/, so we can distinguish yours from those
  - FYI, Linux Kernel uses /**/…
- Function with no args
  - f(void), not f();

Coding Convention (CODING)

- Function definition
  - Insert newline between the return type and function name
  - This will make finding function definition easy
  - E.g., find the definition of mon_kerninfo would be:
  - '^mon_kerninfo' in regexp.

```
int
mon_kerninfo(int argc, char **argv, struct Trapframe *tf)
{
```

```
os2 ~/cs444/s21/os2-lab1-Rogersyp 106% grep -nr ^mon_kerninfo
kern/monitor.c:43:mon_kerninfo(int argc, char **argv, struct Trapframe *tf)
os2 ~/cs444/s21/os2-lab1-Rogersyp 107%
```