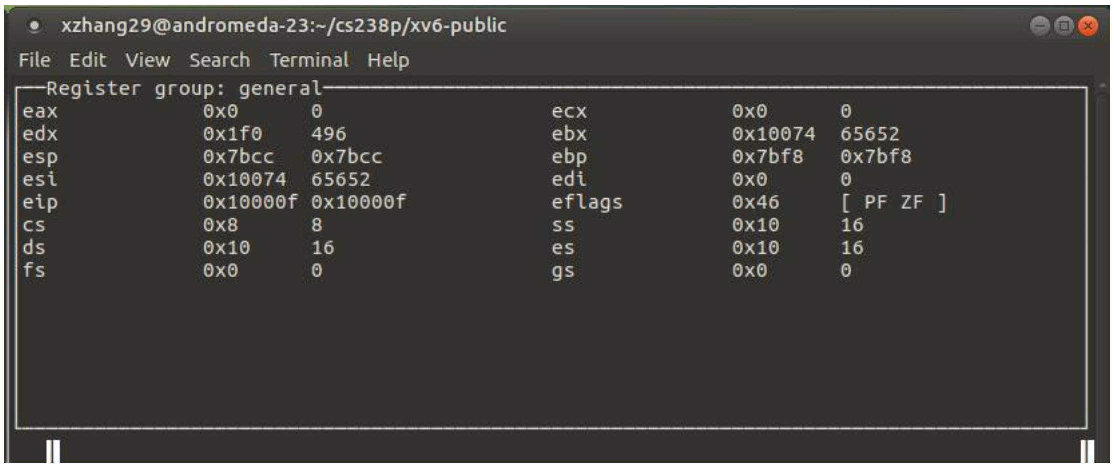
CS238P Fall 2018

HW#2

xzhang29, ID 78369457

Question 1: What is on the stack?

Screenshot from gdb:

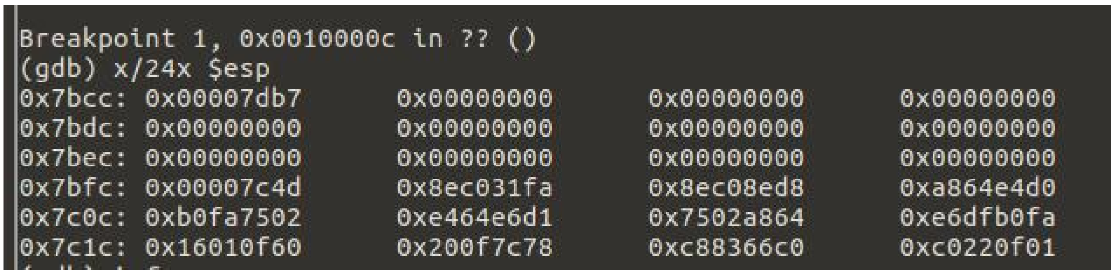


xzhang29@andromeda-23:~/cs238p/xv6-public

File Edit View Search Terminal Help

```
┌─Register group: general──────────────────────────────────────────────────┐
│eax            0x0      0                  ecx            0x0      0      │
│edx            0x1f0    496                ebx            0x10074  65652  │
│esp            0x7bcc   0x7bcc             ebp            0x7bf8   0x7bf8 │
│esi            0x10074  65652              edi            0x0      0      │
│eip            0x10000f 0x10000f           eflags         0x46     [ PF ZF ]│
│cs             0x8      8                  ss             0x10     16     │
│ds             0x10     16                 es             0x10     16     │
│fs             0x0      0                  gs             0x0      0      │
```

Register	Hex Value	Value	Explanation
eax	0x0	0	Return value from last function.
edx	0x1f0	496	One of the genera purpose register.
esp	0x7bcc	0x7bcc	Points to the top of stack.
esi	0x10074	65652	Points to last source address.
eip	0x10000f	0x10000f	Points to the address where stores the next instruction to be execute.
cs	0x8	8	Code segment, current value is set to 8.
ds	0x10	16	Data Segment.
fs	0x0	0	Flag Segment.
ecx	0x0	0	One of the general purpose register.
ebx	0x10074	65652	One of the general purpose register.
ebp	0x7bf8	0x7bf8	Points to the current base address of stack.
edi	0x0	0	Last function's return value.
eflags	0x46	[PF ZF]	Status flags. PF means there are even number of 1s in the result, ZF means the result is zero.
ss	0x10	16	Stack Segment.
es	0x10	16	Extra Segment.
gs	0x0	0	Global Segment.


```
Breakpoint 1, 0x0010000c in ?? ()
(gdb) x/24x $esp
0x7bcc: 0x00007db7      0x00000000      0x00000000      0x00000000
0x7bdc: 0x00000000      0x00000000      0x00000000      0x00000000
0x7bec: 0x00000000      0x00000000      0x00000000      0x00000000
0x7bfc: 0x00007c4d      0x8ec031fa      0x8ec08ed8      0xa864e4d0
0x7c0c: 0xb0fa7502      0xe464e6d1      0x7502a864      0xe6dfb0fa
0x7c1c: 0x16010f60      0x200f7c78      0xc88366c0      0xc0220f01
```

Stack: from address 0x7bcc ($esp) to address 0x7bf8($esp).

From address 0x7bcc to 0x7be8 there are 11 variables, by sequence they are thereturn address from last control flow, which is from entry() with value of 0x00007bd7, $edi (0x000000000), $esi(0x00000000), $ebx(0x00000000), and 7 local variables (they are all zeros).

From address 0x7bf8 to addreee 0x7bfc($ebp), they are old $ebp(which is zero), and current return address (0x00007c4d).

1. Start by setting a break-point at 0x7c00, the start of the boot block (bootasm.S). Single step through the instructions. Where in bootasm.S the stack pointer is initialized?

   Init: movl $start, %esp

2. Single step through the call to bootmain; what is on the stack now?

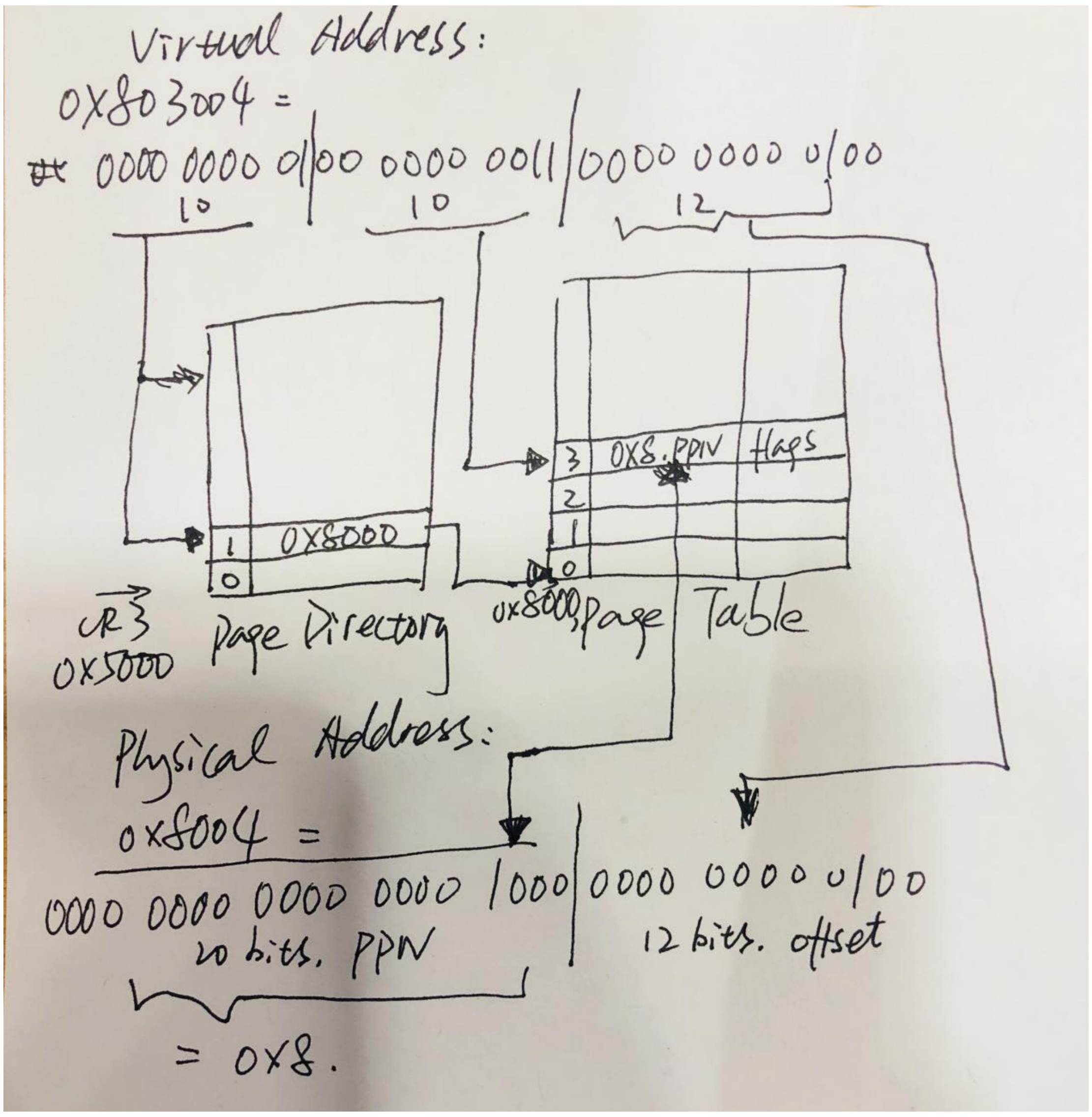
   $eax has the start address of bootmain, which was used to jump to the bootmain.

3. What do the first assembly instructions of bootmain do to the stack? Look for bootmain in bootblock.asm.

   Copy value of ebp to esp, push ebx to the stack, and subtract esp by 12 to make room for local variables.

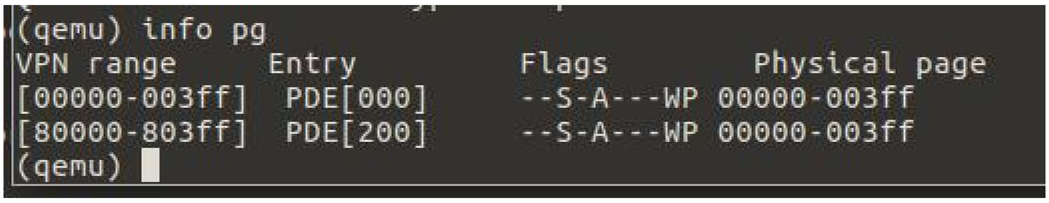
Exercise 2:

Question 1: Explain how logical to physical address translation works

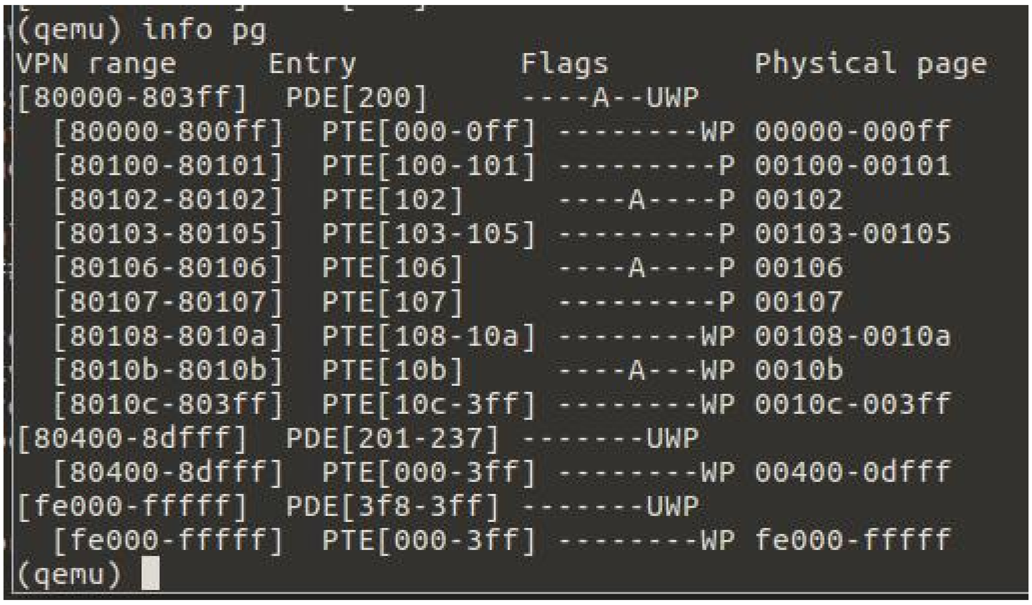


Question 2: What is the state of page tables after xv6 is done initializing the first 4K page table?

Befor kvmalloc():


```
(qemu) info pg
VPN range     Entry         Flags        Physical page
[00000-003ff]  PDE[000]     --S-A---WP 00000-003ff
[80000-803ff]  PDE[200]     --S-A---WP 00000-003ff
(qemu) 
```

After kvmalloc():



```
(qemu) info pg
VPN range     Entry         Flags        Physical page
[80000-803ff]  PDE[200]     ----A--UWP
  [80000-800ff]  PTE[000-0ff] --------WP 00000-000ff
  [80100-80101]  PTE[100-101] ---------P 00100-00101
  [80102-80102]  PTE[102]     ----A----P 00102
  [80103-80105]  PTE[103-105] ---------P 00103-00105
  [80106-80106]  PTE[106]     ----A----P 00106
  [80107-80107]  PTE[107]     ---------P 00107
  [80108-8010a]  PTE[108-10a] --------WP 00108-0010a
  [8010b-8010b]  PTE[10b]     ----A---WP 0010b
  [8010c-803ff]  PTE[10c-3ff] --------WP 0010c-003ff
[80400-8dfff]  PDE[201-237] -------UWP
  [80400-8dfff]  PTE[000-3ff] --------WP 00400-0dfff
[fe000-fffff]  PDE[3f8-3ff] -------UWP
  [fe000-fffff]  PTE[000-3ff] --------WP fe000-fffff
(qemu)
```

VPN range [80000-800ff]: those entries belong to first page directory entry, total size should be 2^8*4096 = 2^20 bytes = 1MB, and those page entries are writable and present. According to the kernel memory structure, the space is for I/O spacing.

VPN range [80100-80107]: those entries belong to first page directory entry, total size is 8*4096 = 2^15 bytes = 32KB. Those pages are present and accessible but not writable, because they are kernel's text data.

VPN range [80108-9dfff]: those entries belong to first and second page directory entries, total size is (3ff+3ff-108) entries, which is equal to 1784*4096 bytes, roughly equals to more than 22MB. Those pages are for kernel read/write data and free memory, hence they are writable and present.

VPN range [fe000-fffff]: those entries belong to third page directory entry, maps to PHYSLIMT. Those pages are for the use of I/O device.