CS5460/6460: Operating Systems

Lecture 9: Finishing system boot, and system init

Anton Burtsev
January, 2014
First stack

Linear
- Code
- Data

Stack

Physical
- Code: 0 - 4GB
- Data: 0 - 4GB

GDT
- NULL: 0x0
- CODE: 0 - 4GB
- DATA: 0 - 4GB

Protected Mode
- CS: 0x8
- SS: 0x10
- GDT: 0x7c78
- EIP: 0x7c1d
- ESP: 0x7c00
- TSS: 0x0
- IDT: 0x0
Invoke first C function

9166 movl $start, %esp
9167 call bootmain

xv6/bootasm.S
void bootmain(void)
{
    struct elfhdr *elf;
    struct proghdr *ph, *eph;
    void (*entry)(void);
    uchar* pa;

    elf = (struct elfhdr*)0x10000; // scratch space

    // Read 1st page off disk
    readseg((uchar*)elf, 4096, 0);

    // Is this an ELF executable?
    if(elf->magic != ELF_MAGIC)
        return; // let bootasm.S handle error

    return; // let bootasm.S handle error
}
bootmain(): read kernel from disk

// Load each program segment (ignores ph flags).
ph = (struct proghdr*)((uchar*)elf + elf->phoff);
eph = ph + elf->phnum;
for(; ph < eph; ph++){
    pa = (uchar*)ph->paddr;
    readseg(pa, ph->filesz, ph->off);
    if(ph->memsz > ph->filesz)
        stosb(pa + ph->filesz, 0, ph->memsz - ph->filesz);
}

// Call the entry point from the ELF header.
// Does not return!
entry = (void(*)(void))(elf->entry);
entry();
1039 .globl entry

1136 # By convention, the _start symbol specifies the ELF entry point.
1137 # Since we haven’t set up virtual memory yet, our entry point is
1138 # the physical address of 'entry'.

1139 .globl _start
1140 _start = V2P_WO(entry)

1141

1142 # Entering xv6 on boot processor, with paging off.

1143 .globl entry
1144 entry:
1145 entry:

1146 # Turn on page size extension for 4Mbyte pages
1147 movl %cr4, %eax
1148 orl $(CR4_PSE), %eax
1149 movl %eax, %cr4
Set up page directory

1149  # Set page directory

1150  movl $(V2P_WO(entrypgdir)), %eax

1151  movl %eax, %cr3
Our goal: 2GB/2GB address space
First page table

- Two 4MB entries (large pages)
- Entry #0
  - 0x0 – 4MB → 0x0:0x400000
- Entry #512
  - 0x0 – 4MB → 0x8000000:0x80400000
The boot page table used in entry.S and entryother.S.

Page directories (and page tables) must start on page boundaries,

hence the __aligned__ attribute.

PTE_PS in a page directory entry enables 4Mbyte pages.

__attribute__((__aligned__(PGSIZE)))

pde_t entrypgdir[NPDENTRIES] = {
    // Map VA’s [0, 4MB) to PA’s [0, 4MB)
    [0] = (0) | PTE_P | PTE_W | PTE_PS,
    // Map VA’s [KERNBASE, KERNBASE+4MB) to PA’s [0, 4MB)
    [KERNBASE>>PDXSHIFT] = (0) | PTE_P | PTE_W | PTE_PS,
};

First page table
The boot page table used in entry.S and entryother.S.

Page directories (and page tables) must start on page boundaries,

hence the __aligned__ attribute.

PTE_PS in a page directory entry enables 4Mbyte pages.

__attribute__((__aligned__((PGSIZE)))

pde_t entrypgdir[NPDENTRIES] = {

// Map VA’s [0, 4MB) to PA’s [0, 4MB)
[0] = (0) | PTE_P | PTE_W | PTE_PS,

// Map VA’s [KERNBASE, KERNBASE+4MB) to PA’s [0, 4MB)
[KERNBASE>>PDXSHIFT] = (0) | PTE_P | PTE_W | PTE_PS,

};

First page table
The boot page table used in entry.S and entryother.S.

Page directories (and page tables) must start on page boundaries,

hence the __aligned__ attribute.

PTE_PS in a page directory entry enables 4Mbyte pages.

__attribute__((__aligned__(PGSIZE)))

pde_t entrypgdir[NPDENTRIES] = {
  // Map VA’s [0, 4MB) to PA’s [0, 4MB)
  [0] = (0) | PTE_P | PTE_W | PTE_PS,
  // Map VA’s [KERNBASE, KERNBASE+4MB) to PA’s [0, 4MB)
  [KERNBASE>>PDXSHIFT] = (0) | PTE_P | PTE_W | PTE_PS,
};

First page table
The boot page table used in entry.S and entryother.S.

Page directories (and page tables) must start on page boundaries,

hence the \_\_aligned\_\_ attribute.

PTE_PS in a page directory entry enables 4Mbyte pages.

__attribute__((__aligned__(PGSIZE)))

pde_t entrypgdir[NPDENTRIES] = {

    // Map VA's \([0, 4MB)\) to PA's \([0, 4MB)\)
    [0] = (0) | PTE_P | PTE_W | PTE_PS

    // Map VA's \([\text{KERNBASE}, \text{KERNBASE}+4MB)\) to PA's \([0, 4MB)\)
    [\text{KERNBASE}>>\text{PDXSHIFT}] = (0) | PTE_P | PTE_W | PTE_PS

};
The boot page table used in entry.S and entryother.S.

Page directories (and page tables) must start on page boundaries,
hence the __aligned__ attribute.
PTE_PS in a page directory entry enables 4Mbyte pages.

__attribute__((__aligned__(PGSIZE)))
pde_t entriypgdir[NPDENTRIES] = {
   // Map VA’s [0, 4MB) to PA’s [0, 4MB)
   [0] = (0) | PTE_P | PTE_W | PTE_PS,
   // Map VA’s [KERNBASE, KERNBASE+4MB) to PA’s [0, 4MB)
   [KERNBASE>>PDXSHIFT] = (0) | PTE_P | PTE_W | PTE_PS,
};

First page table
First page table (cont)

0870 // Page directory and page table constants.

0871 #define NPDENTRIES 1024
# Turn on paging.

```
1152 movl %cr0, %eax
1153 orl $(CRO_PG|CRO_WP), %eax
1155 movl %eax, %cr0
```
High address stack (4K)

1157  # Set up the stack pointer.
1158  movl $(stack + KSTACKSIZE), %esp
1159
1159 ...
1167  .comm stack, KSTACKSIZE

0151  #define KSTACKSIZE 4096  // size of per-process kernel stack
Jump to main()

1160  # Jump to main(), and switch to executing at high addresses. The indirect call is needed because
1161  # the assembler produces a PC-relative instruction
1162  # for a direct jump.
1163  #
1164  mov $main, %eax
1165  jmp *%eax
1166
1313 // Bootstrap processor starts running C code here.
1314 // Allocate a real stack and switch to it, first
1315 // doing some setup required for memory allocator to work.
1316 int
1317 main(void)
1318 {
1319     kinit1(end, P2V(4*1024*1024)); // phys page allocator
1320     kvmalloc(); // kernel page table
1321     mpinit(); // detect other processors
1322     lapicinit(); // interrupt controller
1323     seginit(); // segment descriptors
1324     cprintf("ncpu%d: starting xv6\n\n", cpunum());
... 
1340 }
Recap of the boot sequence

- Setup segments (data and code)
- Switched to protected mode
  - Loaded GDT (segmentation is on)
- Setup stack (to call C functions)
- Loaded kernel from disk
- Setup first page table
  - 2 entries [ 0 : 4MB ] and [ 2GB : (2GB + 4MB) ]
- Setup high-address stack
- Jumped to main()
Conclusion

• We've booted
  • We're running in main()

• Next time:
  • 2-level page tables
  • Process and kernel address space
Thank you!
Thank you!