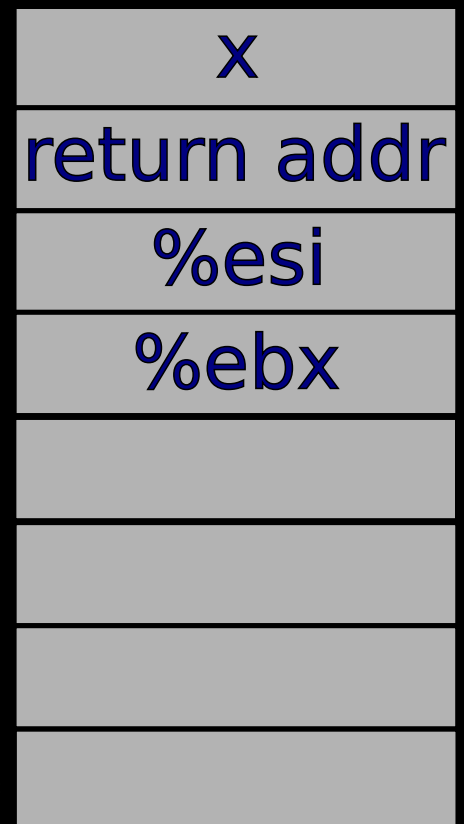


```

1 foo:
2     pushl    %esi
3     pushl    %ebx
4     subl     $4, %esp
5     movl     16(%esp), %ebx
6     cmpl     $1, %ebx
7     jg       .L4           # jg is "jump if greater"
8 .L2:
9     movl     %ebx, %eax
10    addl     $4, %esp
11    popl     %ebx
12    popl     %esi
13    ret
14 .L4:
15    subl     $12, %esp
16    ----- # ebx has the value x.
17    leal     -1(%ebx), %eax
18    pushl    %eax
19    call     foo
20    movl     %eax, %esi
21    subl     $2, %ebx
22    movl     %ebx, (%esp)
23    call     foo
24    leal     (%esi,%eax), %ebx
25    ----- #####
26    addl     $16, %esp
27    jmp      .L2

```

Stack Frame



bootasm.S

```
1  ..
2  ..
3
4  # Start the first CPU: switch to 32-bit protected mode, jump into C.
5  # The BIOS loads this code from the first sector of the hard disk into
6  # memory at physical address 0x7c00 and starts executing in real mode
7  # with %cs=0 %ip=7c00.
8  ..
9  ..
10 lgdt    gdt_desc
11 movl    %cr0, %eax
12 orl     $CR0_PE, %eax
13 movl    %eax, %cr0
14 ..
15 ..
16 movw    $(SEG_KDATA<<3), %ax    # Our data segment selector
17 movw    %ax, %ds                # -> DS: Data Segment
18 ..
19 ..
20 # Set up the stack pointer and call into C.
21 movl    $start, %esp
22 call    bootmain
23 ..
24 ..
25 gdt:
26 SEG_NULLASM                                # null seg
27 SEG_ASM(STA_X|STA_R, 0x0, 0xffffffff)    # code seg
28 SEG_ASM(STA_W, 0x0, 0xffffffff)         # data seg
29 ..
30 ..
31
32
```

bootmain.c

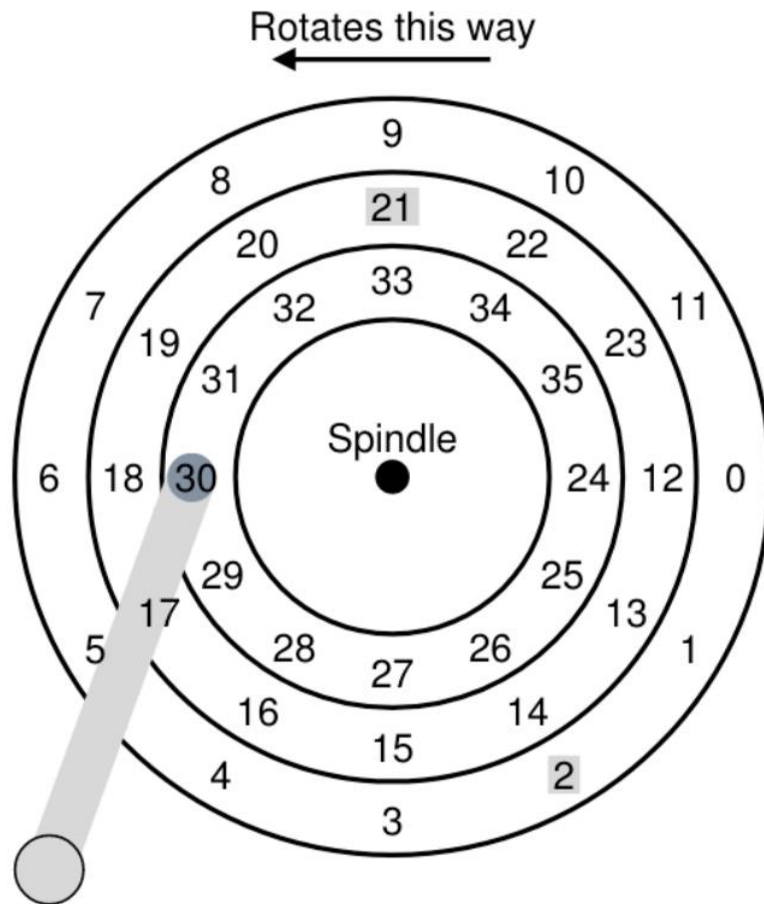
```
1  |..
2  |..
3  |elf = (struct elfhdr*)0x10000; // scratch space
4  |..
5  |..
6  |entry = (void(*)(void))(elf->entry);
7  |entry();
8  |}
9  |..
10 |..
11 |void
12 |waitdisk(void)
13 |{
14 |    // Wait for disk ready.
15 |    while((inb(0x1F7) & 0xC0) != 0x40)
16 |        ;
17 |}
18
19 // Read a single sector at offset into dst.
20 void
21 readsect(void *dst, uint offset)
22 {
23     ..
24     ..
25
26     // Read data.
27     waitdisk();
28     insl(0x1F0, dst, SECTSIZE/4);
29 }
30 ..
31 ..
32
```

C trap.c > ...

```
29 //PAGEBREAK: 41
30 void
31 trap(struct trapframe *tf)
32 {
33     switch(tf->trapno){
34     case T_IRQ0 + IRQ_TIMER:
35         ticks++;
36         lapiceoi();
37         break;
38     case T_IRQ0 + IRQ_IDE:
39         ideintr();
40         lapiceoi();
41         break;
42     case T_IRQ0 + IRQ_COM1:
43         uartintr();
44         lapiceoi();
45         break;
46     case T_IRQ0 + 7:
47     case T_IRQ0 + IRQ_SPURIOUS:
48         cprintf("cpu%d: spurious interrupt at %x:%x\n",
49             ||| cpuid(), tf->cs, tf->eip);
50         lapiceoi();
51         break;
52     default:
53         cprintf("unexpected trap %d from cpu %d eip %x (cr2=0x%x)\n",
54             ||| tf->trapno, cpuid(), tf->eip, rcr2());
55         panic("trap");
56     }
57 }
58 }
59
```

ASM trapasm.S

```
1  #include "mmu.h"
2
3  # vectors.S sends all traps here.
4  .globl alltraps
5  alltraps:
6      # Build trap frame.
7      pushal
8
9  /// #
10     pushl %esp
11     call trap
12     addl $4, %esp
13
14     # Return falls through to trapret...
15     .globl trapret
16     trapret:
17         popal
18     /// addl $0x8, %esp
19     iret
20
```



The disk structure

RAID 0

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4	Disk 5
0	1	2	3	4	5
6	7	8	9	10	11

Raid 0 setup with 6 disk

Block 1 Inode bitmap

1	2	3	4	5	6
---	---	---	---	---	---

Block 2 Data bitmap

6	7	8	9	10	11	12
---	---	---	---	----	----	----

Block 3

Inode = 1 “/”

Type = directory
6

Block 4

Inode = 3

Type = directory
7

Block 5
Inode = 5

Type = file
8
9
10
11

Block 6

File/directory name	Inode number
.	1
foo	3

Block 7

File/directory name	Inode number
.	3
..	1
bar	5

Assumptions

- Blocks 3-5 are inode blocks. There are two inodes in an inode block.
- There are no other files and directories in the system
- There are no other properties in inodes like size, access time, etc.

Block 8

hel

Block 9

lo

Block 10

wor

Block 11

ld
