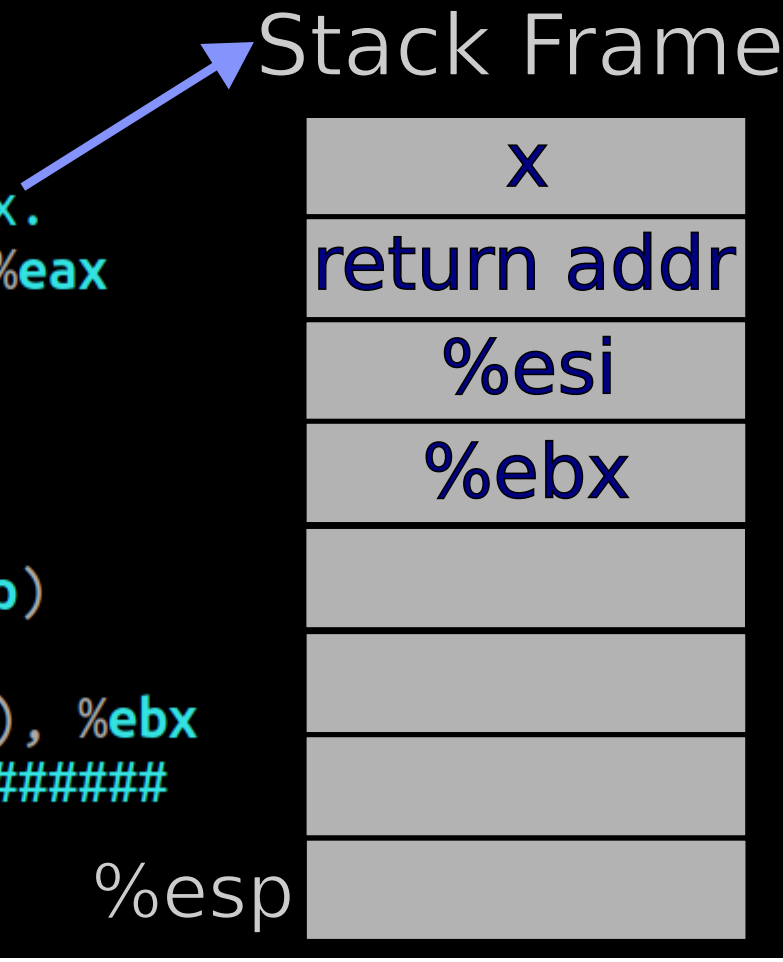
```
 1 foo:
 2         pushl   %esi
 3         pushl   %ebx
 4         subl    $4, %esp
 5         movl    16(%esp), %ebx
 6         cmpl    $1, %ebx
 7         jg      .L4          # jg is "jump if greater"
 8 .L2:
 9         movl    %ebx, %eax
10         addl    $4, %esp
11         popl    %ebx
12         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
```



bootasm.S

```
..
..

# Start the first CPU: switch to 32-bit protected mode, jump into C.
# The BIOS loads this code from the first sector of the hard disk into
# memory at physical address 0x7c00 and starts executing in real mode
# with %cs=0 %ip=7c00.
..
..
  lgdt    gdtdesc
  movl    %cr0, %eax
  orl     $CR0_PE, %eax
  movl    %eax, %cr0
..
..
  movw    $(SEG_KDATA<<3), %ax    # Our data segment selector
  movw    %ax, %ds                # -> DS: Data Segment
..
..
  # Set up the stack pointer and call into C.
  movl    $start, %esp
  call    bootmain
..
..
gdt:
  SEG_NULLASM                             # null seg
  SEG_ASM(STA_X|STA_R, 0x0, 0xffffffff)   # code seg
  SEG_ASM(STA_W, 0x0, 0xffffffff)         # data seg
..
..

```

bootmain.c

```
..
..
  elf = (struct elfhdr*)0x10000;  // scratch space
..
..
  entry = (void(*)(void))(elf->entry);
  entry();
}
..
..
void
waitdisk(void)
{
  // Wait for disk ready.
  while((inb(0x1F7) & 0xC0) != 0x40)
    ;
}

// Read a single sector at offset into dst.
void
readsect(void *dst, uint offset)
{
..
..

  // Read data.
  waitdisk();
  insl(0x1F0, dst, SECTSIZE/4);
}
..
..

```

C trap.c > ...

```c
//PAGEBREAK: 41
void
trap(struct trapframe *tf)
{
  switch(tf->trapno){
  case T_IRQ0 + IRQ_TIMER:
    ticks++;
    lapiceoi();
    break;
  case T_IRQ0 + IRQ_IDE:
    ideintr();
    lapiceoi();
    break;
  case T_IRQ0 + IRQ_COM1:
    uartintr();
    lapiceoi();
    break;
  case T_IRQ0 + 7:
  case T_IRQ0 + IRQ_SPURIOUS:
    cprintf("cpu%d: spurious interrupt at %x:%x\n",
            cpuid(), tf->cs, tf->eip);
    lapiceoi();
    break;

  default:
    cprintf("unexpected trap %d from cpu %d eip %x (cr2=0x%x)\n",
            tf->trapno, cpuid(), tf->eip, rcr2());
    panic("trap");
  }
}

```

ASM trapasm.S

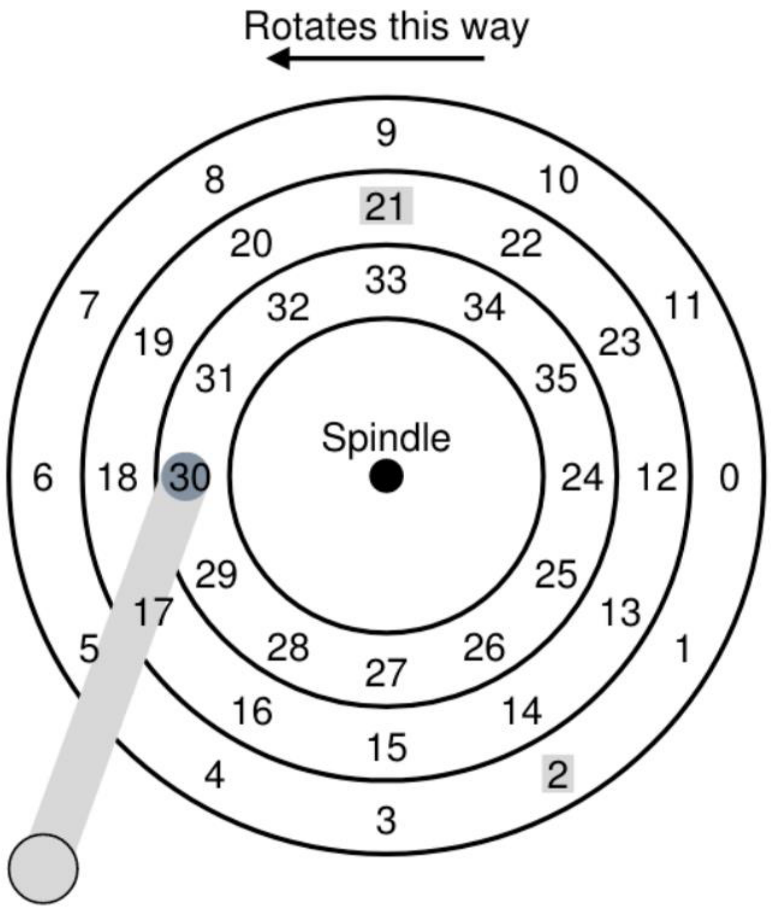
```asm
#include "mmu.h"

  # vectors.S sends all traps here.
.globl alltraps
alltraps:
  # Build trap frame.
  pushal

  #
  pushl %esp
  call trap
  addl $4, %esp

  # Return falls through to trapret...
.globl trapret
trapret:
  popal
  addl $0x8, %esp
  iret

```



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