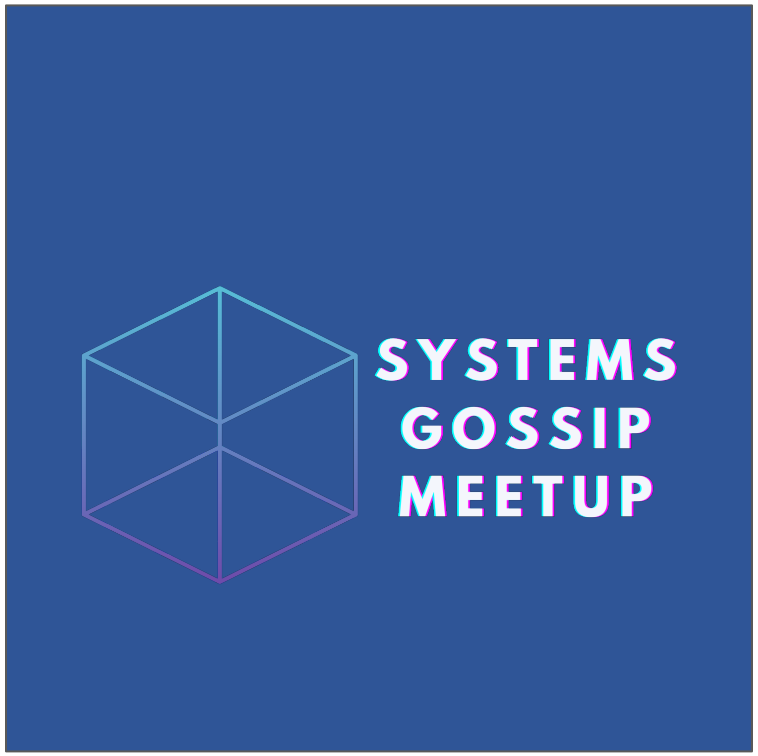


Learning large systems using peer-to-peer gossip

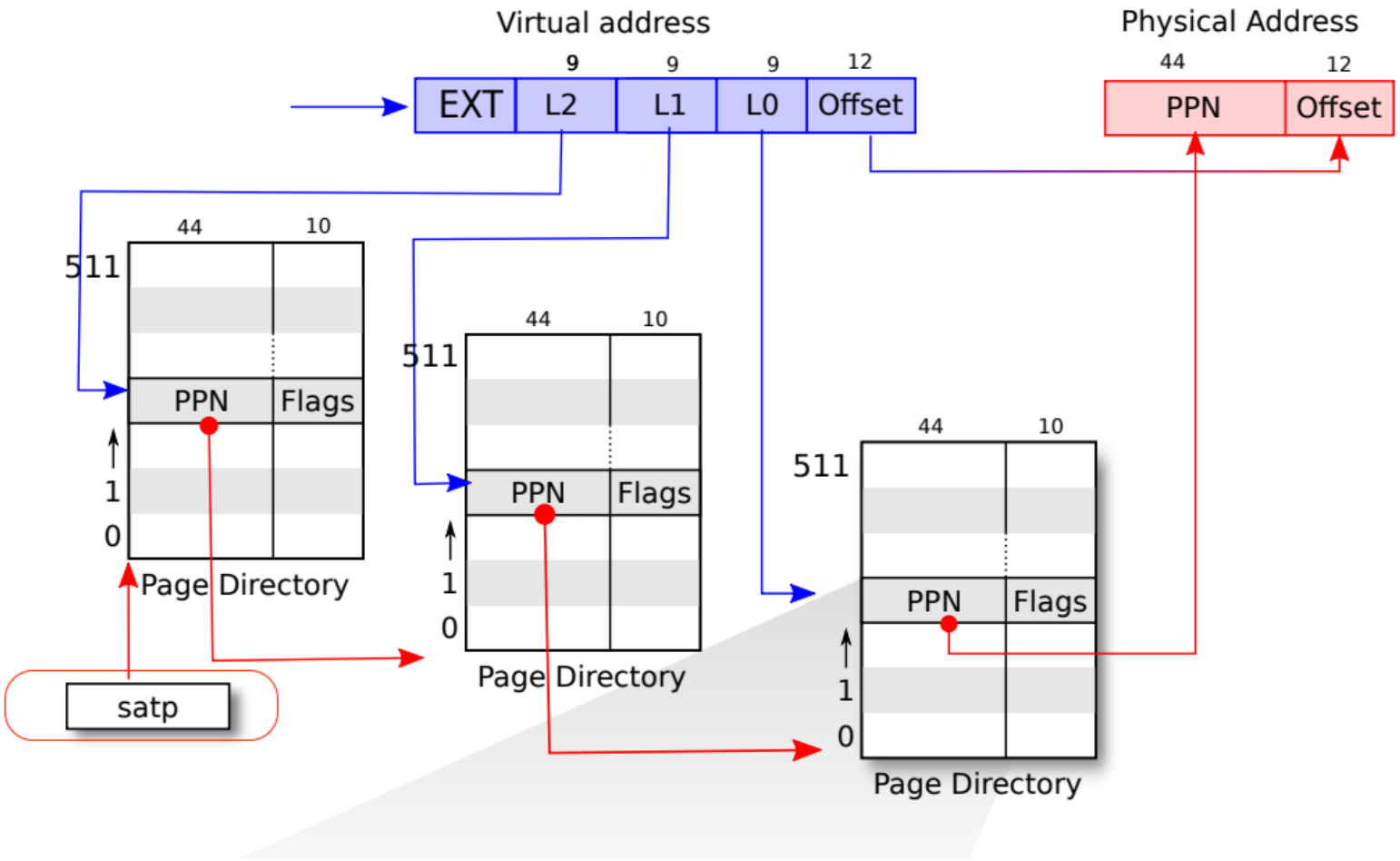
Policy Against Harassment at ACM Activities

OS Meetup wants to encourage and preserve this open exchange of ideas, which requires an environment that enables all to participate without fear of personal harassment. We define harassment to include specific unacceptable factors and behaviors listed in the ACM's policy against harassment. Unacceptable behavior will not be tolerated.
https://www.acm.org/about-acm/policy-against-harassment

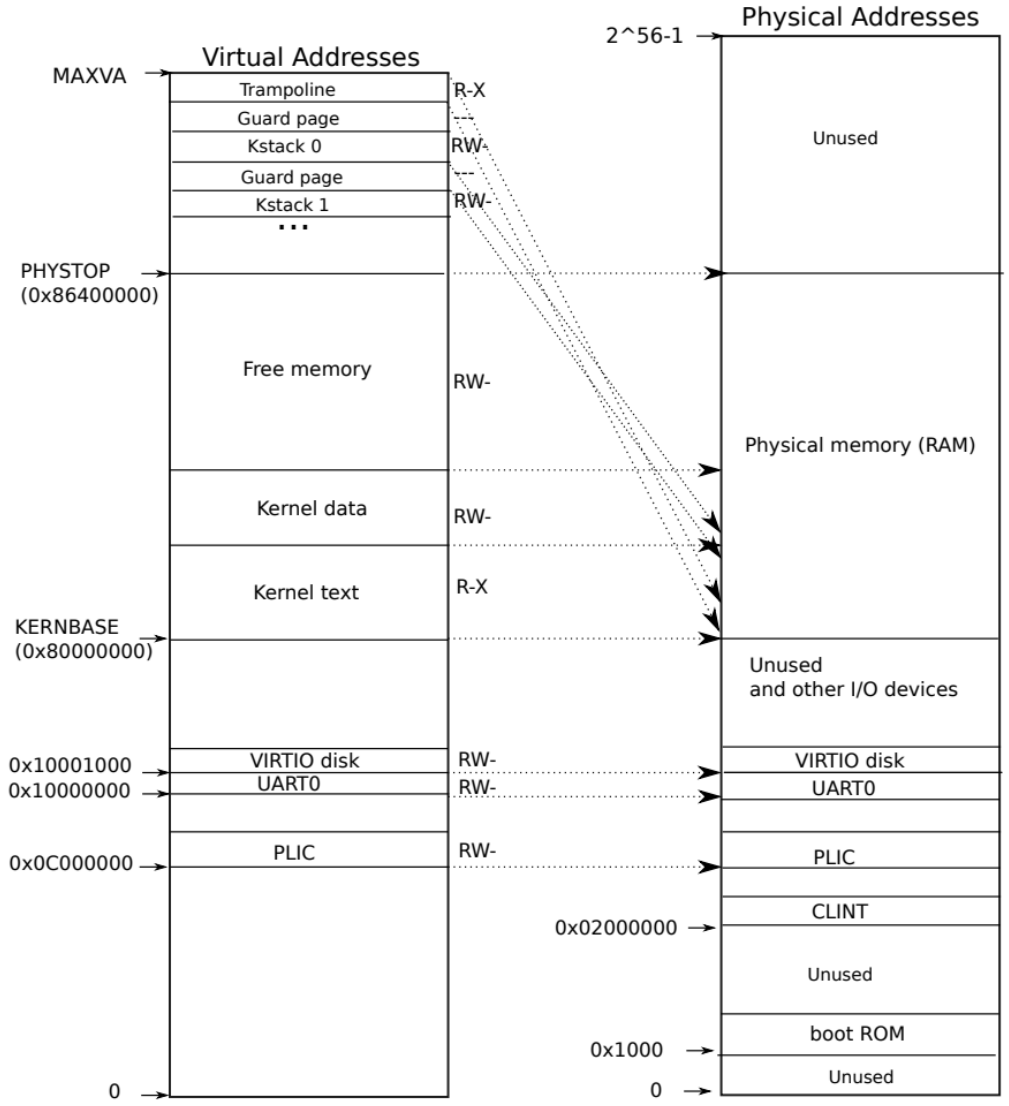
Syscall Preparation: Registers & Memory

- Recall:
  - Page table
- Page table contents
  - Trampoline & Trapframe
- Registers in Risc-V
  - Supervisor Registers
  - General Purpose Registers
- System Calls (User -> Kernel)
  - What CPU does for us?
  - What OS does for us?

Recall: Page Table



Recall: Page Table



Recall: Page Table

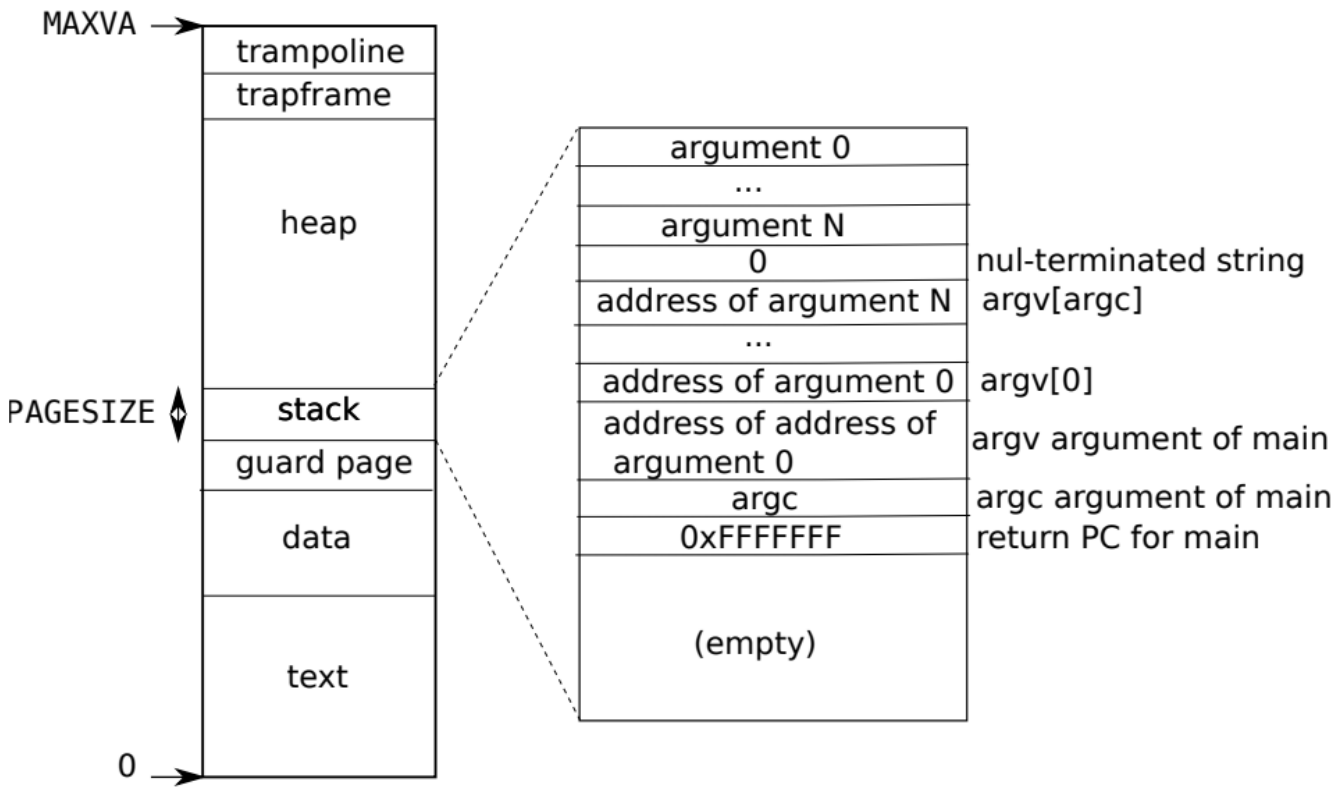
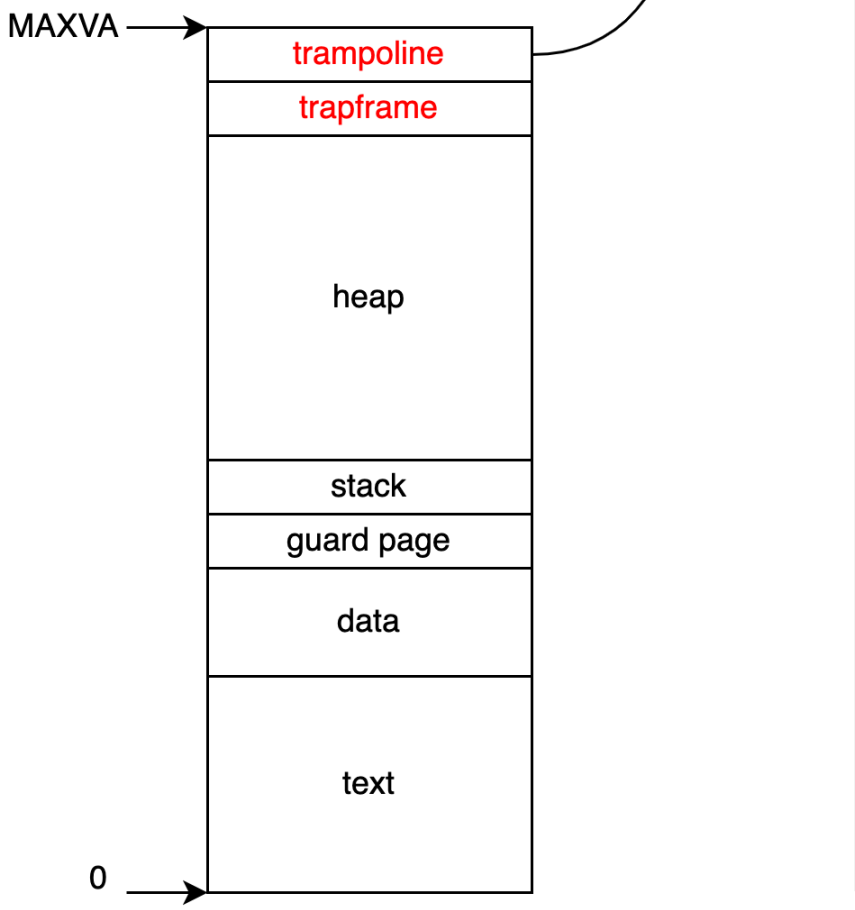


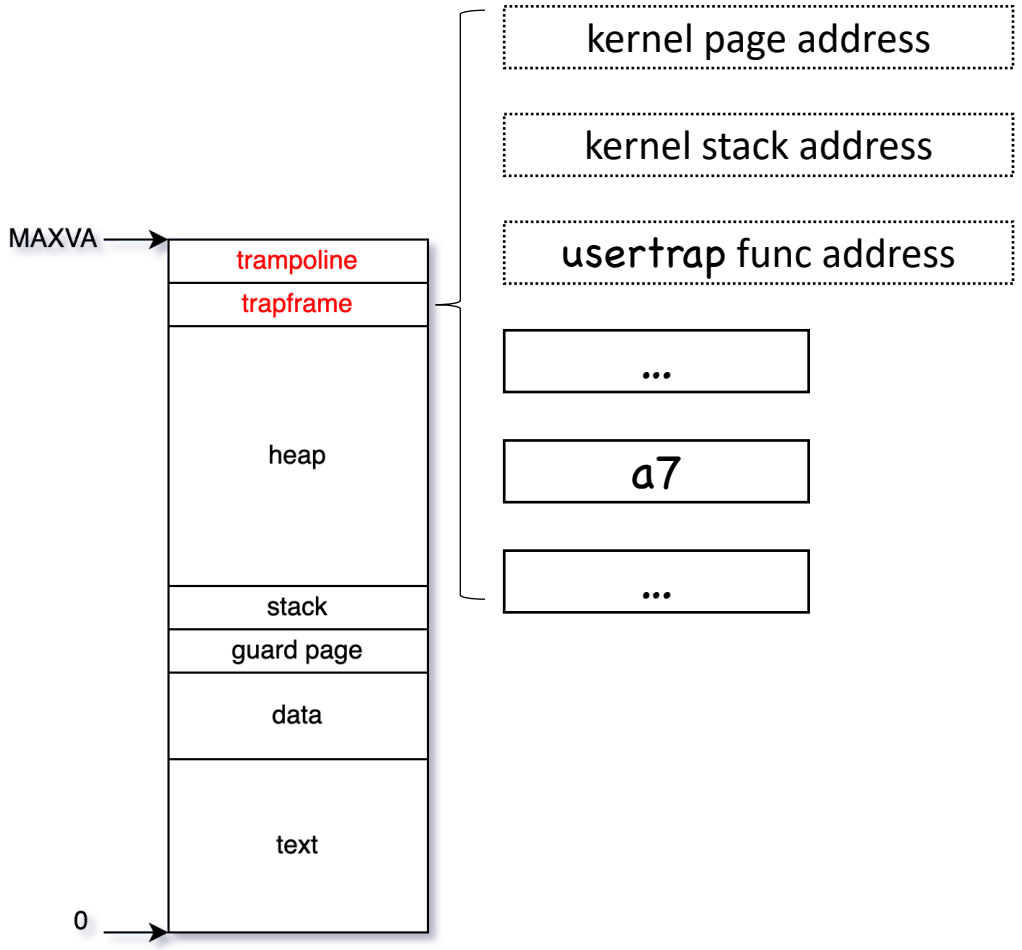
Figure 3.4: A process's user address space, with its initial stack.

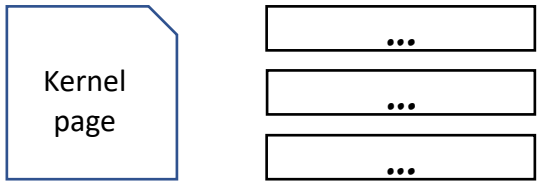
Page: Trampoline




```
16  uservec:
17          #
18          # trap.c sets stvec to point here, so
19          # traps from user space start here,
20          # in supervisor mode, but with a
21          # user page table.
22          #
23          # sscratch points to where the process's p->trapframe is
24          # mapped into user space, at TRAPFRAME.
25          #
26
27          # swap a0 and sscratch
28          # so that a0 is TRAPFRAME
29          csrrw a0, sscratch, a0
30
31          # save the user registers in TRAPFRAME
32          sd ra, 40(a0)
33          sd sp, 48(a0)
34          sd gp, 56(a0)
35          sd tp, 64(a0)
36          sd t0, 72(a0)
37          sd t1, 80(a0)
38          sd t2, 88(a0)
39          sd s0, 96(a0)
40          sd s1, 104(a0)
41          sd a1, 120(a0)
42          sd a2, 128(a0)
43          sd a3, 136(a0)
44          sd a4, 144(a0)
45          sd a5, 152(a0)
46          sd a6, 160(a0)
```

Page: Trapframe





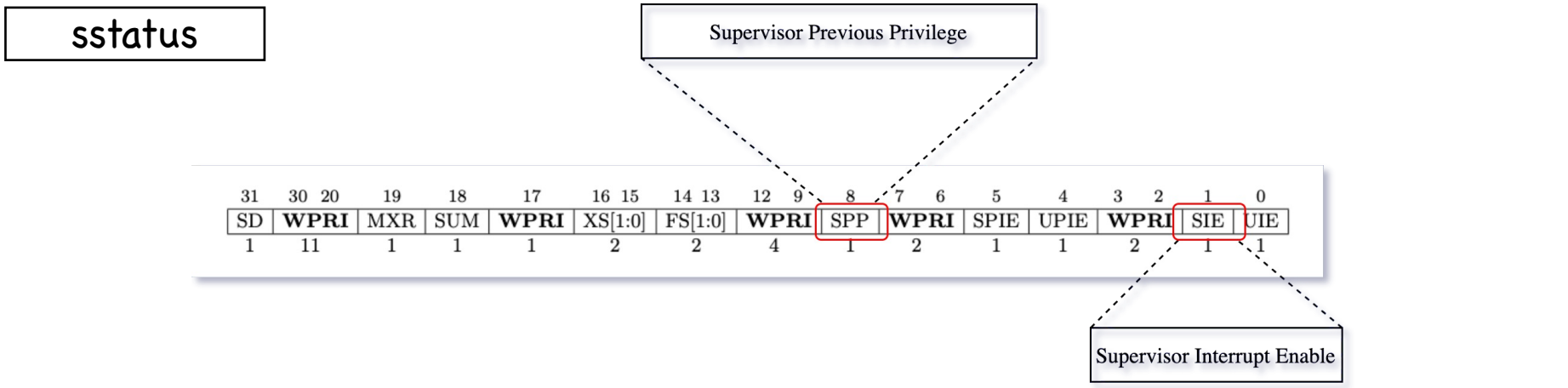

```
36  void
37  usertrap(void)
38  {
39    int which_dev = 0;
40
41    if((r_sstatus() & SSTATUS_SPP) != 0)
42      panic("usertrap: not from user mode");
43
44    // send interrupts and exceptions to kerneltrap(),
45    // since we're now in the kernel.
46    w_stvec((uint64)kernelvec);
47
48    struct proc *p = myproc();
49
50    // save user program counter.
51    p->trapframe->epc = r_sepc();
52
53    if(r_scause() == 8){
54      // system call
55
56      if(p->killed)
57        exit(-1);
58
59      // sepc points to the ecall instruction,
60      // but we want to return to the next instruction.
61      p->trapframe->epc += 4;
62
63      // an interrupt will change sstatus &c registers,
64      // so don't enable until done with those registers.
65      intr_on();
66
67      syscall();
68    } else if((which_dev = devintr()) != 0){
```

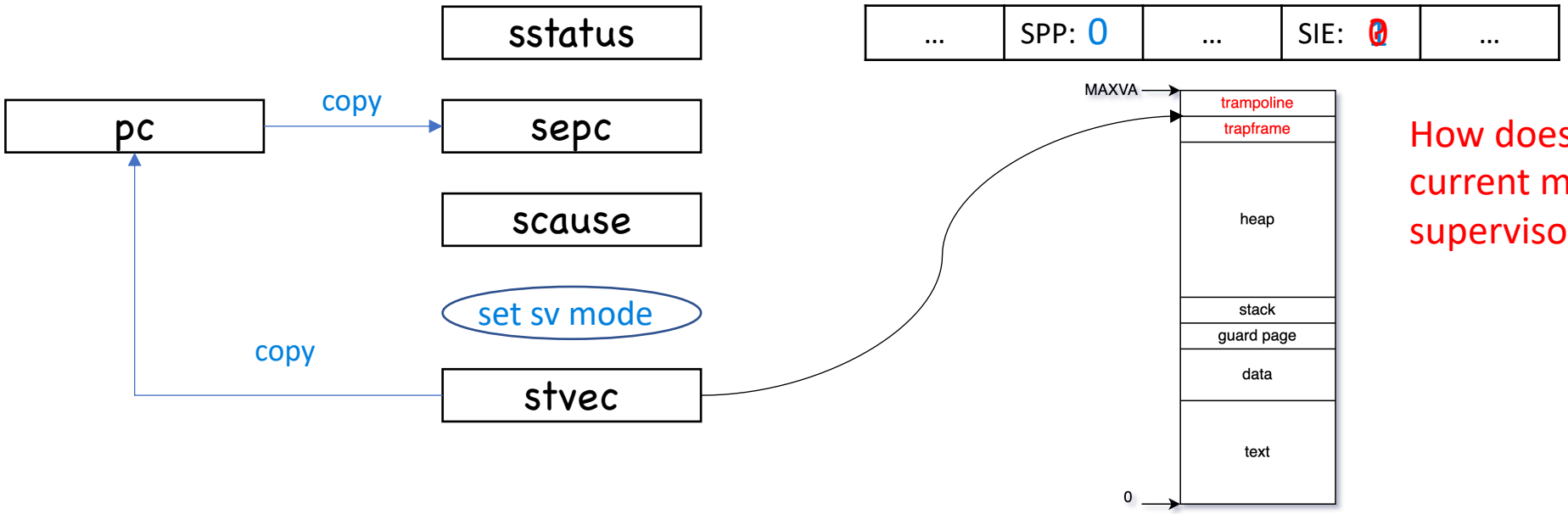
Risc-V Registers

- Supervisor control and status registers
- General purpose registers

Register	ABI Name	Description
x0	zero	hardwired zero
x1	ra	return address
x2	sp	stack pointer
x3	gp	global pointer
x4	tp	thread pointer
x5	t0	temporary register 0
x6	t1	temporary register 1
x7	t2	temporary register 2
x8	s0 / fp	saved register 0 / frame pointer
x9	s1	saved register 1
x10	a0	function argument 0 / return value 0
x11	a1	function argument 1 / return value 1
x12	a2	function argument 2
x13	a3	function argument 3
x14	a4	function argument 4
x15	a5	function argument 5
x16	a6	function argument 6
x17	a7	function argument 7
x18	s2	saved register 2
x19	s3	saved register 3
x20	s4	saved register 4
x21	s5	saved register 5
x22	s6	saved register 6
x23	s7	saved register 7
x24	s8	saved register 8
x25	s9	saved register 9
x26	s10	saved register 10
x27	s11	saved register 11
x28	t3	temporary register 3
x29	t4	temporary register 4
x30	t5	temporary register 5
x31	t6	temporary register 6

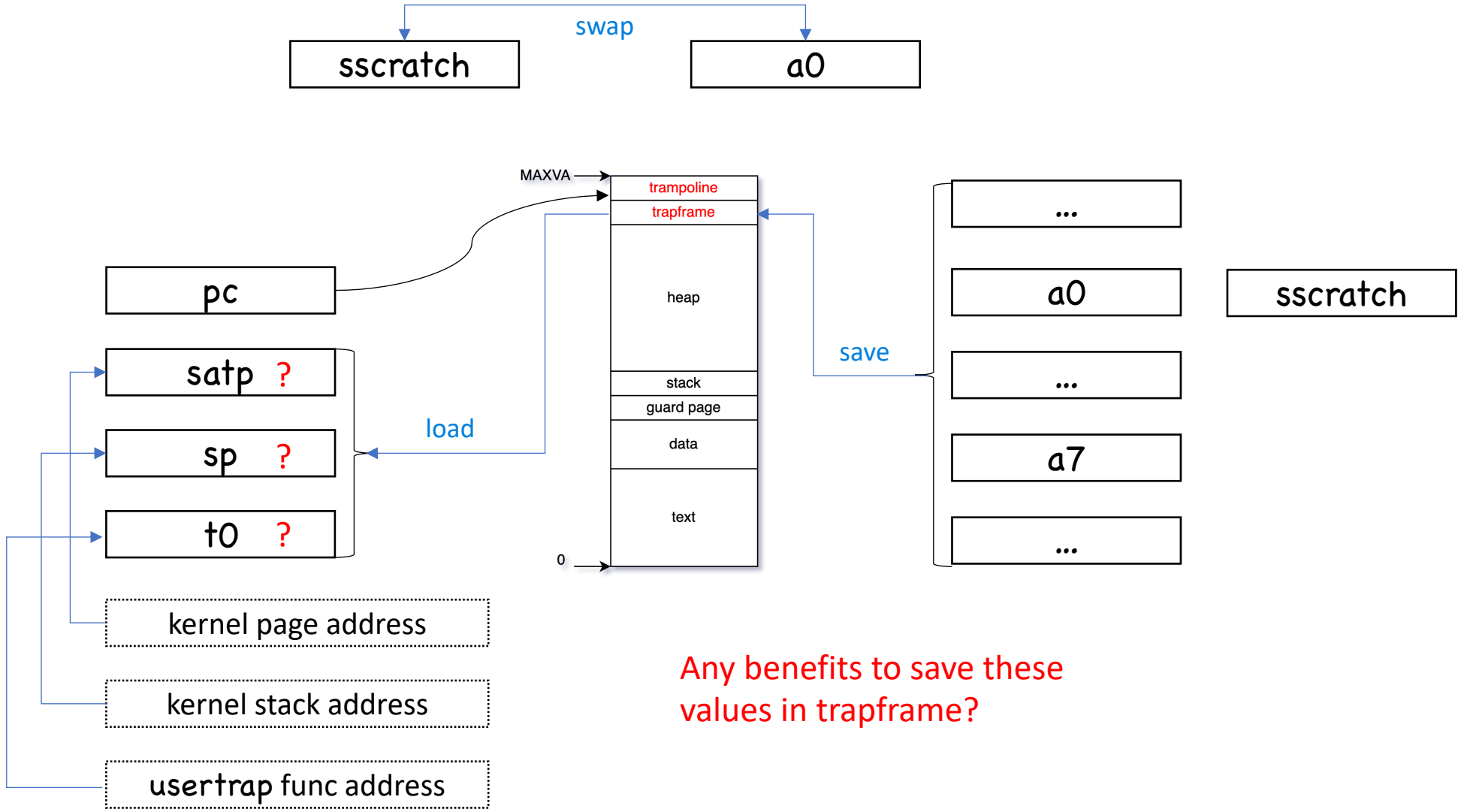
System Calls: CPU





How does CPU set the current mode to supervisor mode?

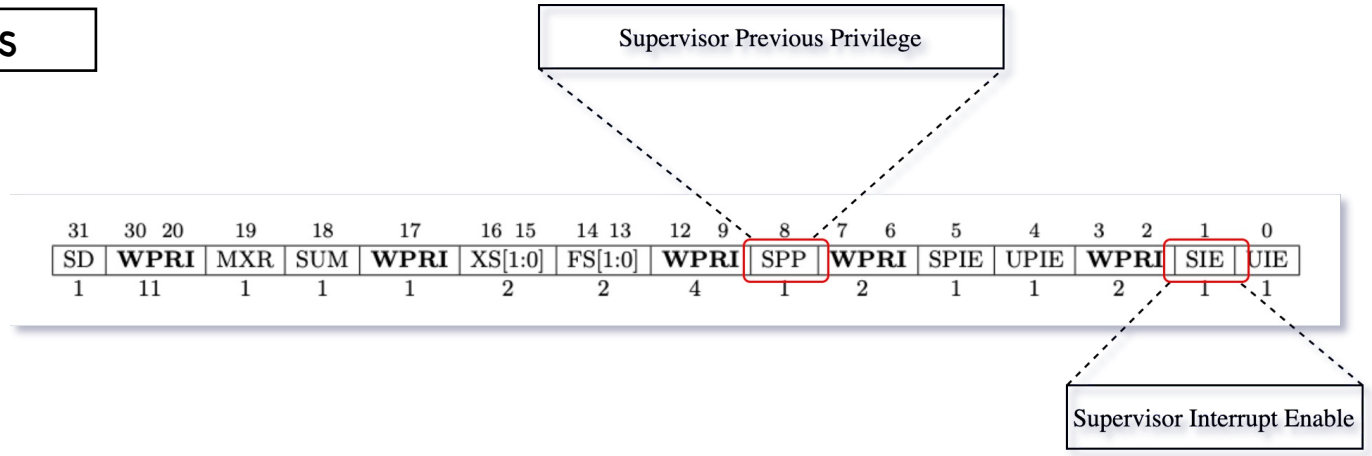
System Calls: OS



Any benefits to save these values in trapframe?

System Calls: OS

sstatus



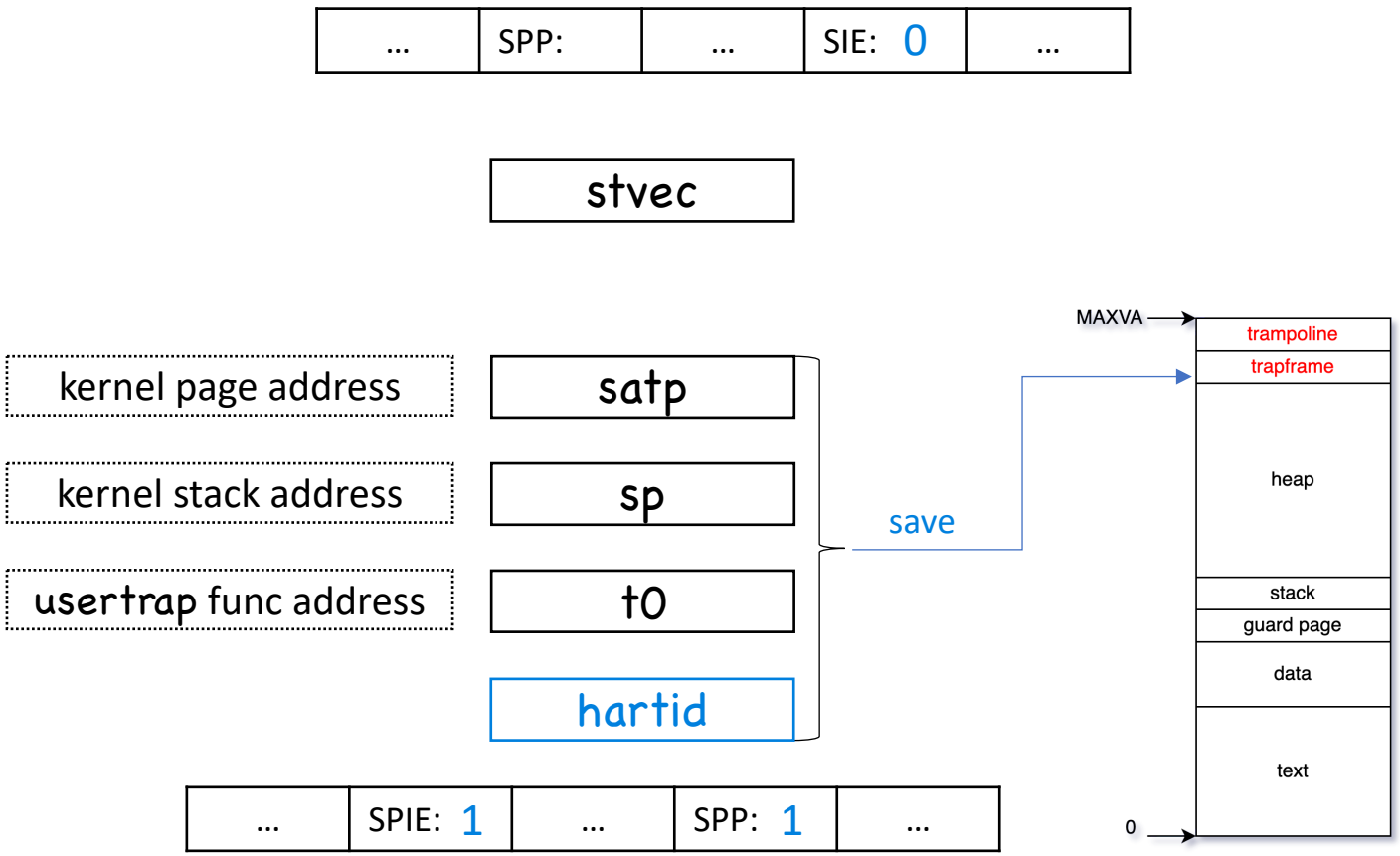
Which part will enable SIE again?

...	SPP: 1	...	SIE: 0	...

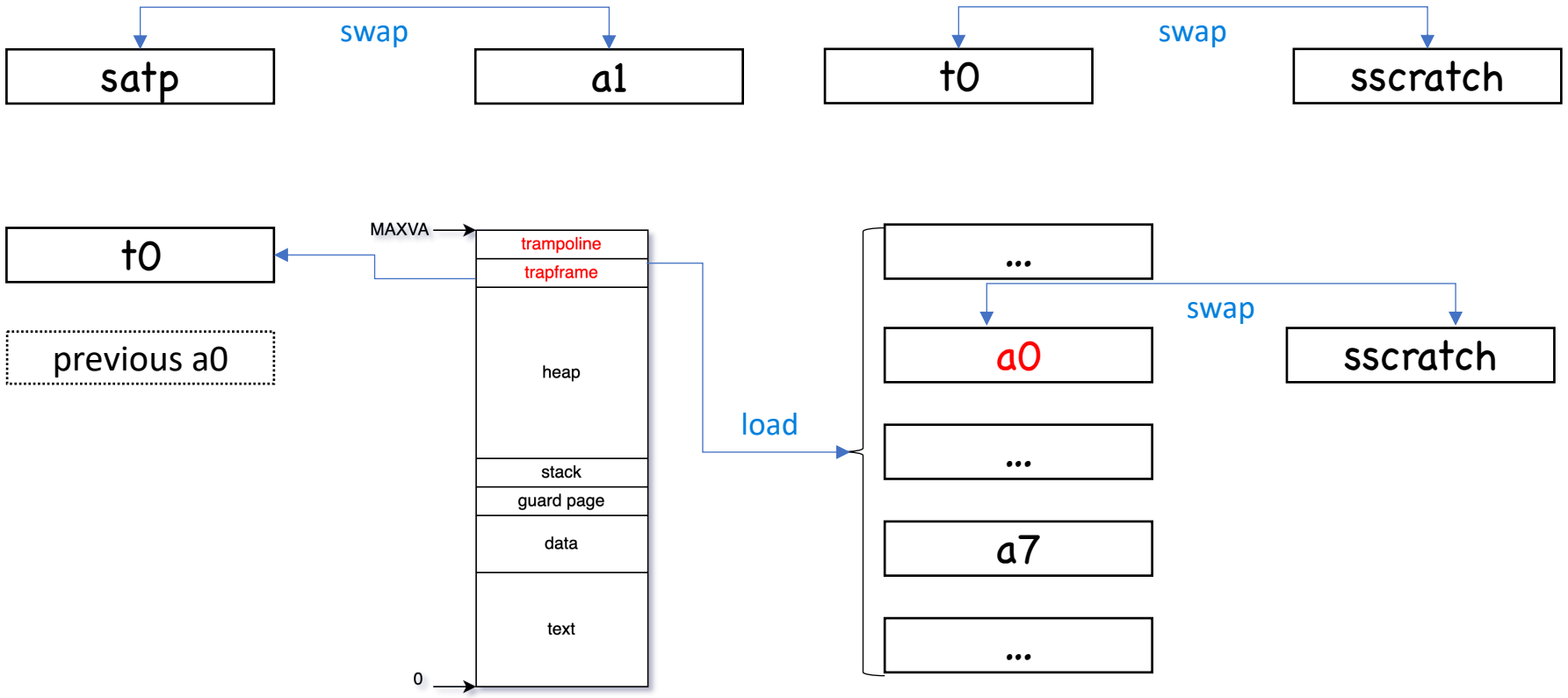
System Calls: OS

```
132  void
133  syscall(void)
134  {
135    int num;
136    struct proc *p = myproc();
137
138    num = p->trapframe->a7;
139    if(num > 0 && num < NELEM(syscalls) && syscalls[num]) {
140      p->trapframe->a0 = syscalls[num]();
141    } else {
142      printf("%d %s: unknown sys call %d\n",
143              p->pid, p->name, num);
144      p->trapframe->a0 = -1;
145    }
146  }
147
```

System Calls: OS



System Calls: OS



Summary

- Trampoline & Trapframe
- Registers in Risc-V
- CPU preparation for system calls
- OS preparation for system calls
- Next
  - Hongwang Li