SYSTEMS
GOSSIP
MEETUP

Learning large systems using peer-to-peer gossip

Policy Against Harassment at ACM Activities

https://www.acm.org/about-acm/policy-against-harassment

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.

Previously…

Time sharing - “allow us to run one process for a few milliseconds, and then another process for a few milliseconds, and so forth.” This mechanism gives the kernel ability to multiplex processes to the same hardware

User/Kernel mode - “OS puts a strict restrictions on what user processes can do and can’t do. Anything that is unsafe should be a privileged op.” This mechanism gives the kernel the ability to strongly isolate processes.

Previously…

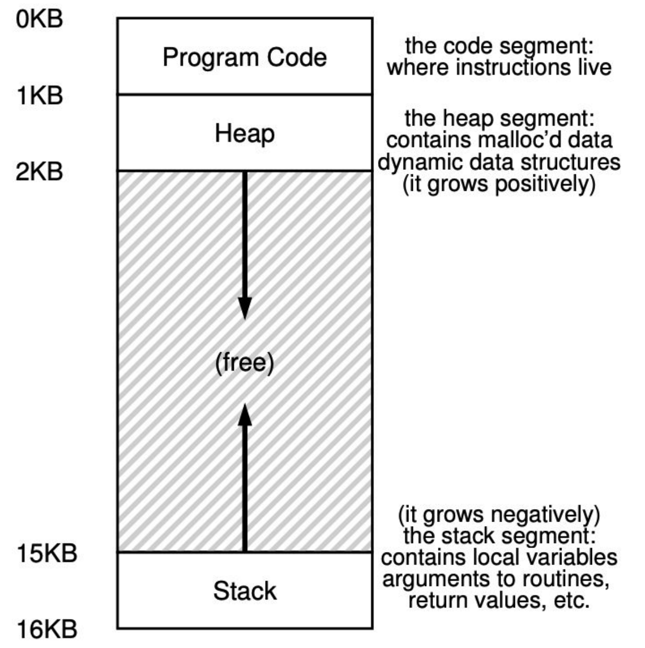
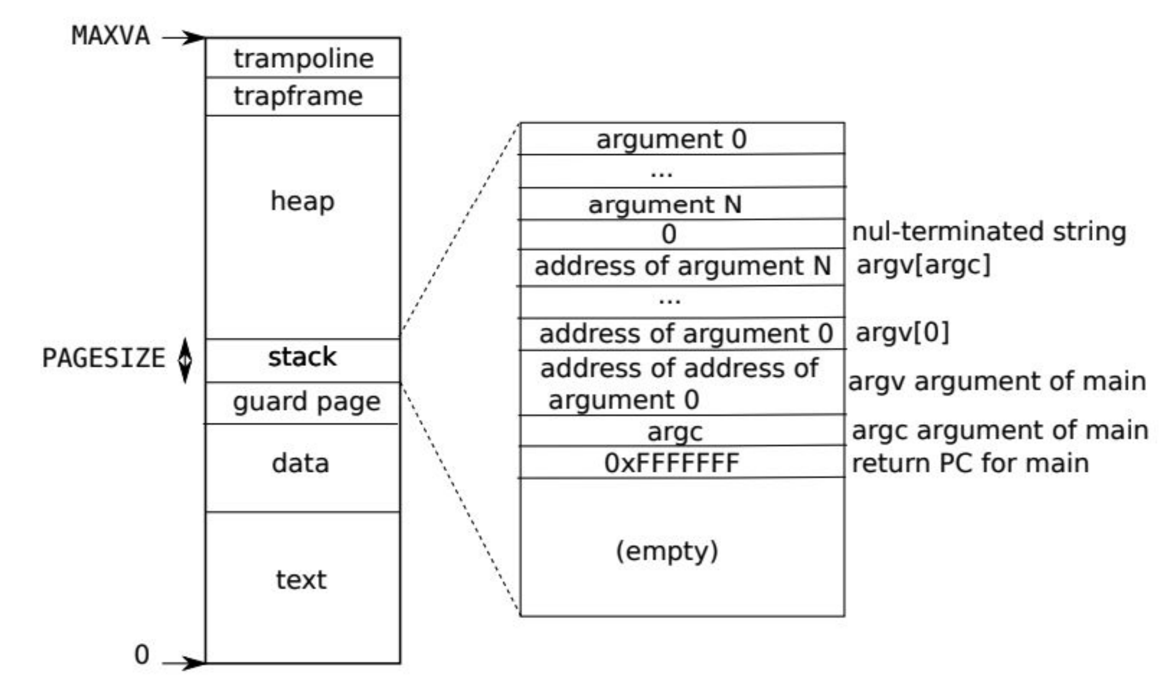
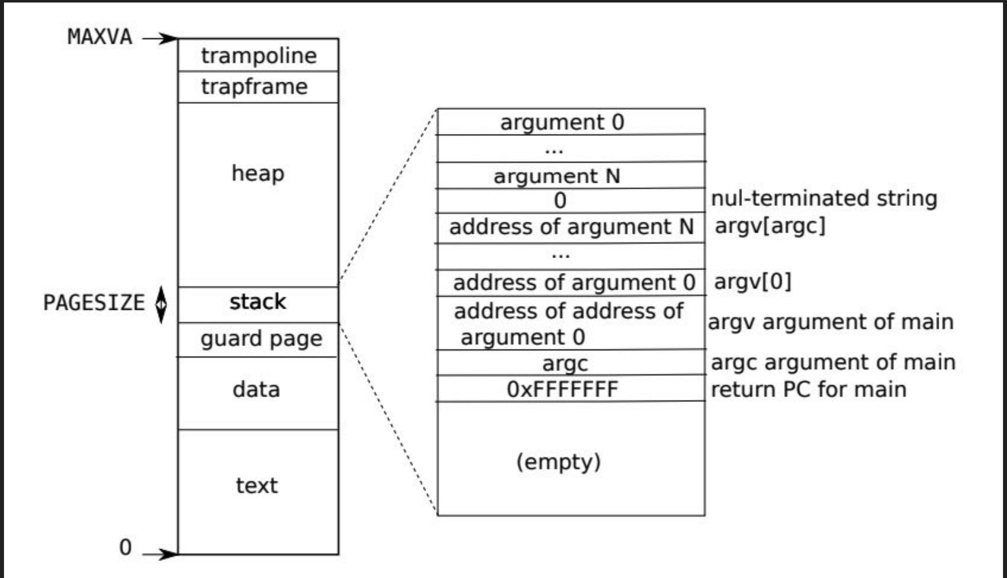


Figure 13.3: **An Example Address Space**



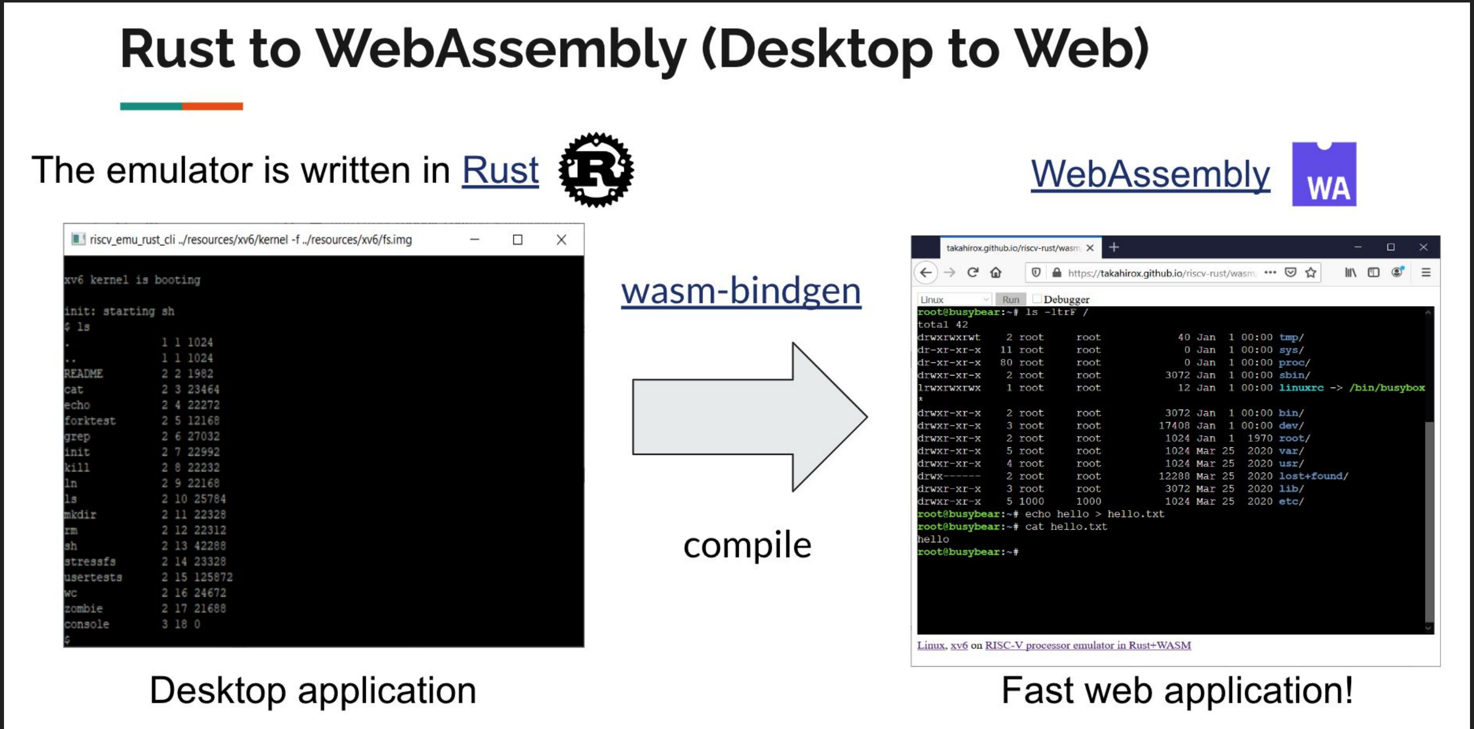
Previously...




```c
int main(int argc, char *argv[])
{
    printf("location of code : %p\n", main);
    printf("location of heap : %p\n", malloc(sizeof(int)));
    printf("location of stack : %p\n", &argc);
    exit(0);
}
```

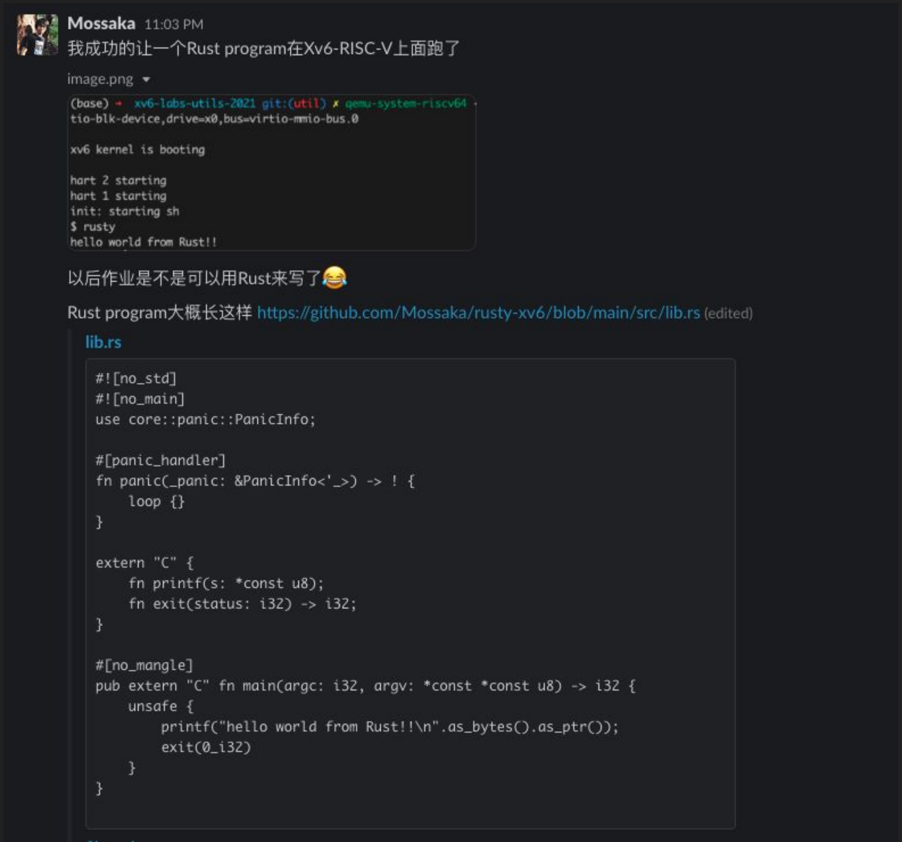
```
xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$ process_memory
location of code : 0x0000000000000000
location of heap : 0x0000000000012FF0
location of stack : 0x0000000000002FCC
```

Xv6 is so fun to hack!



https://docs.google.com/presentation/d/1qeR6KMSaJTR0ZSa2kLxgcBuc_zBo3l-kYbOpq1Wqmi0/edit?usp=sharing

Xv6 is so fun to hack!



https://github.com/Mossaka/rusty-xv6

⚠️ Lab 2 Syscalls is due next Saturday 📅 July 30th

Today

- Lab 1 “Primes”
- Strong isolation revisited
- Page Table
- Page Table in Xv6

Strong Isolation

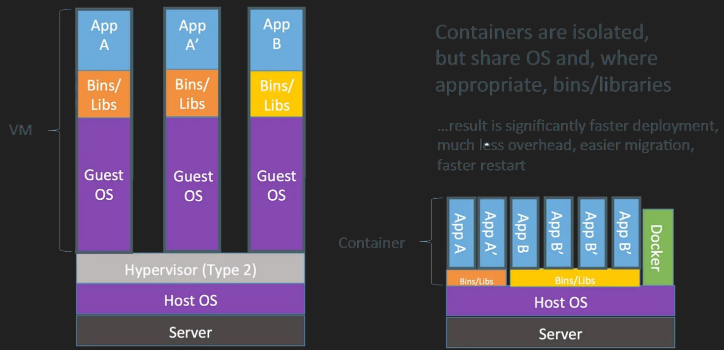
We isolate processes so that one cannot mess up the memory of another process

1. **Hardware approach:** hardware addressing mechanism using virtual memory.
2. **OS approach:** User/Kernel mode to strict what user process can do and can’t do. Microkernel as another example
3. **Software approach:** software-based fault isolation, which transforms one program to another program, which is guaranteed to satisfy security policies (e.g. WebAssembly)

Strong Isolation drives app abstraction

In multi tenant cloud era, how does the notion of strong isolation drive the evolution of application abstractions?

- **Strong security**: Virtual Machine provides OS-level protection to multi tenant applications
- **High density**: Containers share the same underlying OS, isolate application and its dependencies
- **Serverless:** Container + lightweight VM
  - See AWS Firecracker paper
- **Nano processes:** Lighter than containers?



Today, we will focus on hardware approach to achieve strong isolation.

This notion is also noted as a level of indirection

Address Space & Page Tables

38 – 30	29 – 21	20 – 12	11 – 0
VPN[2]	VPN[1]	VPN[0]	page offset
9	9	9	12

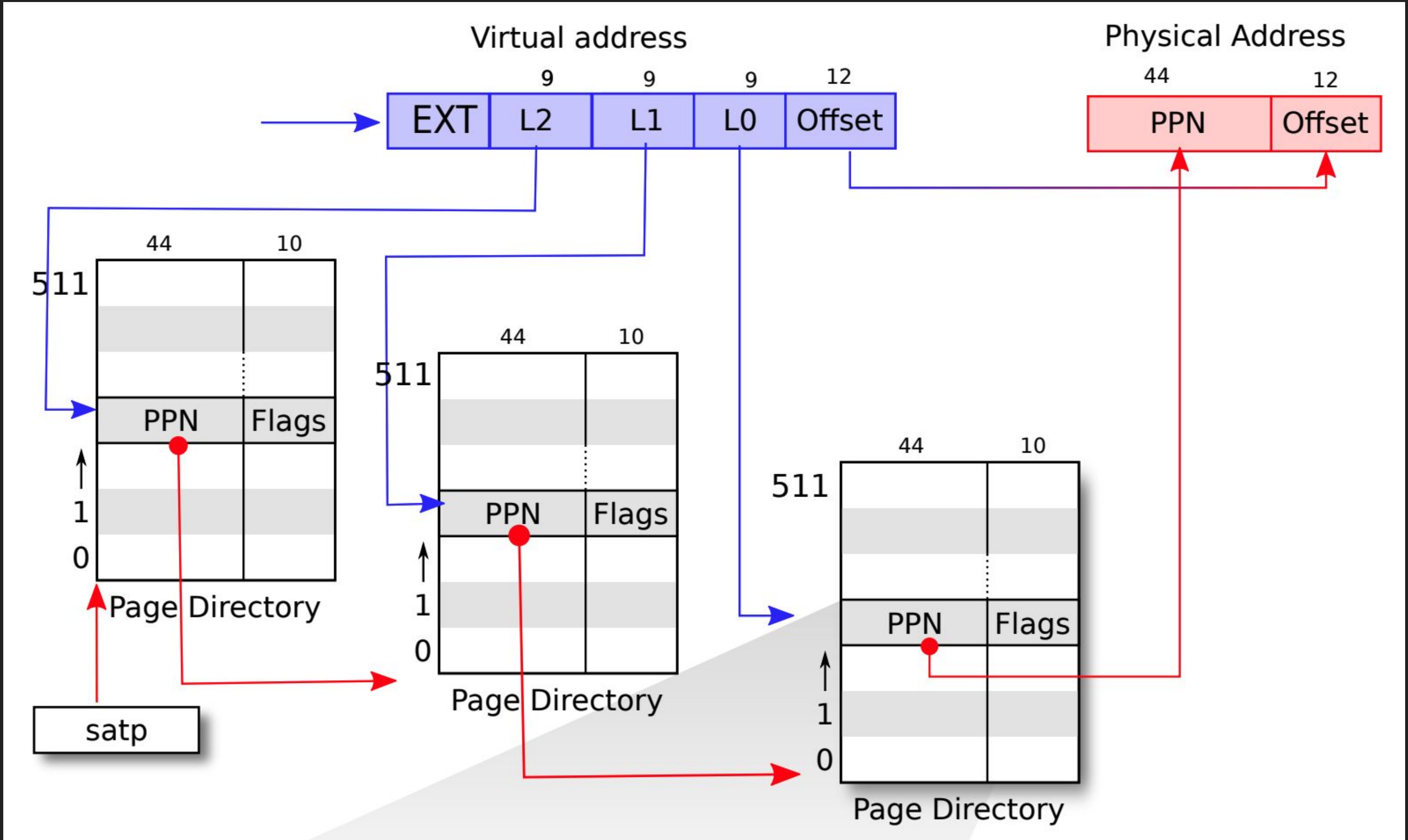
Figure 4.19: Sv39 virtual address.

55 – 30	29 – 21	20 – 12	11 – 0
PPN[2]	PPN[1]	PPN[0]	page offset
26	9	9	12

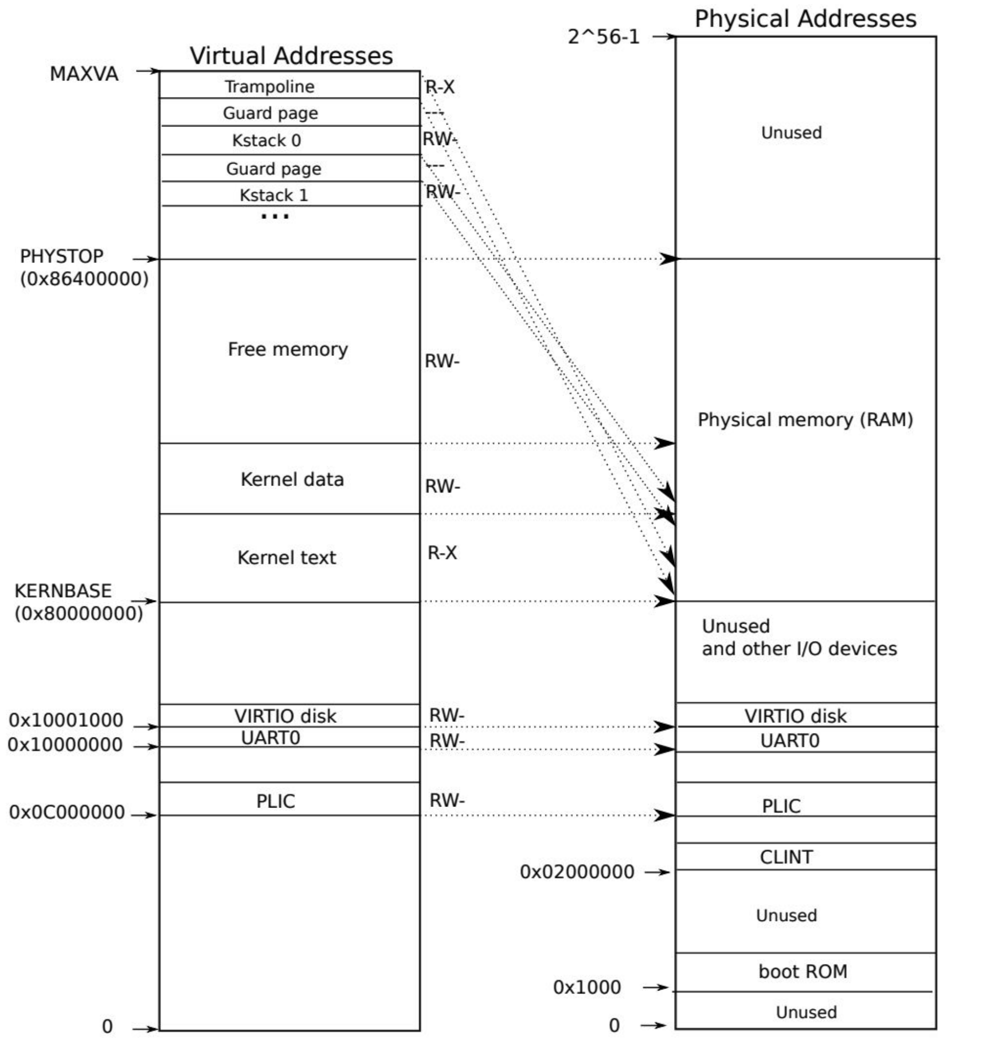
Figure 4.20: Sv39 physical address.

63	62 – 61	60 – 54	53 – 28	27 – 19	18 – 10	9 – 8	7	6	5	4	3	2	1	0
N	PBMT	*Reserved*	PPN[2]	PPN[1]	PPN[0]	RSW	D	A	G	U	X	W	R	V
1	2	7	26	9	9	2	1	1	1	1	1	1	1	1

Figure 4.21: Sv39 page table entry.



Virtual address
9
9
9
12
EXT
L2
L1
L0
Offset
Physical Address
44
12
PPN
Offset
44
10
511
PPN
Flags
1
0
Page Directory
satp
44
10
511
PPN
Flags
1
0
Page Directory
44
10
511
PPN
Flags
1
0
Page Directory



Virtual Addresses
Physical Addresses
MAXVA
Trampoline
R-X
Guard page

Kstack 0
RW-
Guard page

Kstack 1
RW-
...
PHYSTOP
(0x86400000)
Free memory
RW-
Kernel data
RW-
Kernel text
R-X
KERNBASE
(0x80000000)
0x10001000
VIRTIO disk
RW-
0x10000000
UART0
RW-
PLIC
RW-
0x0C000000
0
2^56-1
Unused
Physical memory (RAM)
Unused
and other I/O devices
VIRTIO disk
UART0
PLIC
CLINT
0x02000000
Unused
boot ROM
0x1000
Unused
0

Q&A

Q1: What is an address space?

- Q1.1: Is address space the same as memory space?
- Q1.2: Are addresses the same as indexes?

Q2: How big is a page table?

- Q2.1: How many PTEs does each page table have?
- Q2.2: What's the maximum number of pages could one process have?

Q&A

Q3: Why do we need a three-layer page table for each process?

Q3.1: What are the downsides of 3-layer page table?

Q3.2: How to mitigate the above issues?

Q4: How does one process finds its root page table?

Q5: What's the maximum physical address does RISC-V Sv39 support?

Q6: What's the maximum virtual address does RISC-V Sv39 support?

Q7: How much memory does Xv6 has?

Q&A

Q8: Where does DRAM starts in physical address space?

Q9: What are the use of the virtual address space beyond 128MB of memory?

Q10: What are the use of “guard pages” below kernel stack page?

Q11: What is the use of `sfence.vma` instr?

Q12: How to tell if the page table entry is out-of-date?

Next time

1. Don’t forget to sign up as presenters!
2. Lecture 6: Isolation and syscalls
3. Read chapter 4: traps and syscalls
4. Lab syscalls due next week!