6.S081 : Page faults

Plan: implement VM features using page faults

- lazy allocation
- COW fork
- demand paging
- mmap

Virtual memory benefits

1) Isolation

2) level of indirection

$$\begin{tikzcd} va \arrow[r, "\theta"] \arrow[d, bend right] & pa \arrow[r] \arrow[dr] & \text{trampoline} \\ \text{static} & & \text{guard page} \end{tikzcd}$$

using page faults; we change change the mapping

Information needed.

1) the faulting va

   Stval register $\leftarrow$ va

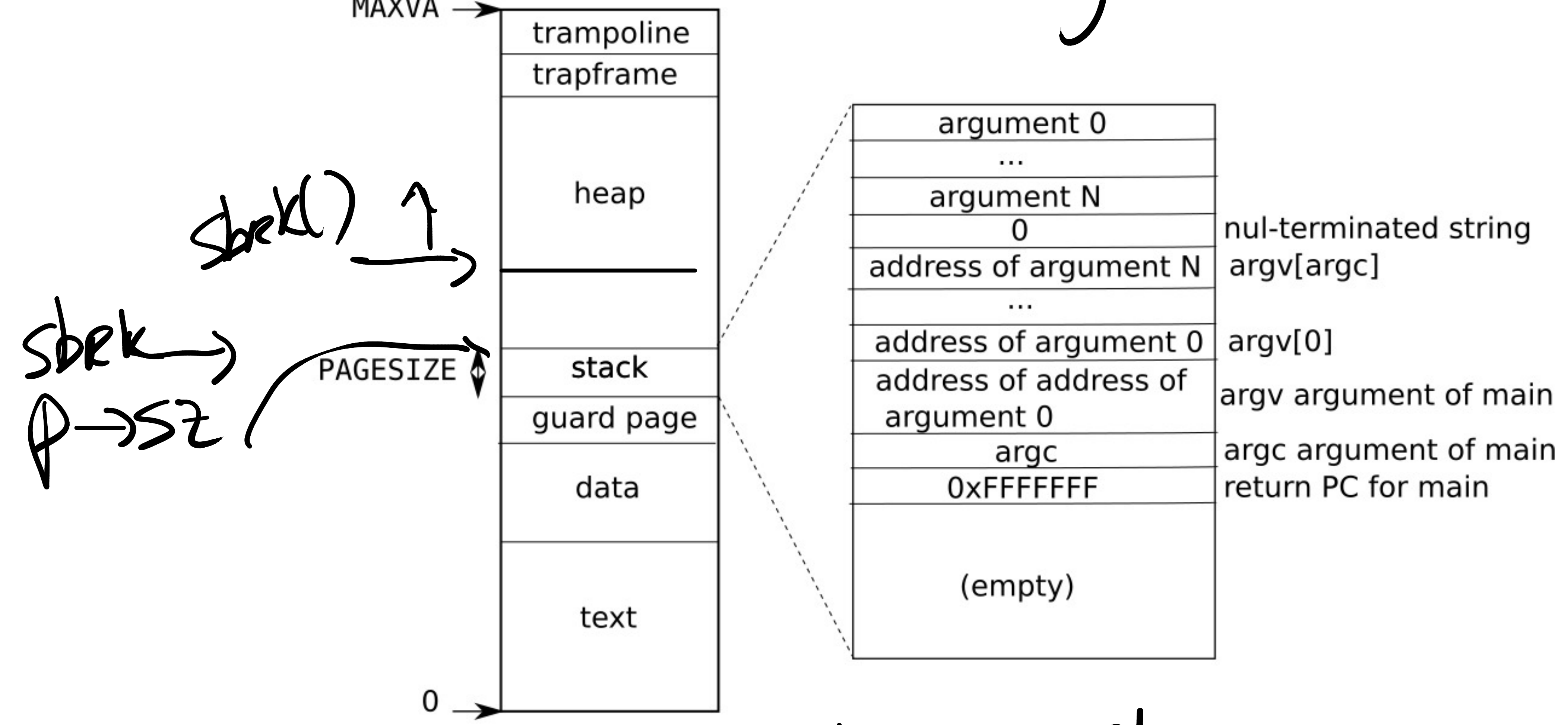
2) the type of page fault

$$\begin{tikzcd}[row sep=small] & R \\ \text{scause} \arrow[ur, dash] \arrow[r, dash] \arrow[dr, dash] & W \\ & I \end{tikzcd}$$

3) the va of instruction that caused the fault

   sepc   tf->epc $\leftarrow$

Allocation : sbrk() $\longrightarrow$ eager allocation



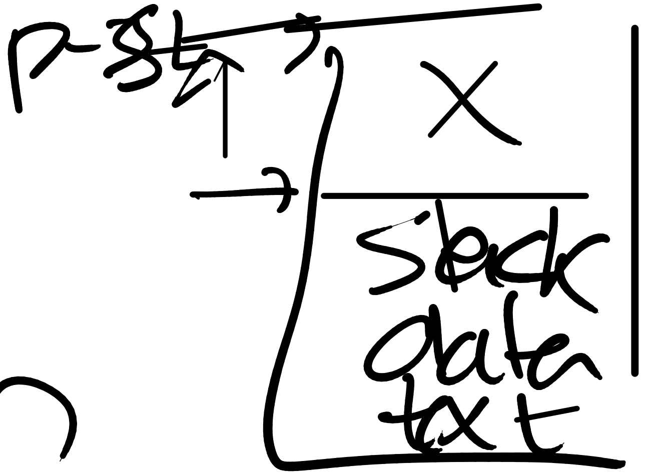
applications tend to over ask

Lazy allocation

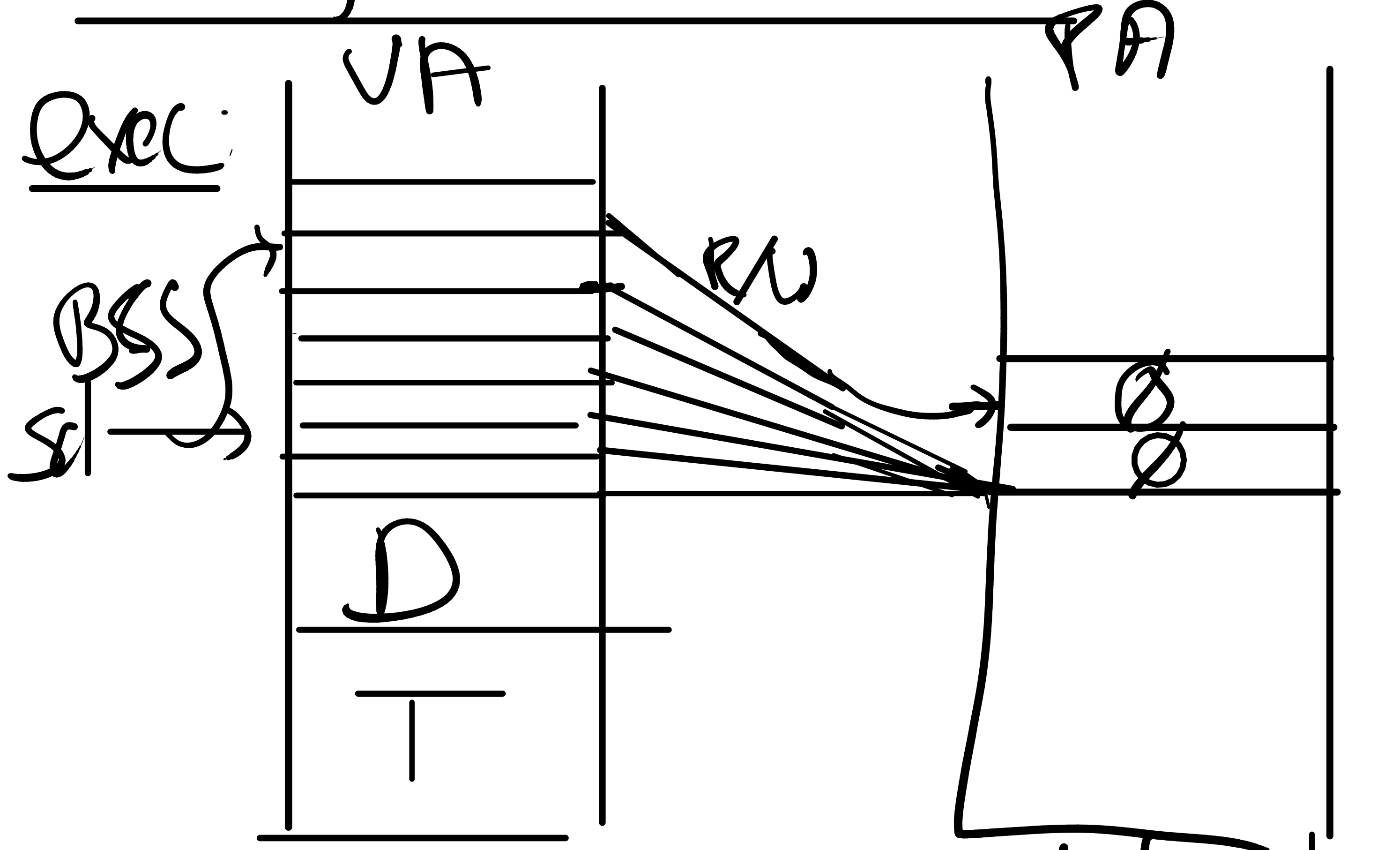
sbrk(): $p \to sz = p \to sz + n$

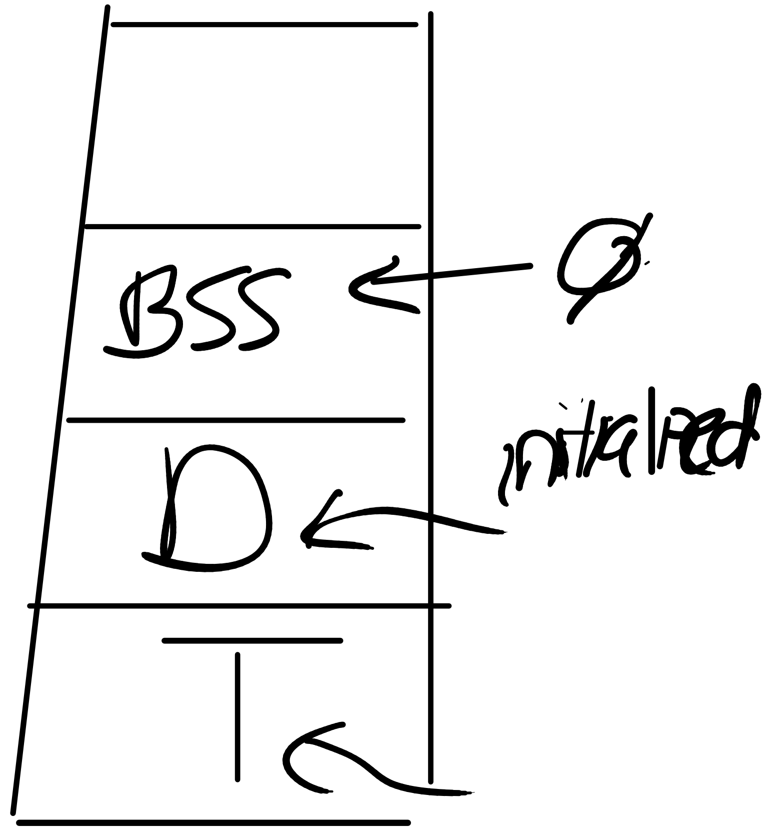
pgfault: $va \not< p \to sz$ > stack

allocate 1 page
zero the page
map the page.
restart instruction



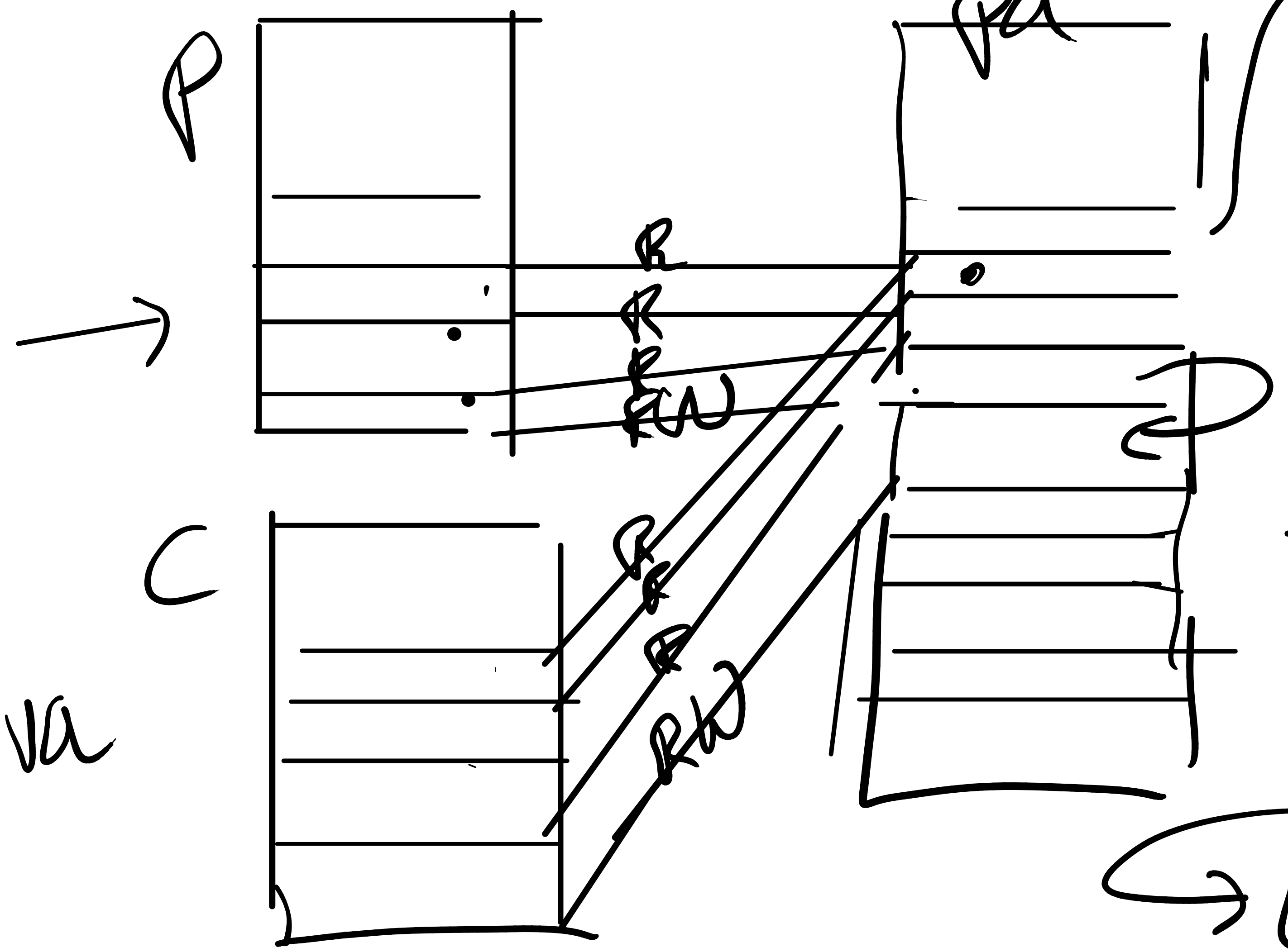
Zero-fill on demand.





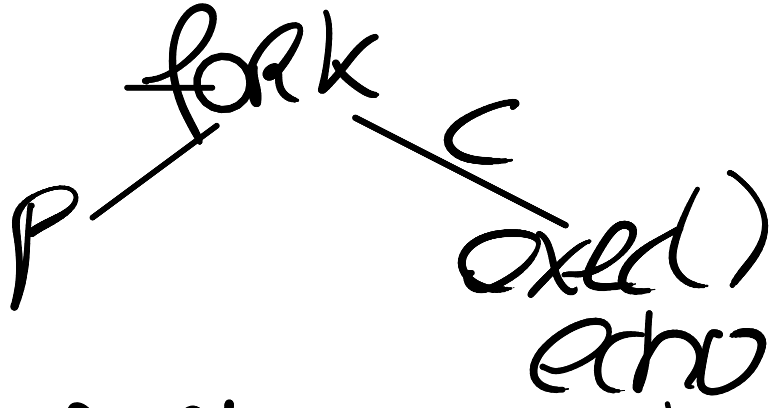
pgfault: copy + update pte
restart instruction

Copy-on-write (COW) fork. ← lab



→ Ref count

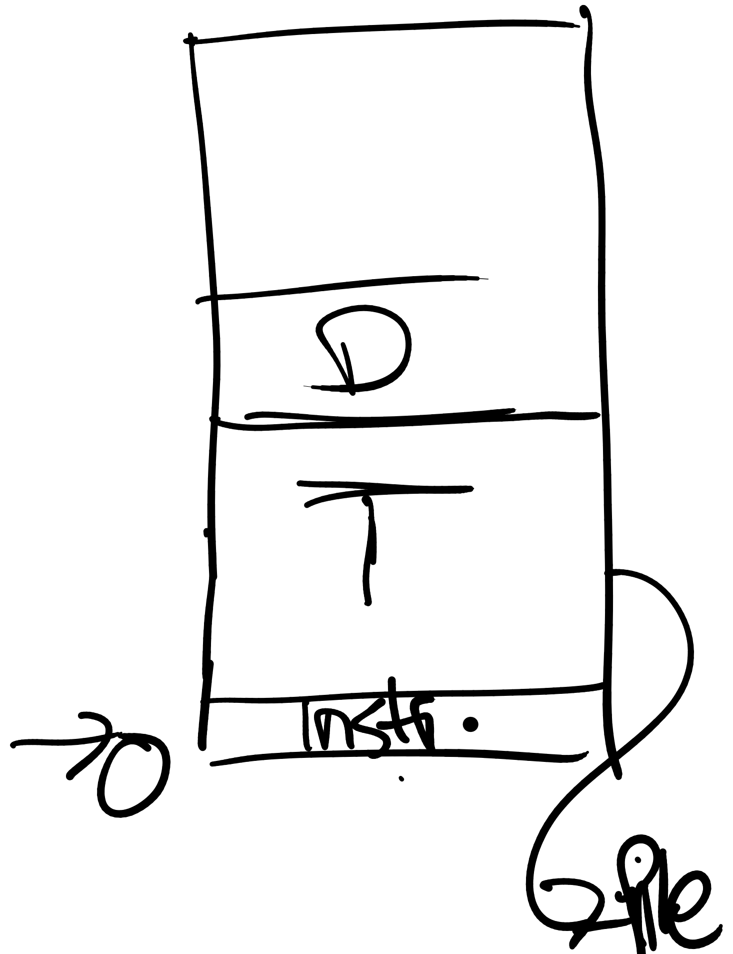
shell: fork

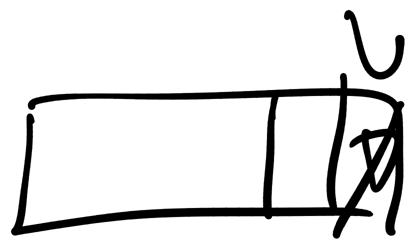


pgfault: COW lab

copy page
map it
restart instruction
→ usertrap()

Demand paging





exec():

load text
data
segment
eagerly → pgtable

pgfault:

- read block/page from F into mem
- map mem into pgtbl.
- restart instr.

Demand paging (2)

If out-of-memory:

evict a page → F

use the-just-free page

restart instruction.

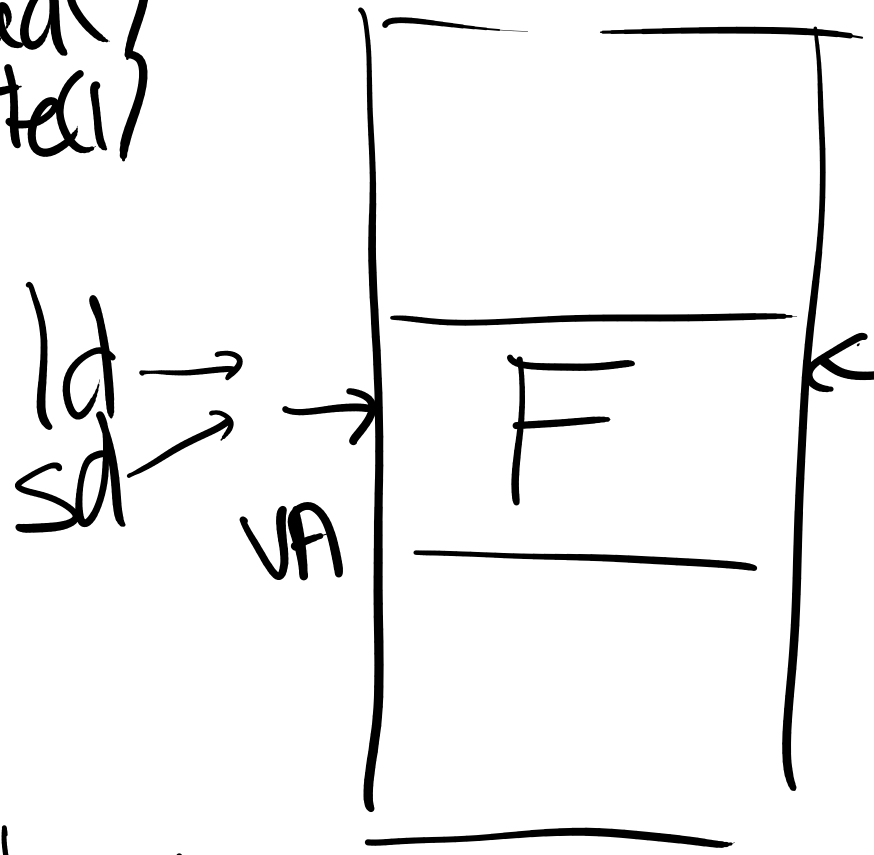
? least-recently-used (LRU).

non-dirty

Memory-mapped files

VMA = virtual memory area

read()
write()



mmap(va, len, prot, flags, fd, off)

(F

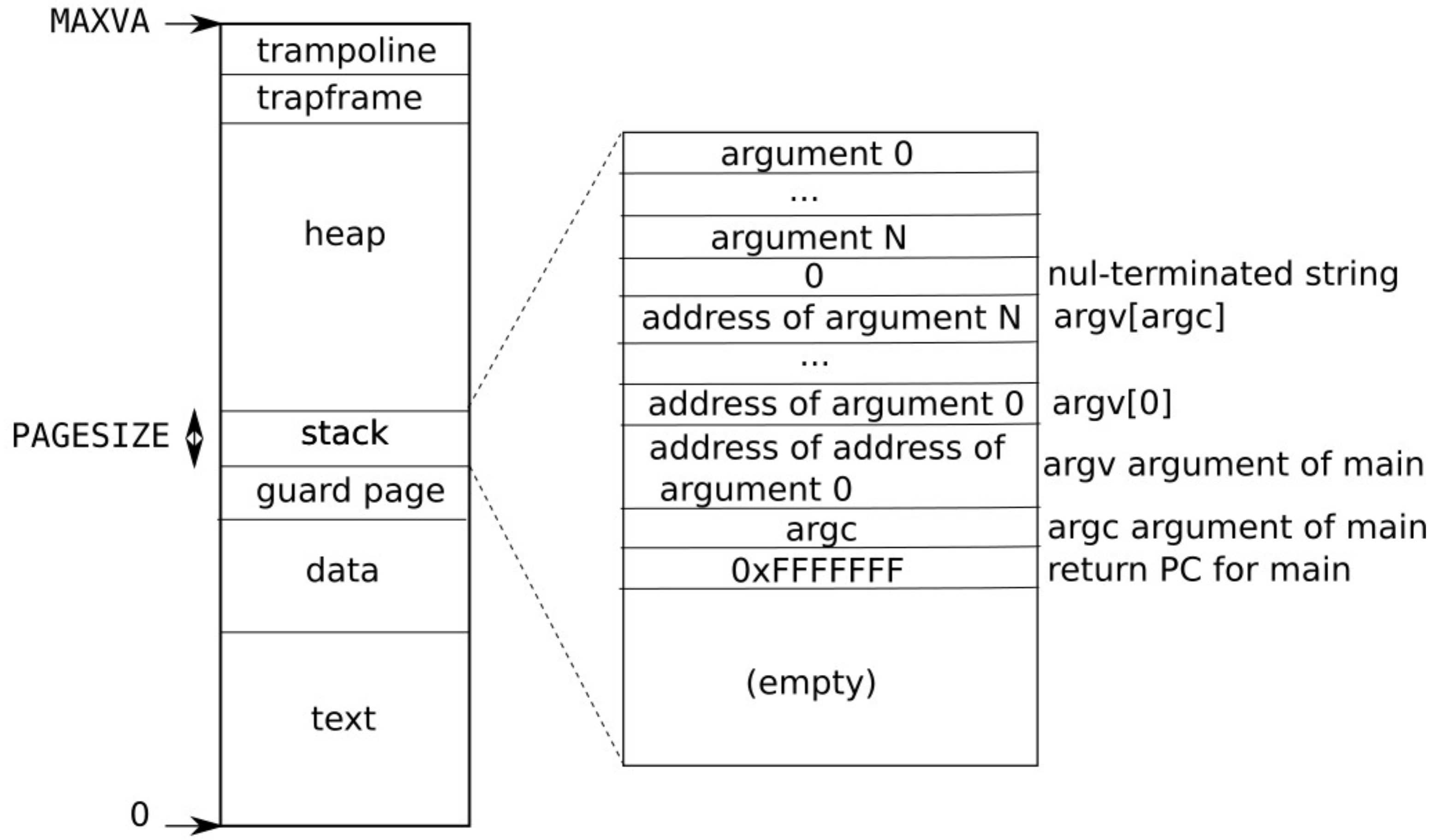
unmap(va, len)

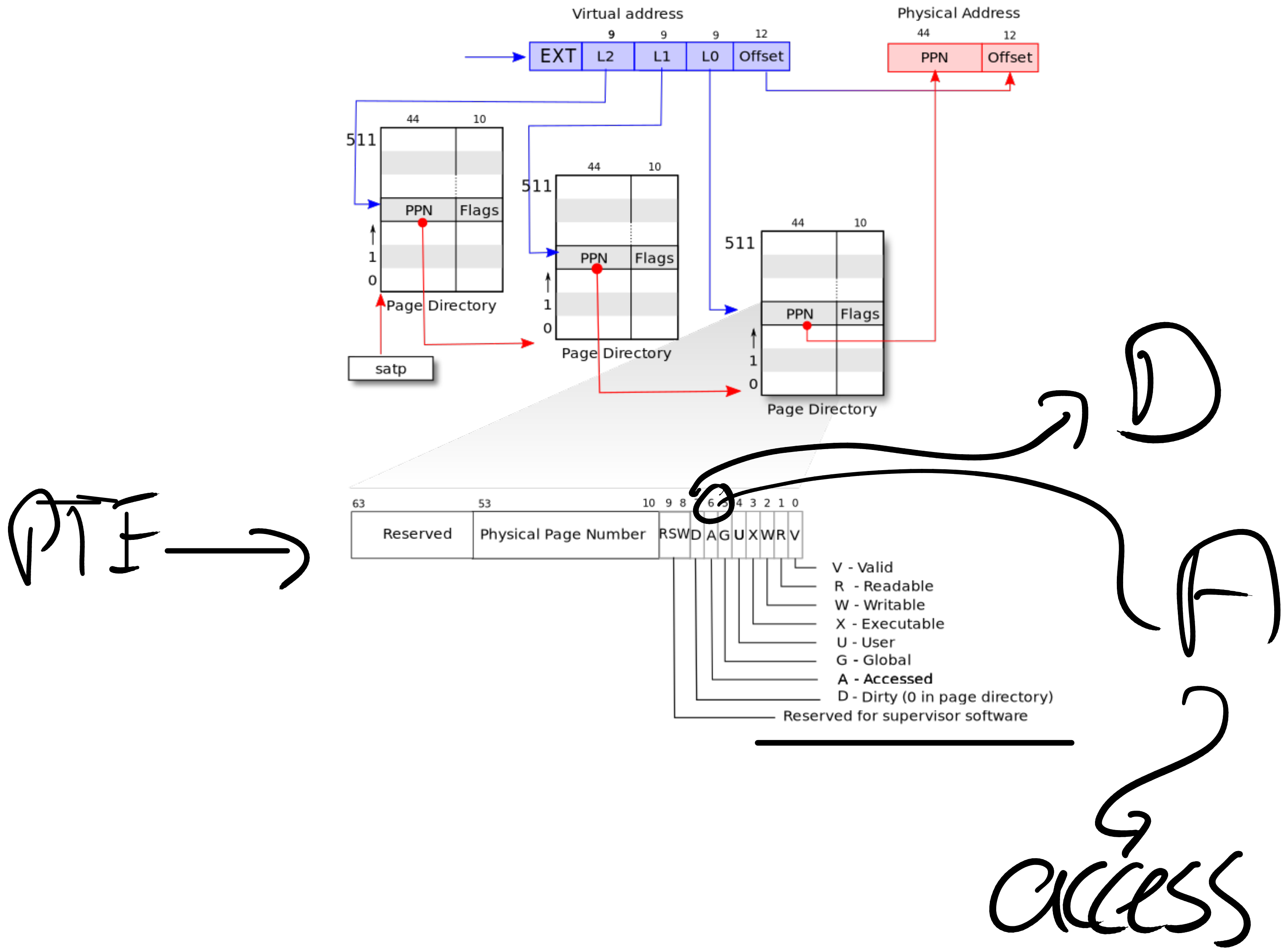
write back dirty blck

Lazily

Summary

$$\begin{tikzcd}
\text{page tables} \arrow[dr, dash] & & \\
& + \arrow[r, Rightarrow] & \text{powerful elegant VM features} \\
\text{traps / page fault} \arrow[ur, dash] & &
\end{tikzcd}$$


MAXVA
trampoline
trapframe
heap
PAGESIZE
stack
guard page
data
text
0
argument 0
...
argument N
0
nul-terminated string
address of argument N
argv[argc]
...
address of argument 0
argv[0]
address of address of argument 0
argv argument of main
argc
argc argument of main
0xFFFFFFF
return PC for main
(empty)


Virtual address
EXT
L2
L1
L0
Offset
9
9
9
12
Physical Address
PPN
Offset
44
12
44
10
511
PPN
Flags
1
0
Page Directory
satp
63
53
10
9
8
6
4
3
2
1
0
Reserved
Physical Page Number
RSW
D
A
G
U
X
W
R
V
V - Valid
R - Readable
W - Writable
X - Executable
U - User
G - Global
A - Accessed
D - Dirty (0 in page directory)
Reserved for supervisor software
PTE
D
A
access