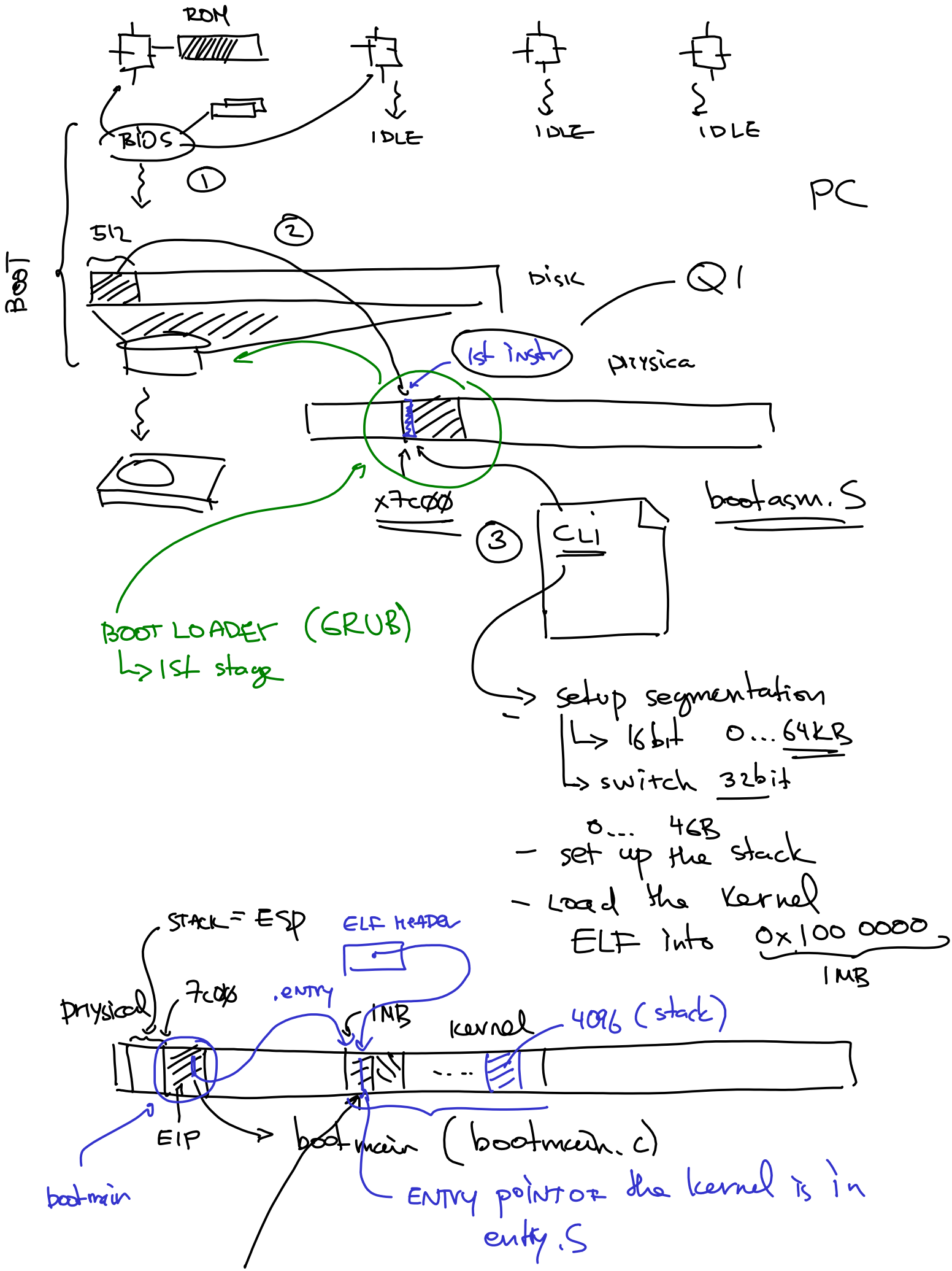
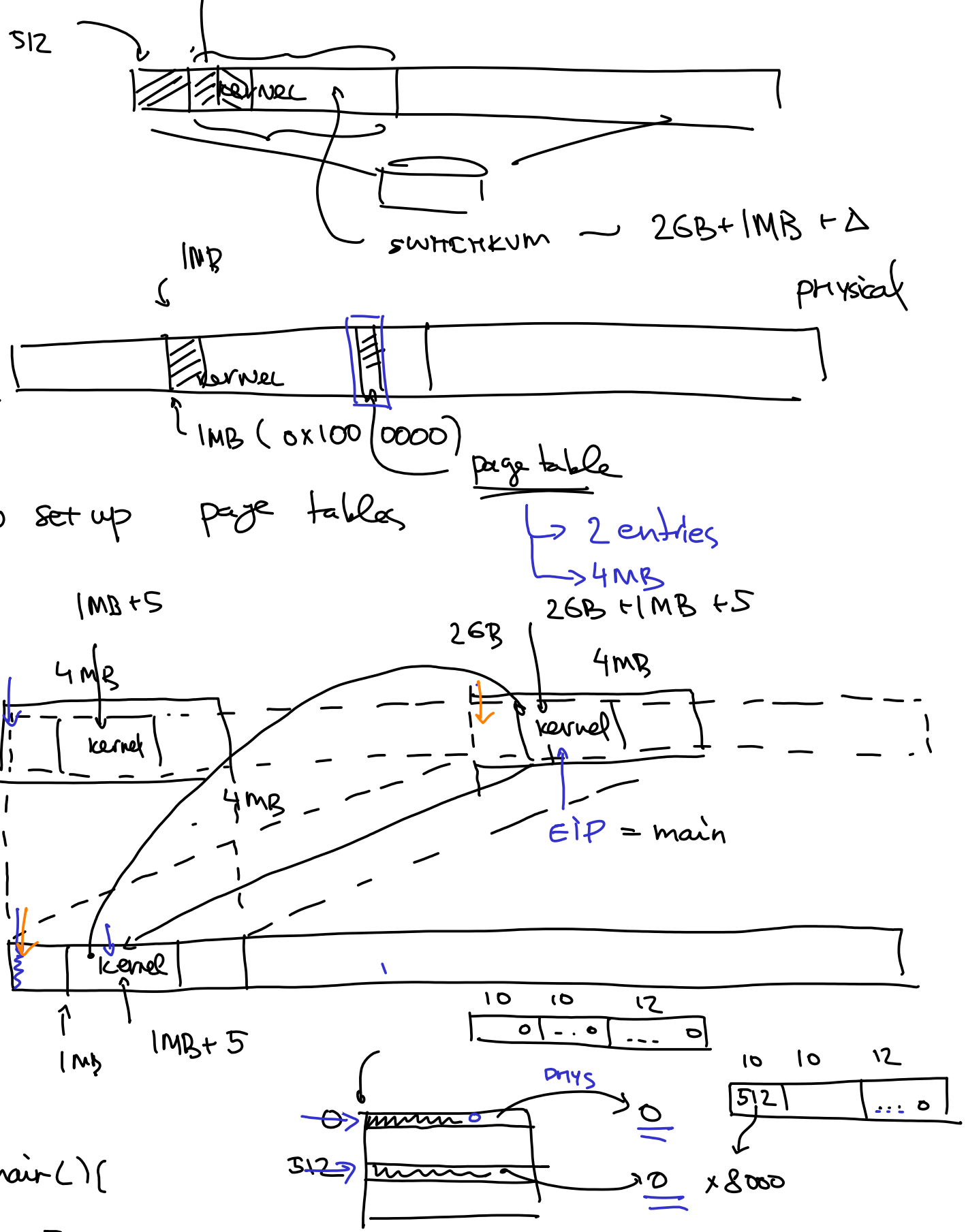
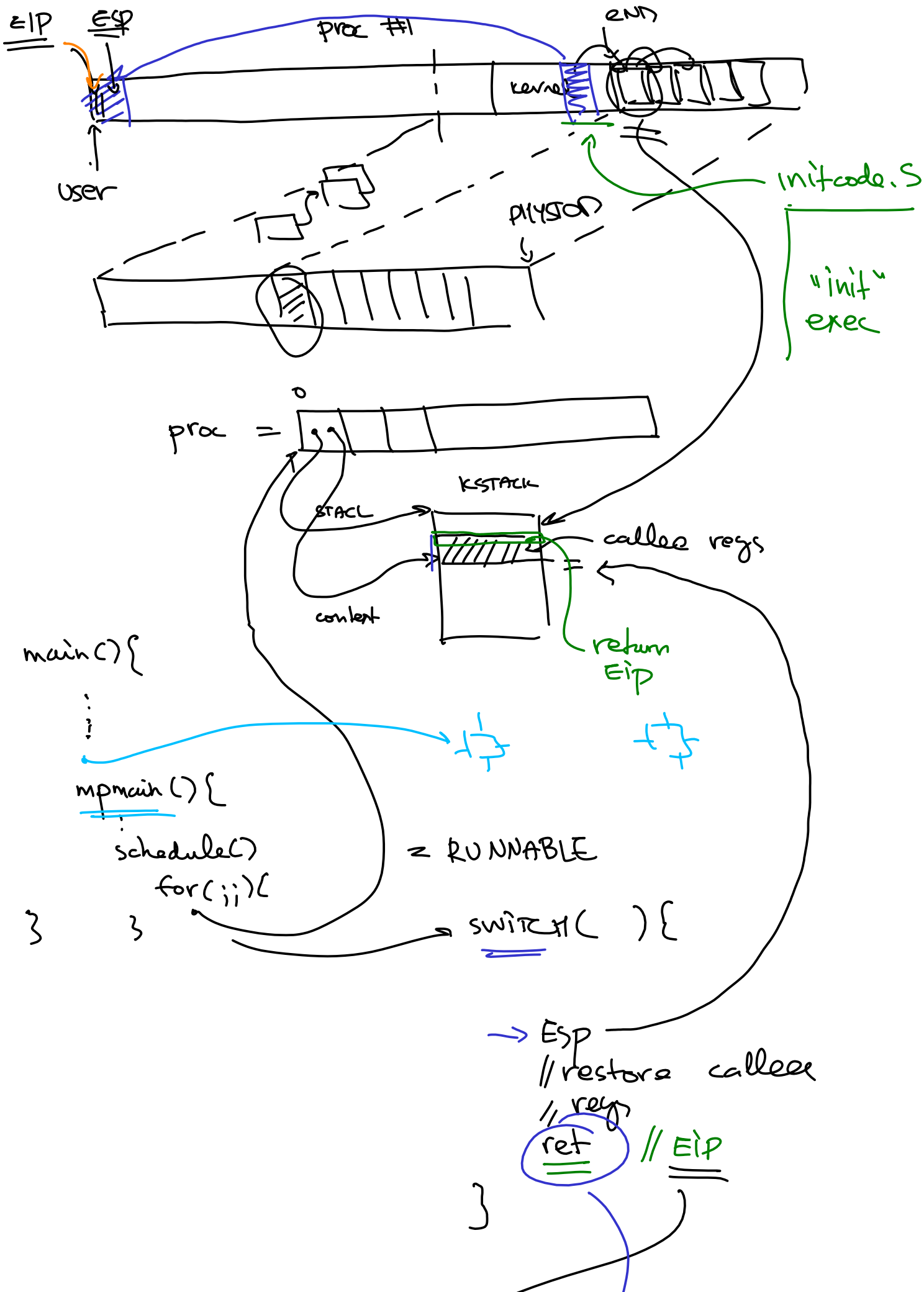
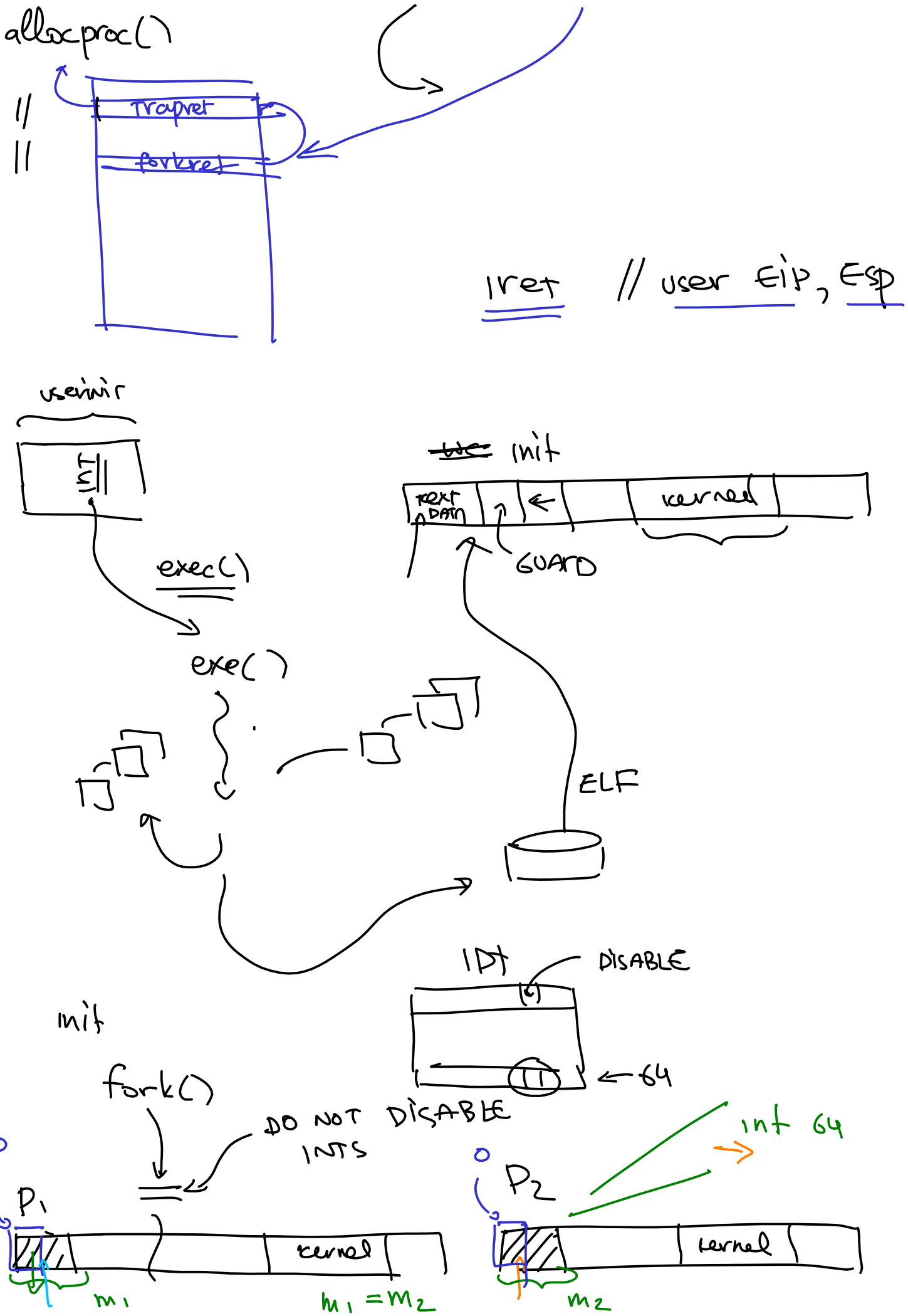
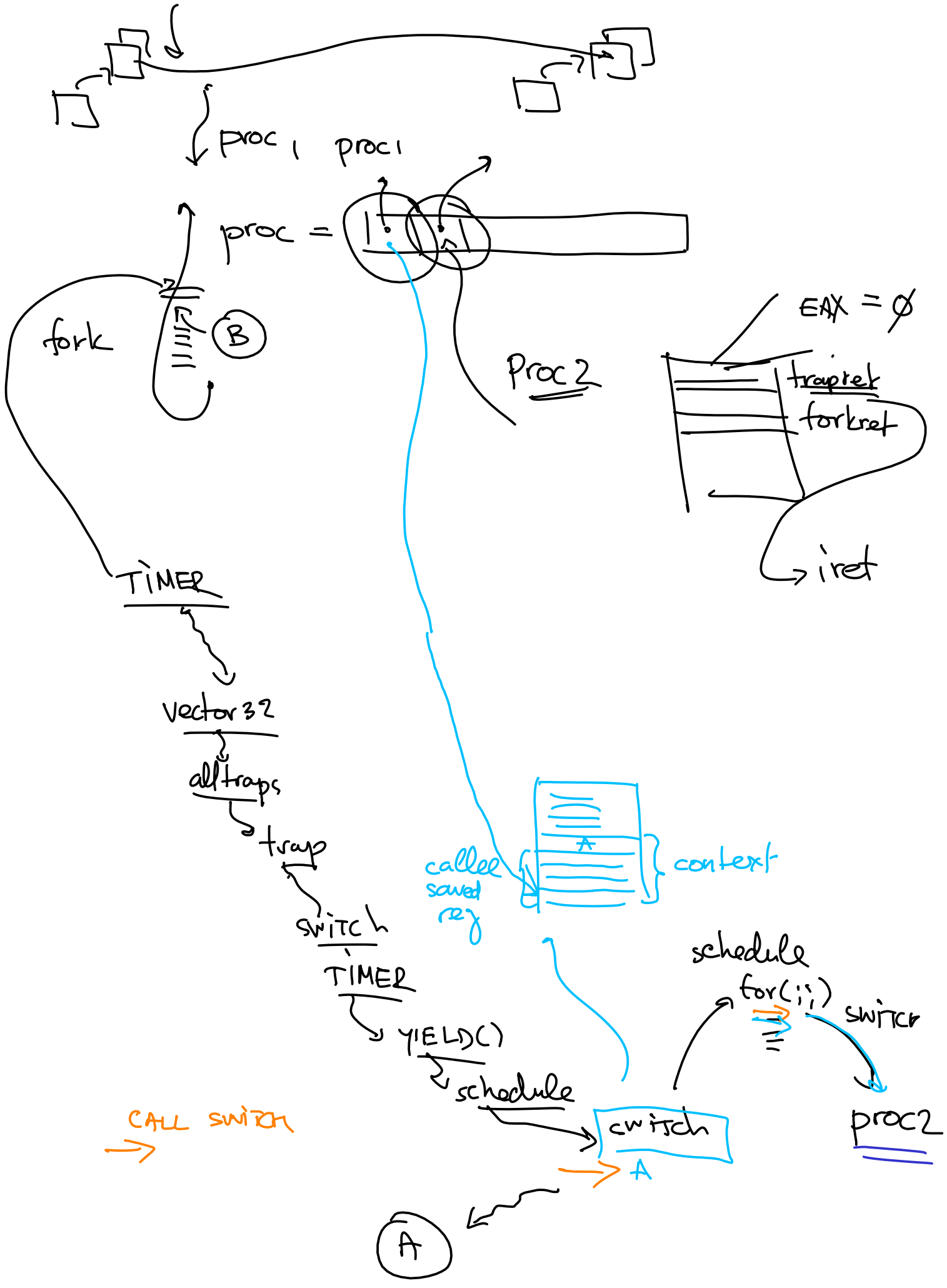
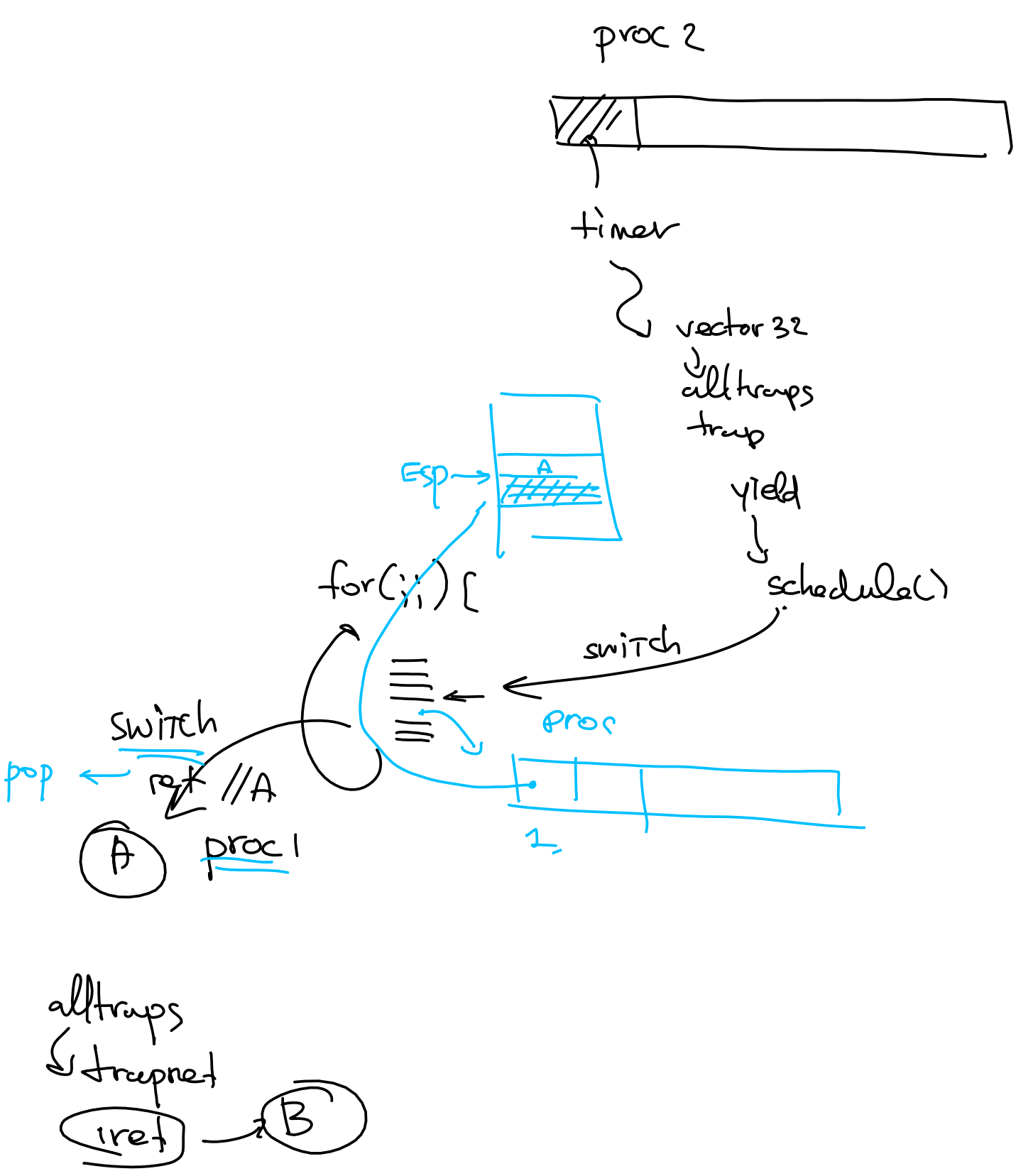

ROM
BIOS
IDLE
IDLE
IDLE
PC
BOOT
512
2
disk
Q1
1st instr
physica
x7c00
bootasm.S
3
CLI
BOOT LOADER (GRUB)
↳ 1st stage
→ setup segmentation
↳ 16bit 0...64KB
↳ switch 32bit
0... 4GB
- set up the stack
- Load the kernel
ELF into 0x100 0000
1MB
STACK = ESP
ELF HEADER
physical
7c00
.entry
1MB
kernel
4096 (stack)
EIP
→ bootmain (bootmain.c)
bootmain
ENTRY point of the kernel is in
entry.S


512
kernel
switchkvm ~ 2GB+1MB+Δ
1MB
physical
kernel
1MB (0x100 0000)
page table
set up page tables
2 entries
4MB
1MB+5
2GB+1MB+5
virt
2GB
4MB
4MB
kernel
kernel
0
4MB
EIP = main
phys
kernel
1MB
1MB+5
10 10 12
0 .. 0 ... 0
phys
0
512
0
10 10 12
512 ... 0
x8000
main(){
≡
← user init
}

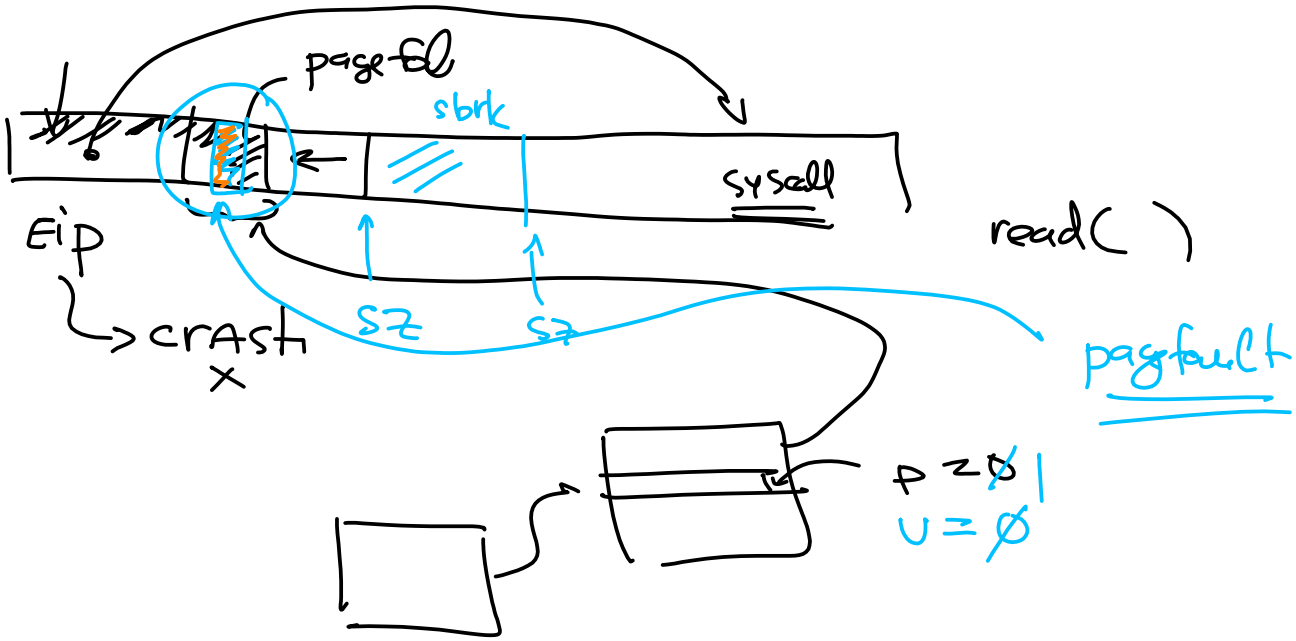

EIP
ESP
proc #1
end
kernel
user
initcode.S
"init"
exec
proc =
0
KSTACK
STACK
callee regs
context
return
Eip
main(){
mpmain(){
scheduler()
for(;;){
= RUNNABLE
}
}
swtch( ){
Esp
// restore callee
// regs
ret
// Eip
}

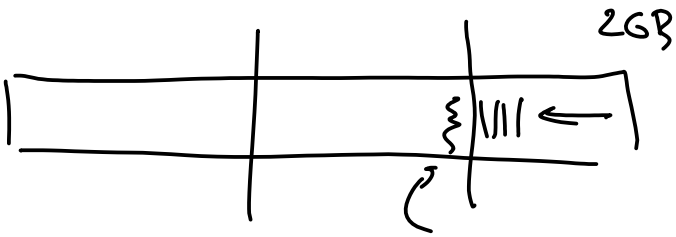

allocproc()
trapret
forkret
iret // user Eip, Esp
userinit
int
init
text
data
kernel
GUARD
exec()
exec()
ELF
IDT
DISABLE
64
init
fork()
DO NOT DISABLE INTS
int 64
P1
P2
kernel
kernel
m1
m1 = m2
m2


proc 1
proc 1
proc =
fork
B
Proc 2
EAX = 0
trapret
forkret
iret
TIMER
vector32
alltraps
trap
switch
TIMER
YIELD()
schedule
callee saved reg
A
context
schedule
for(;;)
switch
CALL SWITCH
switch
A
proc2
A



alltraps
↳ trapret
iret → B

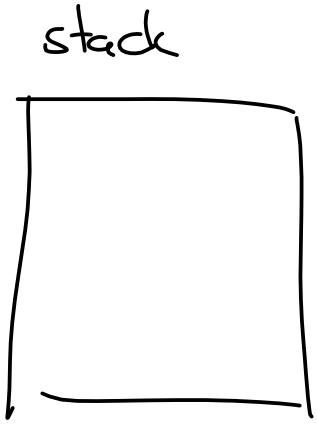




#1) ASM (x86 instruction set)

CALL } PUSH, JUMP } call

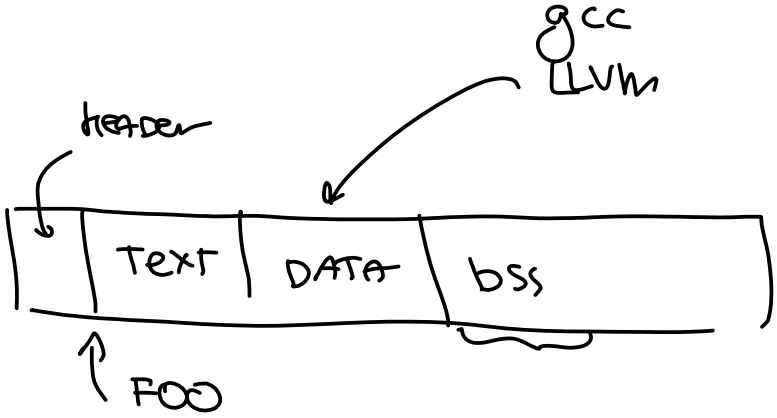
RET } POP, JUMP } ret



SWITCH

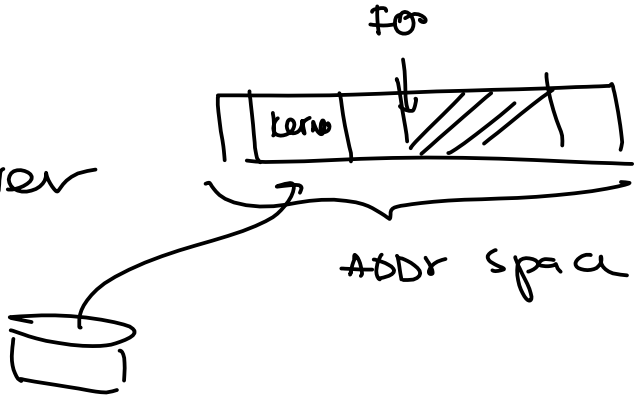
BOOT → stack (main)

#2) ELF



#3) Boot main

↳ Disk driver

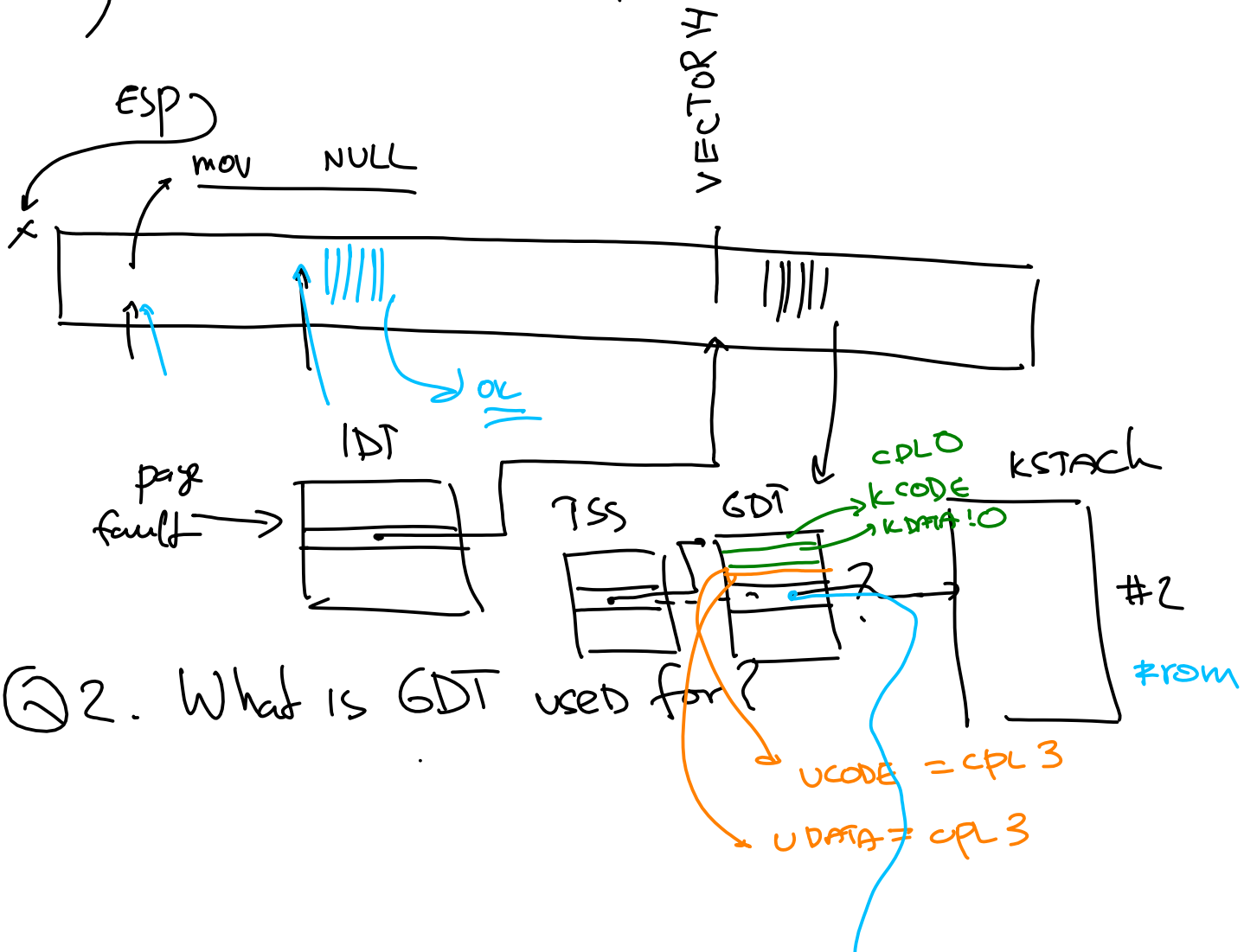


#4) Address spaces

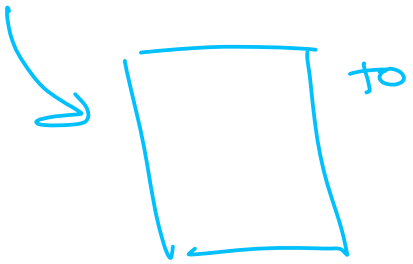
pages + segmentation

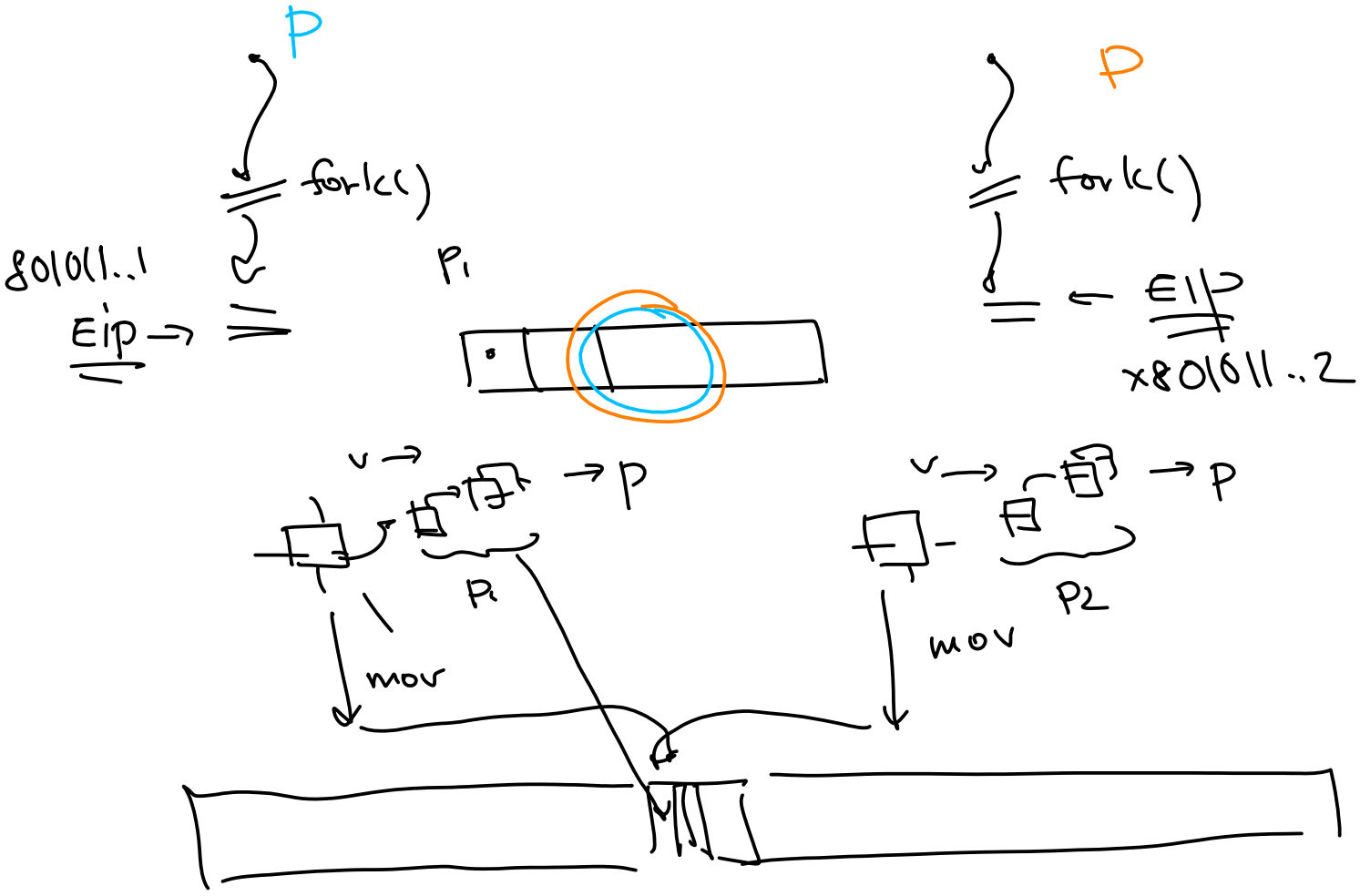
#5) INT

#6) Context switch



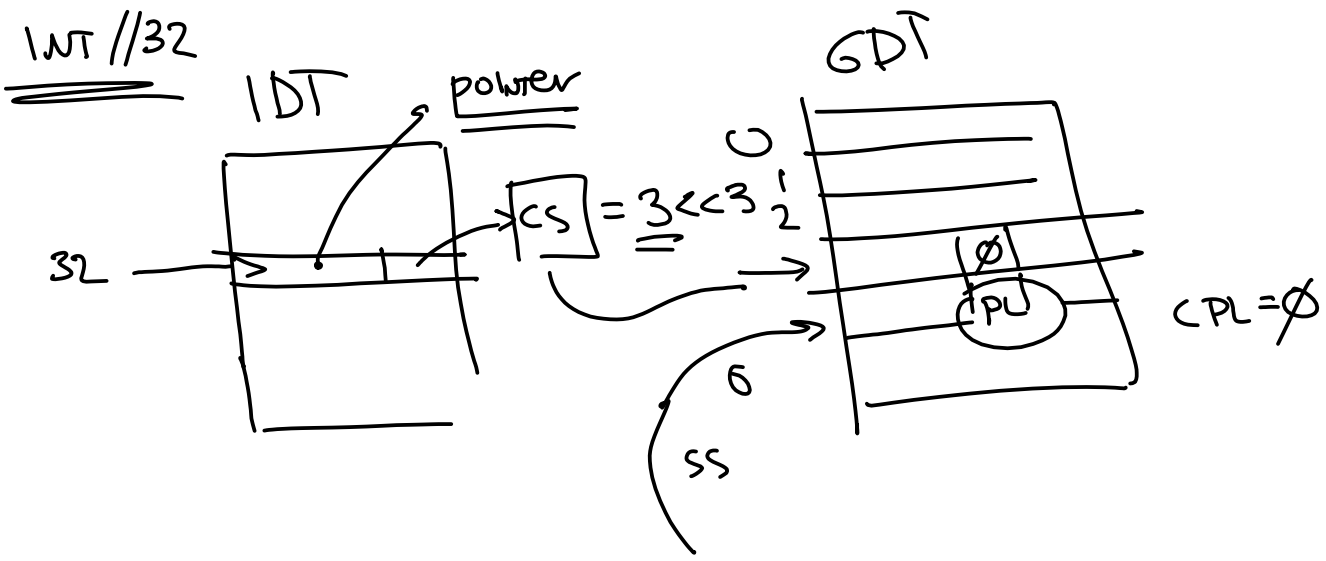
Q2. What is GDT used for?

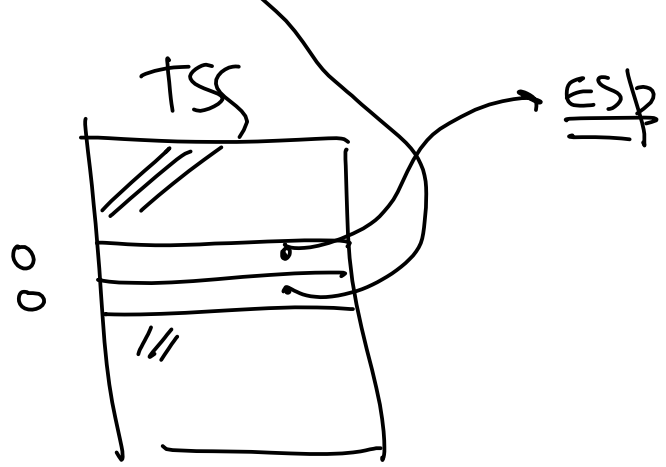

to


P
fork()
&0101..1
Eip →
P1
P
fork()
← EIP
x&0101..2
v →
→ P
P1
mov
v →
→ P
P2
mov

CPL 3

INT //32


IDT
power
GDT
32
CS = 3<<3 | 2
0
PL
CPL=0
6
SS


TSS
esp