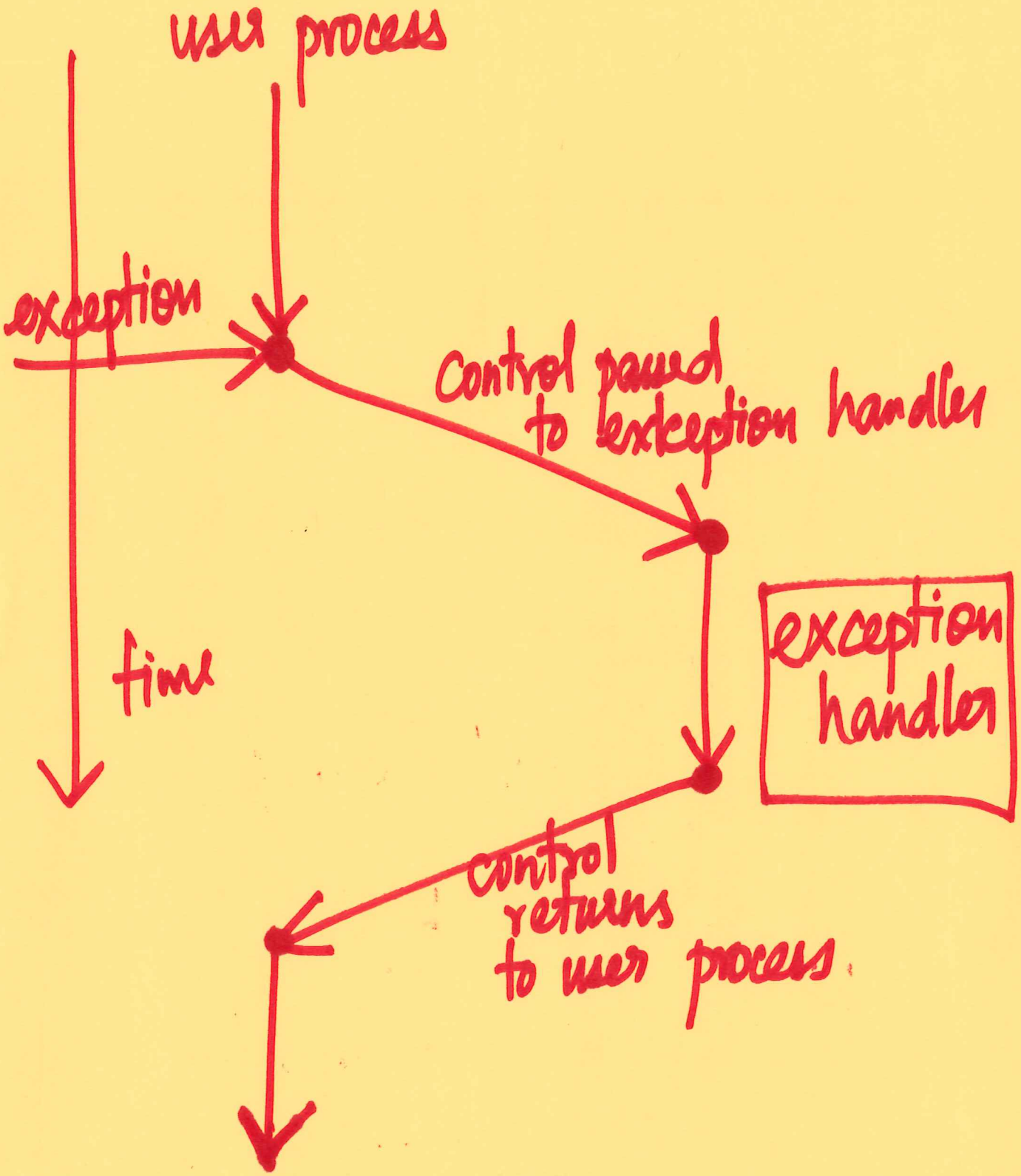
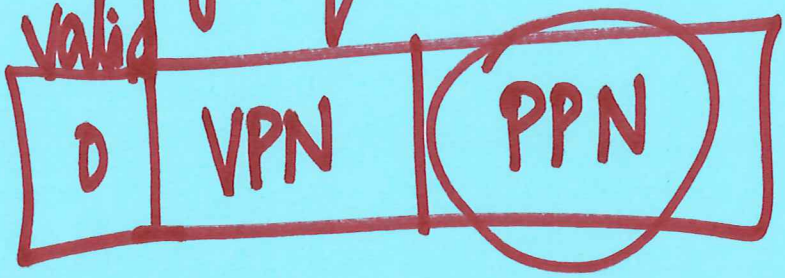
ECF


user process
exception
control passed
to exception handler
time
exception
handler
control
returns
to user process

1. Fault

eg. Page fault.



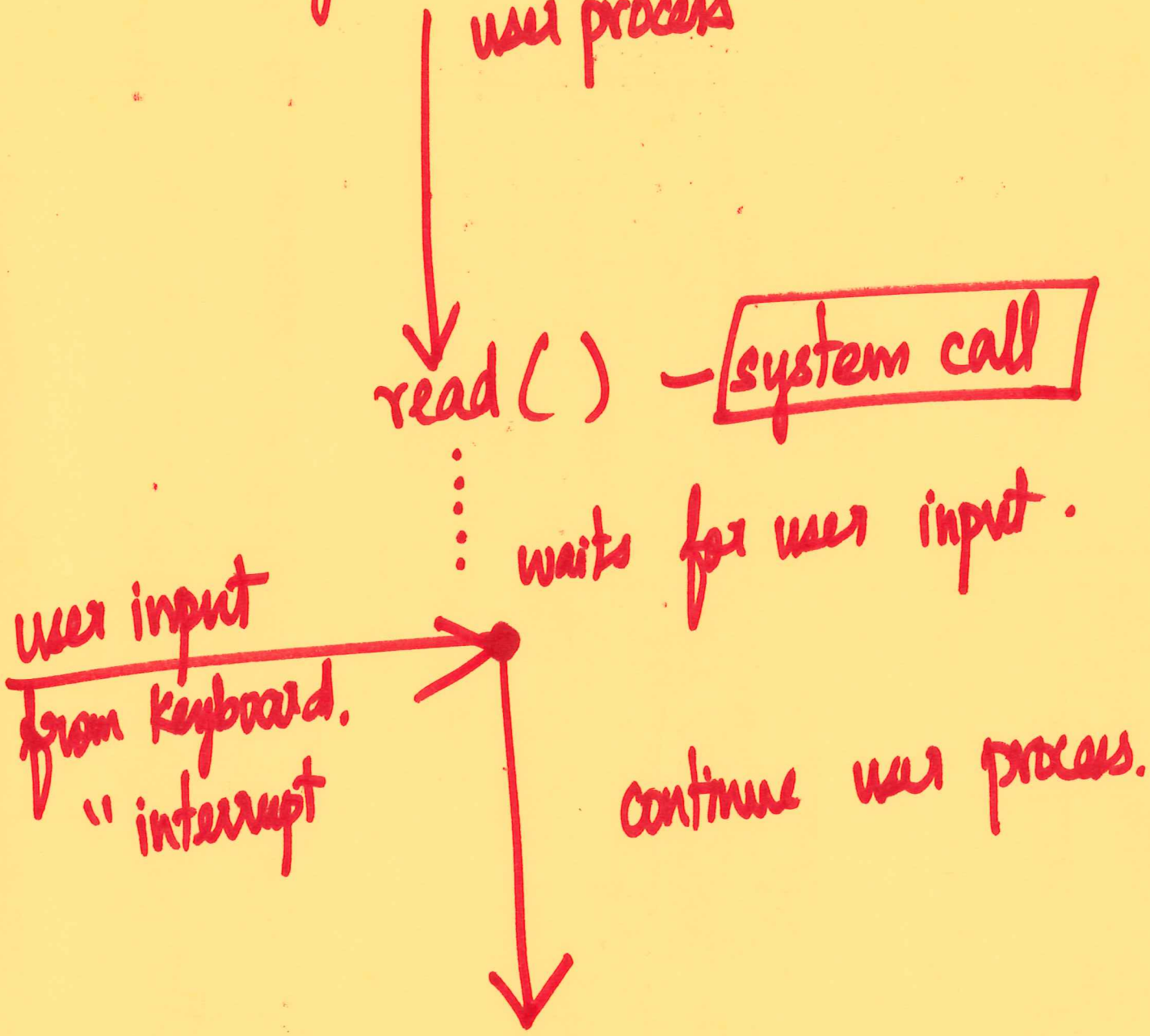
Page fault exception handler.

"Synchronous"

⇒ user process generated the fault.

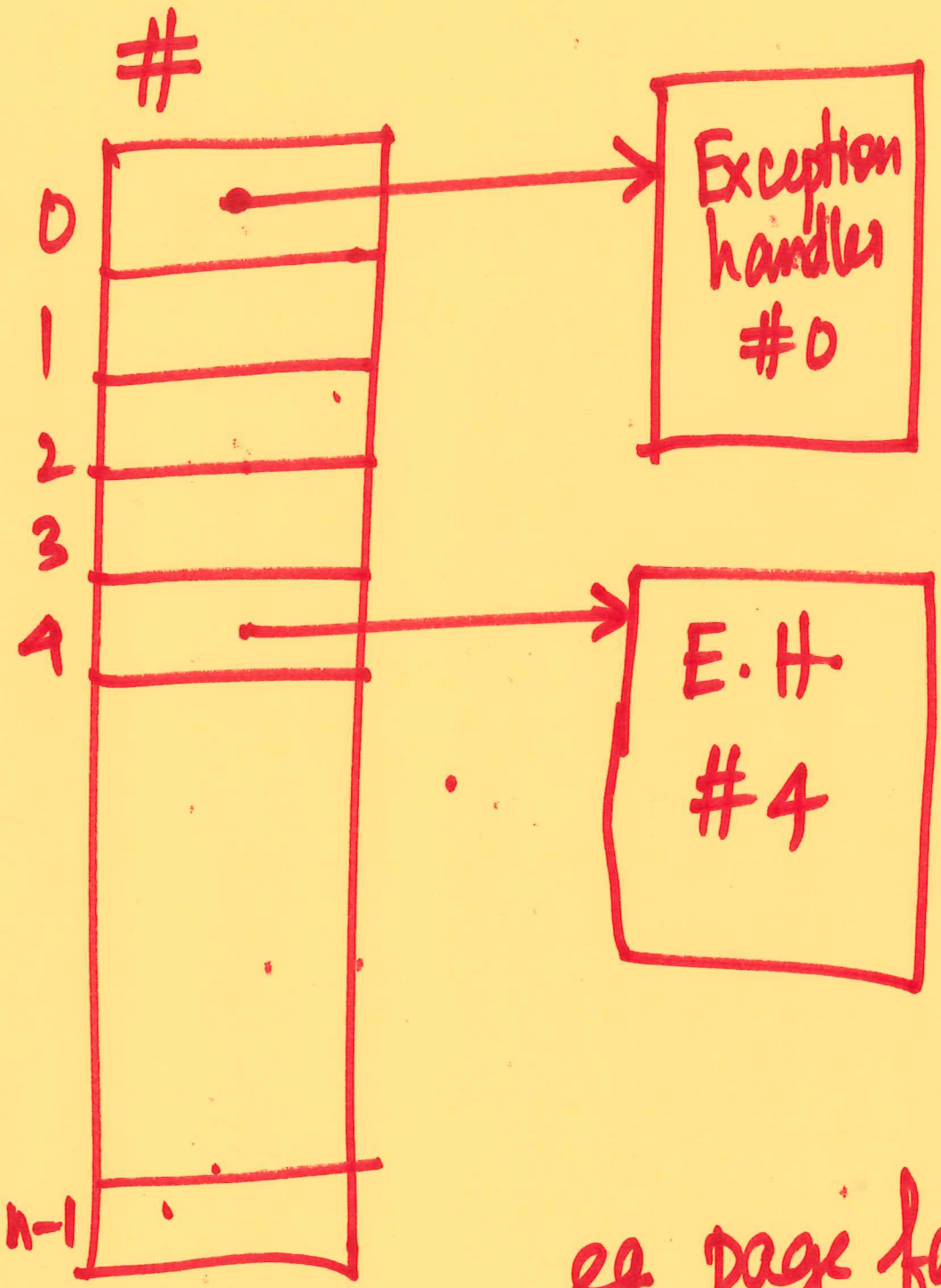
2. Interrupts

"asynchronous"

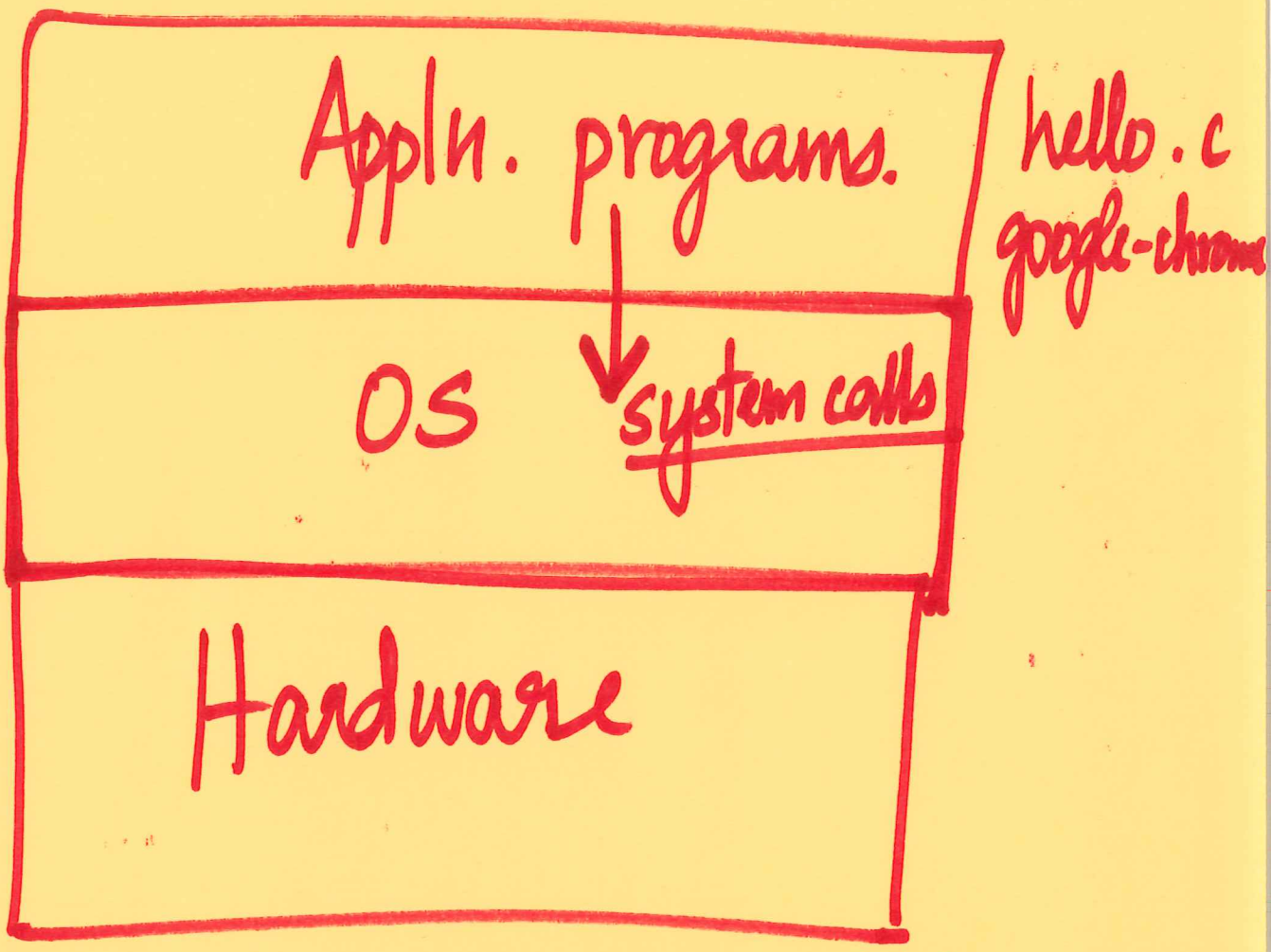


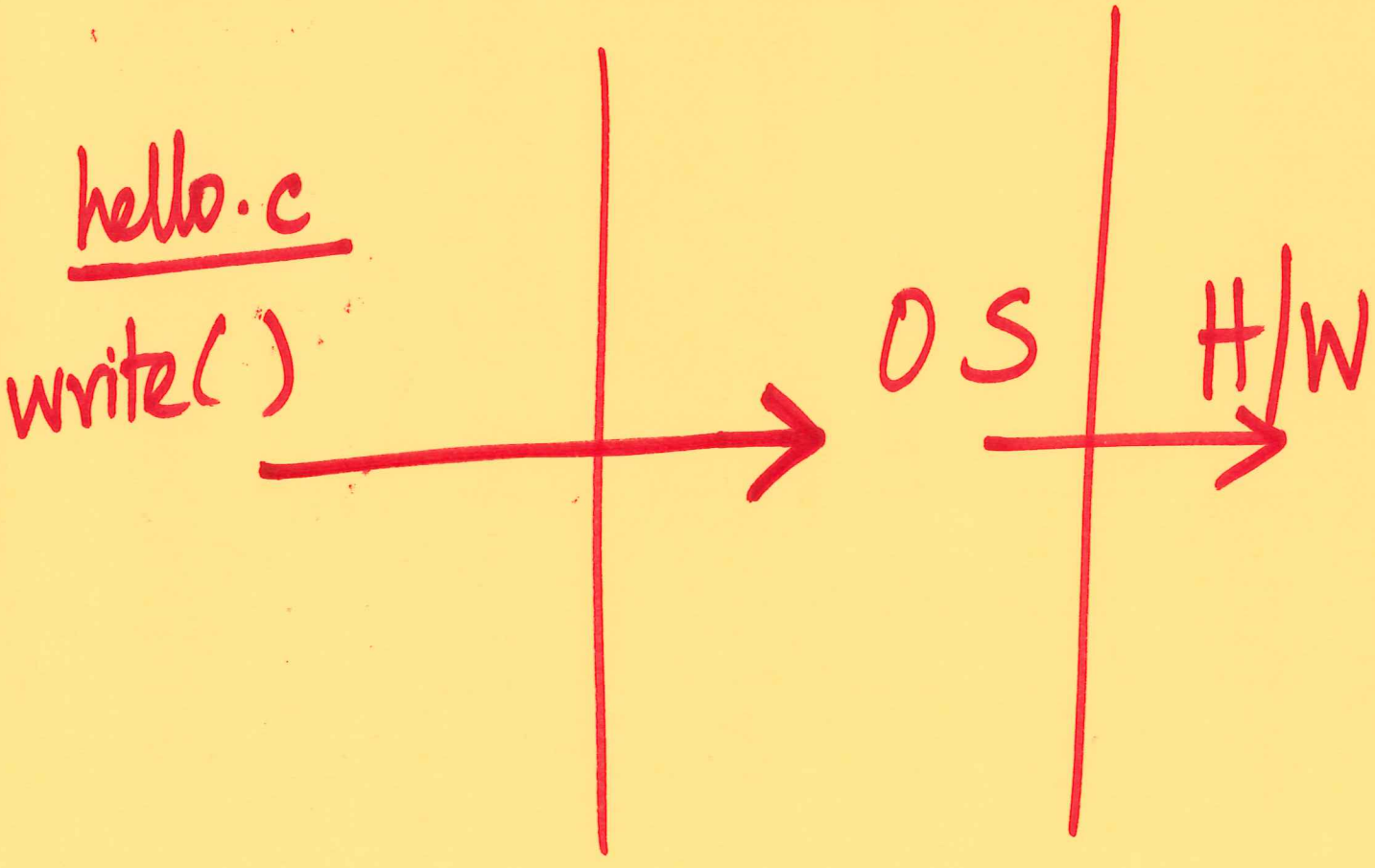
Exception #	Desc.	Type
0	divide by 0.	Fault
13	Protection fault (segfault).	"
14	Page fault	"
128 (0x80)	System Call	Trap.

Exception table



eg page fault exception #14.


Appln. programs.
hello.c
google-chrome
OS
system calls
Hardware


hello.c
write()
OS
H/W

scanf() – function in C library

↓

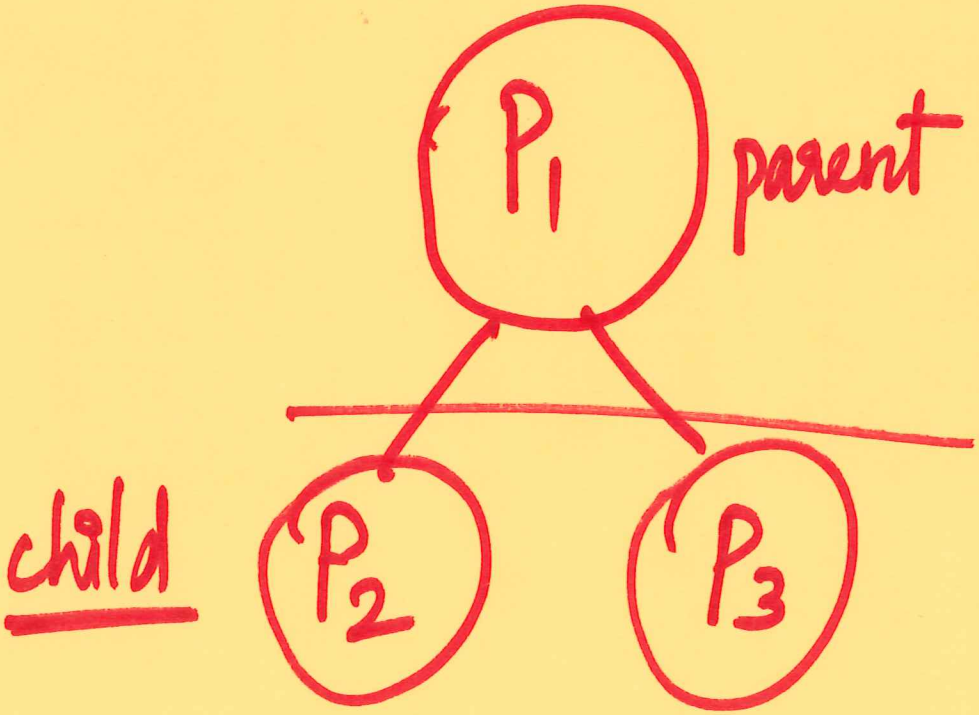
read() – system call in UNIX. OS.

System Calls

1. exit()
2. read()
3. write()
4. brk(), sbrk()
5. open()
6. close()

Syscall #	Name
1	exit
2	fork
3	read
4	write
5	open
6	close

generate / create a new process.

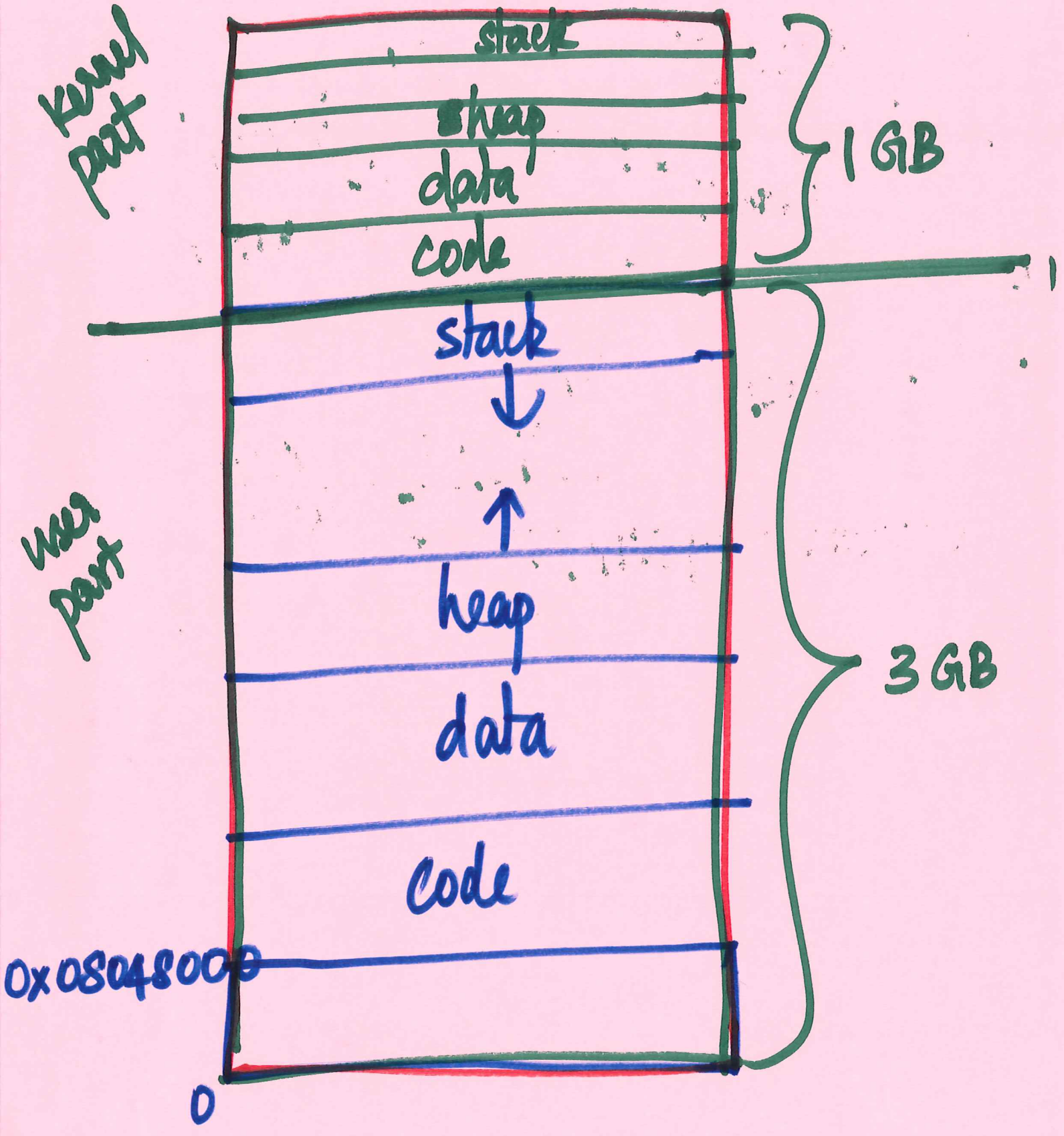


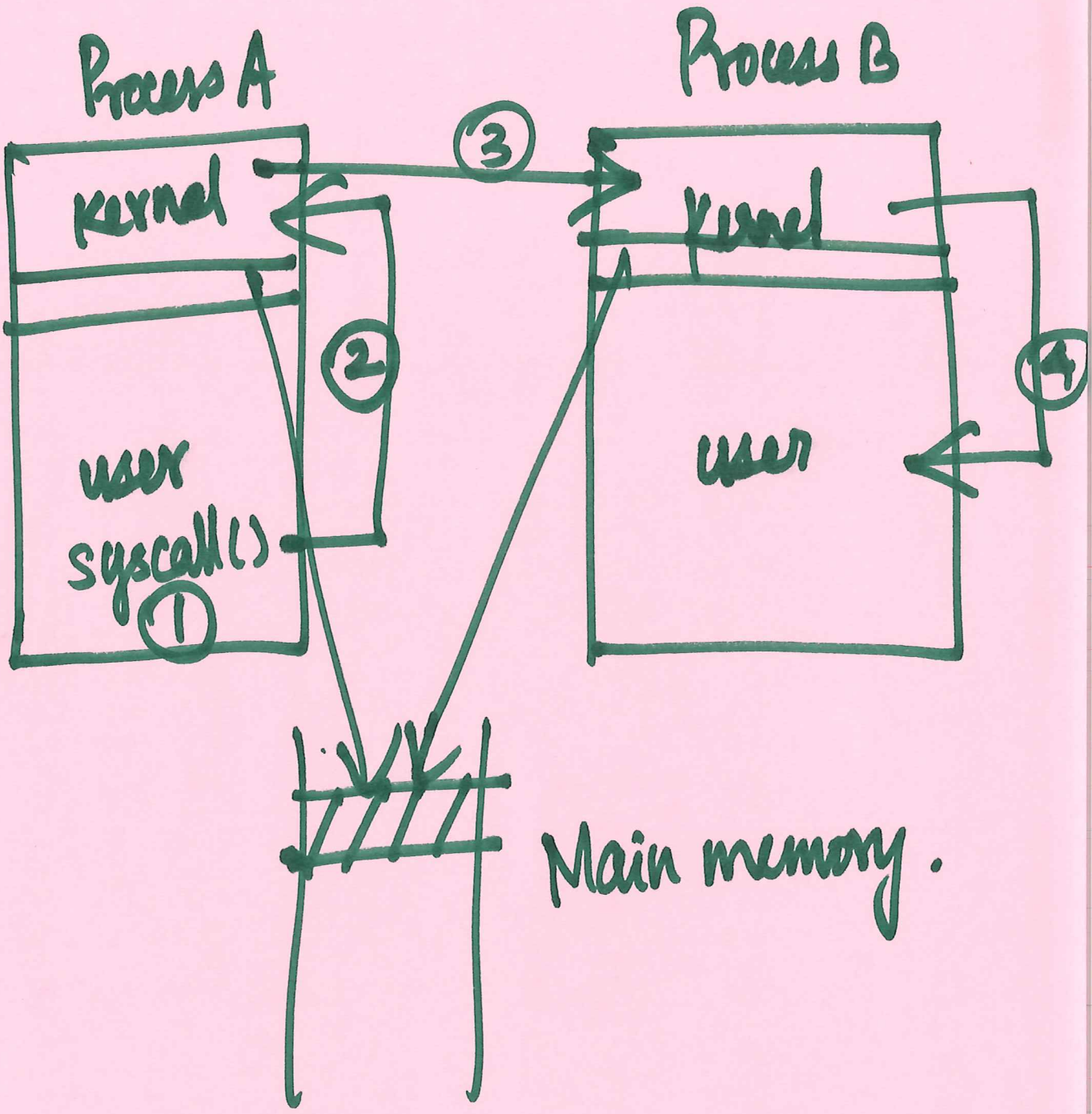
```
int main()
{
    write(1, "hello", 6);
    exit(0);
}
```

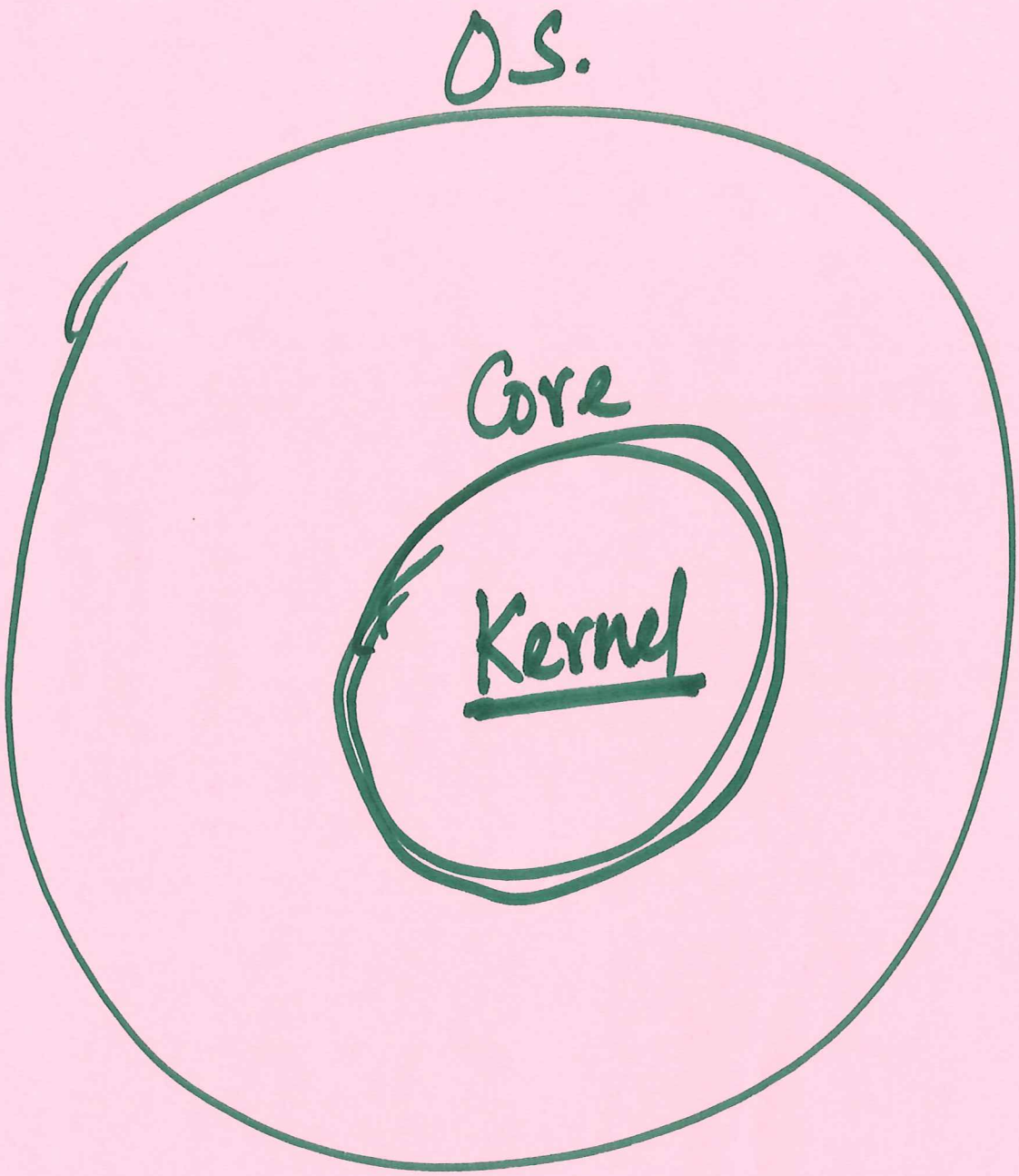
1. syscall # ⟶ %.eax.

2. store the args of the sys call in regs
   - %.ebx
   - %.ecx
   - %.edx
   - %.esi
   - %.edi
   - %.ebp

User mode and Kernel mode




Process A
Process B
kernel
kernel
user
syscall()
user
1
2
3
4
Main memory.


O.S.
Core
Kernel