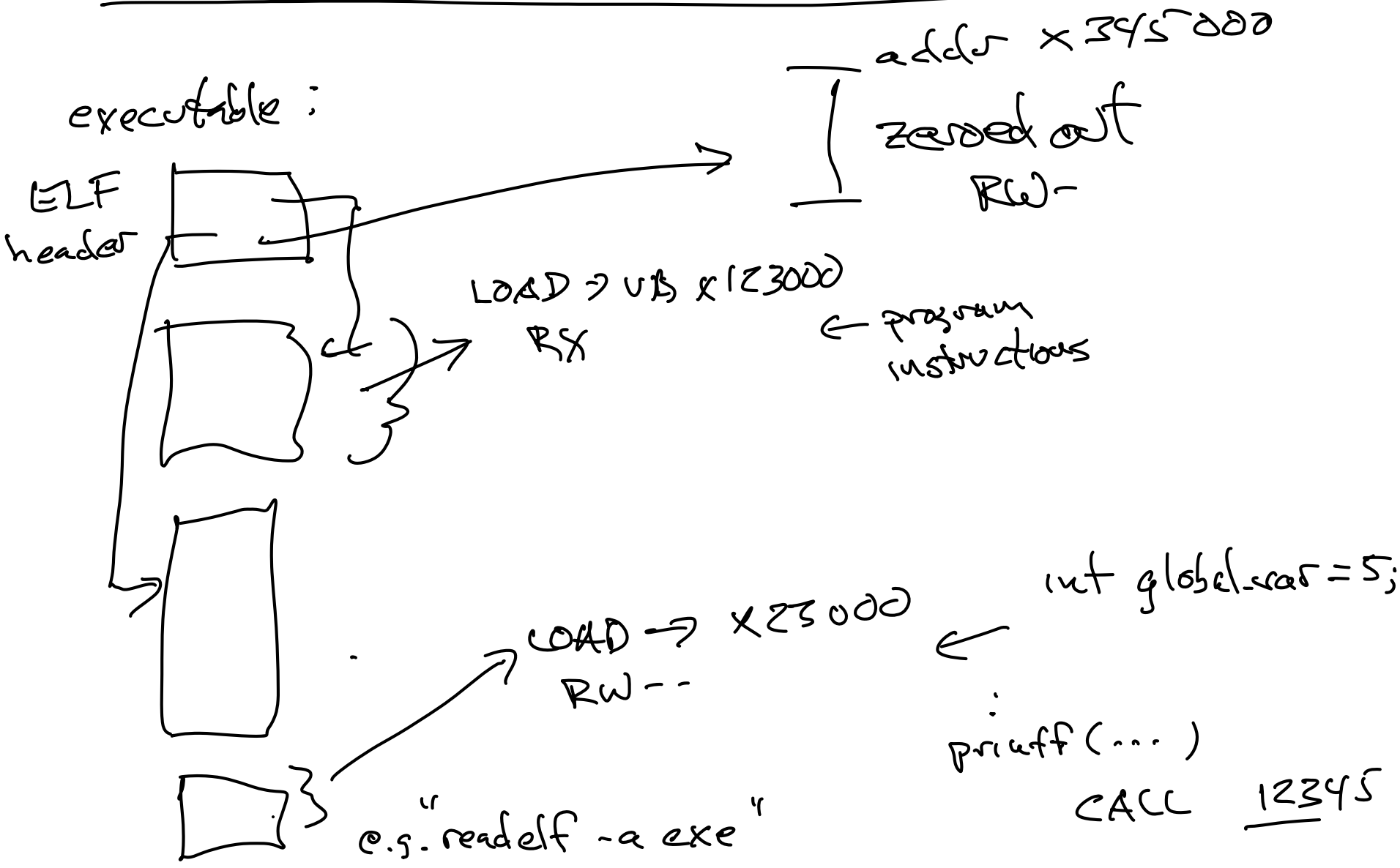
CS 3650 – Computer Systems
Spring 2024
Peter Desnoyers

Lecture 8, Thur Feb 1 2024

Process virtual address space



```
" def fn(a):
     print a
  fn(a)
    :
    "
```

Python example

program representation ≡ program

```
"
void fn(int a){
  printf("%d\n", a);
}"
```

⇒

'%d \n \0'

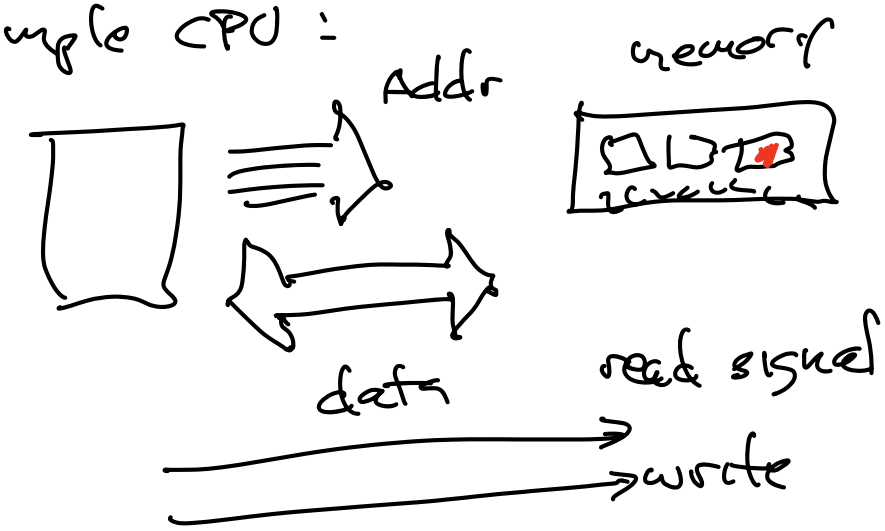
```
x12     PUSH .. (arg a)
x3456   PUSH    (string)
x0012   CALL printf
  :     RET
```

exec : copies sections from file into assigned locations in addr space

What about multiple processes ?

%P

simple CPU :

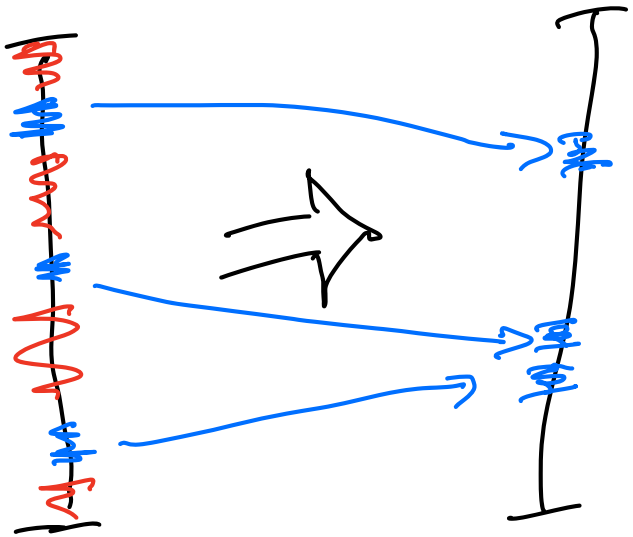


problem:
how to run 2 programs "at once"

LOAD R1 ← x3000

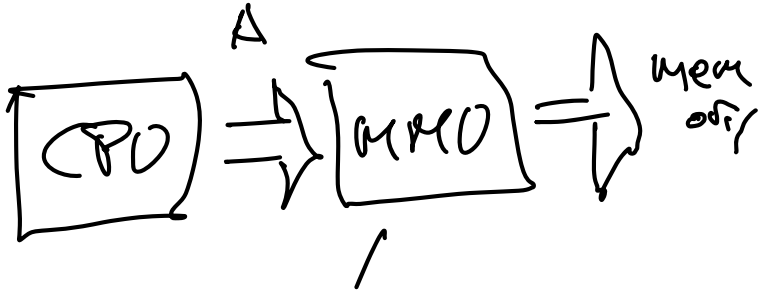
0011 3000 0000 0000

Address translation



Virt addr space

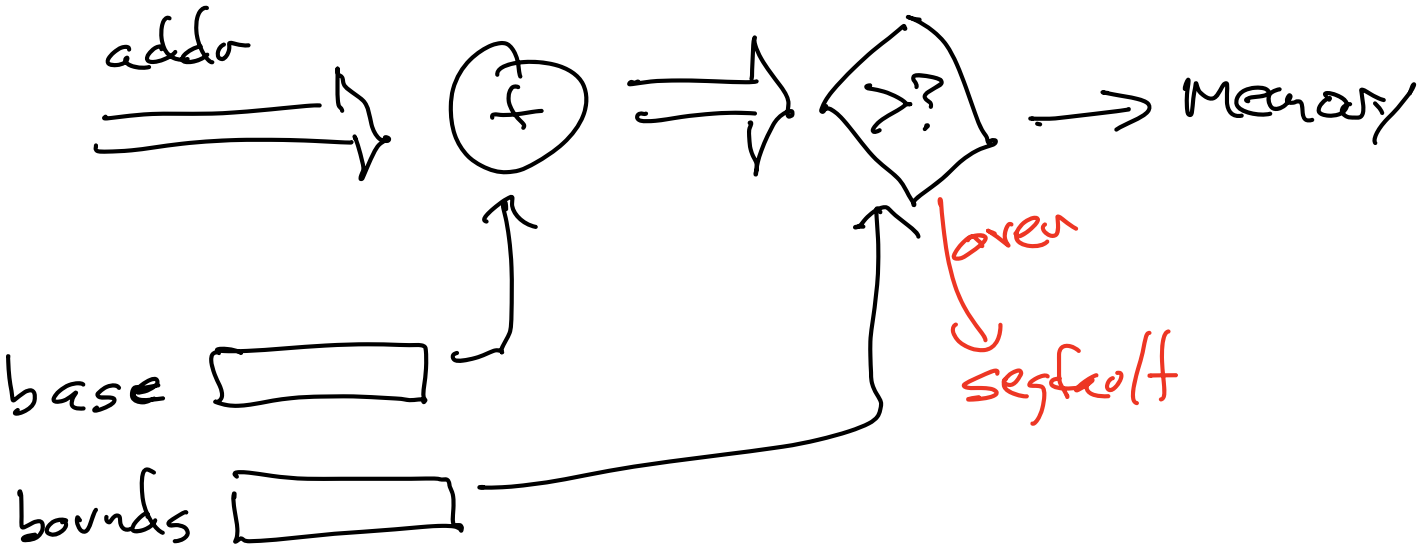
phys. addr space

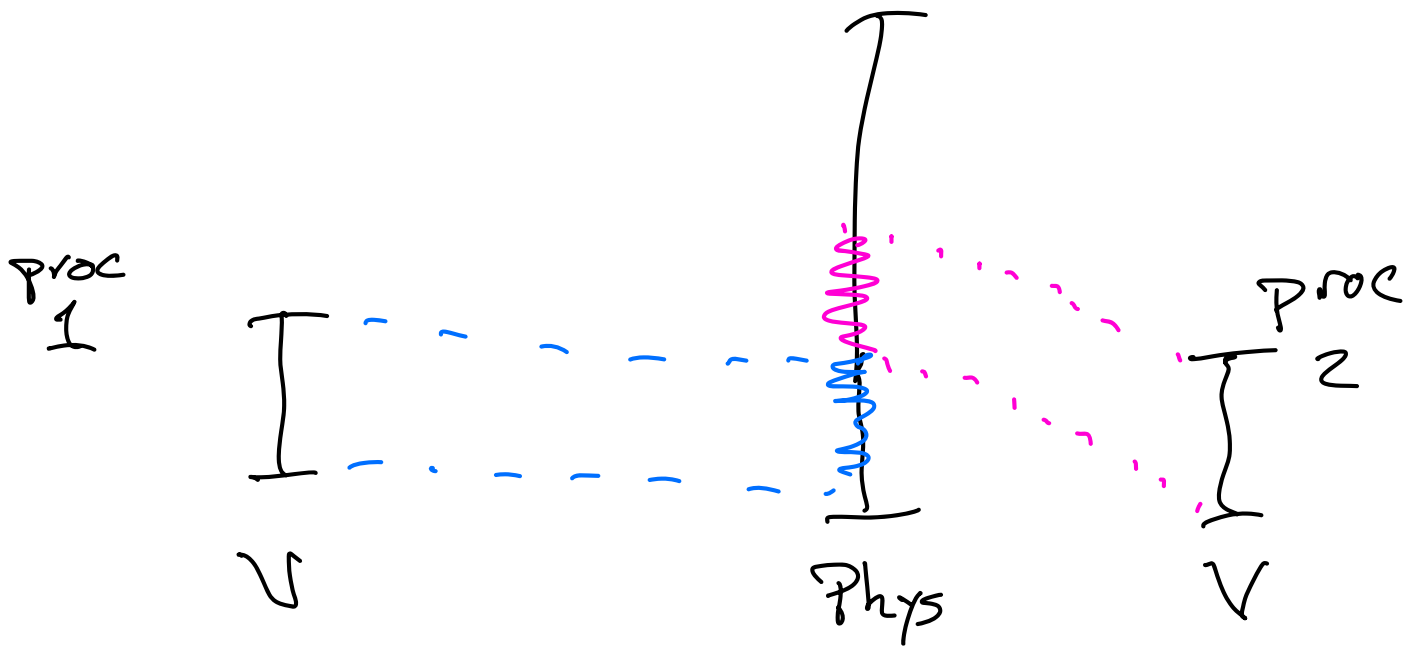


→ all addrs are virtual

Simplest address translation

e.g. PDP-11 early 70s Unix





Privilege levels: user
supervisor

trap, system call instr, INT

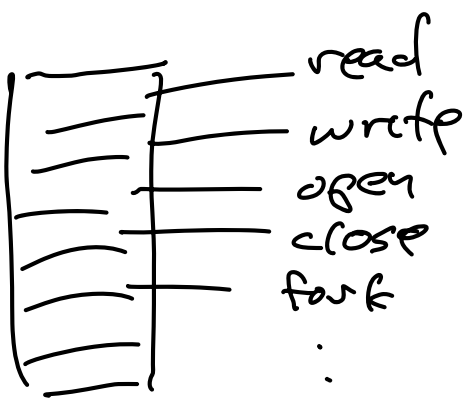
1) change to supervisor mode
2) call specific function
(3 push state on stack)

register:

→ syscall handler

syscall (syscall #, arg, arg, ...)

User (program)



syscall table

→ OS kernel (supervisor mode)

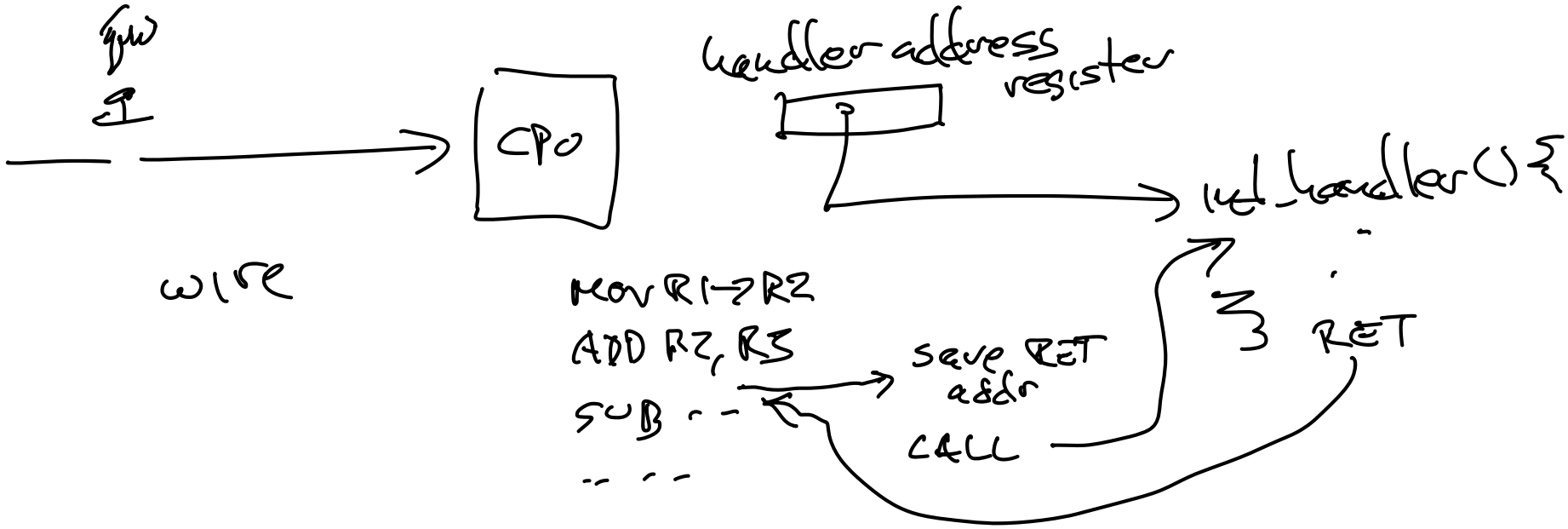
The OS is not a loop

OS is a bunch of functions / exception handlers

→ system calls

→ interrupts ← switch to super mode

[→ page faults]

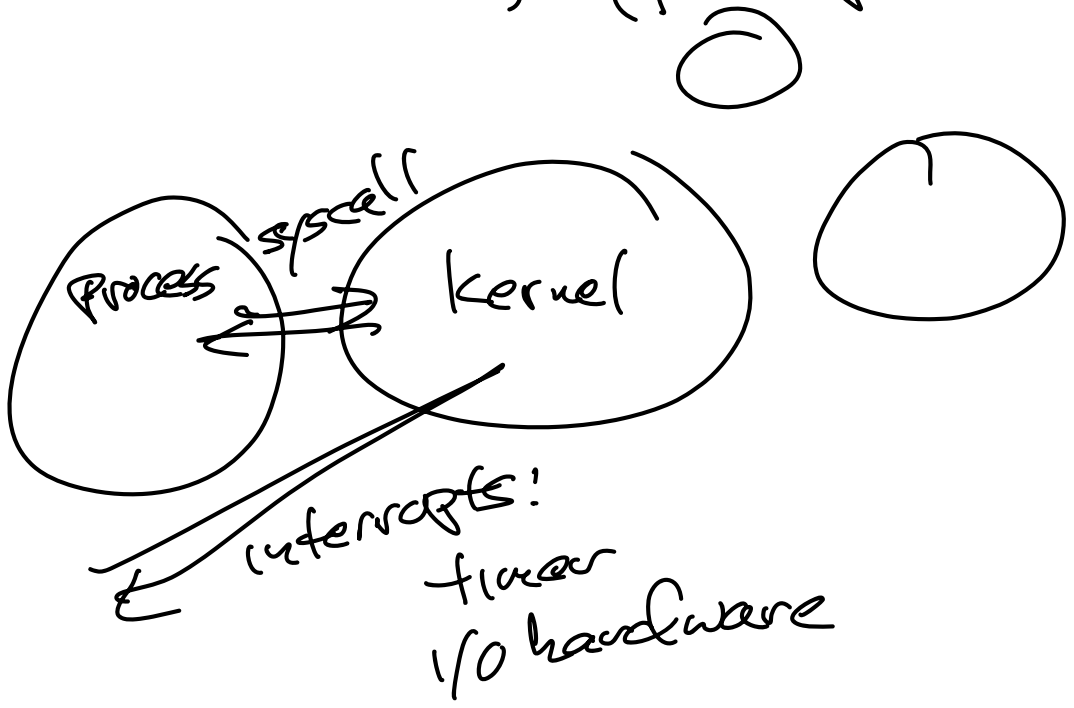


Timer interrupt

config register: timer speed / ..

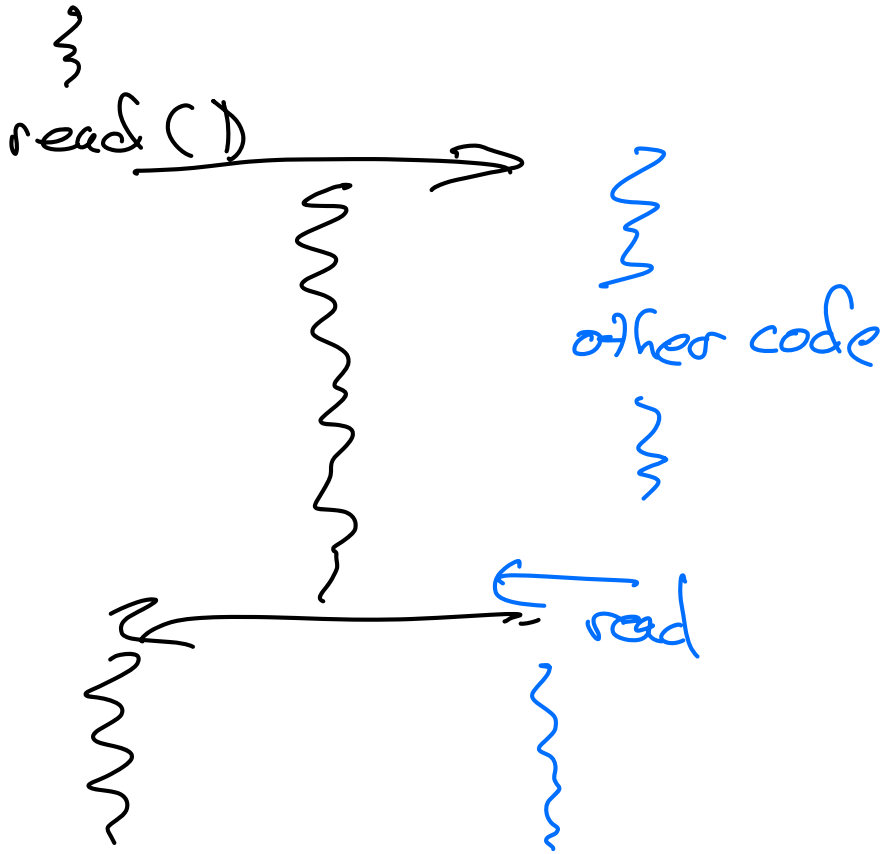
e.g. 1000 times/sec

→ use for timesharing (preemptive multitasking)



Context switch

synchronous:



single-CPU scheduling