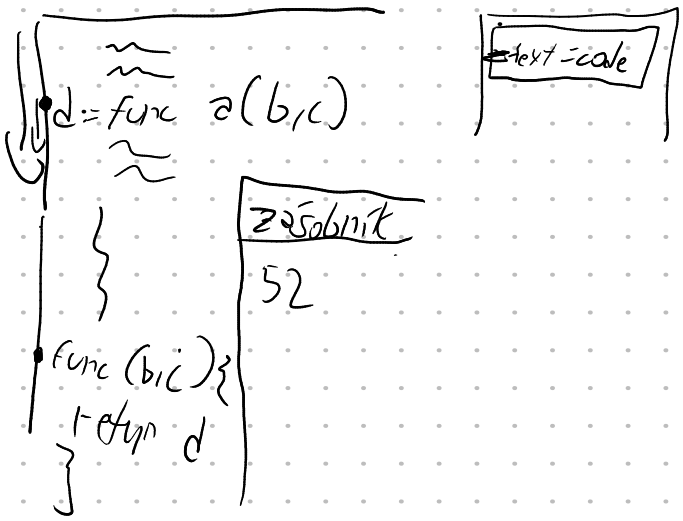
Procesy a vlákna

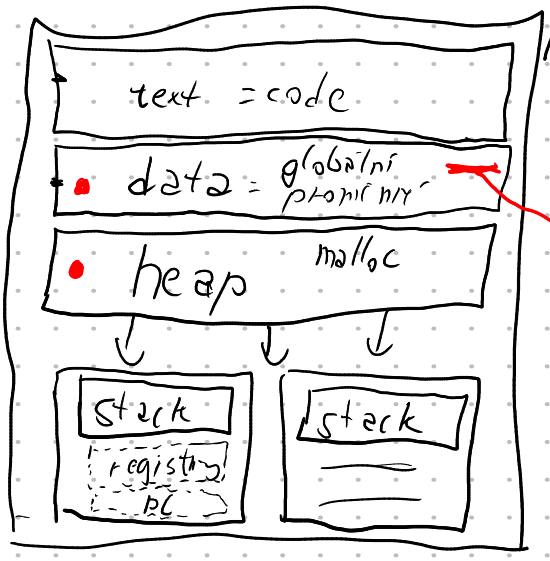
vlákno = posloupnost instrukcí

- PC = program counter
- registry na CPU
- zásobník / stack

proces = instance nějakého programu

PID
- paměťový prostor
- alespoň 1 vlákno





proces

```
int i=0;
int main() {
  printf("Hello world");
}
```

Proč to vadí?

CPU

vlákno 1 (500 → 7000)	vlákno 2 (500 → -500)
x = stav účtu	y = stav účtu
x = x + částka	y = y - výběr
stav účtu = x	stav účtu = y

1. 2. 1. 2.

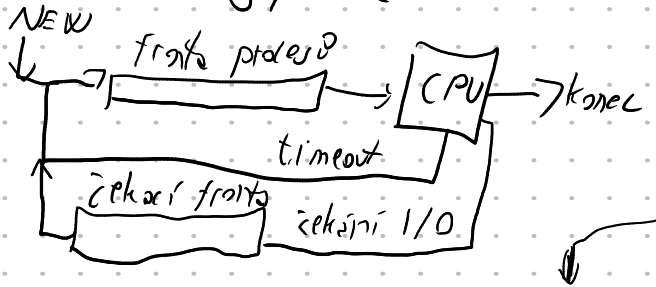
Race condition

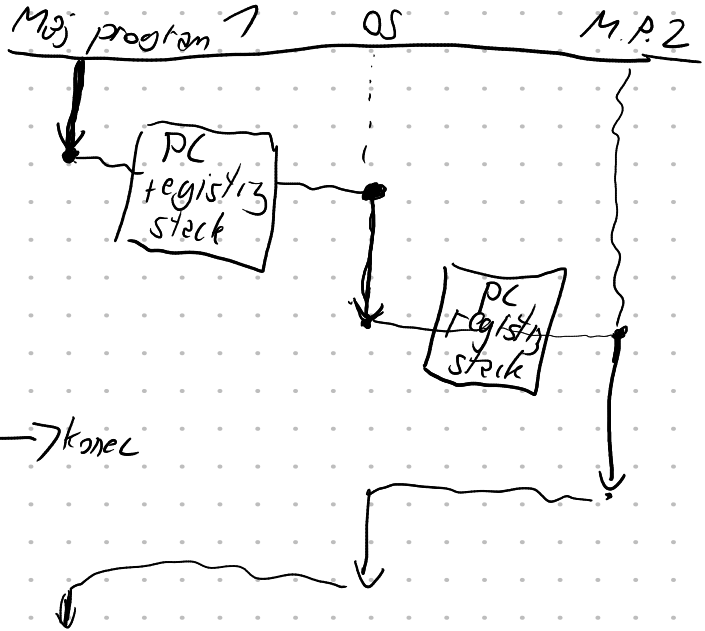
Plánování procesů

CPU - interrupt / přerušení
- timer

Dispatcher = řídí OS plánující kdo kdy poběží

- FCFS
- SJF
- RR





Vlákna v programech

- server ≡ requesty – vlákno per request
- GUI – vlákno pro frontend
  – vlákno pro dlouhotrvající operace
- oddělení dlouhotrvajících operací
- chci využít plného výkonu CPU

---

kritická sekce
– vždy jen jedno vlákno

mutual exclusion
Mutex = zámek

1. pokus

```
bool lock = false
...
while (lock) {}
lock = True
...
lock = False
```

spinlock

2. pokus

```
p1_lock = false
p1_lock = True
turn = 2
while (p2_lock && turn==2) {}
...
p1_lock = false
```

```
p2_lock = False
p2_lock = True
turn = 1
while (p1_lock && turn==1) {}
...
p2_lock = False
```

Petersonův algoritmus

TSL = Test & Set

```
lock = false
while (test_and_set(lock)) { }
[////
lock = false
```

spinlock

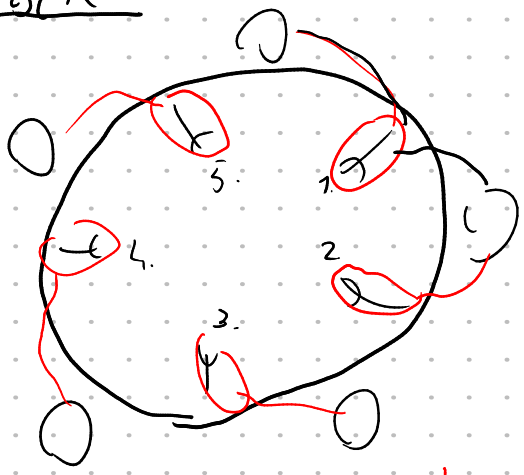
1 instrukce:
1. uloží hodnotu proměnné
2. nastaví True
3. vrátí původní hodnotu

Mutex – zámek
– jedno

Semafor – kapacita 5
– lock → sníží o 1
– unlock → zvýší

Bariéra = funguje naopak
– musí ho zavolat alespoň k vláken než pustí všechny dál

Deadlock



1. vezmu si vidličku vlevo
2. —— " —— vpravo
3. najím se
4. pobírám vidličku vpravo
5. —— " —— vlevo

deadlock | livelock

- Dobrý nápad
- Získávám zámky od nejmenšího