1. review Scheduling

- system call : switch from user process to kernel (process)
- Scheduling : switch from user process A to user process B
- high level intelligence { system call (None)
  scheduling : which process B ?
  ① FIFS ② RR
  # ③ Multi-level feedback Queue
- low level mechanism { hard ware assist. (a little bit)
  Software
  ① kernel mode / user mode
  ② trap / ret-trap instructions

- Scheduler : ISR of TIME Interrupt ? | Sys. call :
  ↓
  Code (almost the most tricky part of xv6)

2. Context Switch

- recall System call procedure :
  ① user process triggers a system. call. (usys.S)
  ② vectors ("int" instruction)
  ③ alltrap.S

④ trap.c

⑤ syscall.c

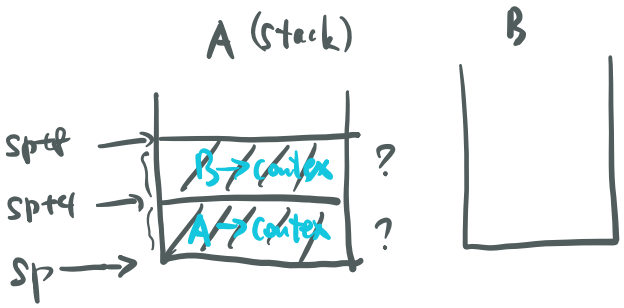
- swtch.S

```
swtch (struct context **A,
       struct context *B)
```

(one of A, B is the scheduler)

another is user process

① start



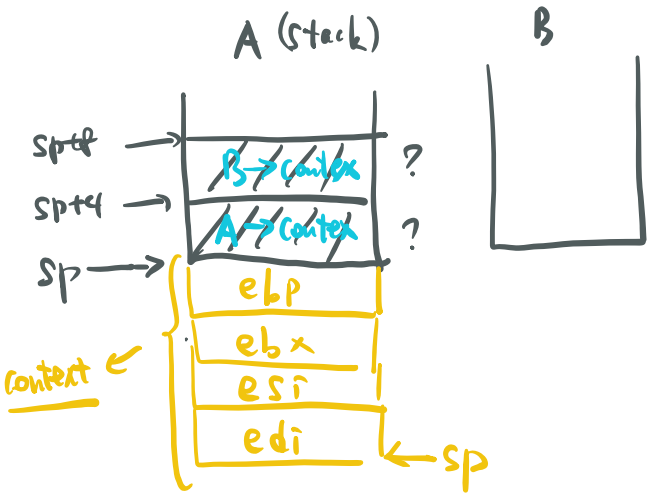
① current process : A.

② eax (register)
edx

eax = sp+4 = A→context

edx = sp+8 = B→context
?

⑤. push (save context of A on its stack)



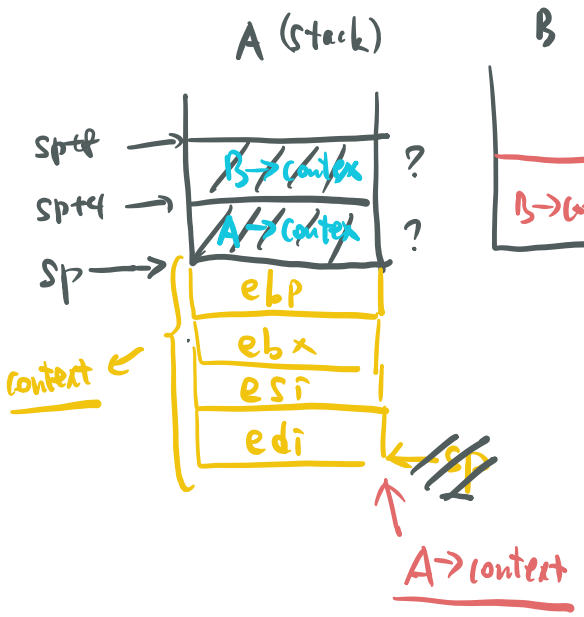
① Current process : A

② What sat on the top of A's stack?

↙ a struct context

```
struct context {
  uint edi;
  uint esi;
  uint ebx;
  uint ebp;
  uint eip
}
```

③ Switch stack



① `mov %esp, (%eax)`

Store "struct context" in (%eax)

<=> A->context = esp

② `mov %edx, %esp`

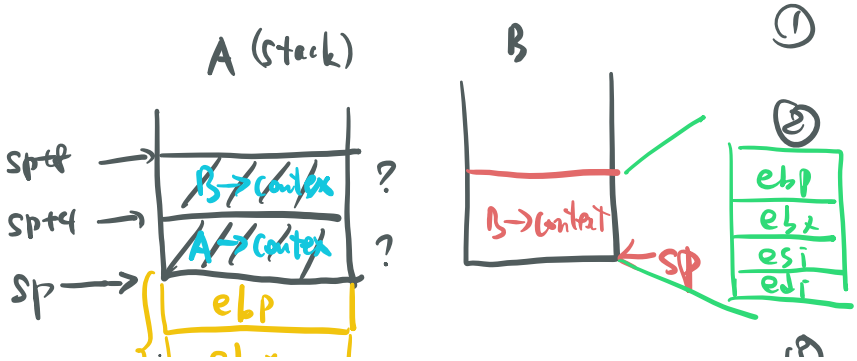
<=> sp = B->context

B->context : the top of process B's stack

③ Current running process:

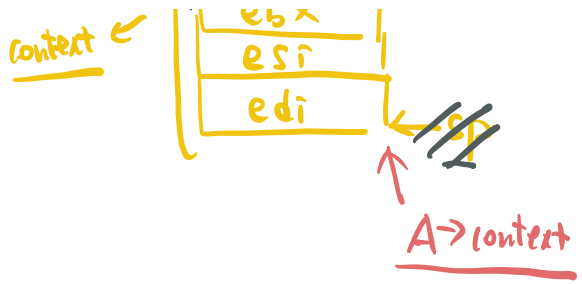
B!!!

④ pop



① current running proc: B

② pop : restore B's context to CPU



① run proc. is from the last time it is been scheduled

↑ Hard. Don't be frustrated.

Don't be panic

Let's move on!