TA. YeouGyu Jeong
(81887821@snu.ac.kr)
System Software & Architecture Lab.
Seoul National University

Spring 2020

4190.307: Operating Systems Lab. 1

What is system call?

- User space applications run with restricted privilege
- They have to request some operations to the OS
- System call is an exception that intentionally made by application for this purpose

3 modes of RISC-V

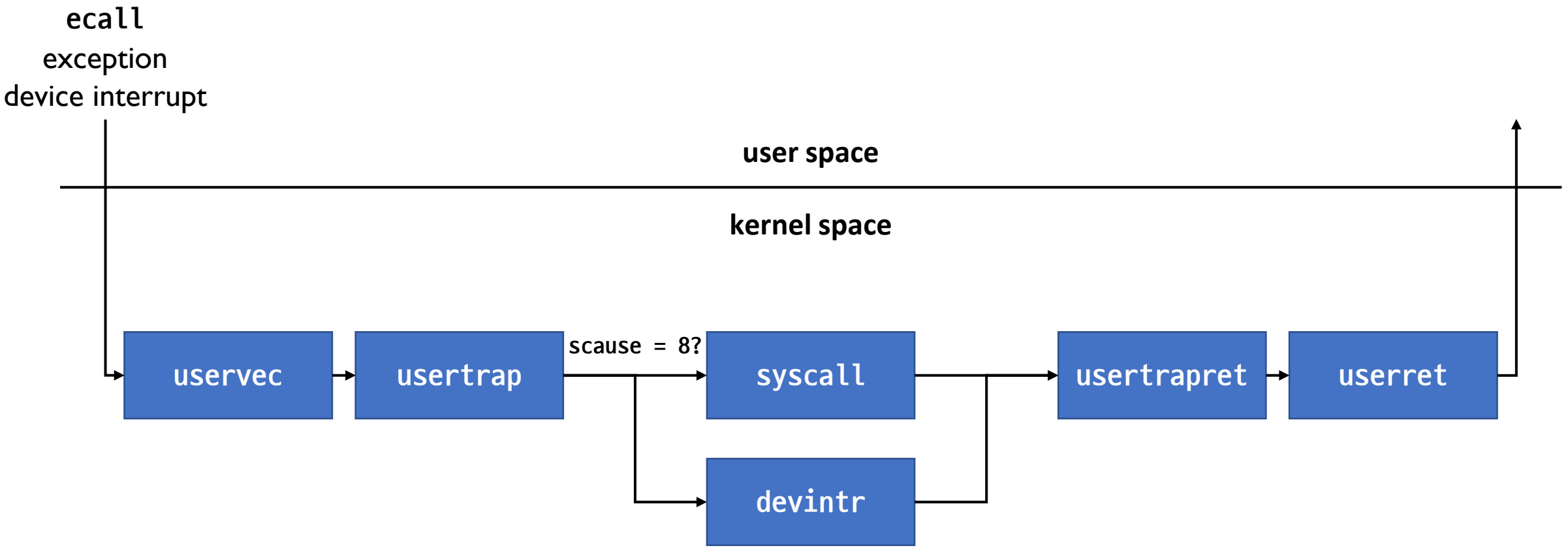
- Machine mode
  - Have full privilege
  - CPU starts in machine mode
- Supervisor mode
  - Allowed to execute privileged instructions
    - Enabling and disabling interrupts
    - Reading and writing the register that holds address of page table
    - …
  - The kernel runs in supervisor mode
- User mode
  - User space applications runs in user mode

ecall instruction

- In RISC-V an application can use “ecall” instruction to invoke system call
- e. g.) Fork function in userspace

```
fork:
    li a7, SYS_fork
    ecall
    ret
```

Traps from user space



Special RISC-V registers

- `stvec`: The address of trap handler
- `sepc`: Register to save program counter when a trap occurs
- `scause`: The reason of a trap
- `sscratch`: The address of trapframe

- `satp`: Current page table

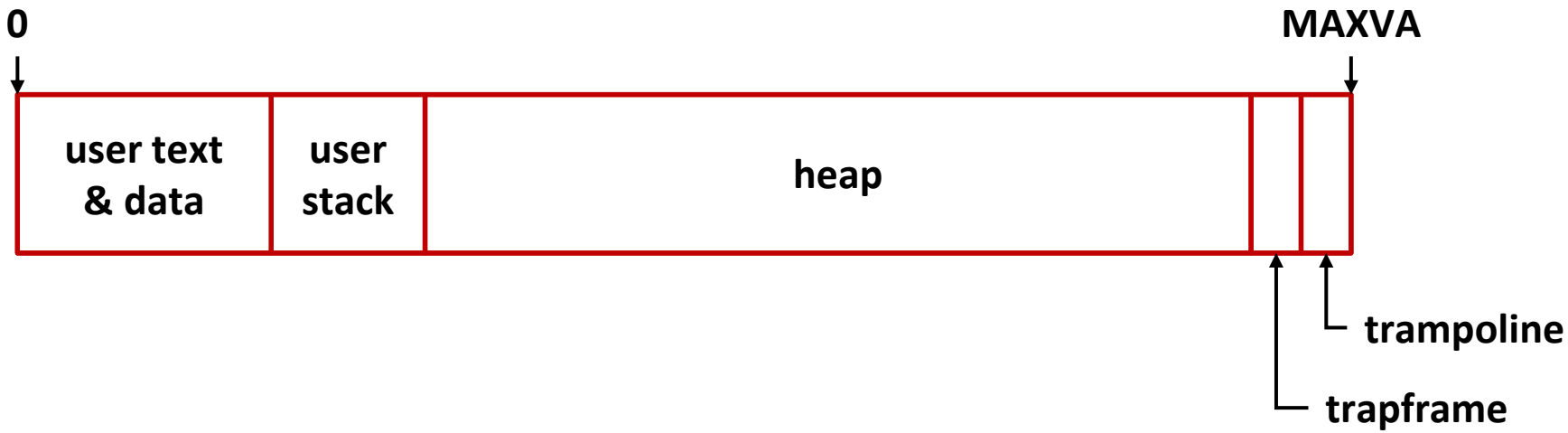
On trap…

- CPU runs trap handler saved in `stvec` register
- Program counter is saved to `sepc` register
- The reason of the trap is saved to `scause` register

uservec

- Saves all register values to `trapframe`
- Change `satp`(page table register) to kernel page table
- `uservec` is located on trampoline page
  because changing page table is needed

Virtual Address Space Layout



usertrap

- `usertrap` determines the cause of the trap, and handles it
- Set `stvec` to `kernelvec`
- Call `syscall` if the trap was caused by system call,
  `devintr` if the trap was caused by device interrupt

`syscall`

- Gets system call number from a7 register saved in trapframe
- Calls system call handler
  - e.g.) `sys_fork` function in kernel space
- Saves return value to a0 register in trapframe

usertrapret

- Saves kernel page table, kernel stack to trapframe
  (To be used for next trap from user space)
- Restore `stvec` to refer uservec
- Restore `sepc` to previously saved user program counter

userret

- Switches `satp` to process's user page table
- Restores all register values from trapframe
- `userret` is also located on trampoline page because changing page table is needed

Project #2 – System call

- Your job is to implement process group and two system calls
- Why process group is needed?
  - e.g.) `tar -c file1 file2 ¦ xz --threads=4 > files.tar.xz`
  - If you want to abort the job, two processes(tar and xz) must be interrupted together
- You have to implement:
  - `setpgid` to set process group ID
  - `getpgid` to get process group ID

`setpgid` system call

- `int setpgid(int pid, int pgid);`
- `setpgid` should change the process group ID of process specified by **`pid`** to **`pgid`**
- **`pid`** and **`pgid`** are non-negative integers
- If **`pid`** is 0, it should change process group ID of the calling process(process that invoked the system call)
- If **`pgid`** is 0, it should change process group ID same as pid

- On success, return 0
- On failure, return -1

`getpgid` system call

- `int getpgid(int pid);`
- `getpgid` should return process group ID of process specified by **`pid`**
- If **`pid`** is 0, it should return process group ID of the calling process

- On success, return **`pgid`**
- On failure, return **`-1`**

Displaying process list

- xv6 prints list of processes when Ctrl+P is pressed
- You have to modify this to print pgid
  - format: `pid pgid state process_name`

```
xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$
1 1 sleep  init
2 1 sleep  sh
```

You may want to see…

- defs.h
  - For function definitions
- proc.h, proc.c
  - For process related functions
- console.c
  - For console input handling
- syscall.c, sysproc.c
  - For system call implementation

When you do your project,

- Please only modify Makefile and files in `kernel` directory
  - Please fill your student id to STUDENTID variable in Makefile
  - Modifications to user directory will be ignored by grading script
- Please remove all the debugging outputs before you submit

- Please read the project description carefully
  - https://github.com/snu-csl/os-pa2
- Skeleton code is on xv6-riscv-snu repository's pa2 branch
  - git clone https://github.com/snu-csl/xv6-riscv-snu
  - git checkout pa2
- Archive your source to tarball and submit it to sys server
  - Type "make submit" to archive your source
  - And upload it to https://sys.snu.ac.kr

Thank you!

- If you have any questions, feel free to ask us in KakaoTalk