Week 4.b
CS6640
09/28 2023
https://naizhengtan.github.io/23fall/

1. Last time: RISC-V timer interrupt
2. Review: OS scheduling
3. Processes in egos-2k+
4. Kernel scheduler

Last time: RISC-V CPUs.

① register handler (mtvec)

② enable timer interrupt (mstatus, mie)

③ set timer (mtime, mtimecmp QUANTUM)

8B 8B

64bit

mtime > mtimecmp

↳ overflow, 0xfff...fff → 0x0

1. Machine-mode exception CSRs

a) **mstatus**
Machine Status, holds the global interrupt enable, along with a plethora of other state.

b) **mie**
Machine Interrupt Enable, lists which interrupts the processor can take and which it must ignore

c) **mcause**
Machine Exception Cause, indicates which exception occurred

d) **mtvec**
Machine Trap Vector, holds the address the processor jumps to when an exception occurs

e) **mepc**
Machine Exception PC, points to the instruction where the exception occurred

f) **mtval**
Machine Trap Value, holds additional trap information: the faulting address for address exceptions, the instruction itself for illegal instruction exceptions, and zero for other exceptions

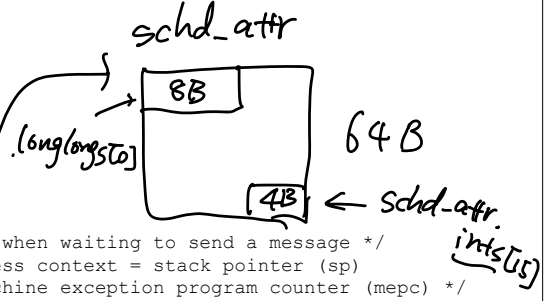
g) **mip**
Machine Interrupt Pending, lists the interrupts currently pending

2. egos-2k+ process management

a) process control block (PCB)

[grass/process.h]

```
struct process {
    int pid;
    int status;
    int receiver_pid; /* used when waiting to send a message */
    void *sp, *mepc;  /* process context = stack pointer (sp)
                       * + machine exception program counter (mepc) */
    // scheduling attributes
    union {
        unsigned char      chars[64];
        unsigned int       ints[16];
        float              floats[16];
        unsigned long long longlongs[8];
        double             doubles[8];
    } schd_attr;
};
```



b) global process data structures

[grass/kernel.c]

```
int proc_curr_idx;
struct process proc_set[MAX_NPROCESS];
```

[grass/process.h]

```
#define curr_pid      proc_set[proc_curr_idx].pid
#define curr_status   proc_set[proc_curr_idx].status
```

c) process life cycles

[grass/scheduler.c]

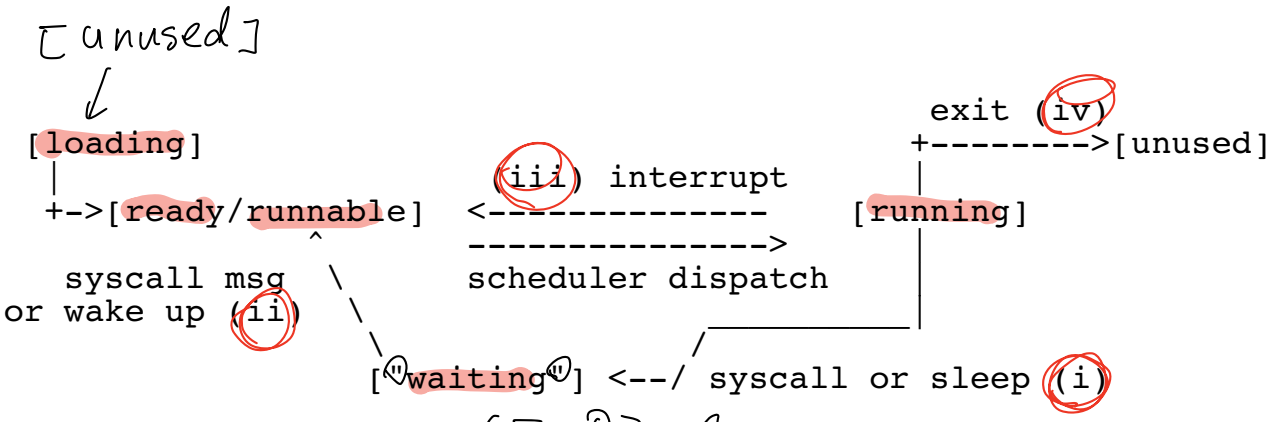
life-cycle functions:

* **proc_on_arrive**(int pid): when creating pid
* **proc_yield**(): when deciding next running process
* **proc_on_stop**(int pid): when destroying pid

a process's life cycle:

```
proc_on_arrive() ->
  proc_yield() -> [other proc] -> [ctx_switch to this proc] ->
  proc_yield() -> [other proc] -> [ctx_switch to this proc] ->
  ...
-> proc_on_stop()
```

turnaround:

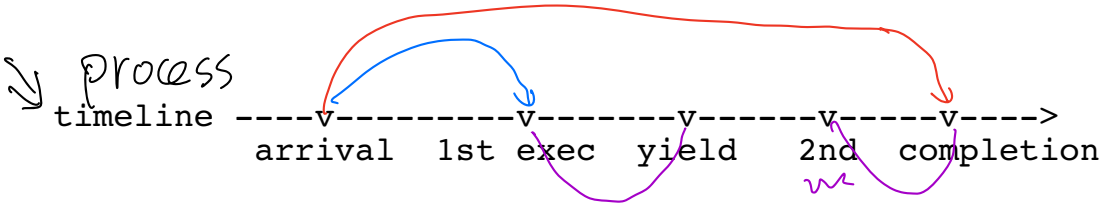


- SLEEPING
- WAIT_TO_SENT/RECV

preemptive: ①-④

non-preemptive: ①, ④

- metrics.



- turnaround time;
- response time
- CPU time;
- #scheduled time (init): 2