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;
    struct syscall syscall;
    enum proc_status status;
    uint mepc, saved_registers[32];

    // 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]

```
uint core_in_kernel;
uint core_to_proc_idx[NCORES];
struct process proc_set[MAX_NPROCESS + 1];
/* proc_set[0] is a place holder for idle cores. */
```

[grass/process.h]

```
#define curr_proc_idx (core_to_proc_idx[core_in_kernel])
#define curr_pid      (proc_set[curr_proc_idx].pid)
#define curr_status   (proc_set[curr_proc_idx].status)
#define curr_saved    (proc_set[curr_proc_idx].saved_registers)
```

c) process life cycle

[grass/scheduler.c]

state-transition callback functions:

* **proc_on_arrive**(int pid): when pid is created

* **proc_on_sched_in**(int pid) when pid is scheduled to run

* **proc_on_sched_out**(int pid) when pid is descheduled

* **proc_on_stop**(int pid): when pid is destroyed

a process's life cycle:

```
proc_on_arrive() ->
  proc_on_sched_in() -> [running] -> proc_on_sched_out() -> [others] ->
  proc_on_sched_in() -> [running] -> proc_on_sched_out() -> [others] ->
  ...
-> proce_on_stop()
```