Operating Systems

CMPT 424 • Fall 2024

– *i*Project Three - 100 points

Goals

To build on the functionality of *i*Project Two (all of which is required) by adding the ability to execute **multiple** user programs at the same time.

Functional Requirements

- □ Allow the user to load three programs into memory at once. [5 points]
- □ Add the following shell commands: [8 points]
  - clearmem — clear all memory segments
  - runall — execute all programs at once
  - ps — display the PID and state of all processes
  - kill <pid> — kill one process
  - killall — kill all process
  - quantum <int> — let the user set the Round Robin quantum (measured in cpu cycles)
- □ Display the Process queue and its contents (including state, location, base, limit, segment, priority, and current quantum) in real time. [5 points]
- □ [challenge] Track and display turnaround time & wait time for each process. [+10 points]

Implementation Requirements

- □ Store multiple programs in memory, each in their own partition / segment, allocated by the client OS (which obviously needs to keep track of available and used partitions/segments in the MMU). [5 points]
- □ Add base and limit registers to your core memory access code in the **host** OS and to your PCB objects in the **client** OS. [5 points]
- □ Enforce memory partition boundaries at all times. [5 points]
- □ Create a Resident list for the loaded processes and a Ready queue for the running processes. These can be combined if you label it carefully. [5 points]
- □ Instantiate a PCB for each loaded program and put it in the queue. [5 points]
- □ Develop a CPU scheduler and dispatcher in the client OS using Round Robin scheduling with the user-specified quantum measured in CPU cycles (default = 6). Be sure that your client OS controls the host CPU with the CPU scheduler and dispatcher in the Kernel. Log all scheduling events. [50 points]
- □ Implement context switches in the dispatcher with software interrupts generated by the scheduler. Only allow context switches after fully completed CPU instruction cycles. Be sure to update the mode bit (if appropriate), the PCBs, and the Process queue. [5 points]
- □ Detect and gracefully handle errors like invalid op codes, missing operands (if you can detect that), and most importantly, memory out of bounds or access violation attempts. [2 points]
- □ Your code must *separate structure from presentation*, be professionally formatted, use and demonstrate best practices, and and be free of compiler errors. If there are compiler errors it's -10 * project number. [−∞ if not]
- □ Continue to cite everything that's not entirely your own original work.
- □ Commit to Git early and often. I want to see many small and descriptive commits, not one or two massive ones. In fact, I will not accept projects with too few commits.

Operating Systems

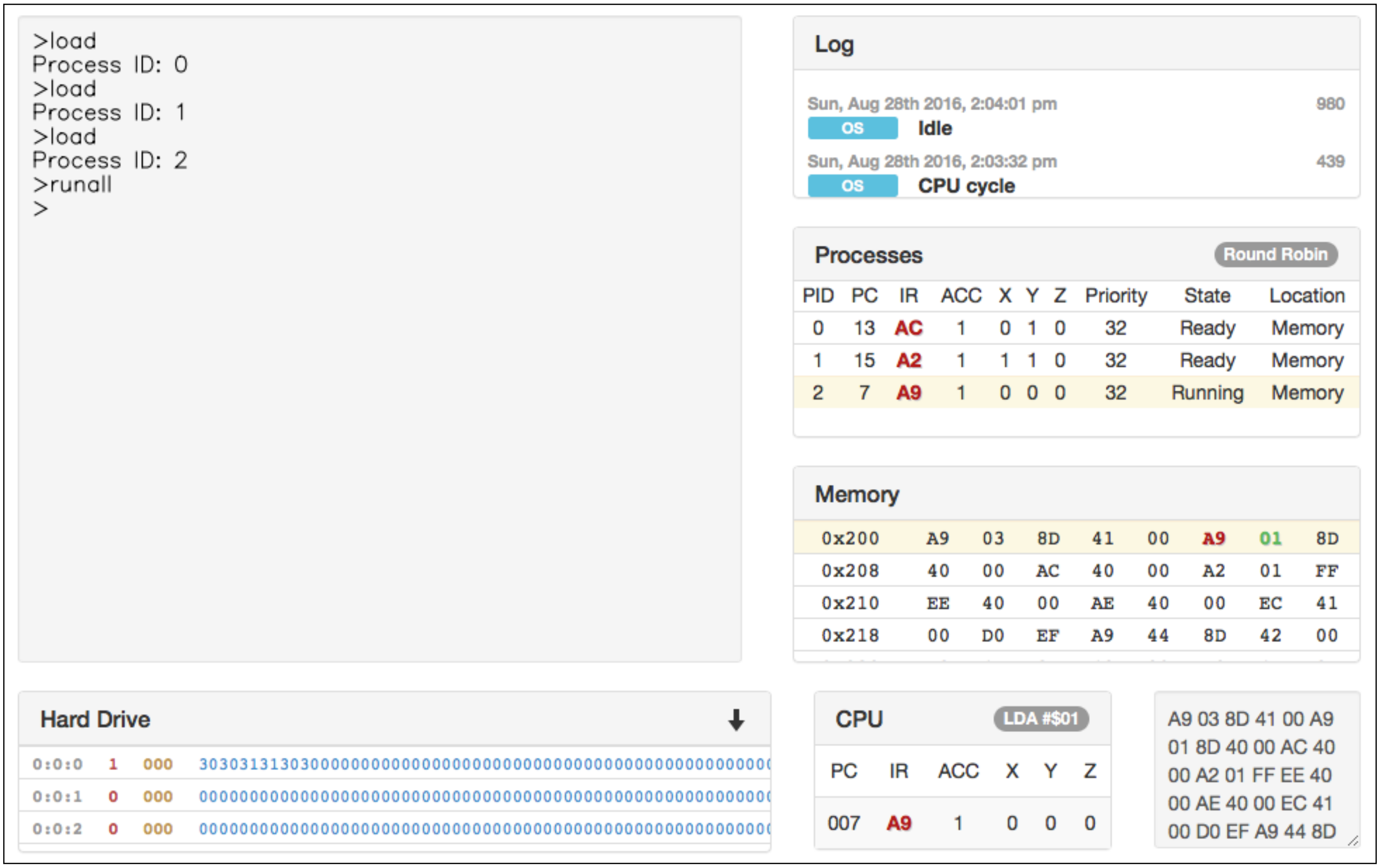
CMPT 424 • Fall 2024

Hints

- Do not reset PIDs as they are used. The PID value should always increase.
- The program counter for any process should never be greater than FF (255).
- Remember that scheduling is done via context switches triggered by the scheduler and implemented through software interrupts that are handled by the dispatcher.
- Updating your Ready Queue only on a scheduling events prevents the initial state from being displayed right away. This is bad.
- Display each processes' state in the Process Queue.
- What should happen if the user tries to clearmem while processes are running?
- Killing a process currently running on the CPU should work and should not create a zombie.
- Speaking of killing, GLaDOS is still alive. Do not break her.

Submitting Your Work

Update GitHub with your final code before the deadline specified in our syllabus. Remember to let me know which branch to grade in the `readme.md` file.



```
>load
Process ID: 0
>load
Process ID: 1
>load
Process ID: 2
>runall
>
```

Log

```
Sun, Aug 28th 2016, 2:04:01 pm                980
OS  Idle
Sun, Aug 28th 2016, 2:03:32 pm                439
OS  CPU cycle
```

Processes

PID	PC	IR	ACC	X	Y	Z	Priority	State	Location
0	13	AC	1	0	1	0	32	Ready	Memory
1	15	A2	1	1	1	0	32	Ready	Memory
2	7	A9	1	0	0	0	32	Running	Memory

Memory

```
0x200   A9  03  8D  41  00  A9  01  8D
0x208   40  00  AC  40  00  A2  01  FF
0x210   EE  40  00  AE  40  00  EC  41
0x218   00  D0  EF  A9  44  8D  42  00
```

Hard Drive

```
0:0:0  1  000  30303131303000000000000000000000000000000000000000000000
0:0:1  0  000  00000000000000000000000000000000000000000000000000000000
0:0:2  0  000  00000000000000000000000000000000000000000000000000000000
```

CPU

PC	IR	ACC	X	Y	Z
007	A9	1	0	0	0

```
A9 03 8D 41 00 A9
01 8D 40 00 AC 40
00 A2 01 FF EE 40
00 AE 40 00 EC 41
00 D0 EF A9 44 8D
```