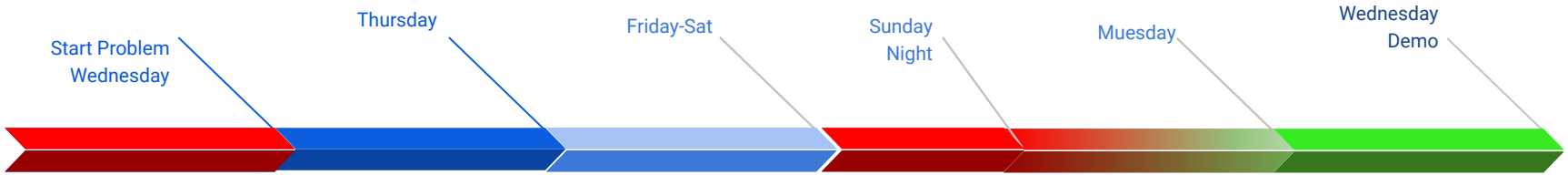
Implementation of Multi-Threading with Semaphore Support!



Understanding The Problem

Read Man pages and drew diagrams of how the system worked.

Diagram Functions

I drew function diagrams and UML and then did my .h file.

Implementation

Wrote all the code doing basic tests on each new module as I went.

Mistakes Realized

Discovered I had a bad loop after a few hours of debug, I also Learned the deadline had been moved up two days.

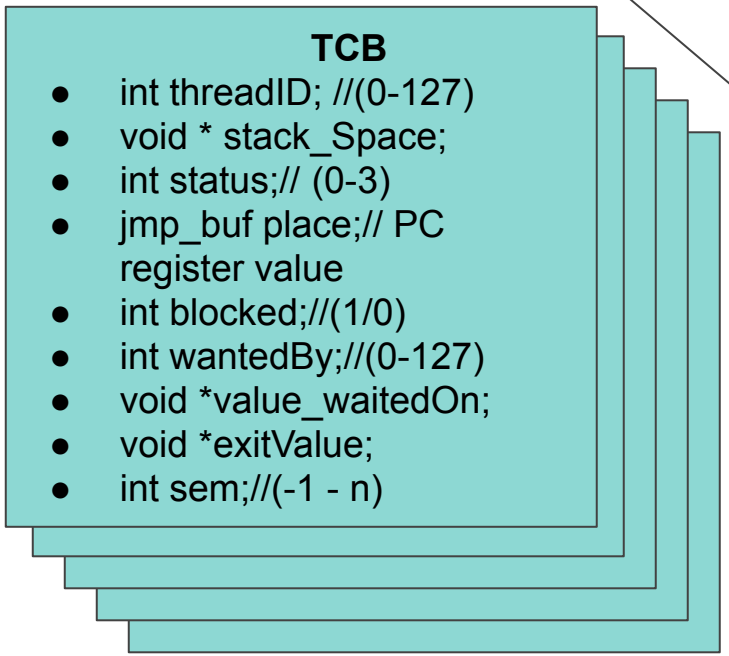
Debug/Complex Behavior Tests

Checked each module fixing problems as i went and then tested for more complex behavior by running a variety of different possible scenarios with as many edge cases as I could think of.

Demo Day

Cleaned up my code, Helped classmates with their issues demoed my code by feeding it through a comprehensive auto grader and explaining it.

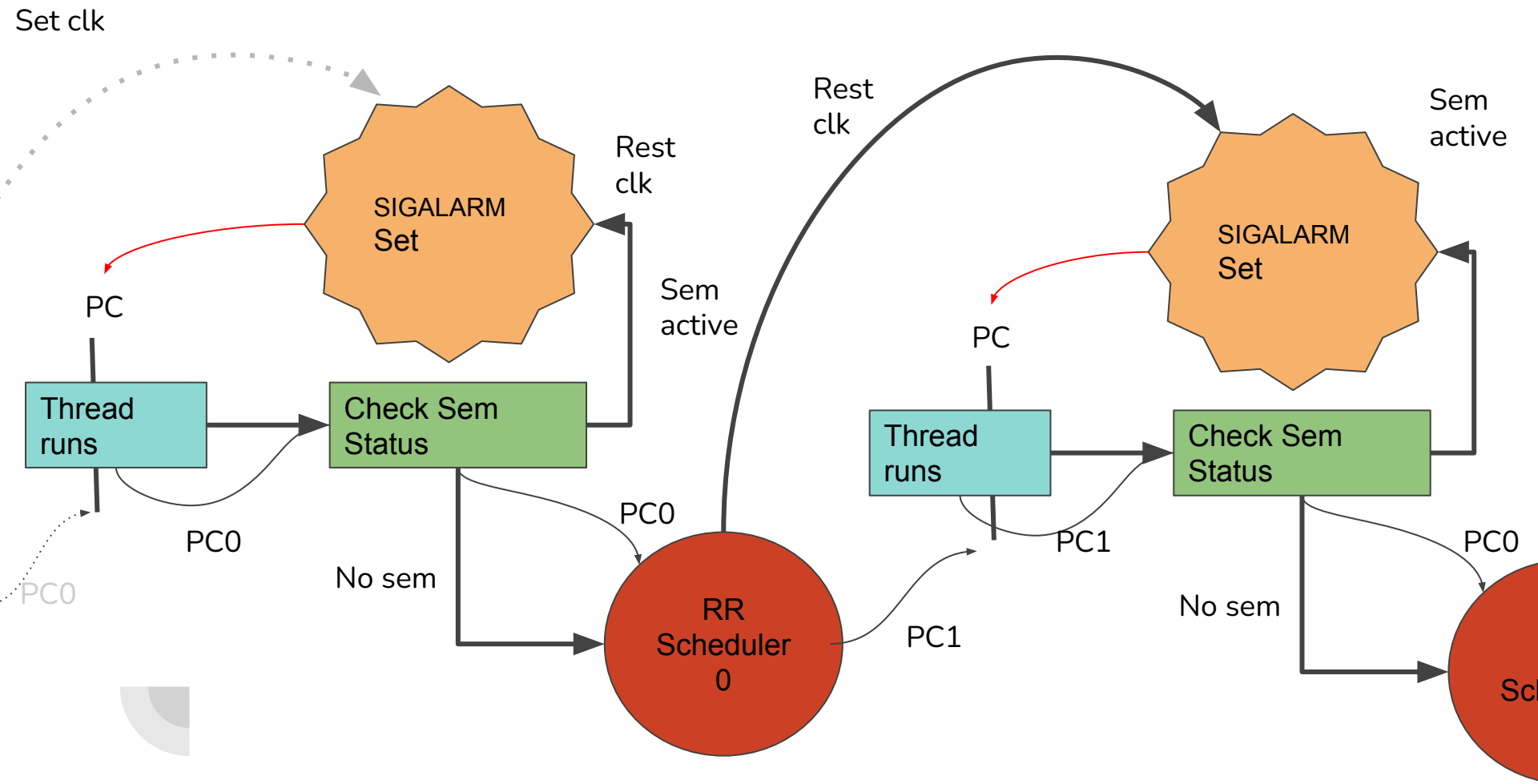
Dev Timeline

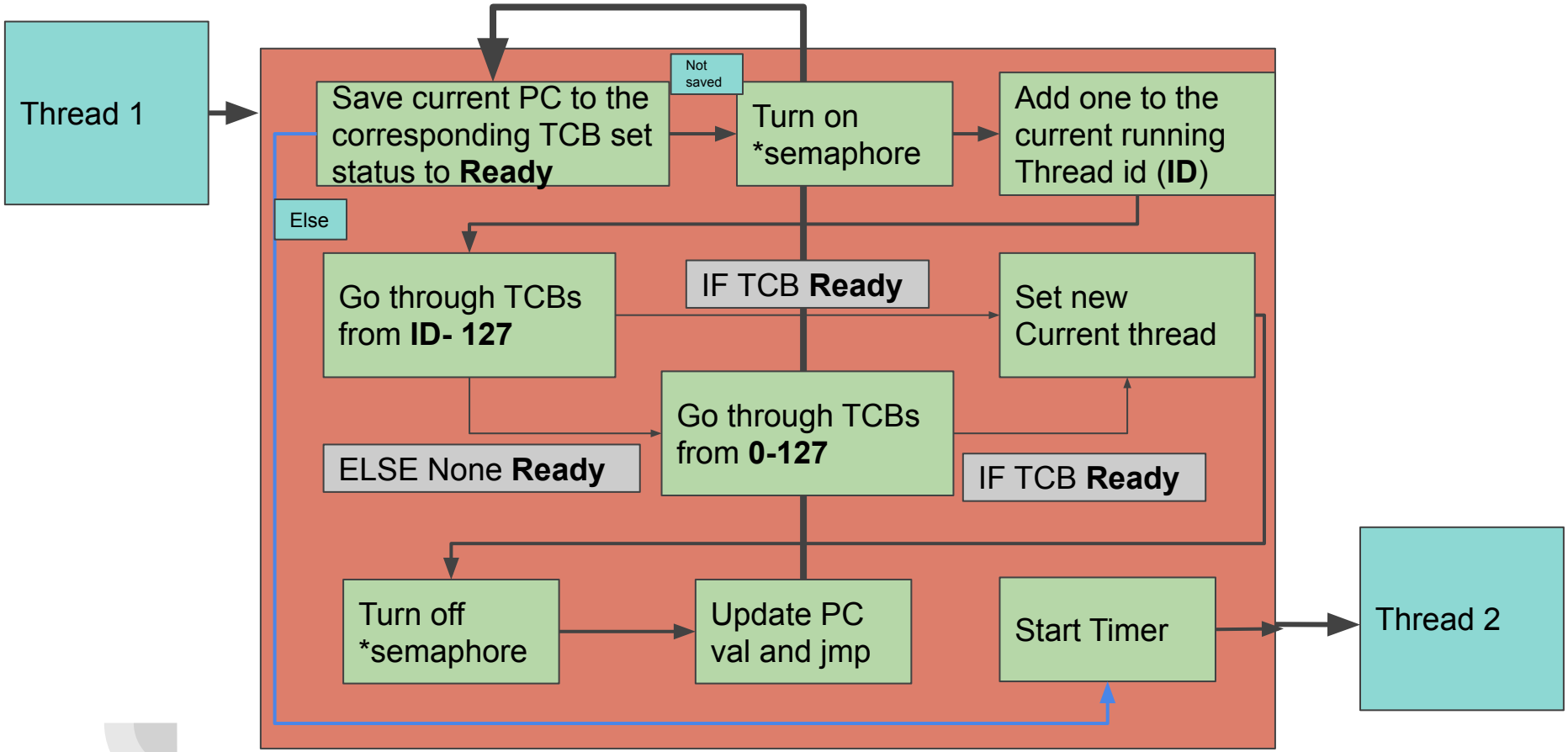


Threads(128 threads supported)
*called **TCBs** for the rest of the PP

Semaphore

- int semid;// (0-127)
- unsigned int currentVal;//(0-n)
- int initialized;// (1/0)


Set clk
SIGALARM
Set
Rest
clk
PC
Sem
active
Thread
runs
Check Sem
Status
PC0
PC0
No sem
RR
Scheduler
0
Rest
clk
SIGALARM
Set
Sem
active
PC
Thread
runs
Check Sem
Status
PC1
PC0
No sem
PC1

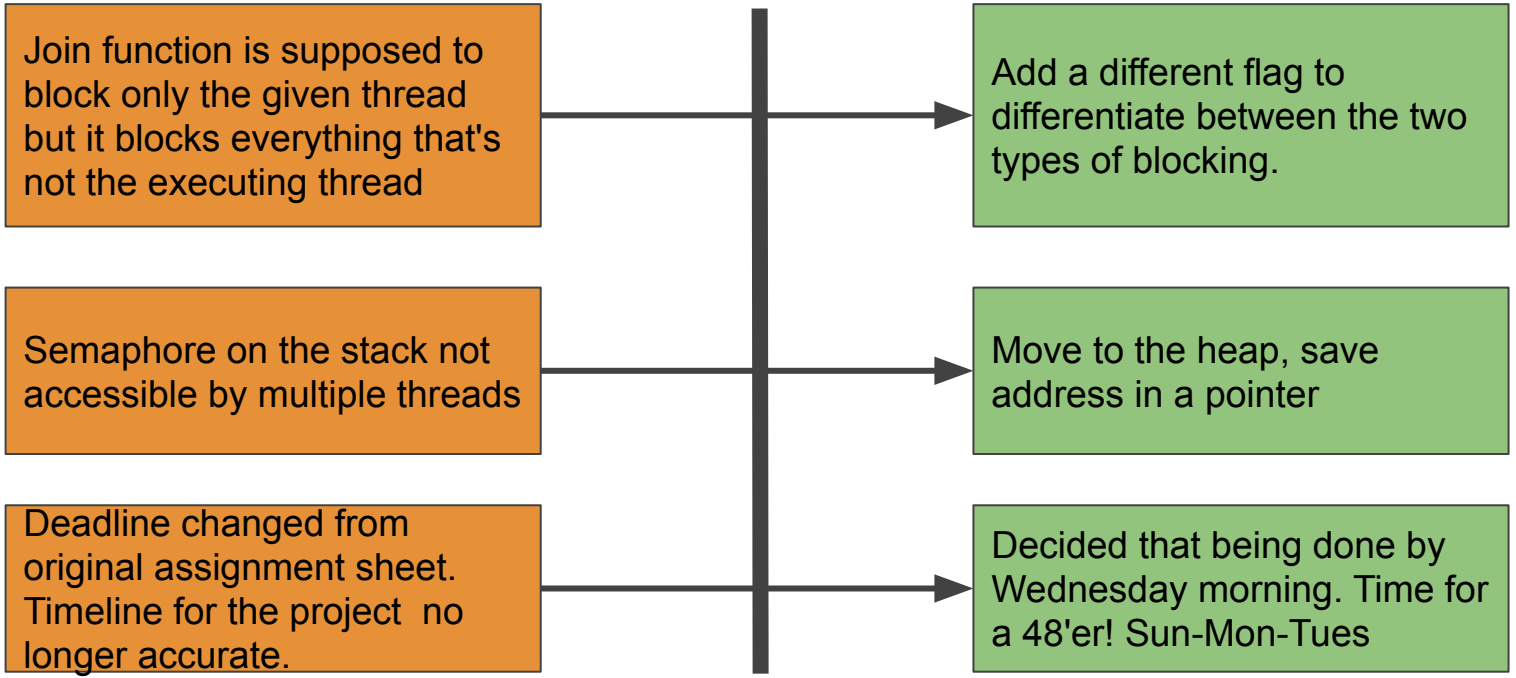


How the Scheduler works

Functionality

- pthread_create(pthread_t *thread,const pthread_attr_t *attr,void *(*start_routine)(void *),void *arg);
- pthread_join(pthread_t thread, void **value_ptr);
- sem_init(sem_t *sem, int pshared, unsigned value);
- sem_wait(sem_t *sem);
- sem_post(sem_t *sem);
- sem_destroy(sem_t *sem);

Issues and solutions



Demo!!