# REPORT PPC5

***Parking - 2 Slot 5 Cars Scheduler using Timer and Semaphores***

***8051 Microcontroller Programming Run on Edsim-51***

***CS 342302 Operating Systems***

***Fall Semester 2022***

***Prof. Pai H. Chou***

**Kevin Richardson Halim 林之耀**

**109006277**

## I. Code: *delay()* and *now()* –Timer Handler

Timer-0 ISR to support our second counter. This timer is sync across the 8051 and can be used to support timer delay and now in the preemptive.c.

Delay() and happen to finish their delays all at the same time by choosing timer windows to be more than or equal than the number of MAXTHREAD. Moreover ensuring the delay accuracy is less than n+0.5, select the timer windows to be twice or more than the MAXTHREAD.

The worst case scenario is a 4 time cycle in which the delay between a process timestamp is 3 timer cycle.

## II. Code: *ThreadExit()* and *ThreadCreate()* –Thread Handler

ThreadExit's job is to remove the "live" indicator in thread bitmap and cycle to the next available thread or be stuck in a while loop. Meanwhile, ThreadCreate will store the original SP and prepare the resources needed for each thread, where each thread needs its own bank, and other values of SPF and push this value to Thread's stack. Thread's stack has been predetermined, so each thread has their specific location.

## III. Code: *testparking.c* –Testbench for Semaphore with 2 Slots and 5 Cars

File testparking.c contains main and Producer.. Each of these functions help simplify the test of semaphore concept with its name and value each are empty: 2, mutex: 1, and full: 0. Each new creation exceeding 4 threads will have to do semaphore wait full label. The mutex label ensures nobody accessing memory together at the same time and the empty label at most there are only 2 concurrent threads running.

To achieve print log in a readable human format, function log helps the job with help from iPrint, cPrint and sPrint; each handles integer(i), character(c), and string(s). The output format are:

Log: CAR-{i} SLOT-{j} {"arrives" or "leaves"} at time {t}\n

This format tells the car with a label i has to do with the j-th slot, either 1 or 2. It has to do with entering or leaving action at t-th time stamp.

## IV. Typescript: SDCC C Compile

These are the typescript **make clean** and **make all** commands using SDCC C compiler.

```
root@LAPTOP-9RQA3LLT:/mnt/e/{-} Mingw CodeBlock/[OS]/5# make clean
rm *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym
rm: cannot remove '*.ihx': No such file or directory
rm: cannot remove '*.lnk': No such file or directory

make: *** [Makefile:25: clean] Error 1
root@LAPTOP-9RQA3LLT:/mnt/e/{-} Mingw CodeBlock/[OS]/5#
```

The command **make clean** removes *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym files. In this syntax * means all files with each criteria where it ends with .(something).

```
root@LAPTOP-9RQA3LLT:/mnt/e/{-} Mingw CodeBlock/[OS]/5# ls
Makefile  preemptive.c  preemptive.h  testparking.c
root@LAPTOP-9RQA3LLT:/mnt/e/{-} Mingw CodeBlock/[OS]/5# make
sdcc -c  testparking.c
sdcc -c  preemptive.c
preemptive.c:140: warning 85: in function ThreadCreate unreferenced function argument : 'fp'
sdcc  -o testparking.hex testparking.rel preemptive.rel
root@LAPTOP-9RQA3LLT:/mnt/e/{-} Mingw CodeBlock/[OS]/5# ls
Makefile        preemptive.h    preemptive.rst   testparking.c    testparking.lst  testparking.rel
preemptive.asm  preemptive.lst  preemptive.sym   testparking.hex  testparking.map  testparking.rst
preemptive.c    preemptive.rel  testparking.asm  testparking.lk   testparking.mem  testparking.sym
root@LAPTOP-9RQA3LLT:/mnt/e/{-} Mingw CodeBlock/[OS]/5#
```

On the other hand, the command **make all** compiles **testcoop.c** and **cooperative.c** first into **testcoop.rel** and **cooperative.rel** and link those two compiled with output format **testcoop.hex**.