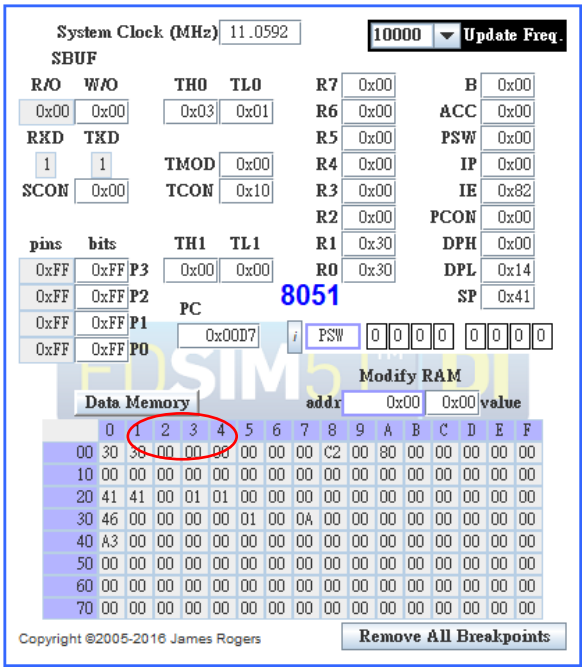
Programming Project Checkpoint 3

IPNS 106010018 Yen-Fong Li

```
      Value  Global                          Global Defined In Module
      -----  -------------------------------  -----------------------
C:    00000014  _Producer                      testpreempt
C:    00000052  _Consumer                      testpreempt
C:    0000008E  _main                          testpreempt
C:    000000A6  __sdcc_gsinit_startup          testpreempt
C:    000000AA  __mcs51_genRAMCLEAR            testpreempt
C:    000000AB  __mcs51_genXINIT               testpreempt
C:    000000AC  __mcs51_genXRAMCLEAR           testpreempt
C:    000000AD  _timer0_ISR                    testpreempt
C:    000000B1  _Bootstrap                     preemptive
C:    000000D7  _ThreadCreate                  preemptive
C:    00000156  _ThreadYield                   preemptive
C:    000001EC  _myTimer0Handler               preemptive
C:    0000027C  _ThreadExit                    preemptive
```

From testpreempt.map, we can find out that the address of ThreadCreate() function is at 000000D7, so set checkpoint at 00D7 and then run Edsim51.



Semaphore:

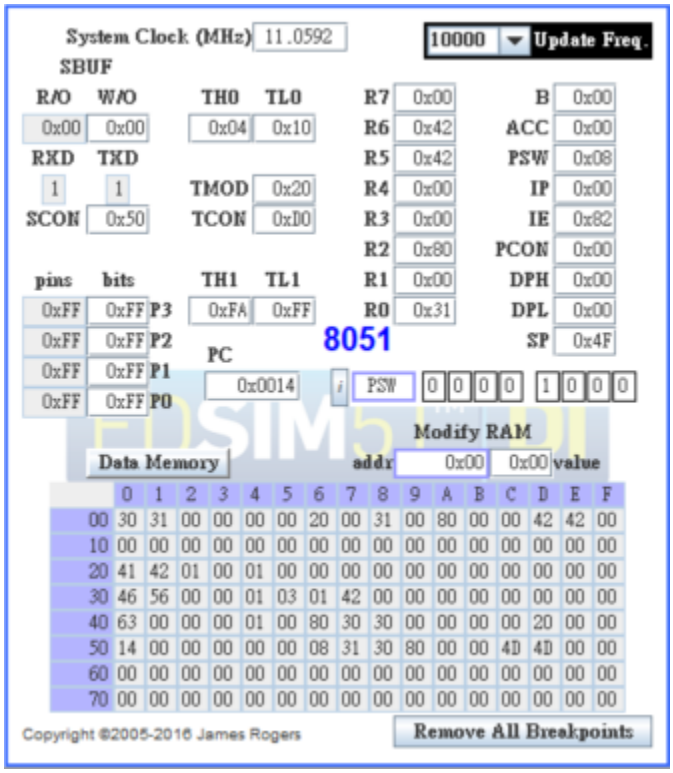
0x21 for full is 0

0x23 for empty is 1

0x24 for mutex is 1

They are correctly initialized.

Producer



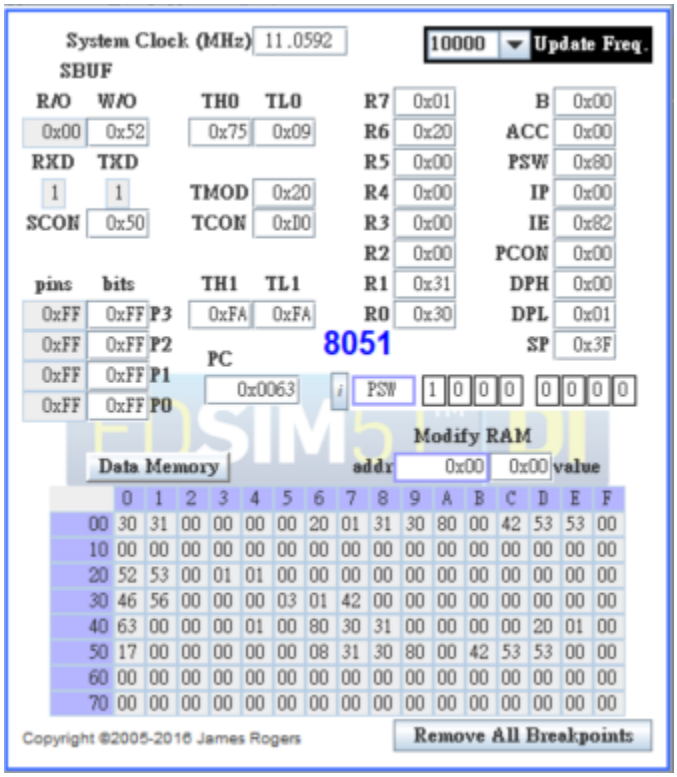
After Calling Producer()

Since empty and mutex were initialized as 1

It can run through and signal full

So 0x22 become 01

Consumer



After Calling Consumer ()

Since full became 01

It can run through and signal empty

So 0x23 become 01

Explanations

```
#define SemaphoreWait(s) \
    SemaphoreWaitBody(s, L(__COUNTER__)) \

#define SemaphoreWaitBody(s, label) \
    { __asm \
      label: \
      MOV ACC, CNAME(s) \
      JZ label \
      JB ACC.7, label \
      dec  CNAME(s) \
     __endasm; }
```

WAIT()

No need to separate to two functions. Just Call SemaphoreWaitBody and give the right parameters can also do the work.

In SemaphoreWaitBody, We first move s to accumulator Then we decide whether we should jump back to label by checking whether it is zero.

```
#define SemaphoreSignal(s)  \
    __asm \
       INC CNAME(s) \
    __endasm;\
```

SIGNAL()

In this part, the only thing we need to do is increment the semaphore.