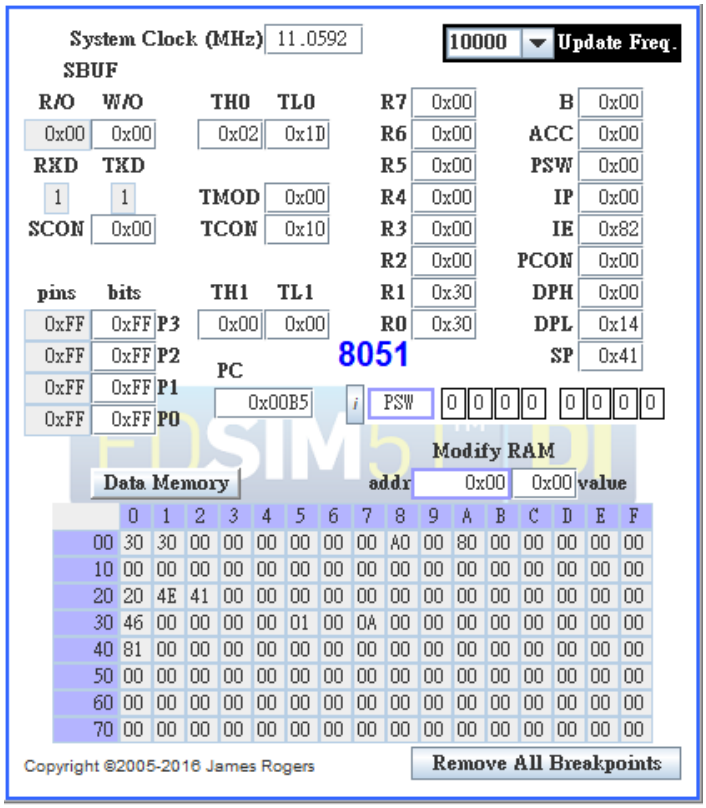
Programming Project Checkpoint 2

IPNS 106010018 Yen-Fong Li

Testpreempt.map

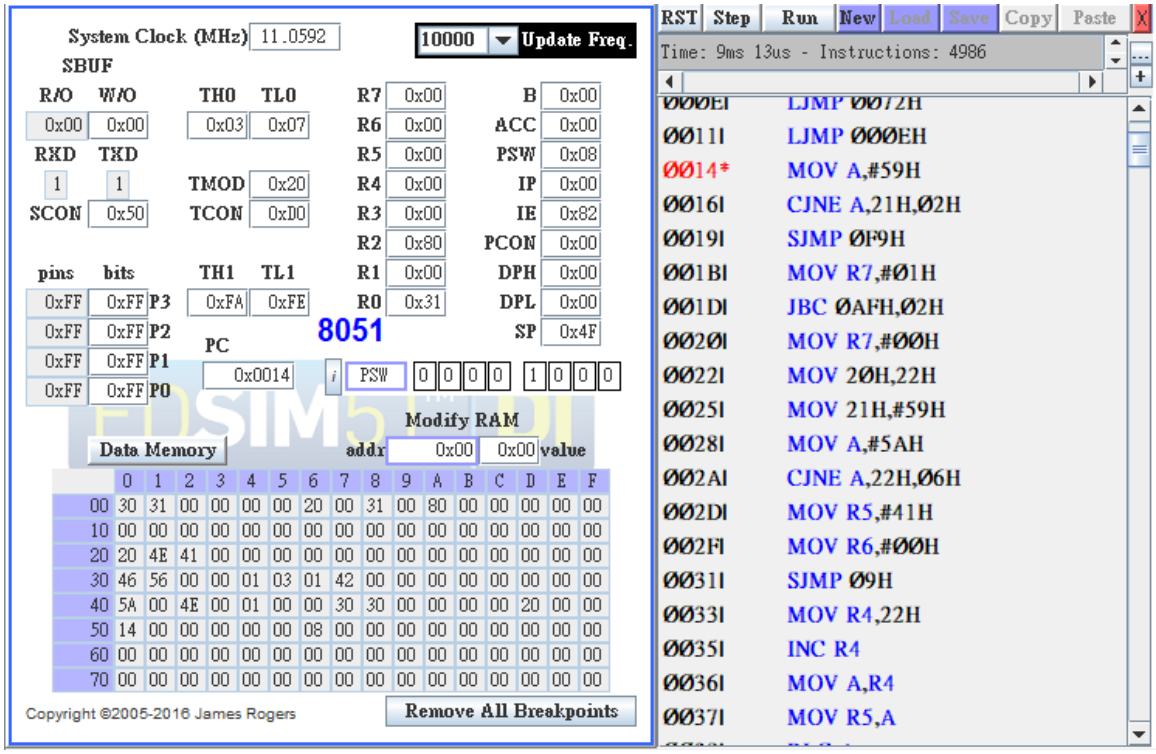
```
      Value  Global                          Global Defined In Module
      -----  -------------------------------  -----------------------
C:    00000014  _Producer                      testpreempt
C:    00000044  _Consumer                      testpreempt
C:    00000072  _main                          testpreempt
C:    00000084  __sdcc_gsinit_startup          testpreempt
C:    00000088  __mcs51_genRAMCLEAR            testpreempt
C:    00000089  __mcs51_genXINIT               testpreempt
C:    0000008A  __mcs51_genXRAMCLEAR           testpreempt
C:    0000008B  _timer0_ISR                    testpreempt
C:    0000008F  _Bootstrap                     preemptive
C:    000000B5  _ThreadCreate                  preemptive
C:    00000134  _ThreadYield                   preemptive
C:    000001CA  _myTimer0Handler               preemptive
C:    0000025A  _ThreadExit                    preemptive
```

From testpreempt.map, we can find out that the address of ThreadCreate() function is at 000000B5, so set checkpoint at 0085 and then run Edsim51.



0x40 save the return address, after ThreadCreate(), the program can go back and continue

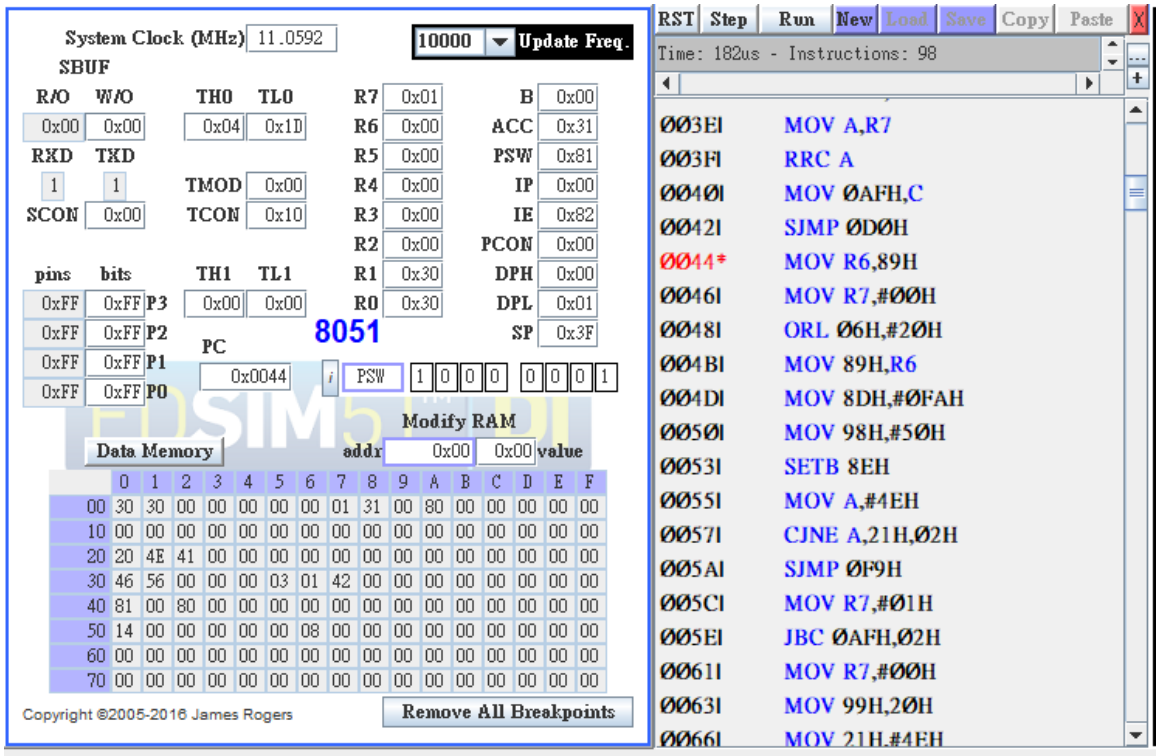
Producer



From testcoop.map, we can tell that the address of Producer is at 0009. Set checkpoint to it and run Edsim51.

To know whether the Producer is running, it can be observed from the address 0x34, which points to the current_id. (current thread id) The value of it is 01, which means Producer is running.

Consumer



From testcoop.map, we can tell that the address of Producer is at 0009. Set checkpoint to it and run Edsim51.

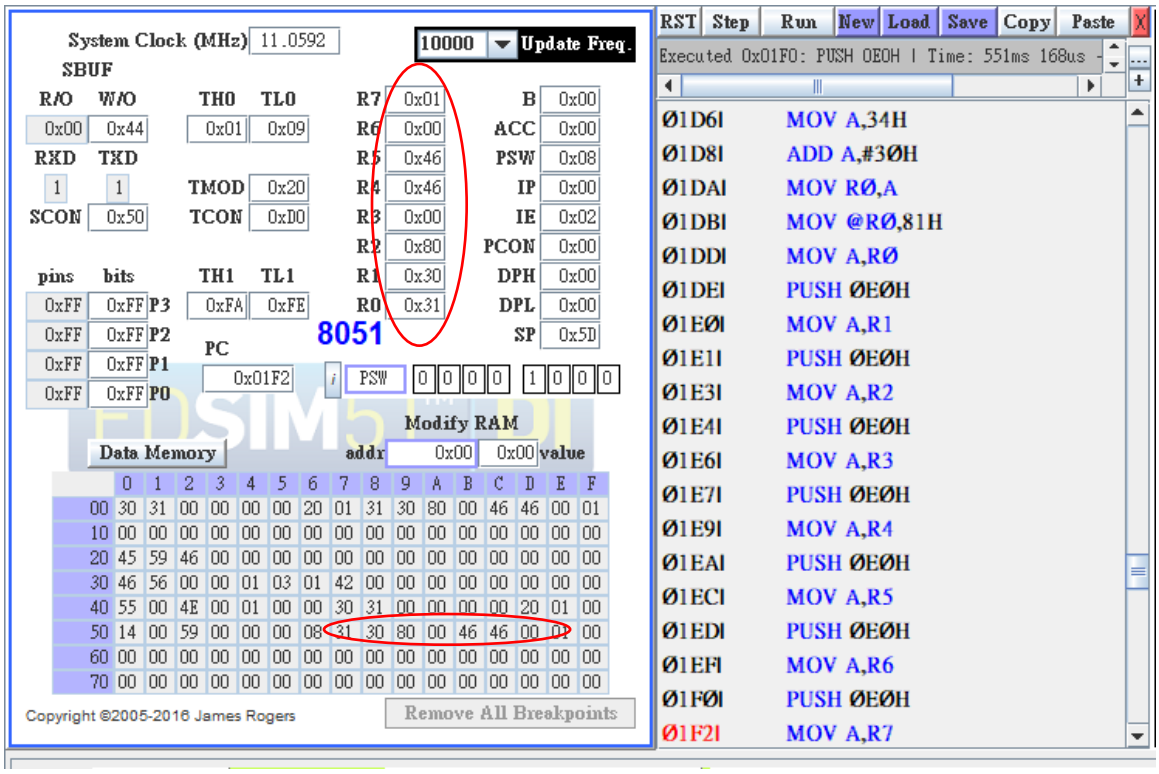
To know whether the Consumer is running, it can be observed from the address 0x34, which points to the current_id. (current thread id) The value of it is 00, which means Producer is running.

myTimer0Handler (Interrupt triggered)

When an interrupt is triggered, the function myTimer0Handler() will be called. From testpreempt.map, we know that the address of myTimer0Handler() is 01cA. Set a breakpoint to it and then run edsim51.

Since I wrote parts of my code of myTimer0Handler() in C between SAVESTATE and RESTORESTATE, it is likely to use R0 and R1, any code that modifies R0-R7 will trash them. I had to save them to the stack so I could restore their value.

In that case, when interrupt is triggering, we can notice that the top of the stack will have the value of R0-R7.



Explanations

This checkpoint is similar with the last checkpoint. There are some parts to change.

1. Don’t need ThreadYield()
2. Plus __critical
   Since we implement Producer-Consumer Problem by preemptive way, we need to add __critical to the part that are shared, preventing from multiple accesses.
3. Plus EA
   EA is used to disable interrupt and enable interrupt by changing its value to 0 or 1.