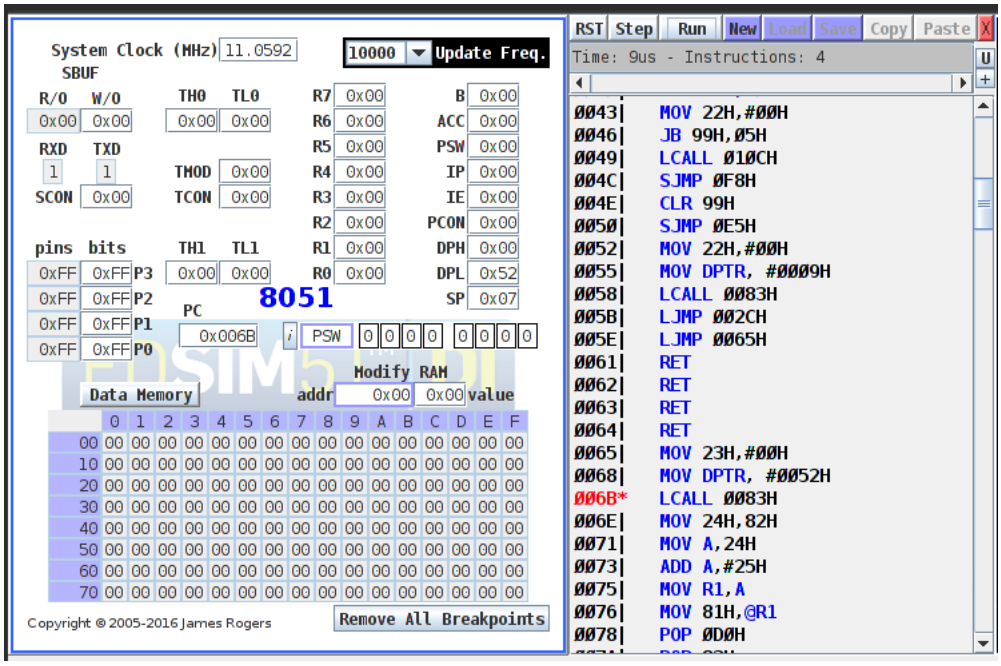
110060035 黃振寧

There are only two threads in my program. The first one is for main and the second one is for producer. Please note that the consumer uses the same thread create for main at first.

From the memory mapping as shown below, we can find out that “00000083” is where the function ThreadCreate lies. Therefore, we set the breakpoint on line 006B, which is the line for “LCALL 0083H”.

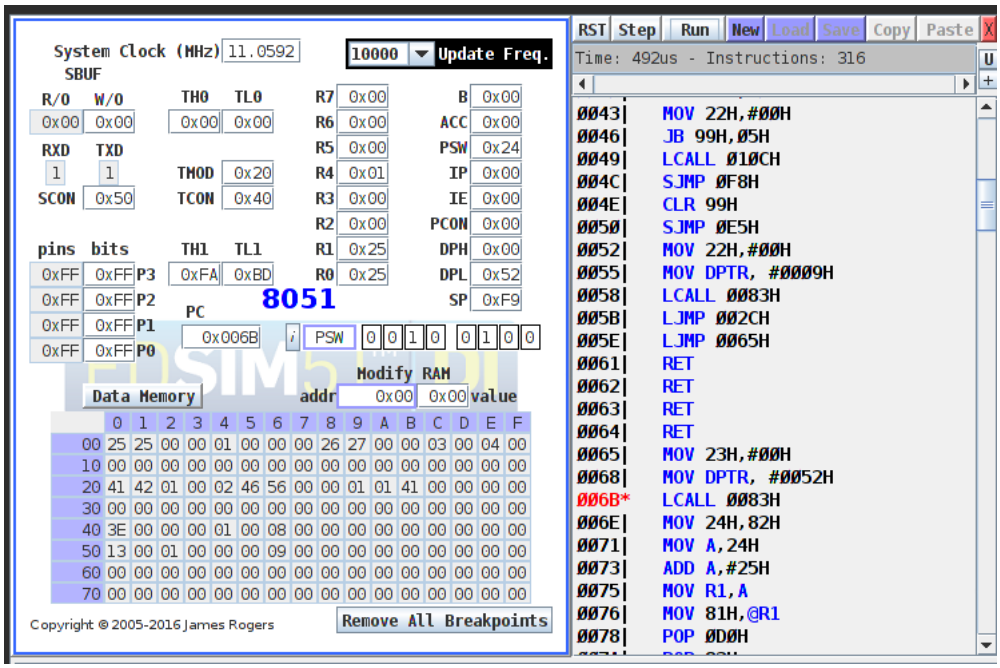
```
      Value  Global                                 Global Defined In Module
      -----  --------------------------------   -------------------------
C:    00000009  _Producer                           testcoop
C:    0000002C  _Consumer                           testcoop
C:    00000052  _main                               testcoop
C:    0000005E  __sdcc_gsinit_startup               testcoop
C:    00000062  __mcs51_genRAMCLEAR                 testcoop
C:    00000063  __mcs51_genXINIT                    testcoop
C:    00000064  __mcs51_genXRAMCLEAR                testcoop
C:    00000065  _Bootstrap                          cooperative
C:    00000083  _ThreadCreate                       cooperative
C:    0000010C  _ThreadYield                        cooperative
C:    00000166  _ThreadExit                         cooperative
```

1. Threadcreate(main)



As shown in the graph, there are nothing in the stack before creating the thread for main.

2. Threadcreate(producer)



Then when jumping to 0083H, we push 006E on the stack so we can return back to continue executing our code. In ThreadCreate(), we move the SP to 0X3F to use 0X40~0X4F as stack 0. Next, we push the address of main(0X52) on the stack. Next, we pushes DPL(0X08), DPH(0X09), the pointer to main, ACC(0X0A), B(0X0B), DPL(0X0C), DPH(0X0D), all of them are set to zero, and PSW(0X0E), indicating that we are using bank 0.

After that we move sp back to 0X09 using sp_temp. And then, when finishing ThreadYield(), we return to 0X6E, popping two values out of the stack.

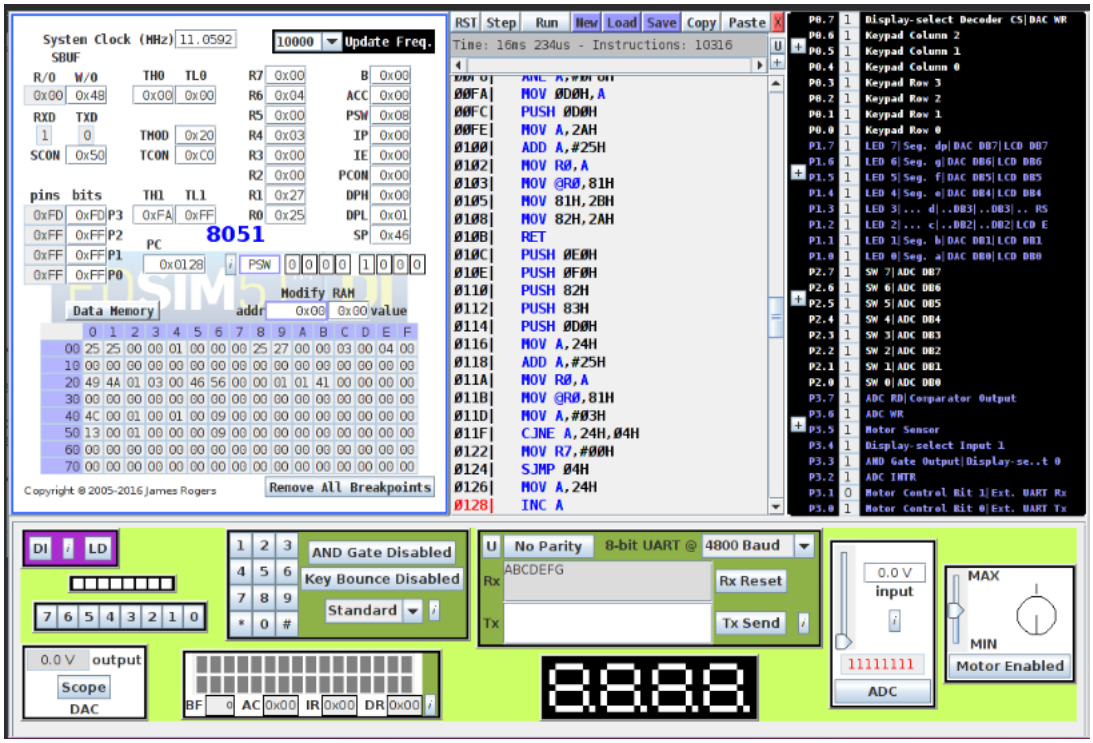
After that we assign SP as the saved_sp[0](0X46). We pop PSW, DPH, DPL, B, and ACC. Then, we make return to main(). Now there is no value on the stack with SP pointing to 0X3F..

Next, in main, we called ThreadCreate() again.

We map the variable current_thread to “00000024”. Therefore, by looking at the value at 0X24, we can know the current running node.

```
00000020  _buffer                    testcoop
00000021  _msg                       testcoop
00000022  _full                      testcoop
00000023  _thread_bitmap             cooperative
00000024  _current_thread            cooperative
00000025  _saved_sp                  cooperative
00000029  _i                         cooperative
0000002A  _new_thread                cooperative
0000002B  _sp_temp                   cooperative
```

A screenshot when producer is running (current_thread = 0).



A screenshot when consumer is running (current_thread = 1).