

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".

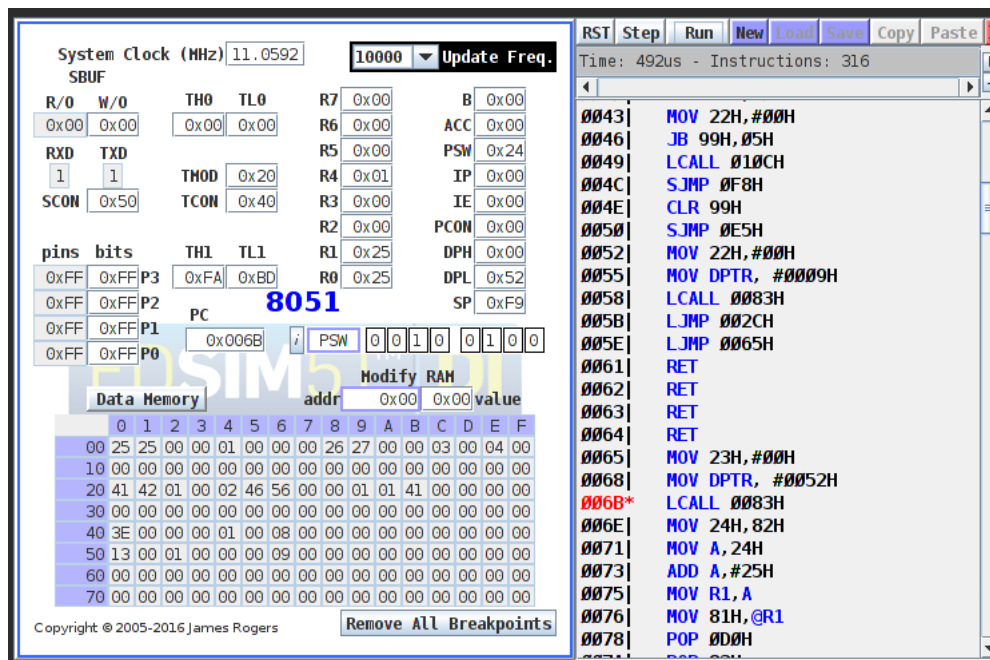
Value	Global	Global Defined In Module
C: 00000009	_Producer	testcoop
C: 0000002C	_Consumer	testcoop
C: 00000052	_main	testcoop
C: 0000005E	_sdcc_gsinit_startup	testcoop
C: 00000062	_mcs51_genRAMCLEAR	testcoop
C: 00000063	_mcs51_genXINIT	testcoop
C: 00000064	_mcs51_genXRAMCLEAR	testcoop
C: 00000065	_Bootstrap	cooperative
C: 00000083	_ThreadCreate	cooperative
C: 0000010C	_ThreadYield	cooperative
C: 00000166	_ThreadExit	cooperative

1. Threadcreate(main)

The screenshot shows the SIM51 IDE interface. On the left, the register window displays the PC (Program Counter) at 0x006B. The assembly code window on the right shows the instruction at address 006B: **LCALL 0083H**. Below the code, the memory dump shows the stack is empty, with all memory locations containing 00.

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 push 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.

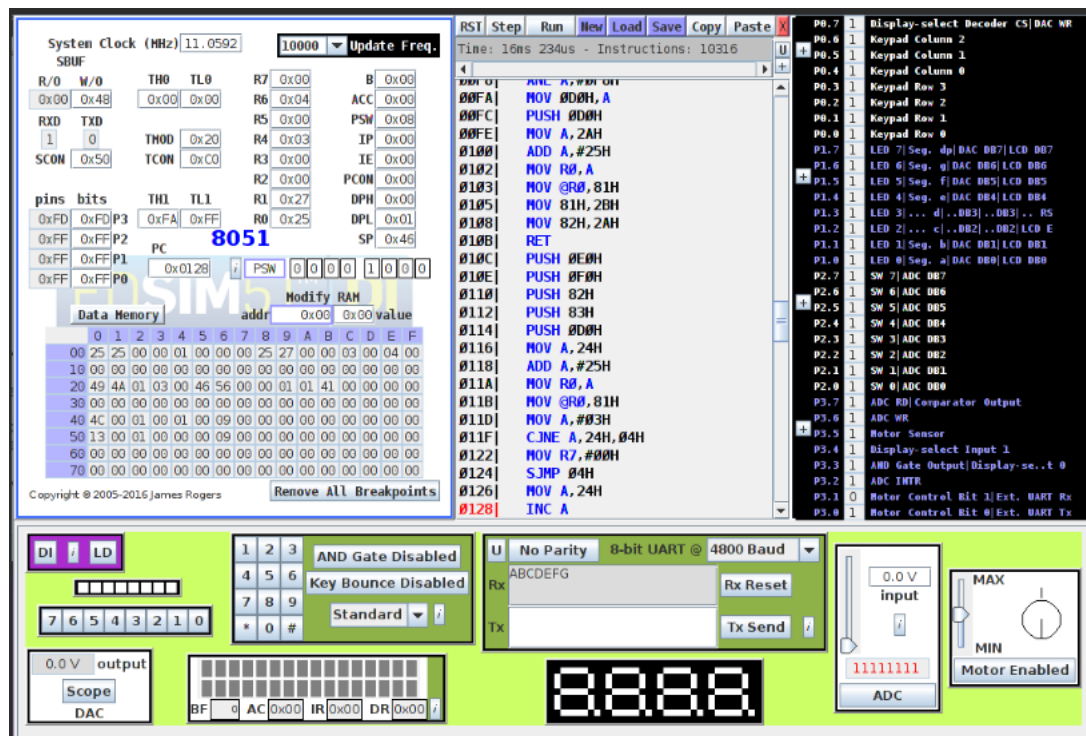
After that we assign SP as the saved_sp[0](0X46). We pop PSW, DPH, DPL, B, and ACC. Then, we make return to main(). Now there is no value on the stack with SP pointing to 0X3F..

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).

The screenshot displays the Proteus ISIS simulation environment for an AVR microcontroller. The main CPU window shows the following registers and values:

R/W	M/W	TH0	TL0	R7	B
0x00	0x55	0x00	0x00	0x00	0x01

Other registers shown include R6 (0x01), R5 (0x00), R4 (0x03), R3 (0x00), R2 (0x00), R1 (0x27), R0 (0x25), PC (0x0138), and SP (0x46). The Program Counter (PC) is highlighted at 8051.

The instruction list shows the following assembly code:

```
0110] PUSH 82H
0112] PUSH 83H
0114] PUSH 000H
0116] MOV A, 24H
0118] ADD A, 25H
011A] MOV R0, A
011B] MOV @R0, 81H
011D] MOV A, #03H
011F] CJNE A, 24H, 04H
0122] MOV R7, #00H
0124] SJMP 04H
0126] MOV A, 24H
0128] INC A
0129] MOV R7, A
012A] MOV 24H, R7
012C] MOV A, 24H
012E] MOV 0F0H, 24H
0131] INC 0F0H
0133] MOV R6, #01H
0135] MOV R7, #00H
0137] SJMP 06H
0139] MOV A, R6
013A] ADD A, R6
013B] MOV R6, A
```

The I/O window on the right shows the state of various peripherals, including the Display-select Decoder CS, DAC WR, Keypad Column 2, Keypad Column 1, Keypad Column 0, Keypad Row 3, Keypad Row 2, Keypad Row 1, Keypad Row 0, LED 7 Seg. dp, LED 6 Seg. g, LED 5 Seg. f, LED 4 Seg. e, LED 3 Seg. d, LED 2 Seg. c, LED 1 Seg. b, LED 0 Seg. a, SM 7 ADC 0B7, SM 6 ADC 0B6, SM 5 ADC 0B5, SM 4 ADC 0B4, SM 3 ADC 0B3, SM 2 ADC 0B2, SM 1 ADC 0B1, SM 0 ADC 0B0, ABC RD Comparator Output, ABC WR, Motor Sensor, Display-select Input 1, AND Gate Output, ADC INTR, Motor Control Bit 1, and Motor Control Bit 0.

The bottom panel shows the I/O window with the following components:

- DI: 1, LD: 1
- AND Gate Disabled
- Key Bounce Disabled
- Standard
- 0.0 V output
- Scope
- DAC
- 0.0 V input
- 11111111
- ADC
- Motor Enabled