- Take one screenshot before each `ThreadCreate` call. Explain how the stack changes.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	30	00	00	01	00	00	00	91	00	00	00	00	46	46	00
10	00	30	00	00	00	00	00	10	00	00	00	00	00	00	00	00
20	07	00	00	00	00	00	00	45	46	00	00	00	00	00	00	00
30	46	56	00	00	00	01	00	41	01	02	41	01	00	00	00	00
40	4F	00	00	00	01	00	88	30	32	00	00	00	45	45	00	00
50	19	00	01	00	00	00	89	31	31	00	00	00	46	46	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

1.

The initial stack is in 0x07, and after call LCALL it saves the return address in stack, so it becomes 0x09. Then, the stack thread 1 use start from 0x40 to 0x4F. After push DPL, DPH, the stack is in 0x41. Later it push 0 into stack 4 times and PSW. In the end, it return to 0x09 to get the return address.

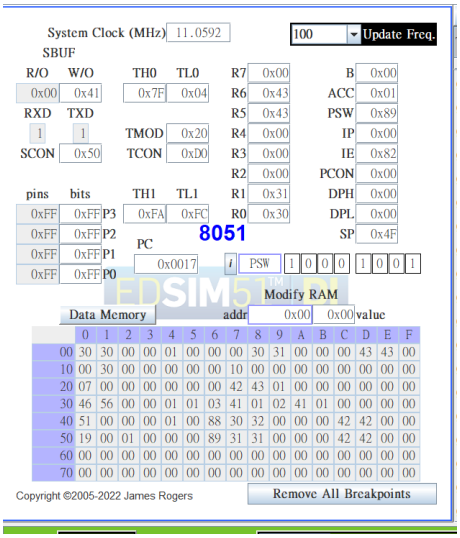
P 0x3F -> SP 0x41 -> SP 0x45 -> SP 0x46 -> SP 0x09 -> SP 0x07

2.

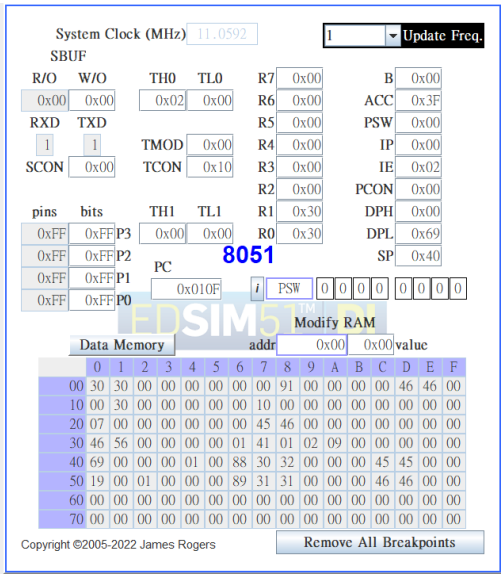
SP 0x3F -> SP 0x41 -> SP 0x51 -> SP 0x55 -> SP 0x56 ->

SP 0x41 -> SP 0x3F

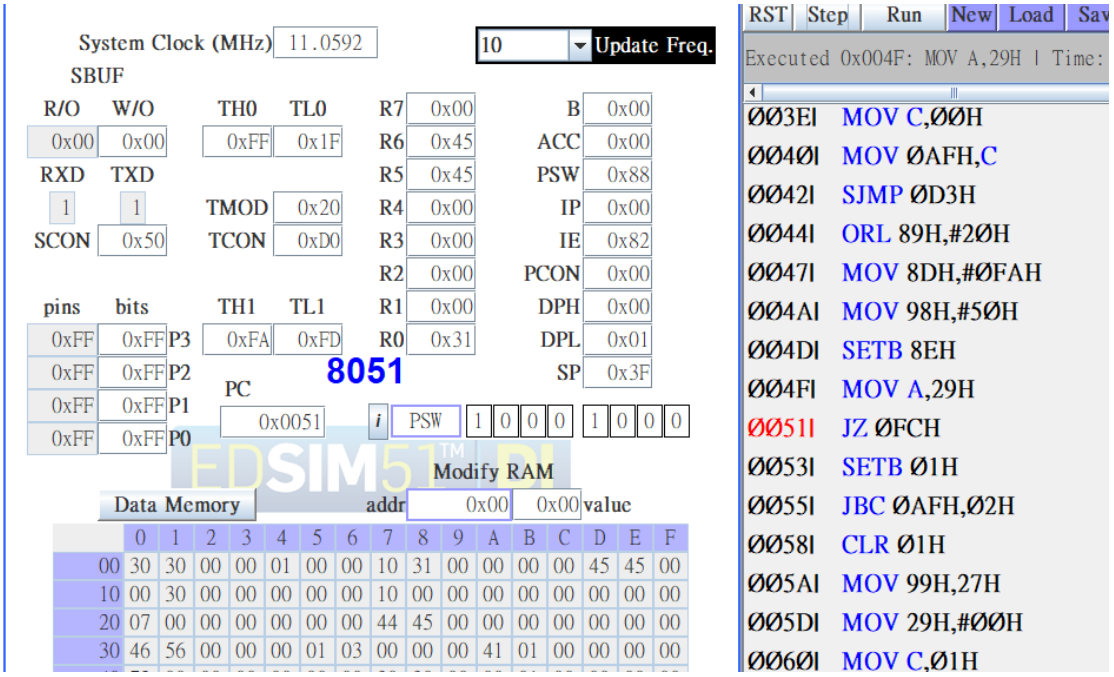
- Take one screenshot when the Producer is running. How do you know? Since my cur_thread is in 0x34, when 0x34 is 1, it means that Producer is running.



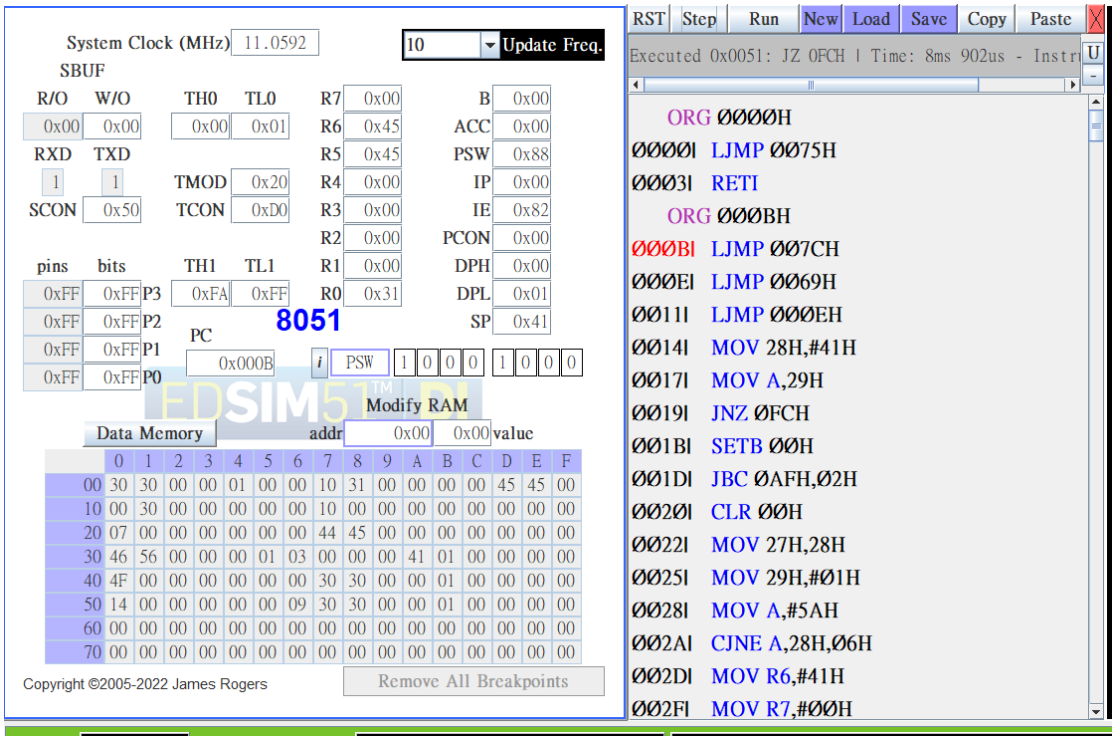
- Take one screenshot when the Consumer is running. How do you know? Since my cur_thread is in 0x34, when 0x34 is 0, it means that Producer is running.



- How can you tell that the interrupt is triggering on a regular basis?



TH0 = 0xFF, TL0 = 0x1F 的時候會進到 timer0handler 去做 preemptive 的動作。



007C 是 void timer0_ISR(void) __interrupt(1)的位置，裡面再去 call timer0handler。

Typescript

```
PS C:\Users\josh9\OneDrive\桌面\大三上\OS\109062119-ppc2> make
sdcc -c  preemptive.c
preemptive.c:155: warning 85: in function ThreadCreate unreferenced function argument : 'fp'
preemptive.c:249: warning 158: overflow in implicit constant conversion
sdcc  -o testpreempt.hex testpreempt.rel preemptive.rel
PS C:\Users\josh9\OneDrive\桌面\大三上\OS\109062119-ppc2>
```