

- 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.

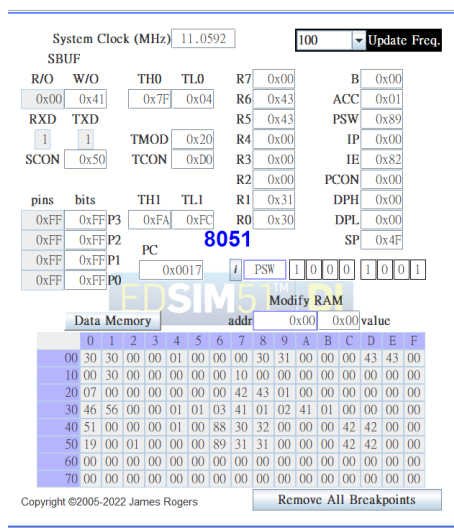
The initial stack is in 0x07, and after call LCALL it saves the return address in stack, so it becomes 0x09. Then, the stack thread 1 use start from 0x40 to 0x4F. After push DPL, DPH, the stack is in 0x41. Later it push 0 into stack 4 times and PSW. In the end, it return to 0x09 to get the return address.

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

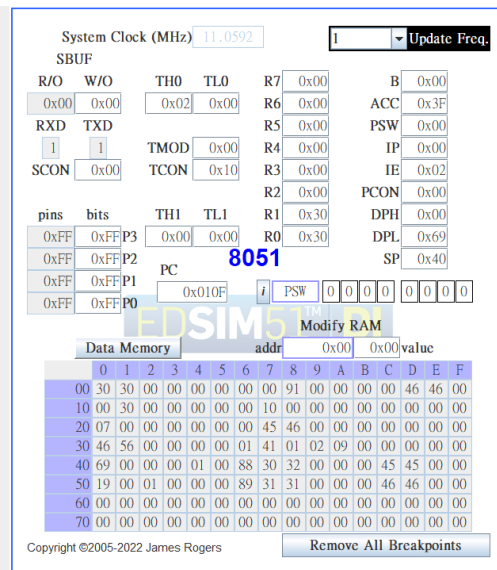
2.

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?

System Clock (MHz): 11.0592 | Update Freq. 10

SBUF: R/O 0x00, W/O 0x00, TH0 0xFF, TL0 0x1F, R7 0x00, B 0x00, R6 0x45, ACC 0x00, R5 0x45, PSW 0x88, R4 0x00, IP 0x00, R3 0x00, IE 0x82, R2 0x00, PCON 0x00, R1 0x00, DPH 0x00, R0 0x31, DPL 0x01, SP 0x3F.

pins: P3 0xFF, P2 0xFF, P1 0xFF, P0 0xFF. bits: P3 0xFF, P2 0xFF, P1 0xFF, P0 0xFF.

TH1 0xFA, TL1 0xFD. PC 0x0051. PSW 10000100.

Data Memory: addr 0x00 to 0x0F, value 00 30 30 00 00 01 00 00 10 31 00 00 00 00 45 45 00.

RST | Step | Run | New | Load | Save

Executed 0x004F: MOV A,29H | Time: 00:00:00

```

003E MOV C,00H
0040 MOV 0AFH,C
0042 SJMP 0D3H
0044 ORL 89H,#20H
0047 MOV 8DH,#0FAH
004A MOV 98H,#50H
004D SETB 8EH
004F MOV A,29H
0051 JZ 0FCH
0053 SETB 01H
0055 JBC 0AFH,02H
0058 CLR 01H
005A MOV 99H,27H
005D MOV 29H,#00H
0060 MOV C,01H

```

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

System Clock (MHz) 11.0592 10 Update Freq.

SBUF

R/O	W/O	TH0	TL0	R7	0x00	B	0x00
0x00	0x00	0x00	0x01	R6	0x45	ACC	0x00
RXD	TXD	TMOD	0x20	R5	0x45	PSW	0x88
1	1	TCON	0xD0	R4	0x00	IP	0x00
SCON	0x50			R3	0x00	IE	0x82
				R2	0x00	PCON	0x00
pins	bits	TH1	TL1	R1	0x00	DPH	0x00
0xFF	0xFF	P3	0xFA	0xFF		DPL	0x01
0xFF	0xFF	P2				SP	0x41
0xFF	0xFF	P1					
0xFF	0xFF	P0					

PC 8051 0x000B PSW 1 0 0 0 1 0 0 0

Data Memory

addr	0x00	0x00	value
0	30	30	00 00 01 00 00 10 31 00 00 00 00 45 45 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 44 45 00 00 00 00 00 00 00
30	46	56	00 00 01 03 00 00 00 41 01 00 00 00 00 00
40	4F	00	00 00 00 00 30 30 00 00 01 00 00 00 00 00
50	14	00	00 00 00 09 30 30 00 00 01 00 00 00 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

Copyright ©2005-2022 James Rogers Remove All Breakpoints

RST Step Run New Load Save Copy Paste

Executed 0x0051: JZ 0FCH | Time: 8ms 902us - Instr

```

ORG 0000H
0000H LJMP 0075H
0003H RETI
ORG 000BH
000BH LJMP 007CH
000EH LJMP 0069H
0011H LJMP 000EH
0014H MOV 28H,#41H
0017H MOV A,29H
0019H JNZ 0FCH
001BH SETB 00H
001DH JBC 0AFH,02H
0020H CLR 00H
0022H MOV 27H,28H
0025H MOV 29H,#01H
0028H MOV A,#5AH
002AH CJNE A,28H,06H
002DH MOV R6,#41H
002FH MOV R7,#00H

```

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>

```