3.1 [2 points] Typescript for compilation

```
$ make clean
```

del *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym *.asm *.lk

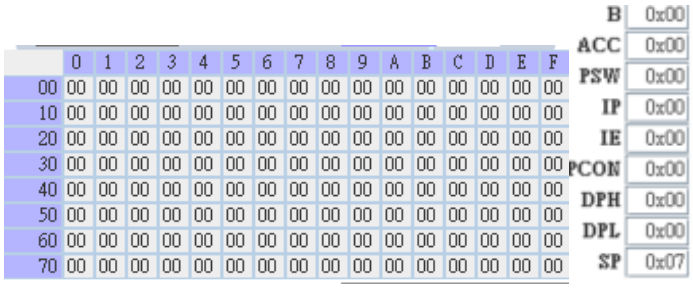
```
$ make
```

sdcc -c testpreempt.c
testpreempt.c:55: warning 158: overflow in implicit constant conversion
sdcc -c preemptive.c
preemptive.c:143: warning 85: in function ThreadCreate unreferenced function argument : 'fp'
preemptive.c:177: warning 158: overflow in implicit constant conversion
sdcc -o testpreempt.hex testpreempt.rel preemptive.rel

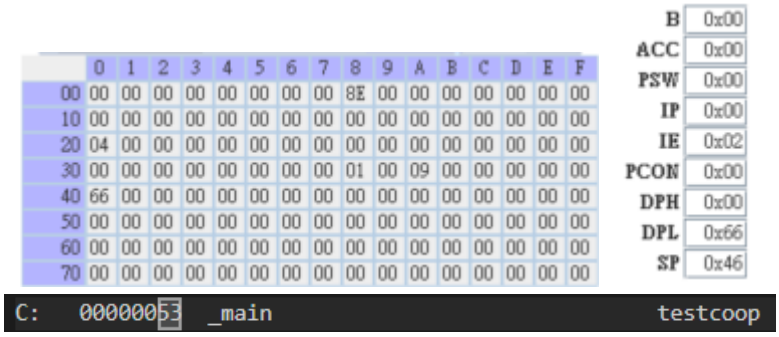
3.2 [18 points] Screenshots and explanation

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

before ThreadCreate(main)

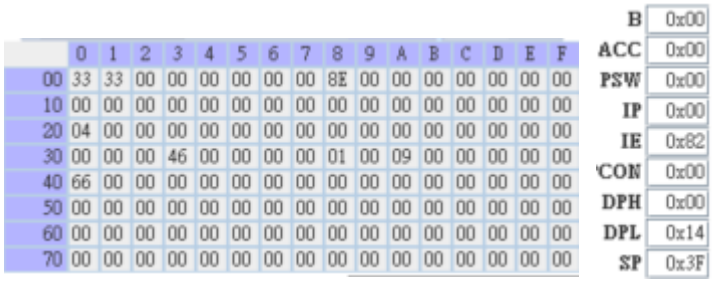


after ThreadCreate(main)



40 is set to 66 and SP is 0x46, so after RESTORESTATE is done, 66 will be pop and we will start to run on main. 3A, which is the temp that store the original SP is set to 09. So, after creating is done, SP can be set back to 09.

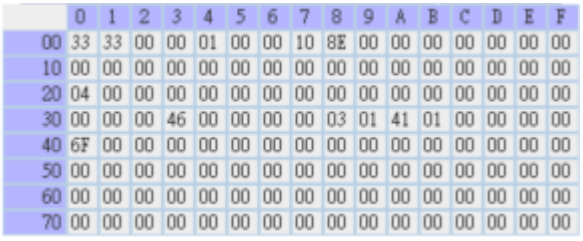
before ThreadCreate(Producer)



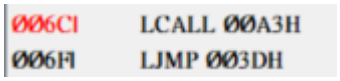
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	33	33	00	00	00	00	00	00	8E	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	04	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	46	00	00	00	00	01	00	09	00	00	00	00	00
40	66	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	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

DPL is set to 14 because it's calling ThreadCreate(Producer).

after ThreadCreate(Producer)



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	33	33	00	00	01	00	00	10	8E	00	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	04	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	00	00	00	46	00	00	00	00	03	01	41	01	00	00	00	00
40	6F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	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



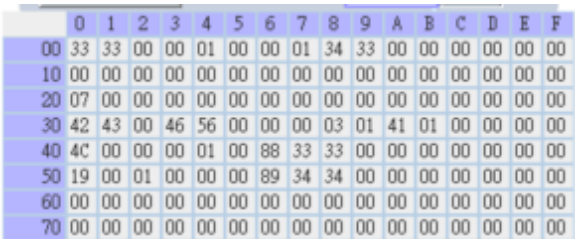
40 is set to 6F because 6F is the next instruction after ThreadCreate(Producer), SP is 0x4F because the new thread is found. 3A, which is the temp that store the original SP is set to 41. So, after creating is done, SP can be set back to 41. And 38 which is mask, is set to 03, indicating that the bit map is now 0011.

- Take one screenshot when the `Producer` is running. How do you know?

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	33	33	00	00	01	00	00	01	33	34	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	07	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	42	43	01	46	56	00	00	01	03	01	41	01	00	00	00	00
40	4C	00	00	00	01	00	88	33	33	00	00	00	00	00	00	00
50	17	00	01	00	00	00	89	34	34	00	00	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

By observing the 37, which is set to 01, I can tell that the current thread is now 1. And 32 is set to 1, which means the buffer is full, so it's polling.

- Take one screenshot when the `Consumer` is running. How do you know?



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	33	33	00	00	01	00	00	01	34	33	00	00	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	07	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	42	43	00	46	56	00	00	00	03	01	41	01	00	00	00	00
40	4C	00	00	00	01	00	88	33	33	00	00	00	00	00	00	00
50	19	00	01	00	00	00	89	34	34	00	00	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

Observing the 37, which is set to 00, I can tell that the current thread is now 0, which is `Consumer`. And 32 is set to 0, which means the buffer is empty, so it's polling.

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

Set break point here, which is the point where timer0 ISR. It triggers a interrupt every time PC reach here.