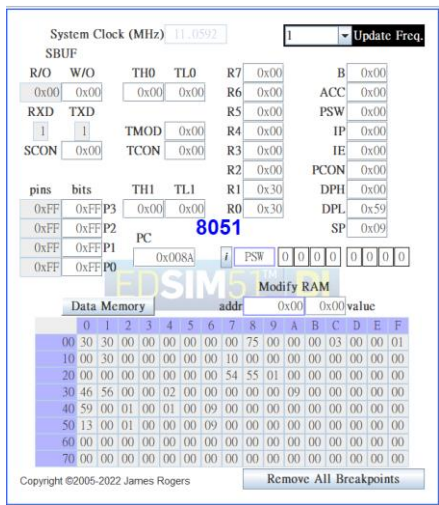
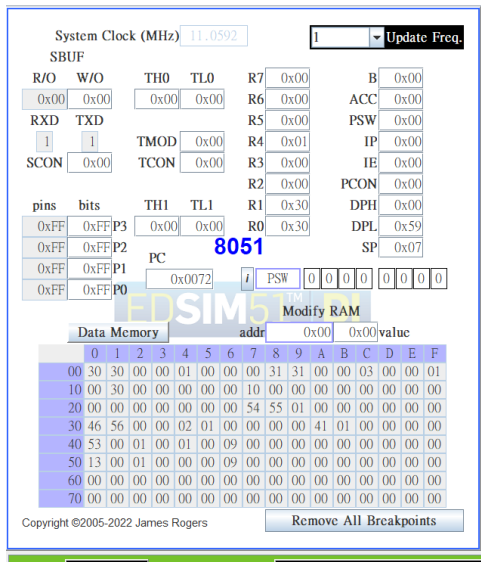


- Take one screenshot before each ThreadCreate call. Explain how the stack changes.

(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)

System Clock (MHz)

11.0592

100

Update Fre...

SBUF

R/O

W/O

TH0

TL0

R7

0x00

B

0x00

0x00

0x00

0x00

0x00

R6

0x00

ACC

0x00

RXD

TXD

TMOD

0x00

R5

0x00

PSW

0x00

1

1

R4

0x00

IP

0x00

SCON

0x00

TCON

0x00

R3

0x00

IE

0x00

PCON

0x00

pins

bits

TH1

TL1

R1

0x30

DPH

0x00

0xFF

0xFF

P3

PC

8051

R0

0x00

DPL

0x09

0xFF

0xFF

P2

0x005F

i

PSW

0

0

0

0

0

0

0

0

0

0

0xFF

0xFF

P1

RAM

SP

0x3F

0xFF

0xFF

P0

DATA MEMORY

addr

RAM

value

00

30

30

00

00

00

00

00

00

75

00

00

03

00

00

01

10

00

30

00

00

00

00

00

10

00

00

00

00

00

00

00

20

00

00

00

00

00

00

53

54

00

00

00

00

00

00

00

30

46

56

00

00

01

00

00

00

09

00

00

00

00

00

00

40

59

00

00

00

00

00

00

00

00

00

00

00

00

00

00

50

13

00

01

00

00

09

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

70

00

00

00

00

00

00

00

00

00

00

00

00

00

00

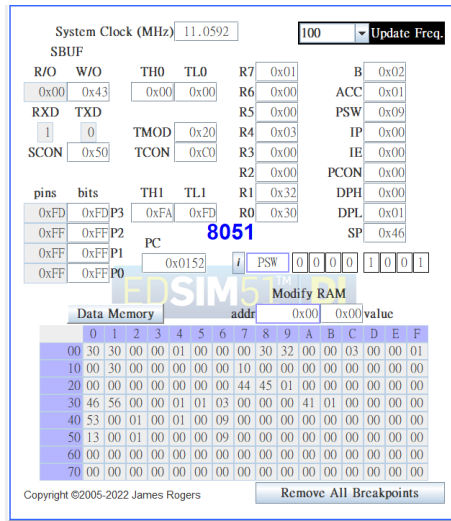
00

Copyright ©2005-2022 James Rogers

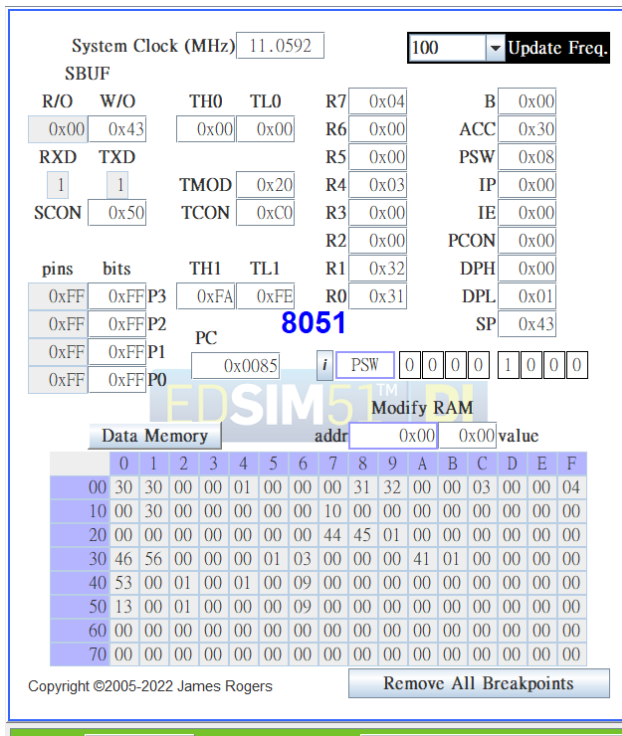
Remove All Breakpoints

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.



Typescript

Make clean other files, and make execute sdcc -c and execute sdcc -o to output a file testcoop.hex to be loaded into edsim.

```
PS C:\Users\josh9\OneDrive\桌面\大三上\OS\checkpoint1> make clean
del *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym
PS C:\Users\josh9\OneDrive\桌面\大三上\OS\checkpoint1> make
sdcc -c testcoop.c
testcoop.c:56: warning 158: overflow in implicit constant conversion
sdcc -c cooperative.c
cooperative.c:149: warning 85: in function ThreadCreate unreferenced function argument : 'fp'
cooperative.c:240: warning 158: overflow in implicit constant conversion
sdcc -o testcoop.hex testcoop.rel cooperative.rel
PS C:\Users\josh9\OneDrive\桌面\大三上\OS\checkpoint1>
```