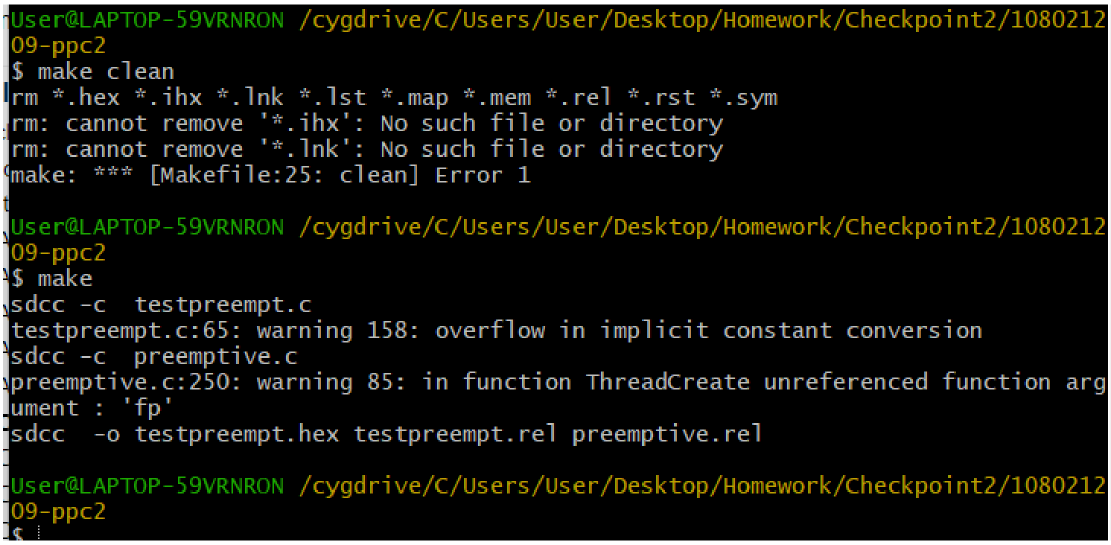
108021209 李思諭

3.1 Typescript for compilation :



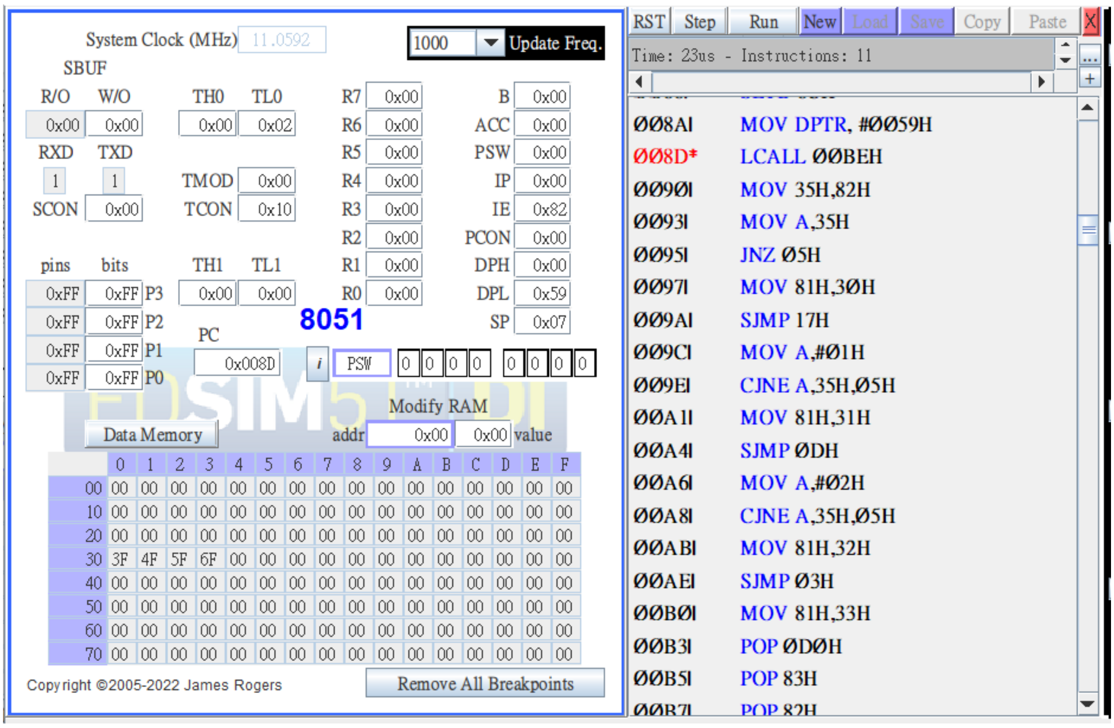
```
User@LAPTOP-59VRNRON /cygdrive/C/Users/User/Desktop/Homework/Checkpoint2/1080212
09-ppc2
$ make clean
rm *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym
rm: cannot remove '*.ihx': No such file or directory
rm: cannot remove '*.lnk': No such file or directory
make: *** [Makefile:25: clean] Error 1

User@LAPTOP-59VRNRON /cygdrive/C/Users/User/Desktop/Homework/Checkpoint2/1080212
09-ppc2
$ make
sdcc -c  testpreempt.c
testpreempt.c:65: warning 158: overflow in implicit constant conversion
sdcc -c  preemptive.c
preemptive.c:250: warning 85: in function ThreadCreate unreferenced function arg
ument : 'fp'
sdcc  -o testpreempt.hex testpreempt.rel preemptive.rel

User@LAPTOP-59VRNRON /cygdrive/C/Users/User/Desktop/Homework/Checkpoint2/1080212
09-ppc2
$
```

3.2 Screenshots and explanation

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

From testpreempt.map，we can know the function ThreadCreate is start from the line 00BE.

The ThreadCreate function in Bootstrap：
CurrentThreadID = ThreadCreate(main);



Time: 23us - Instructions: 11

System Clock (MHz) 11.0592 | 1000

SBUF R/O 0x00 W/O 0x00 | RXD 1 TXD 1 | SCON 0x00

TH0 0x00 TL0 0x02 | TMOD 0x00 | TCON 0x10 | TH1 0x00 TL1 0x00

R7 0x00 R6 0x00 R5 0x00 R4 0x00 R3 0x00 R2 0x00 R1 0x00 R0 0x00

B 0x00 ACC 0x00 PSW 0x00 IP 0x00 IE 0x82 PCON 0x00 DPH 0x00 DPL 0x59 SP 0x07

pins bits: 0xFF 0xFF P3 | 0xFF 0xFF P2 | 0xFF 0xFF P1 | 0xFF 0xFF P0

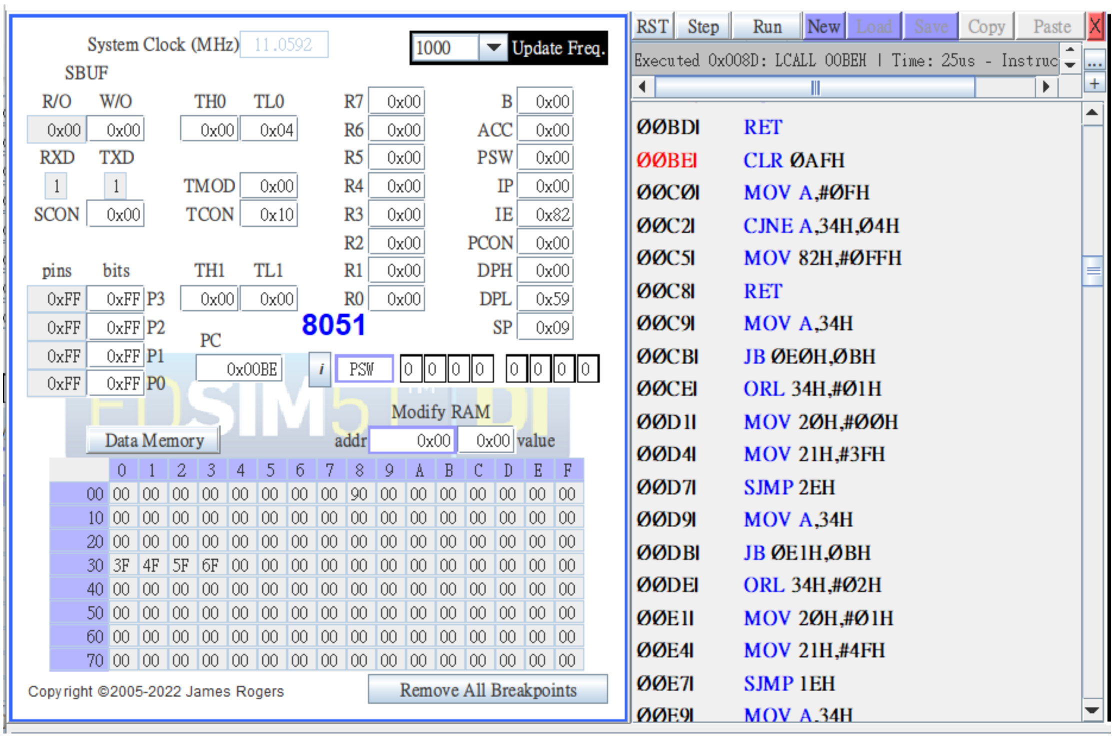
PC 0x008D | PSW 0 0 0 0 0 0 0 0

addr 0x00 value 0x00

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	00	00	00	00	00	00	00	00	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	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	3F	4F	5F	6F	00	00	00	00	00	00	00	00	00	00	00	00
40	00	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

Copyright ©2005-2022 James Rogers

```
ØØ8AI   MOV DPTR, #ØØ59H
ØØ8D*   LCALL ØØBEH
ØØ9ØI   MOV 35H,82H
ØØ93I   MOV A,35H
ØØ95I   JNZ Ø5H
ØØ97I   MOV 81H,3ØH
ØØ9AI   SJMP 17H
ØØ9CI   MOV A,#Ø1H
ØØ9EI   CJNE A,35H,Ø5H
ØØA1I   MOV 81H,31H
ØØA4I   SJMP ØDH
ØØA6I   MOV A,#Ø2H
ØØA8I   CJNE A,35H,Ø5H
ØØABI   MOV 81H,32H
ØØAEI   SJMP Ø3H
ØØBØI   MOV 81H,33H
ØØB3I   POP ØDØH
ØØB5I   POP 83H
ØØB7I   POP 82H
```



Executed 0x008D: LCALL 00BEH | Time: 25us - Instruc

System Clock (MHz) 11.0592 | 1000

SBUF R/O 0x00 W/O 0x00 | RXD 1 TXD 1 | SCON 0x00

TH0 0x00 TL0 0x04 | TMOD 0x00 | TCON 0x10 | TH1 0x00 TL1 0x00

R7 0x00 R6 0x00 R5 0x00 R4 0x00 R3 0x00 R2 0x00 R1 0x00 R0 0x00

B 0x00 ACC 0x00 PSW 0x00 IP 0x00 IE 0x82 PCON 0x00 DPH 0x00 DPL 0x59 SP 0x09

pins bits: 0xFF 0xFF P3 | 0xFF 0xFF P2 | 0xFF 0xFF P1 | 0xFF 0xFF P0

PC 0x00BE | PSW 0 0 0 0 0 0 0 0

addr 0x00 value 0x00

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	00	00	00	00	00	00	00	90	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	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	3F	4F	5F	6F	00	00	00	00	00	00	00	00	00	00	00	00
40	00	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

Copyright ©2005-2022 James Rogers

```
ØØBDI   RET
ØØBEI   CLR ØAFH
ØØCØI   MOV A,#ØFH
ØØC2I   CJNE A,34H,Ø4H
ØØC5I   MOV 82H,#ØFFH
ØØC8I   RET
ØØC9I   MOV A,34H
ØØCBI   JB ØEØH,ØBH
ØØCEI   ORL 34H,#Ø1H
ØØD1I   MOV 2ØH,#ØØH
ØØD4I   MOV 21H,#3FH
ØØD7I   SJMP 2EH
ØØD9I   MOV A,34H
ØØDBI   JB ØE1H,ØBH
ØØDEI   ORL 34H,#Ø2H
ØØE1I   MOV 2ØH,#Ø1H
ØØE4I   MOV 21H,#4FH
ØØE7I   SJMP 1EH
ØØE9I   MOV A,34H
```

The thread we created now is thread0. And the stack for thread0 is from 40H to 4FH.

In the line 010D, when executing PUSH 82H, it push DPL to 40H and the value in 40H becomes 59.

In the line 010F, when executing PUSH 83H, it push DPH to 41H and the value in 41H is 0.

In the line 0112, when executing PUSH 0E0H, it push ACC to 42H and the value in 42H is 0.

In the line 0114, when executing PUSH 0E0H, it push ACC to 43H and the value in 43H is 0.

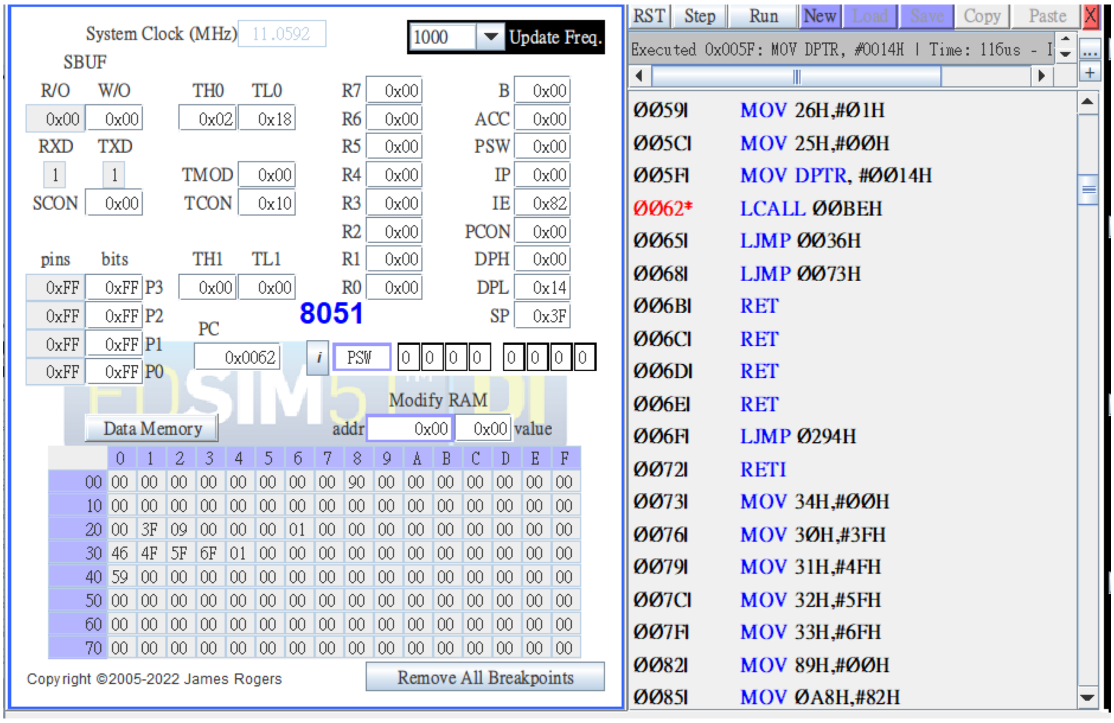
In the line 0116, when executing PUSH 0E0H, it push ACC to 44H and the value in 44H is 0.

In the line 0118, when executing PUSH 0E0H, it push ACC to 45H and the value in 45H is 0.

In the line 0122, when executing PUSH 0D0H, it push PSW to 46H and the value in 46H is 0.

The ThreadCreate function in main:

ThreadCreate(Producer);



System Clock (MHz) 11.0592

1000

SBUF

R/O	W/O	TH0	TL0
0x00	0x00	0x02	0x18

RXD 1 TXD 1

TMOD 0x00

SCON 0x00 TCON 0x10

R7	0x00
R6	0x00
R5	0x00
R4	0x00
R3	0x00
R2	0x00
R1	0x00
R0	0x00

B	0x00
ACC	0x00
PSW	0x00
IP	0x00
IE	0x82
PCON	0x00
DPH	0x00
DPL	0x14
SP	0x3F

pins	bits	
0xFF	0xFF	P3
0xFF	0xFF	P2
0xFF	0xFF	P1
0xFF	0xFF	P0

TH1	TL1
0x00	0x00

PC 0x0062

PSW 0 0 0 0 0 0 0 0

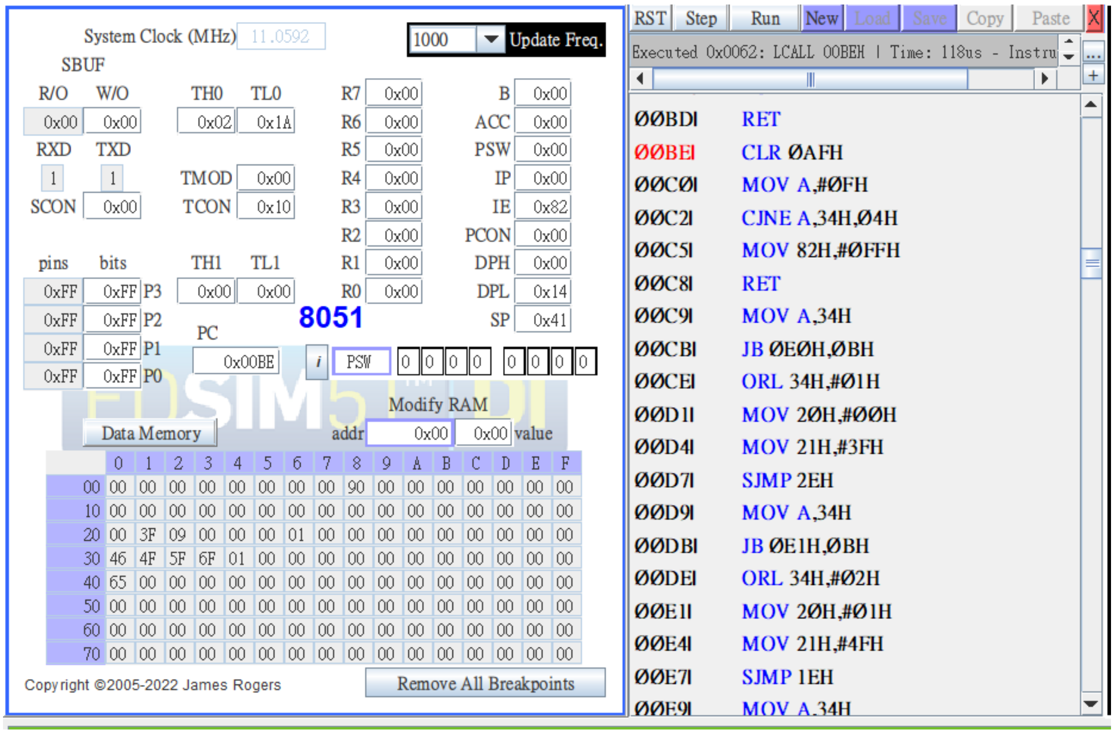
addr 0x00 0x00 value

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	00	00	00	00	00	00	00	90	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	00	3F	09	00	00	00	01	00	00	00	00	00	00	00	00	00
30	46	4F	5F	6F	01	00	00	00	00	00	00	00	00	00	00	00
40	59	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

Copyright ©2005-2022 James Rogers

```
Executed 0x005F: MOV DPTR, #0014H | Time: 116us - I
```

```
0059|      MOV 26H,#01H
005C|      MOV 25H,#00H
005F|      MOV DPTR, #0014H
0062*      LCALL 00BEH
0065|      LJMP 0036H
0068|      LJMP 0073H
006B|      RET
006C|      RET
006D|      RET
006E|      RET
006F|      LJMP 0294H
0072|      RETI
0073|      MOV 34H,#00H
0076|      MOV 30H,#3FH
0079|      MOV 31H,#4FH
007C|      MOV 32H,#5FH
007F|      MOV 33H,#6FH
0082|      MOV 89H,#00H
0085|      MOV 0A8H,#82H
```



System Clock (MHz) 11.0592

1000 Update Freq.

SBUF

R/O 0x00 W/O 0x00

RXD 1 TXD 1

SCON 0x00

TH0 0x02 TL0 0x1A

TMOD 0x00

TCON 0x10

TH1 0x00 TL1 0x00

pins	bits	
0xFF	0xFF	P3
0xFF	0xFF	P2
0xFF	0xFF	P1
0xFF	0xFF	P0

R7 0x00
R6 0x00
R5 0x00
R4 0x00
R3 0x00
R2 0x00
R1 0x00
R0 0x00

B 0x00
ACC 0x00
PSW 0x00
IP 0x00
IE 0x82
PCON 0x00
DPH 0x00
DPL 0x14
SP 0x41

PC 0x00BE

PSW 0 0 0 0 0 0 0 0

Modify RAM addr 0x00 0x00 value

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	00	00	00	00	00	00	00	90	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	00	3F	09	00	00	00	01	00	00	00	00	00	00	00	00	00
30	46	4F	5F	6F	01	00	00	00	00	00	00	00	00	00	00	00
40	65	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

Copyright ©2005-2022 James Rogers

```
Executed 0x0062: LCALL 00BEH | Time: 118us - Instru
```

```
ØØBD|    RET
ØØBE|    CLR ØAFH
ØØCØ|    MOV A,#ØFH
ØØC2|    CJNE A,34H,Ø4H
ØØC5|    MOV 82H,#ØFFH
ØØC8|    RET
ØØC9|    MOV A,34H
ØØCB|    JB ØEØH,ØBH
ØØCE|    ORL 34H,#Ø1H
ØØD1|    MOV 2ØH,#ØØH
ØØD4|    MOV 21H,#3FH
ØØD7|    SJMP 2EH
ØØD9|    MOV A,34H
ØØDB|    JB ØE1H,ØBH
ØØDE|    ORL 34H,#Ø2H
ØØE1|    MOV 2ØH,#Ø1H
ØØE4|    MOV 21H,#4FH
ØØE7|    SJMP 1EH
ØØE9|    MOV A,34H
```

The thread we created now is thread1. And the stack for thread1 is from 50H to 5FH.

In the line 010D, when executing PUSH 82H, it push DPL to 50H and the value in 50H becomes 0x14.

In the line 010F, when executing PUSH 83H, it push DPH to 51H and the value in 51H is 0.

In the line 0112, when executing PUSH 0E0H, it push ACC to 52H and the value in 52H is 0.

In the line 0114, when executing PUSH 0E0H, it push ACC to 53H and the value in 53H is 0.

In the line 0116, when executing PUSH 0E0H, it push ACC to 54H and the value in 54H is 0.

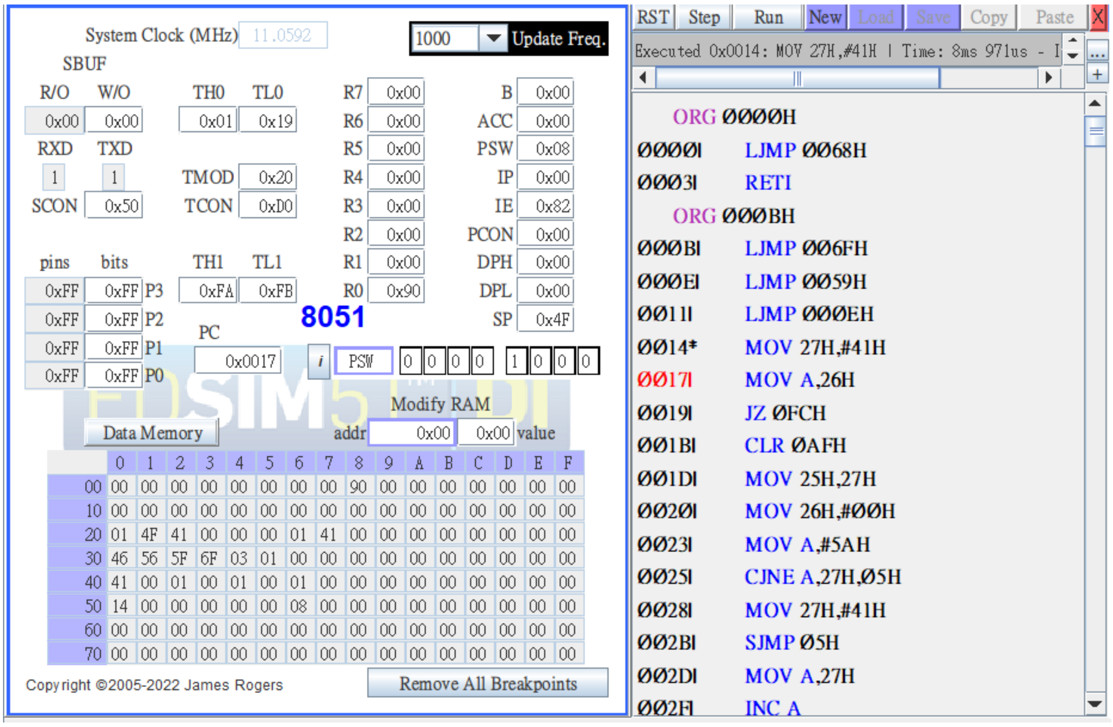
In the line 0118, when executing PUSH 0E0H, it push ACC to 55H and the value in 55H is 0.

In the line 0132, when executing PUSH 0D0H, it push PSW to 56H and the value in 56H is 0x08.

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

From testpreempt.map, we can know the function Producer is start from the line 0014.

In the picture, the line 0014 is being executed. We can know the Producer is running.



System Clock (MHz) 11.0592 1000 Update Freq.

```
SBUF
R/O   W/O     TH0   TL0     R7 0x00     B    0x00
0x00  0x00    0x01  0x19    R6 0x00     ACC  0x00
RXD   TXD                   R5 0x00     PSW  0x08
1     1       TMOD  0x20    R4 0x00     IP   0x00
SCON  0x50    TCON  0xD0    R3 0x00     IE   0x82
                            R2 0x00     PCON 0x00
pins  bits    TH1   TL1     R1 0x00     DPH  0x00
0xFF  0xFF P3 0xFA  0xFB    R0 0x90     DPL  0x00
0xFF  0xFF P2                           SP   0x4F
0xFF  0xFF P1 PC
0xFF  0xFF P0 0x0017   PSW 0 0 0 0 1 0 0 0
```

Modify RAM addr 0x00 0x00 value

```
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 00 00 00 00 00 00 00 00 90 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 01 4F 41 00 00 00 01 41 00 00 00 00 00 00 00 00
30 46 56 5F 6F 03 01 00 00 00 00 00 00 00 00 00 00
40 41 00 01 00 01 00 01 00 00 00 00 00 00 00 00 00
50 14 00 00 00 00 00 08 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
```

Copyright ©2005-2022 James Rogers

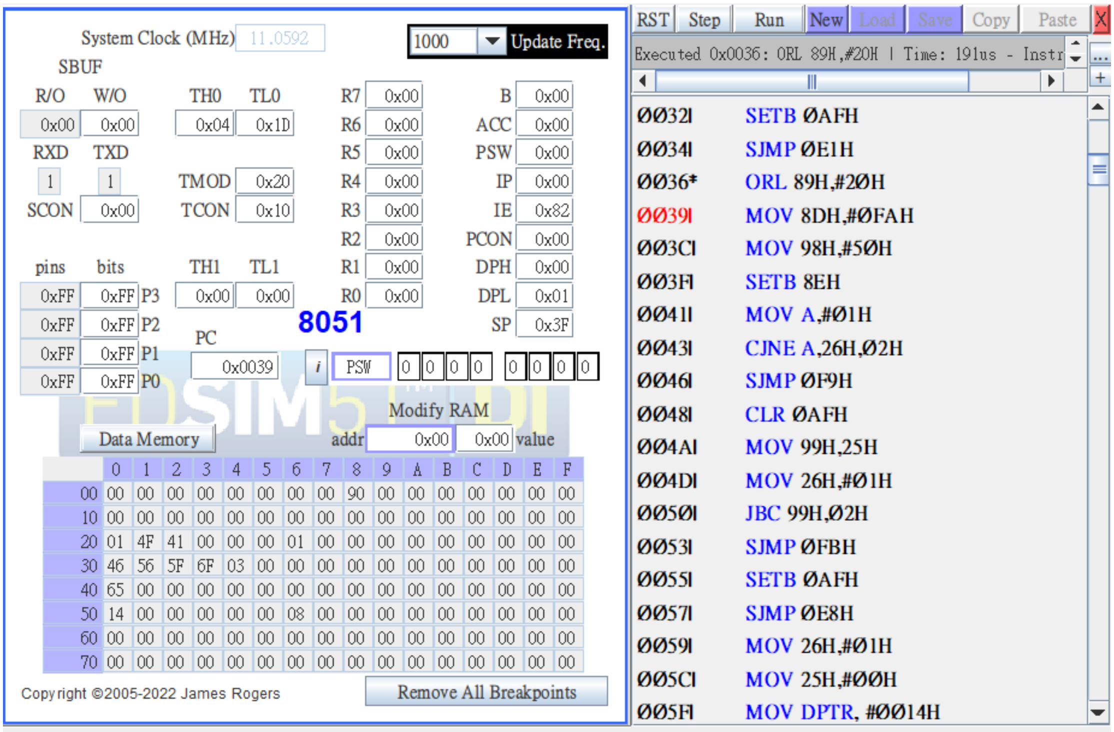
Executed 0x0014: MOV 27H,#41H | Time: 8ms 971us - I

```
      ORG ØØØØH
ØØØØ| LJMP ØØ68H
ØØØ3| RETI
      ORG ØØØBH
ØØØB| LJMP ØØ6FH
ØØØE| LJMP ØØ59H
ØØ11| LJMP ØØØEH
ØØ14* MOV 27H,#41H
ØØ17| MOV A,26H
ØØ19| JZ ØFCH
ØØ1B| CLR ØAFH
ØØ1D| MOV 25H,27H
ØØ2Ø| MOV 26H,#ØØH
ØØ23| MOV A,#5AH
ØØ25| CJNE A,27H,Ø5H
ØØ28| MOV 27H,#41H
ØØ2B| SJMP Ø5H
ØØ2D| MOV A,27H
ØØ2F| INC A
```

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

From testpreempt.map, we can know the function Consumer is start from the line 0036.

In the picture, the line 0036 is being executed. We can know the Consumer is running.



System Clock (MHz) 11.0592 1000 Update Freq.

```
SBUF
R/O   W/O     TH0   TL0     R7 0x00     B    0x00
0x00  0x00    0x04  0x1D    R6 0x00     ACC  0x00
RXD   TXD                   R5 0x00     PSW  0x00
1     1       TMOD  0x20    R4 0x00     IP   0x00
SCON  0x00    TCON  0x10    R3 0x00     IE   0x82
                            R2 0x00     PCON 0x00
pins  bits    TH1   TL1     R1 0x00     DPH  0x00
0xFF  0xFF P3 0x00  0x00    R0 0x00     DPL  0x01
0xFF  0xFF P2                           SP   0x3F
0xFF  0xFF P1 PC
0xFF  0xFF P0 0x0039   PSW 0 0 0 0 0 0 0 0
```

Modify RAM addr 0x00 0x00 value

```
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 00 00 00 00 00 00 00 00 90 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 01 4F 41 00 00 00 01 00 00 00 00 00 00 00 00 00
30 46 56 5F 6F 03 00 00 00 00 00 00 00 00 00 00 00
40 65 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
50 14 00 00 00 00 00 08 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
```

Copyright ©2005-2022 James Rogers

Executed 0x0036: ORL 89H,#20H | Time: 191us - Instr

```
ØØ32| SETB ØAFH
ØØ34| SJMP ØE1H
ØØ36* ORL 89H,#2ØH
ØØ39| MOV 8DH,#ØFAH
ØØ3C| MOV 98H,#5ØH
ØØ3F| SETB 8EH
ØØ41| MOV A,#Ø1H
ØØ43| CJNE A,26H,Ø2H
ØØ46| SJMP ØF9H
ØØ48| CLR ØAFH
ØØ4A| MOV 99H,25H
ØØ4D| MOV 26H,#Ø1H
ØØ5Ø| JBC 99H,Ø2H
ØØ53| SJMP ØFBH
ØØ55| SETB ØAFH
ØØ57| SJMP ØE8H
ØØ59| MOV 26H,#Ø1H
ØØ5C| MOV 25H,#ØØH
ØØ5F| MOV DPTR, #ØØ14H
```

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

From the testpreempt.map, we can know the function timer0_ISR is start from the line 006F, and the function myTimer0Handler is start from the line 0294.

I set breakpoint on the line 0294 which shows that myTimer0Handler is going to executing.

And the time that stop at are 8ms914us, 17ms803us, 26ms692us, 35ms582us, and 44ms469us, … .

Then we know that the interrupt is triggering almost every 9ms.

Thus, we know that the interrupt is triggering on a regular basis.