

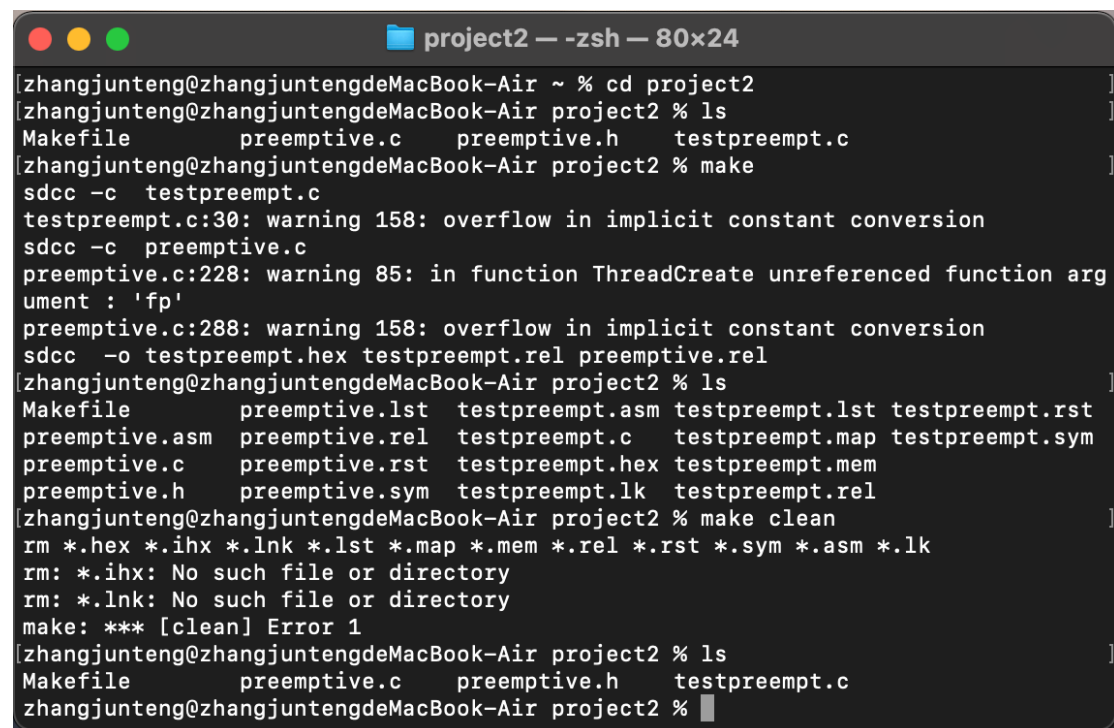
### 3. [20 points] Typescript and screenshots

#### 3.1 [2 points] Typescript for compilation

Turn in a typescript showing compilation of your code using a modified Makefile (same as for cooperative except the file names are changed to the preemptive version). You should use the following two commands (Note: \$ is the prompt displayed by the shell and is not part of the command that you type.) The first one deletes all the compiled files so it forces a rebuild if you have compiled before. The second one compiles it.

```
$ make clean
```

```
$ make
```



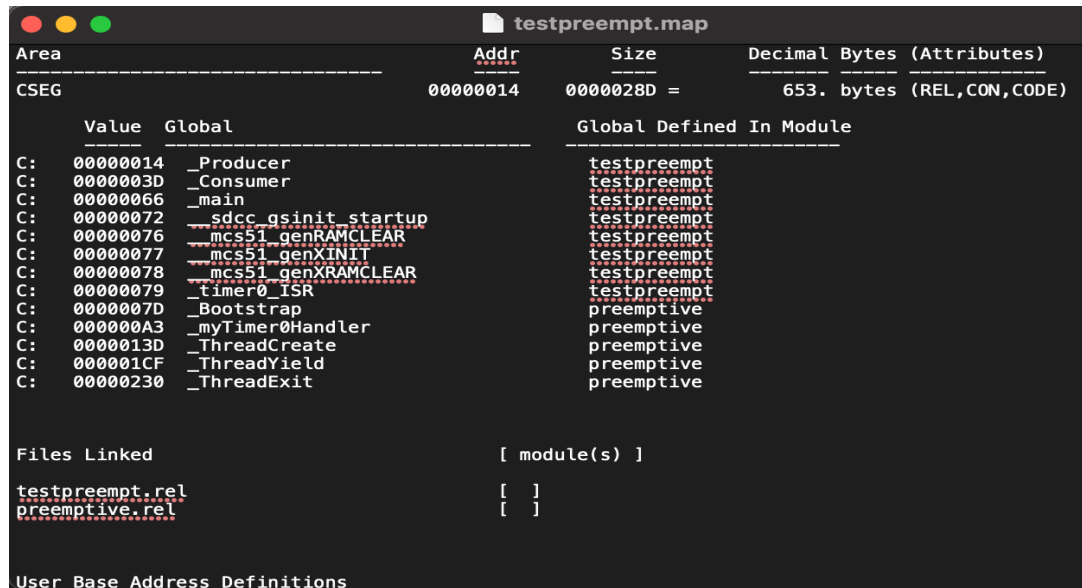
```
project2 — -zsh — 80x24
[zhangjunteng@zhangjuntengdeMacBook-Air ~ % cd project2
[zhangjunteng@zhangjuntengdeMacBook-Air project2 % ls
Makefile      preemptive.c  preemptive.h  testpreempt.c
[zhangjunteng@zhangjuntengdeMacBook-Air project2 % make
sdcc -c testpreempt.c
testpreempt.c:30: warning 158: overflow in implicit constant conversion
sdcc -c preemptive.c
preemptive.c:228: warning 85: in function ThreadCreate unreferenced function arg
ument : 'fp'
preemptive.c:288: warning 158: overflow in implicit constant conversion
sdcc -o testpreempt.hex testpreempt.rel preemptive.rel
[zhangjunteng@zhangjuntengdeMacBook-Air project2 % ls
Makefile      preemptive.lst  testpreempt.asm  testpreempt.lst  testpreempt.rst
preemptive.asm  preemptive.rel  testpreempt.c    testpreempt.map  testpreempt.sym
preemptive.c    preemptive.rst  testpreempt.hex  testpreempt.mem
preemptive.h    preemptive.sym  testpreempt.lk   testpreempt.rel
[zhangjunteng@zhangjuntengdeMacBook-Air project2 % make clean
rm *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym *.asm *.lk
rm: *.ihx: No such file or directory
rm: *.lnk: No such file or directory
make: *** [clean] Error 1
[zhangjunteng@zhangjuntengdeMacBook-Air project2 % ls
Makefile      preemptive.c  preemptive.h  testpreempt.c
[zhangjunteng@zhangjuntengdeMacBook-Air project2 % ]
```

It should show actual compilation, warning, or error messages. Note that not all warnings are errors. The compiler should generate several `testpreempt.*` files with different extensions:

- the `.hex` file can be opened directly in EdSim51
- the `.map` file shows the mapping of the symbols to their addresses after linking

### 3.2 [18 points] Screenshots and explanation

Look up the addresses for your symbols (i.e., functions, variables, etc) in the file `testpreempt.map`. Set one or more breakpoints in EdSim51's assembly code window after you have assembled it.



Area	Addr	Size	Decimal	Bytes	(Attributes)
CSEG	00000014	0000028D =	653.	bytes	(REL,CON,CODE)
Value	Global	Global	Defined	In	Module
C:	00000014	_Producer	testpreempt		
C:	0000003D	_Consumer	testpreempt		
C:	00000066	_main	testpreempt		
C:	00000072	_sdcc_gsinit_startup	testpreempt		
C:	00000076	_mcs51_genRAMCLEAR	testpreempt		
C:	00000077	_mcs51_genXINIT	testpreempt		
C:	00000078	_mcs51_genXRAMCLEAR	testpreempt		
C:	00000079	_timer0_ISR	testpreempt		
C:	0000007D	_Bootstrap	preemptive		
C:	000000A3	_myTimer0Handler	preemptive		
C:	0000013D	_ThreadCreate	preemptive		
C:	000001CF	_ThreadYield	preemptive		
C:	00000230	_ThreadExit	preemptive		

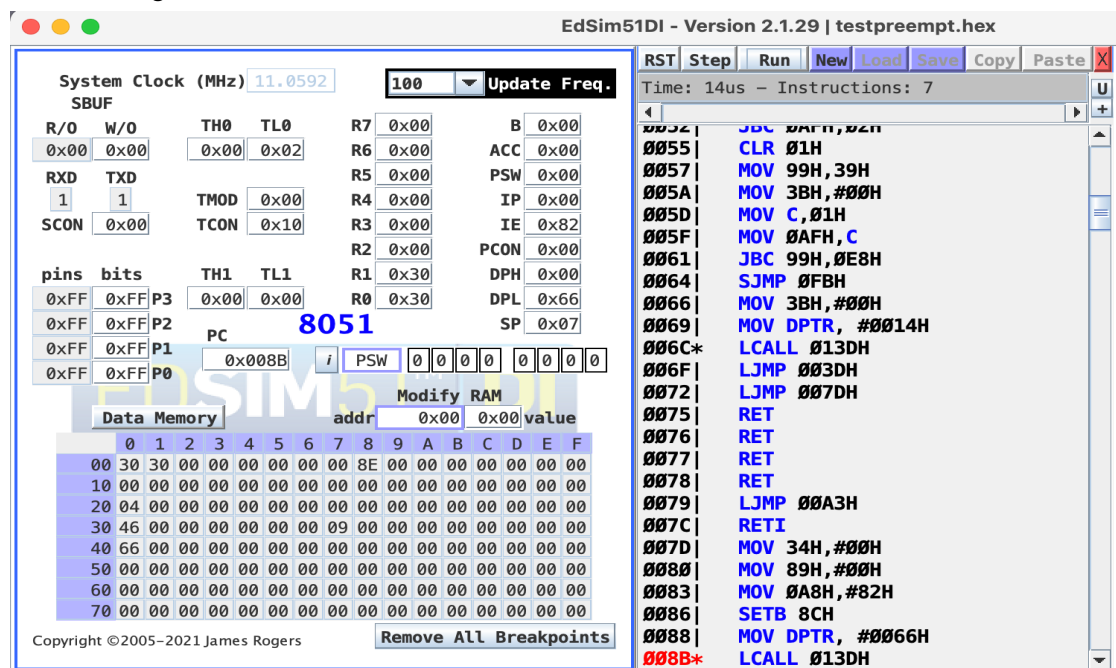
Files Linked [ module(s) ]

testpreempt.rel [ ]

preemptive.rel [ ]

User Base Address Definitions

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



EdSim51DI - Version 2.1.29 | testpreempt.hex

System Clock (MHz) 11.0592 100 Update Freq.

SBUF R/O W/O TH0 TL0 R7 0x00 B 0x00

RXD TXD 0x00 0x02 R6 0x00 ACC 0x00

1 1 TMOD 0x00 R5 0x00 PSW 0x00

SCON 0x00 TCON 0x10 R4 0x00 IP 0x00

pins bits TH1 TL1 R3 0x00 IE 0x82

0xFF 0xFF P3 0x00 0x00 R2 0x00 PCON 0x00

0xFF 0xFF P2 PC 8051 R1 0x30 DPH 0x00

0xFF 0xFF P1 0x00 0x00 R0 0x30 DPL 0x66

0xFF 0xFF P0 0x00 0x00 SP 0x07

Data Memory addr 0x00 0x00 value

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

00 30 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

20 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00

30 46 00 00 00 00 00 00 00 00 00 00 00 00 00 00

40 66 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

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 © 2005–2021 James Rogers Remove All Breakpoints

RST Step Run New Load Save Copy Paste

Time: 14us - Instructions: 7

0052 JBC 0AFH, 02H

0055 CLR 01H

0057 MOV 99H, 39H

005A MOV 3BH, #00H

005D MOV C, 01H

005F MOV 0AFH, C

0061 JBC 99H, 0E8H

0064 SJMP 0FBH

0066 MOV 3BH, #00H

0069 MOV DPTR, #0014H

006C\* LCALL 013DH

006F LJMP 003DH

0072 LJMP 007DH

0075 RET

0076 RET

0077 RET

0078 RET

0079 LJMP 00A3H

007C RETI

007D MOV 34H, #00H

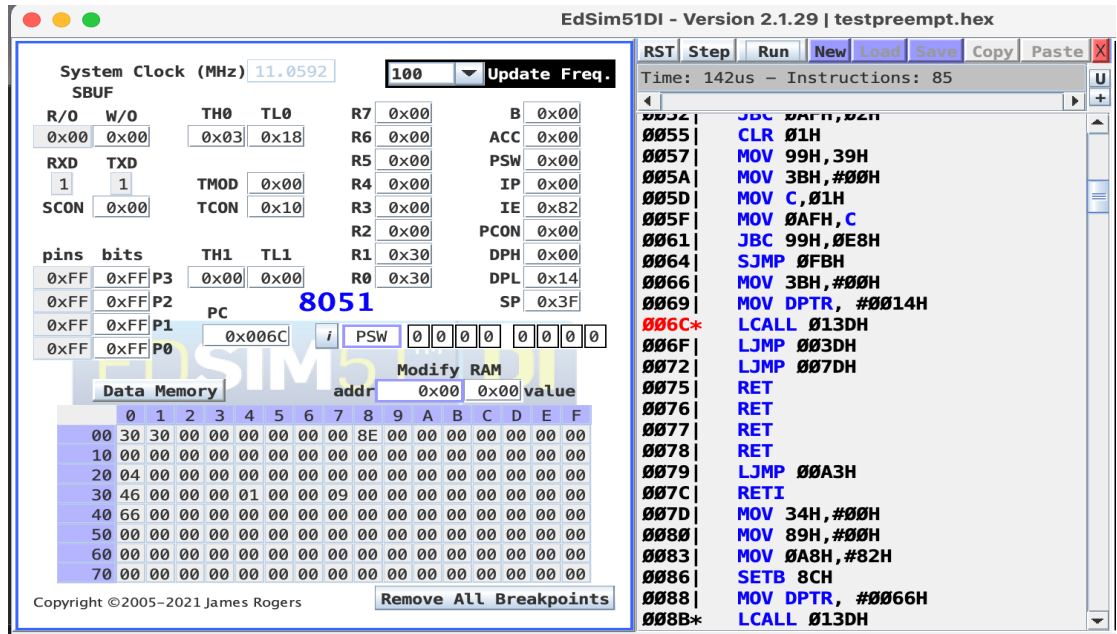
0080 MOV 89H, #00H

0083 MOV 0A8H, #82H

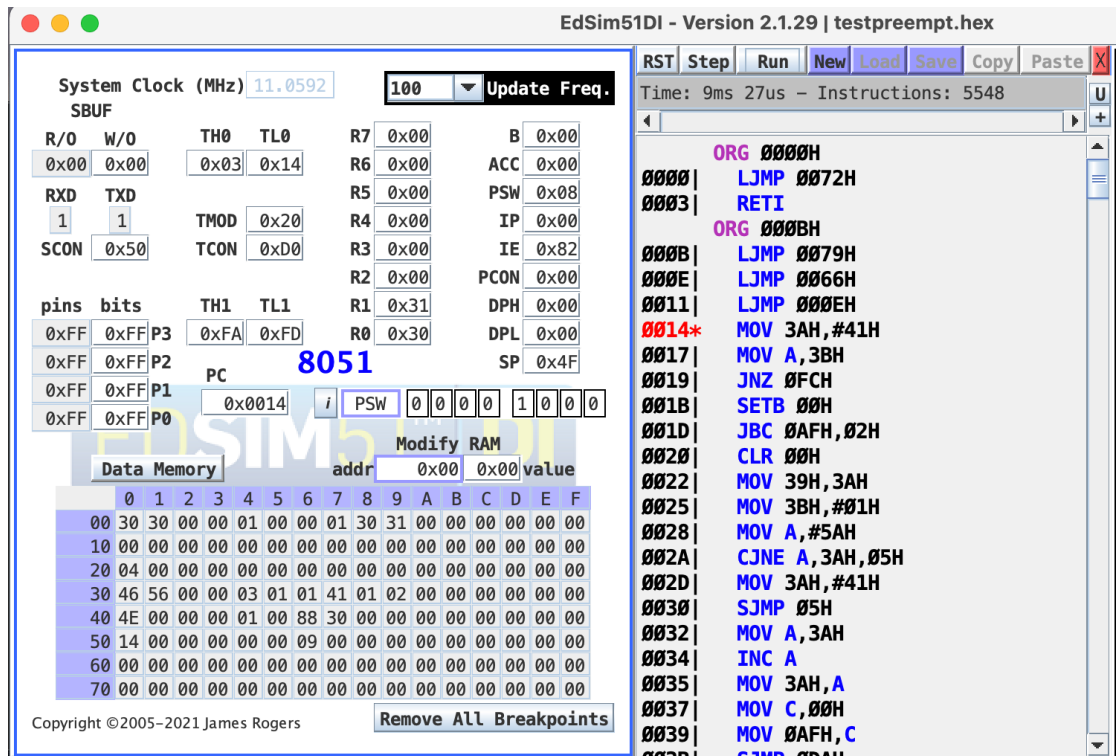
0086 SETB 8CH

0088 MOV DPTR, #0066H

008B\* LCALL 013DH



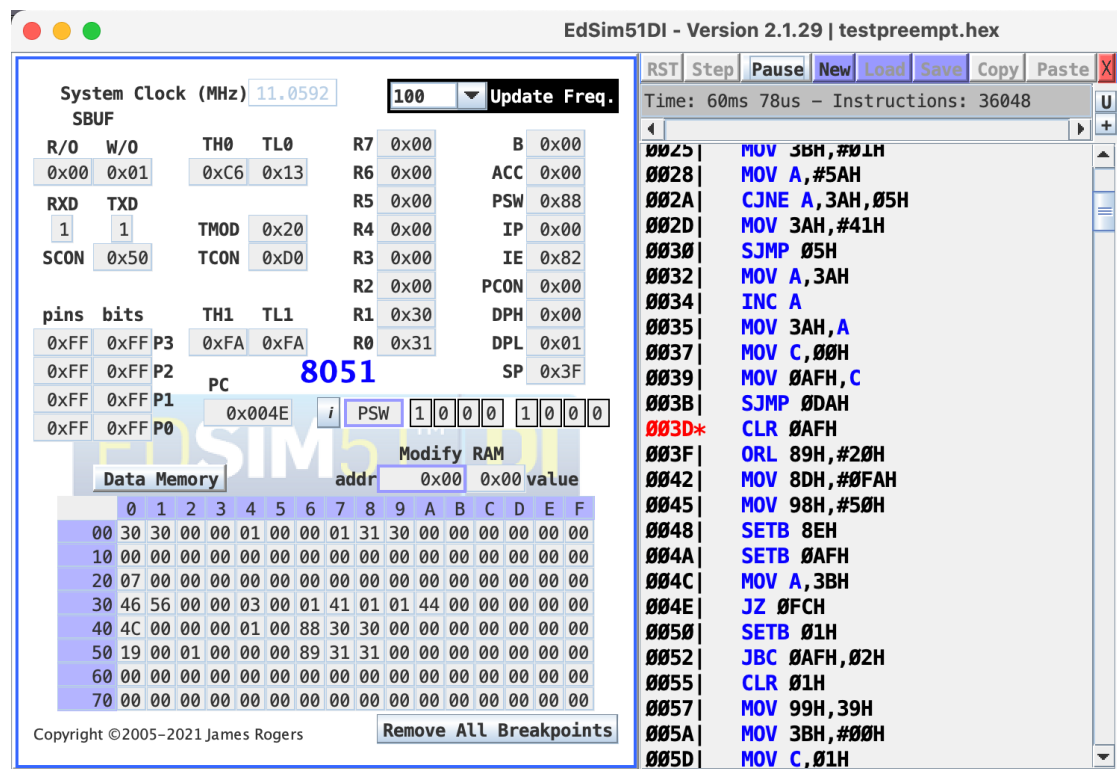
- Stack was set up by ThreadCreate(main), RESTORESTATE sets SP to the savedSP for stack-0, restore its PSW (which selects register bank 0), DPTR, B, ACC using stack value. Stack 0 now has the return address of main.
- Take one screenshot when the Producer is running. How do you know?



- Stack was set up by ThreadCreate(Producer), and PC point to ThreadCreate.

Because we want to run producer. That means the address of producer is passed as a parameter, and that is in DPL and DPH as the address that you want the new thread to run.

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



- I set a breakpoint for Consumer() .By observing the stack address, we knew that while calling the Consumer(). The stack space for stack1 which is pointed to 5\_H was changed and we knew that Consumer is running.
- How can you tell that the interrupt is triggering on a regular basis?
- When an interrupt is triggered, the following actions are taken automatically by the microcontroller: The current Program Counter is saved on the stack, low-byte first. Interrupts of the same and lower priority are blocked. In the case of Timer and External interrupts, the corresponding interrupt flag is cleared.