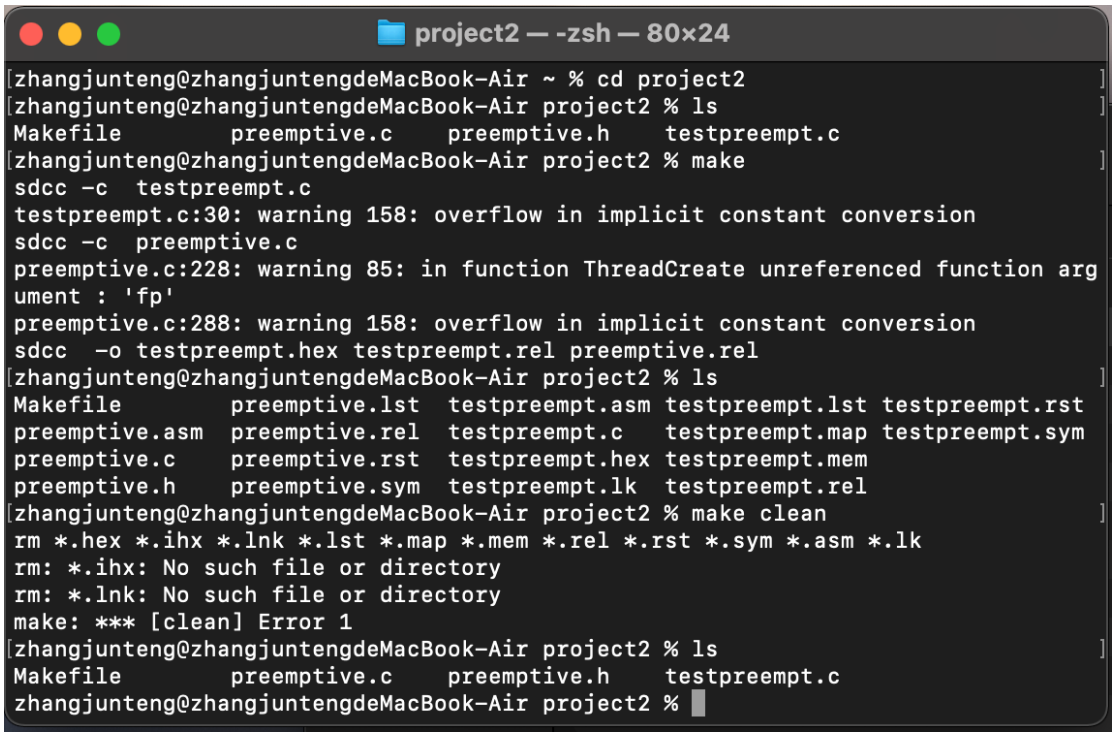
3. [20 points] Typescript and screenshots

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
```



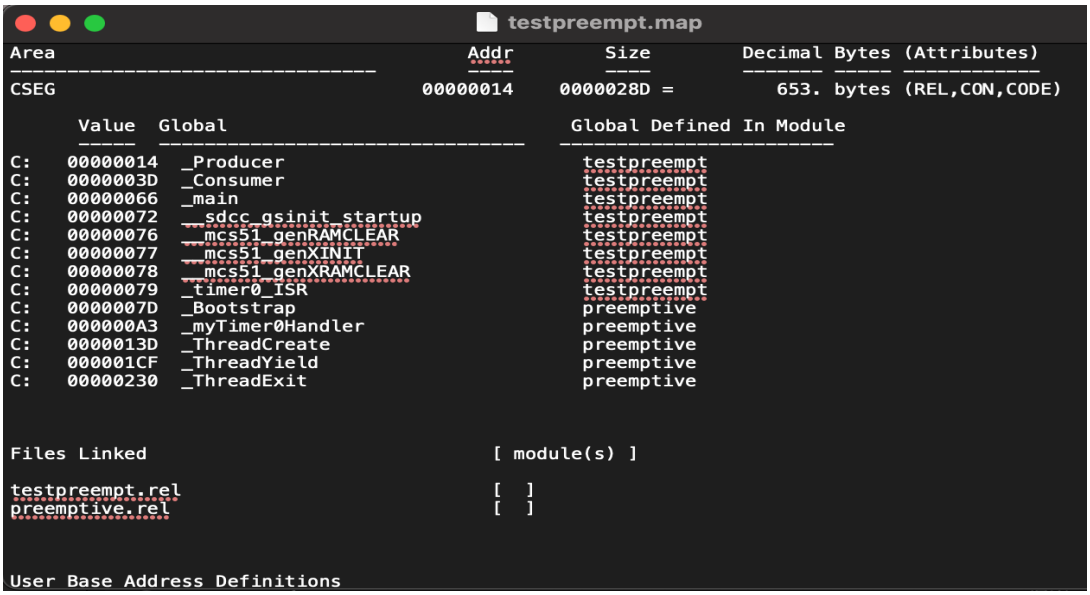
```
project2 — -zsh — 80×24
zhangjunteng@zhangjuntengdeMacBook-Air ~ % cd project2
zhangjunteng@zhangjuntengdeMacBook-Air project2 % ls
Makefile        preemptive.c    preemptive.h    testpreempt.c
zhangjunteng@zhangjuntengdeMacBook-Air project2 % make
sdcc -c  testpreempt.c
testpreempt.c:30: warning 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

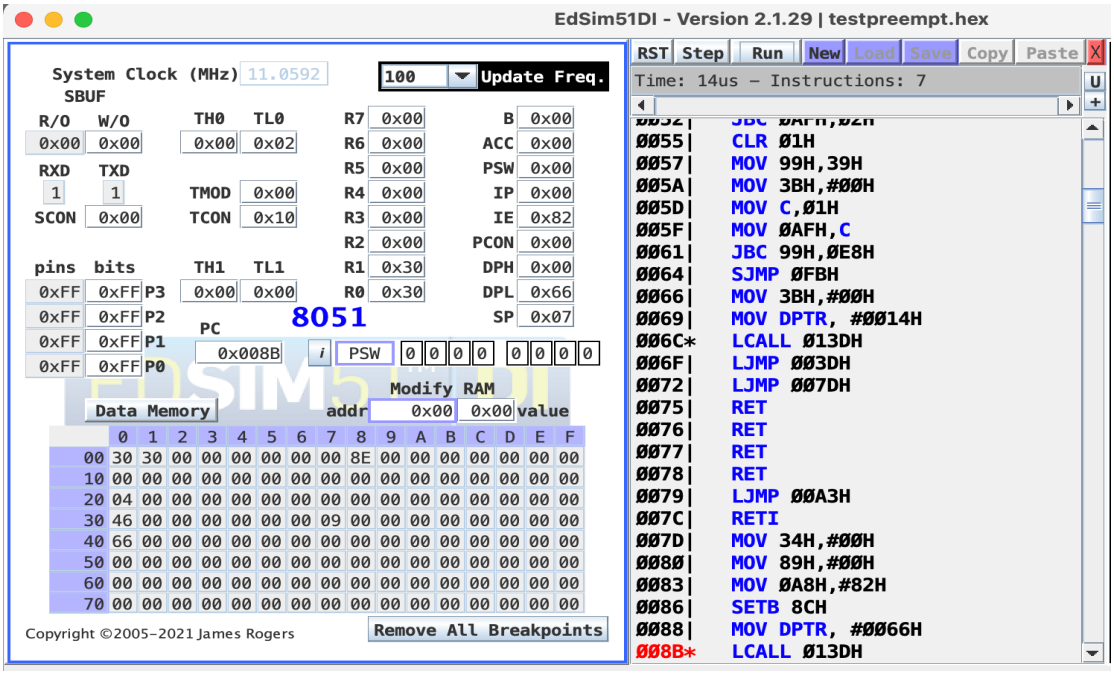
3.2 [18 points] Screenshots and explanation

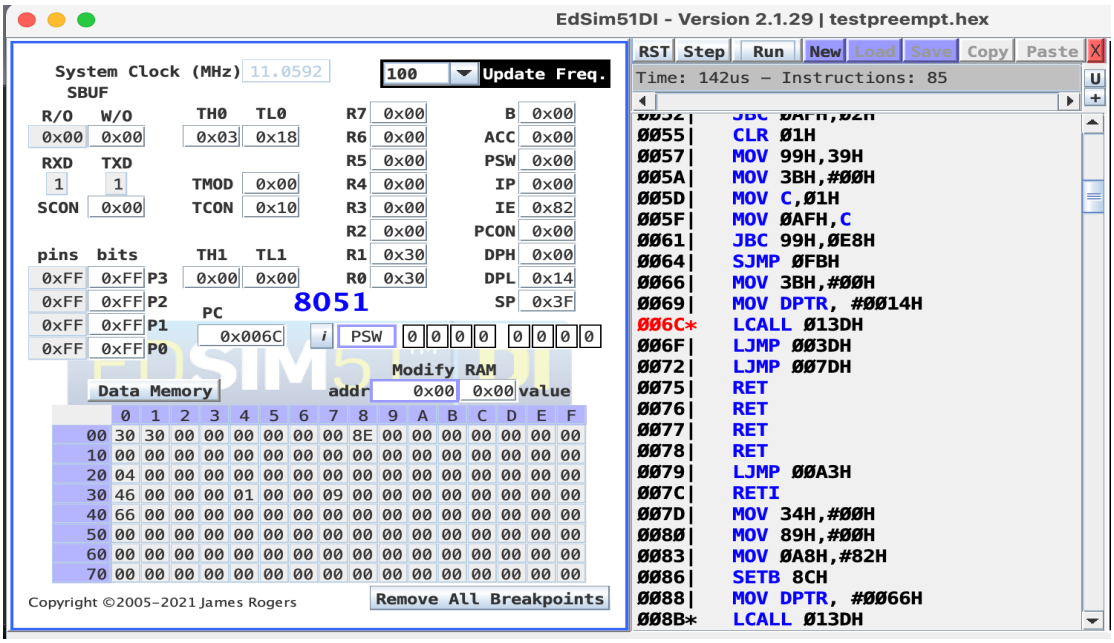
Look up the addresses for your symbols (i.e., functions, variables, etc) in the file `testpreempt.map`. Set one or more breakpoints in EdSim51's assembly code window after you have assembled it.



```
testpreempt.map
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
```

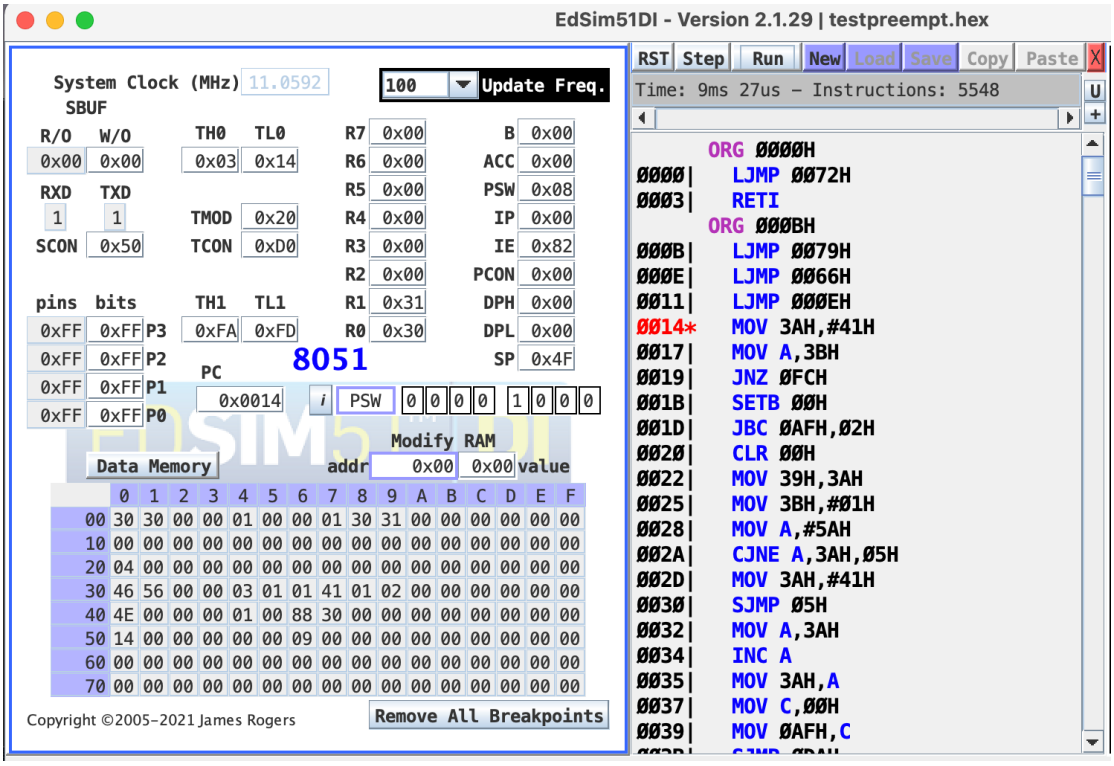
- Take one screenshot before each `ThreadCreate` call. Explain how the stack changes.





- 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?

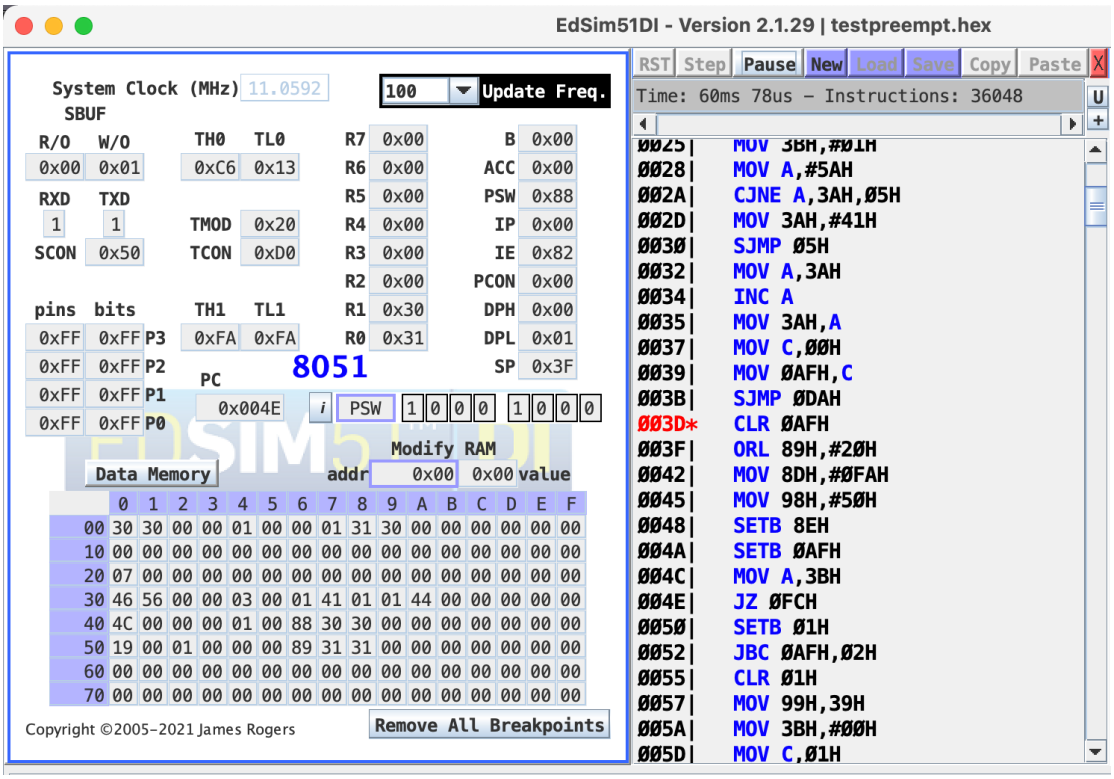


-

- 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.