

# Programming Project Checkpoint 1 Report

## Typescript for Compiling

```
wilbertallen@MacBook-Air-2016 ppc2 % make clean
rm *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym
rm: *.ihx: No such file or directory
rm: *.lnk: No such file or directory
make: *** [clean] Error 1
wilbertallen@MacBook-Air-2016 ppc2 % make
sdcc -c testpreempt.c
sdcc -c preemptive.c
preemptive.c:220: warning 85: in function ThreadCreate unreferenced function argument : 'fp'
preemptive.c:285: warning 283: function declarator with no prototype
sdcc -o testpreempt.hex testpreempt.rel preemptive.rel
wilbertallen@MacBook-Air-2016 ppc2 %
```

Fig.1 Typescript for compiling using the given makefile

## Before Each ThreadCreate Call

Based on my understanding, ThreadCreate is called 2 times, one is for ThreadCreate(main), and the other is for ThreadCreate(Producer)

ThreadCreate(main);

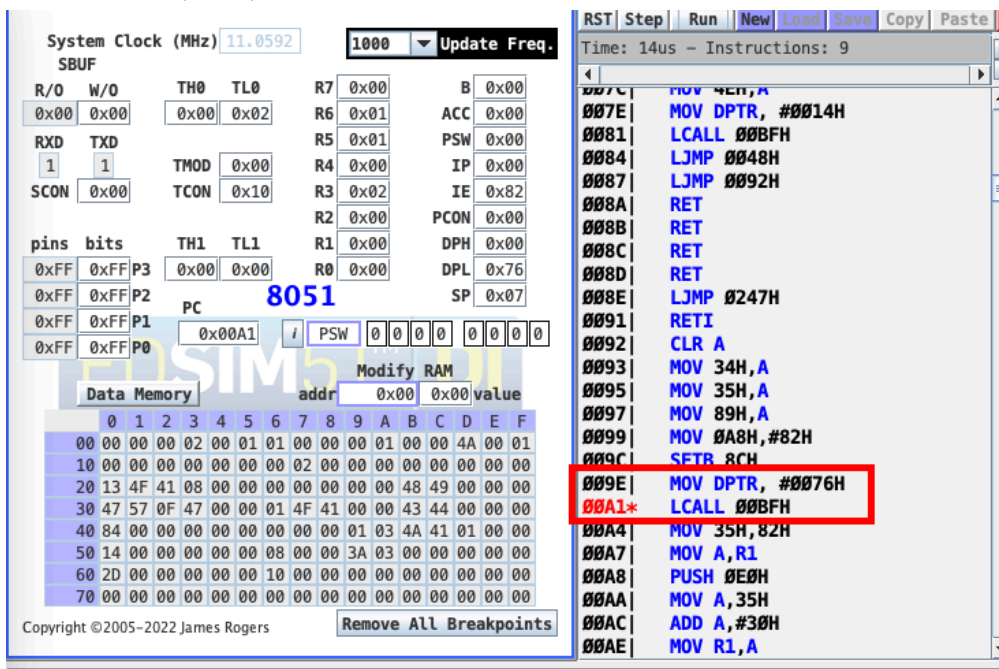


Fig2.1 Screenshot before ThreadCreate(main)

When the breakpoint reached LCALL, we can see that the address of main which is 0x76 can be seen in the DPL. It will then be pushed into SP, which will change 0x07 to 0x09.

ThreadCreate(Producer);

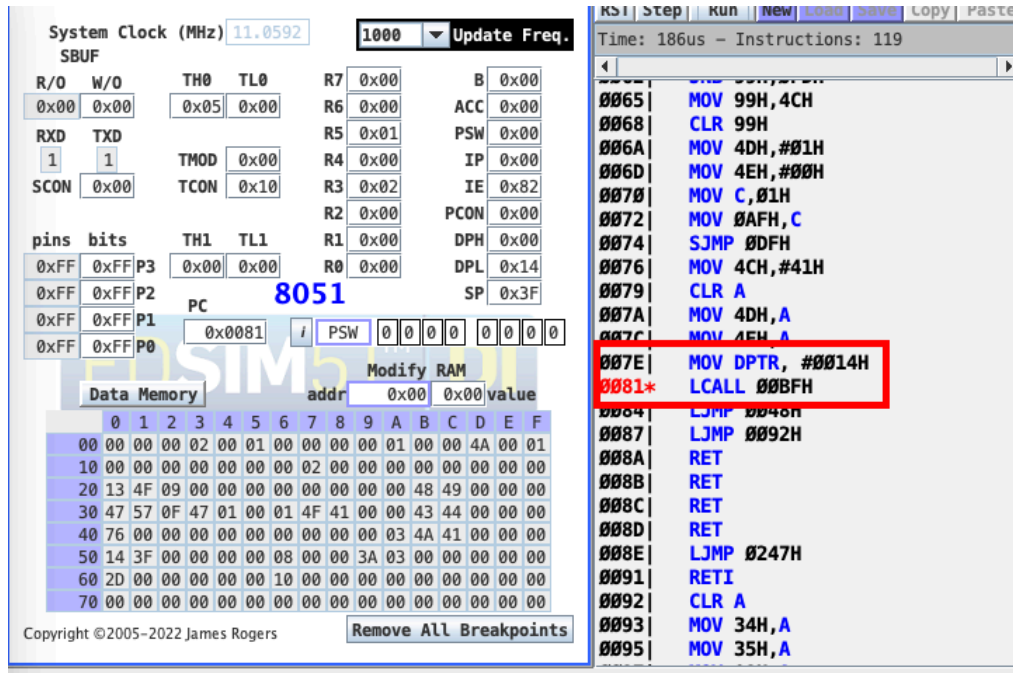


Fig.2.2 Screenshot before ThreadCreate(Producer)

Same goes for ThreadCreate(Producer). At the breakpoint, the address of Producer which is 0x14 is pushed into the Stack Pointer, which will result in a change in the SP from 0x3F to 0x41.

Value	Global	Global Defined In Module
C: 00000014	_Producer	testpreempt
C: 00000048	_Consumer	testpreempt
C: 00000076	_main	testpreempt
C: 00000087	__sdcc_gsinit_startup	testpreempt
C: 0000008B	__mcs51_genRAMCLEAR	testpreempt
C: 0000008C	__mcs51_genXINIT	testpreempt
C: 0000008D	__mcs51_genXRAMCLEAR	testpreempt
C: 0000008E	_timer0_ISR	testpreempt
C: 00000092	_Bootstrap	preemptive
C: 000000BF	_ThreadCreate	preemptive
C: 00000182	_ThreadYield	preemptive
C: 00000200	_ThreadExit	preemptive
C: 00000247	_myTimer0Handler	preemptive
C: 000002C6	__moduint	_moduint
C: 00000313	__modsint	_modsint

ASxxx Linker V03.00/V05.40 + sld, page 14.

Fig.2.3 Function addresses value

## Running Producer

addr	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
00	00	00	00	00	02	00	01	01	00	02	00	00	00	00	4F	00
10	00	00	00	00	00	00	00	00	02	00	00	00	00	00	00	00
20	13	4F	41	01	00	00	00	00	00	00	00	48	49	00	00	00
30	47	57	0F	47	03	00	01	4F	41	00	00	43	44	00	00	00
40	59	00	01	00	01	00	81	00	00	3A	03	4F	4E	01	00	00
50	1B	00	00	00	00	00	88	00	00	3A	03	00	00	00	00	00
60	2D	00	00	00	00	00	10	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

Remove All Breakpoints

We can see that the Producer function is running by observing the addresses of *shared\_buff* and *next\_buff*. In my implementation, I store *shared\_buff* on 0x4C and *next\_buff* on 0x4B. At each iteration, the value from 0x4B will be copied to 0x4C. At the above example, we know that the current value of the shared buffer is 4E which is HEX for the ASCII character 'N', and the next buffer has a value of 4F, which translates to 'O'.

Link of additional producer video : [Drive](#)

## Running Consumer

System Clock (MHz) 11.0592

1000 Update Freq.

SBUF

R/O	W/O	TH0	TL0	R7	B
0x00	0x4E	0xC5	0x0A	0x00	0x00

RXD TXD

RXD	TXD	TMOD	TCON	R6	ACC
1	1	0x20	0xD0	0x01	0x01

SCON

SCON	bits	TH1	TL1	R5	PSW
0x52	0xFF	0xFA	0xFD	0x01	0x81

pins

P3	P2	P1	P0	R4	IP
0xFF	0xFF	0xFF	0xFF	0x00	0x00

PC 8051

PC	PSW	R3	IE
0x0057	10000001	0x02	0x82

PCON

PCON	DPH	DPL	SP	R2	PCON
0x00	0x00	0x01	0x3F	0x00	0x00

Modify RAM

addr	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
00	00	00	00	02	00	01	01	00	02	00	00	00	00	4F	00	01
10	00	00	00	00	00	00	00	00	02	00	00	00	00	00	00	00
20	13	4F	41	01	00	00	00	00	00	00	00	48	49	00	00	00
30	47	57	0F	47	03	00	01	4F	41	00	00	43	44	00	00	00
40	59	00	01	00	01	00	81	00	00	3A	03	4F	4E	01	00	00
50	1B	00	00	00	00	00	88	00	00	3A	03	00	00	00	00	00
60	2D	00	00	00	00	00	10	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

Remove All Breakpoints

We can know Consumer is running by observing the SBUF. SBUF is currently writing out 0x4E to the received\_data part on Edsim, from the shared\_buffer.

## Interrupt

B	0x00	0087	LJMP 0092H
ACC	0x01	008A	RET
PSW	0x81	008B	RET
IP	0x00	008C	RET
IE	0x82	008D	RET
PCON	0x00	008E*	LJMP 0247H
DPH	0x00	0091	RETI
DPL	0x01	0092	CLR A
SP	0x41	0093	MOV 34H,A
		0095	MOV 35H,A

When the timer interrupt (0x8E) is triggered, the code jumps to 0x247, which is the address for the function myTimer0Handler(), described by the LJMP above.

B	0x00	0242	MOV C,03H
ACC	0x01	0244	MOV 0AFH,C
PSW	0x81	0246	RET
IP	0x00	0247	SETB 04H
IE	0x02	0249	JBC 0AFH,02H
PCON	0x00	024C	CLR 04H
DPH	0x00	024E	PUSH 0E0H
DPL	0x01	0250	PUSH 0F0H
SP	0x41	0252	PUSH 82H

A few steps later, we can see that an interrupt has happened by observing the IE that changes from 0x82 to 0x02. We can tell that the interrupt is triggering on a regular basis by simply observing the changes in the IE value.