

## Setup

Software	Purpose
Cygwin (3.5.4)	For Unix-like Environment
SDCC (4.4.0)	compiler suite that targets the Intel MCS51 based microprocessors
Notepad++ (8.7.1)	Write and edit .c files
EdSim51DI (2.1.36)	Simulator for 8051

Table 1: Setup describing Software with respective version and purpose.

## Compiling Makefile

Running the following commands in Cygwin (3.5.4)

```
$ make clean
```

```
$ make
```

as shown in Fig. 1. *make clean* will clear the files generated from previous execution (if any) and then *make* command will create new require file as per the code written in .c files. Table 2 shows the result of respective make command.

```

Snehit@LAPTOP-V8N83JAP /cygdrive/d/PhD/NTHU/OS/2024/Project/cp1/test_3.1
$ make clean
rm *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym *.asm *.lk
rm: cannot remove '*.ihx': No such file or directory
rm: cannot remove '*.lnk': No such file or directory
make: *** [Makefile:25: clean] Error 1

Snehit@LAPTOP-V8N83JAP /cygdrive/d/PhD/NTHU/OS/2024/Project/cp1/test_3.1
$ make
sdcc -c testcoop.c
sdcc -c cooperative.c
sdcc -o testcoop.hex testcoop.rel cooperative.rel

Snehit@LAPTOP-V8N83JAP /cygdrive/d/PhD/NTHU/OS/2024/Project/cp1/test_3.1
$

```

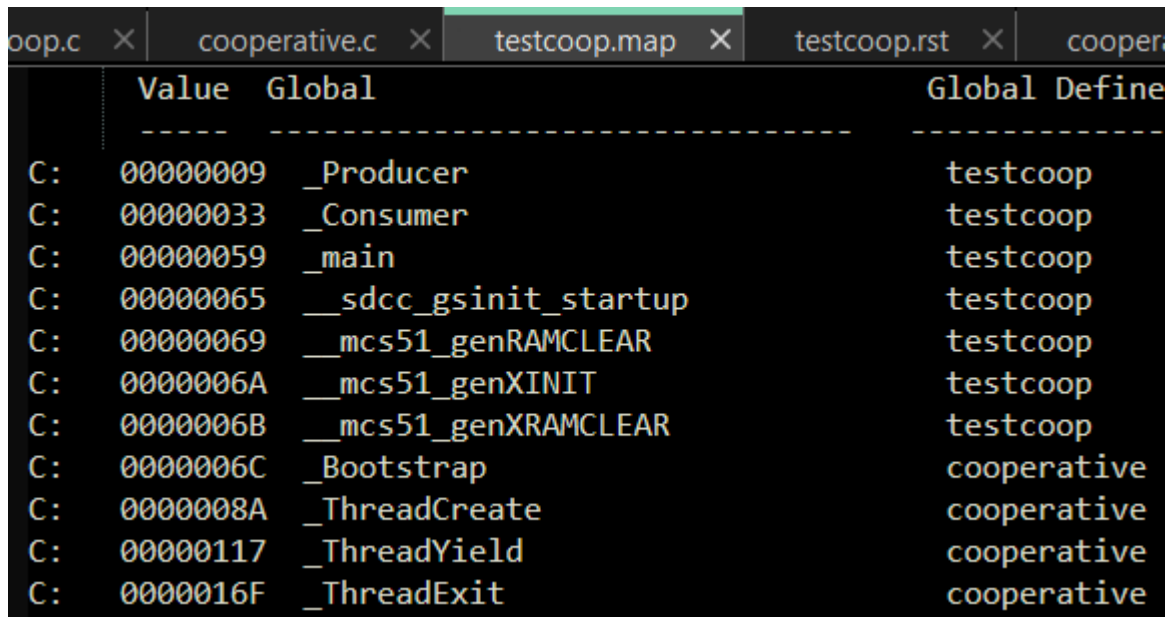
Fig. 1: Screenshot of Cygwin after running *make clean* and *make* command.

After \$ make clean	After \$ make

Table 2: results of Makefile compilation

## ThreadCreate calls

There are two thread create calls one for main and other for Producer.

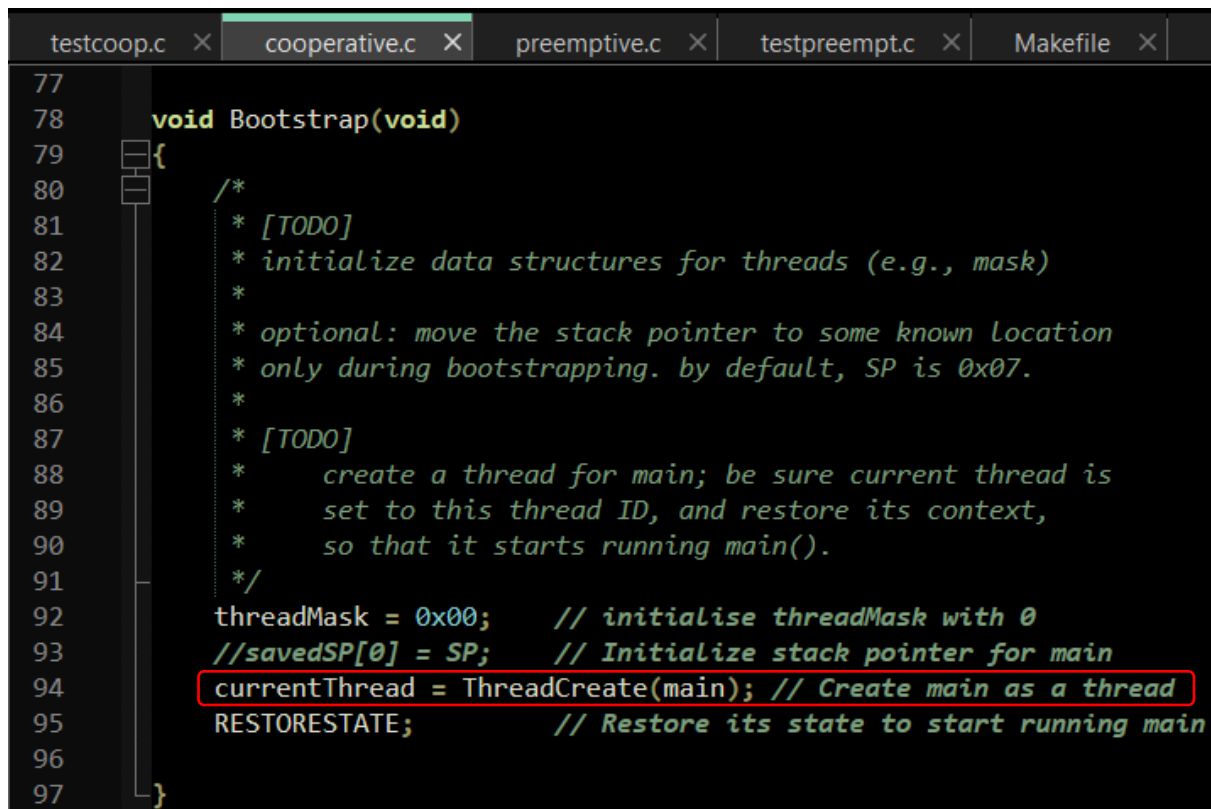


	Value	Global	Global Define
C:	00000009	_Producer	testcoop
C:	00000033	_Consumer	testcoop
C:	00000059	_main	testcoop
C:	00000065	__sdcc_gsinit_startup	testcoop
C:	00000069	__mcs51_genRAMCLEAR	testcoop
C:	0000006A	__mcs51_genXINIT	testcoop
C:	0000006B	__mcs51_genXRAMCLEAR	testcoop
C:	0000006C	_Bootstrap	cooperative
C:	0000008A	_ThreadCreate	cooperative
C:	00000117	_ThreadYield	cooperative
C:	0000016F	_ThreadExit	cooperative

Fig 2: Address of respective functions in cooperative and testcoop for reference.

### 1. ThreadCreate(main)

ThreadCreate for main form cooperative is called in during startup using Bootstrap (refer Fig. 3)



```
77
78 void Bootstrap(void)
79 {
80     /*
81     * [TODO]
82     * initialize data structures for threads (e.g., mask)
83     *
84     * optional: move the stack pointer to some known location
85     * only during bootstrapping. by default, SP is 0x07.
86     *
87     * [TODO]
88     * create a thread for main; be sure current thread is
89     * set to this thread ID, and restore its context,
90     * so that it starts running main().
91     */
92     threadMask = 0x00; // initialise threadMask with 0
93     //savedSP[0] = SP; // Initialize stack pointer for main
94     currentThread = ThreadCreate(main); // Create main as a thread
95     RESTORESTATE; // Restore its state to start running main
96 }
97
```

Fig. 3: ThreadCreate(main) call made in Bootstrap

```

operative.c x testcoop.map x testcoop.rst x cooperative.h x Makefile x cooperative.rst x
000065          427 ; -----
          428 __sdcc_gsinit_startup:
          429 ; testcoop.c:114: endasm;
000065 02 00 6C      [24] 430 LJMP _Bootstrap
          431 ; testcoop.c:115: }
000068 22          [24] 432 ret

```

Fig. 4: \_Bootstrap call in testcoop at address 0065H

```

testcoop.map x testcoop.rst x cooperative.h x Makefile x cooperative.rst x
          316 ; cooperative.c:92: threadMask = 0x00; // initialise
00006C 75 36 00      [24] 317 mov _threadMask,#0x00
          318 ; cooperative.c:94: currentThread = ThreadCreate(main);
00006F 90 00 59      [24] 319 mov dptr,#_main
000072 12 00 8A      [24] 320 lcall _ThreadCreate
000075 85 82 34      [24] 321 mov _currentThread,dpl

```

Fig. 5: ThreadCreate call in cooperative in \_Bootstrap at address 0072H

As can be seen from Fig. 5 ThreadCreate(main) is called at address 0072H, hence BreakPoint on 0072H is added.

EdSim51DI - Version 2.1.36 | testcoop.hex

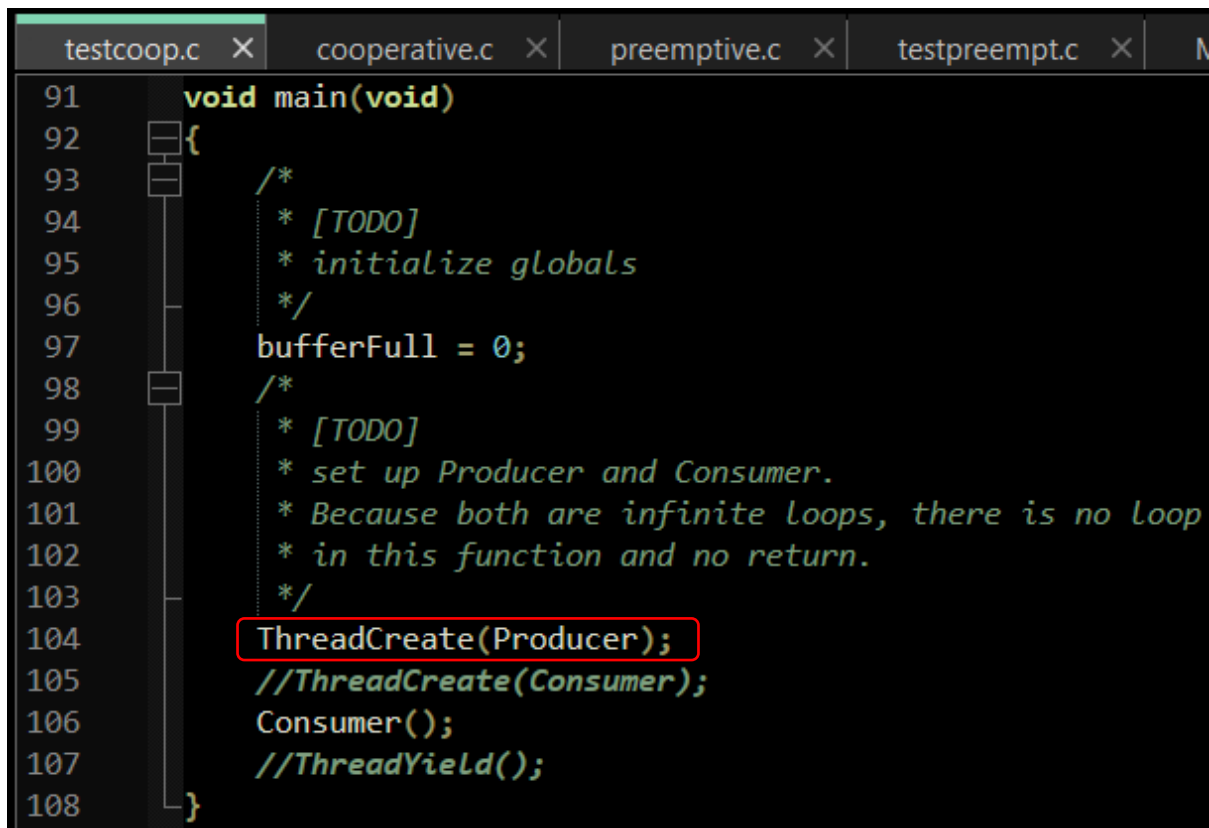
The screenshot displays the EdSim51DI interface. On the left, the register window shows the Program Counter (PC) at 0x0072 and the Stack Pointer (SP) at 0x0007. The right pane shows the instruction list, with the instruction at address 0072H, 'LCALL 008AH', highlighted with a red box. The bottom pane shows the data memory, which is currently empty (all zeros).

Fig. 6: Screenshot EdSim51 BreakPoint at 0072H

As shown in Figure 6, DPTR (i.e. DPH and DPL) is loaded with 0059 which is address of main in testcoop (refer Fig. 2) and SP is at 07H which is default value at SP depicting nothing is loaded into SP yet.

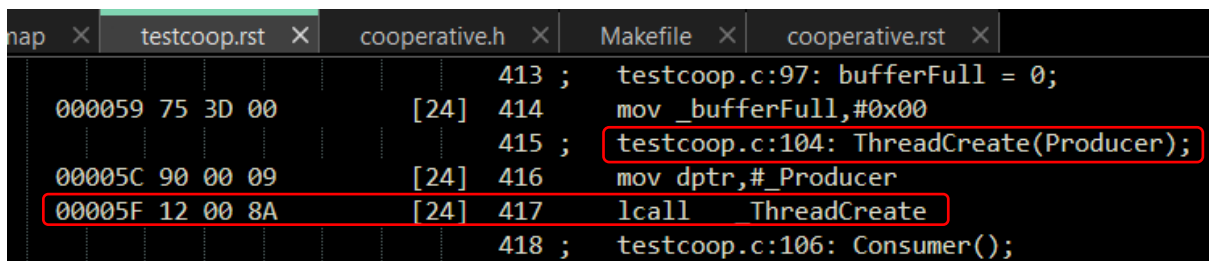
## 2. ThreadCreate(Producer)

ThreadCreate for Producer is called in main of testcoop (refer Fig. 7), where Producer maps to 0009H (refer Fig. 2) and ThreadCreate call for Producer is at 005FH can be observed in Fig. 8.



```
testcoop.c x cooperative.c x preemptive.c x testpreempt.c x M
91 void main(void)
92 {
93     /*
94     * [TODO]
95     * initialize globals
96     */
97     bufferFull = 0;
98     /*
99     * [TODO]
100    * set up Producer and Consumer.
101    * Because both are infinite loops, there is no loop
102    * in this function and no return.
103    */
104    ThreadCreate(Producer);
105    //ThreadCreate(Consumer);
106    Consumer();
107    //ThreadYield();
108 }
```

Fig. 7: ThreadCreate(Producer) call made in main



Address	Disassembly	Comment
000059	75 3D 00	[24] 413 ; testcoop.c:97: bufferFull = 0;
00005C	90 00 09	[24] 414 mov _bufferFull,#0x00
00005F	12 00 8A	[24] 415 ; testcoop.c:104: ThreadCreate(Producer);
		416 mov dptr,#_Producer
		417 lcall ThreadCreate
		418 ; testcoop.c:106: Consumer();

Fig. 8: ThreadCreate(Producer) call indicating at address 005FH.

The screenshot displays the EdSim51DI interface. On the left, the system clock is set to 11.0592 MHz, and the update frequency is 100. The register window shows R0-R7, ACC, PSW, IP, IE, PCON, DPH, DPL, and SP. The PC is 0x005F. The instruction list on the right shows the sequence of instructions, with 005F\* LCALL 008AH highlighted. The instruction list includes:

- 0053 | SJMP 0F8H
- 0055 | CLR 99H
- 0057 | SJMP 0E5H
- 0059 | MOV 3DH, #00H
- 005C | MOV DPTR, #0009H
- 005F\* | LCALL 008AH
- 0062 | LJMP 0033H
- 0065 | LJMP 006CH
- 0068 | RET
- 0069 | RET
- 006A | RET
- 006B | RET
- 006C | MOV 36H, #00H
- 006F | MOV DPTR, #0059H
- 0072\* | LCALL 008AH
- 0075 | MOV 34H, 82H
- 0078 | MOV A, 34H
- 007A | ADD A, #30H
- 007C | MOV R1, A
- 007D | MOV 81H, @R1
- 007F | POP 0D0H

Fig. 9: Screenshot of EdSim51 for BreakPoint at 005FH (ThreadCreate(Producer))

As shown in Figure 9, DPTR (i.e. DPH and DPL) having 0009H which is Producer's address (refer Fig. 2), and SP is at 3FH. On the BreakPoint at 005FH it's going to call for 008AH which is address of ThreadCreate (refer Fig. 2).

## Producer in Run

In Producer, variable "currentChar" is used to loop through the character "A" to "Z" and repeat the cycle again and it's going to be assigned to variable "sharedBuffer" to later which will be then transferred to SBUF in EdSim51. Variable "bufferFull" is a common variable for Producer and consumer to update, if sharedBuffer contains new character "bufferFull" is triggered to 1 and once character transferred SBUF "bufferFull" is set to 0.

```
testcoop.c X cooperative.c X testcoop.map X testcoop.rst X cooperative.h X Makefile
16  __data __at (0x3D) char bufferFull; // buffer status (0: empty, 1: full)
17  __data __at (0x3E) char sharedBuffer; // Shared buffer
18  __data __at (0x3F) char currentChar; // current character: A - Z
```

Fig. 10: Screenshot testcoop.c indicating address for respective variables

```

testcoop.c x cooperative.c x preemptive.c x testpreempt.c x Makefile x
26 void Producer(void)
27 {
28     /*
29     * [TODO]
30     * initialize producer data structure, and then enter
31     * an infinite loop (does not return)
32     */
33     currentChar = 'A';
34     while (1)
35     {
36         /* [TODO]
37         * wait for the buffer to be available,
38         * and then write the new data into the buffer */
39         while (bufferFull!=0) ThreadYield();
40         sharedBuffer = currentChar;
41         bufferFull = 1;
42         currentChar = (currentChar == 'Z') ? 'A' : currentChar + 1;
43         ThreadYield();
44     }
45 }

```

Fig. 11: Produce code snippet

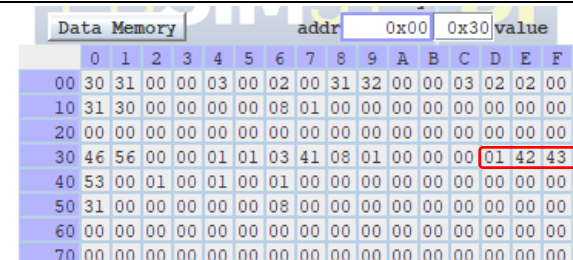
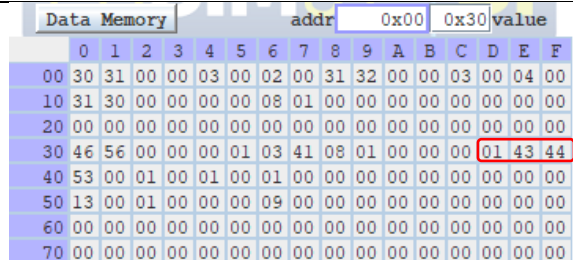
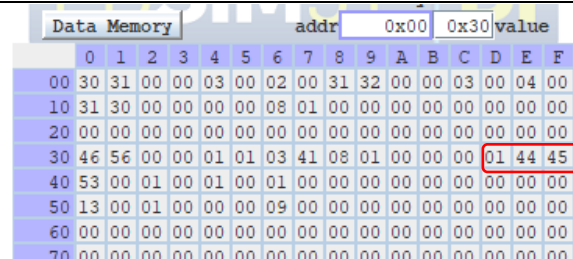
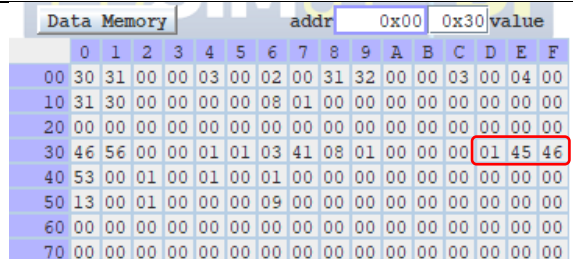
<p>sharedBuffer = "B" (i.e. 42H) currentChar = "C" (i.e. 43H)</p> 	<p>sharedBuffer = "C" (i.e. 43H) currentChar = "D" (i.e. 44H)</p> 
<p>sharedBuffer = "D" (i.e. 44H) currentChar = "E" (i.e. 45H)</p> 	<p>sharedBuffer = "E" (i.e. 45H) currentChar = "F" (i.e. 46H)</p> 

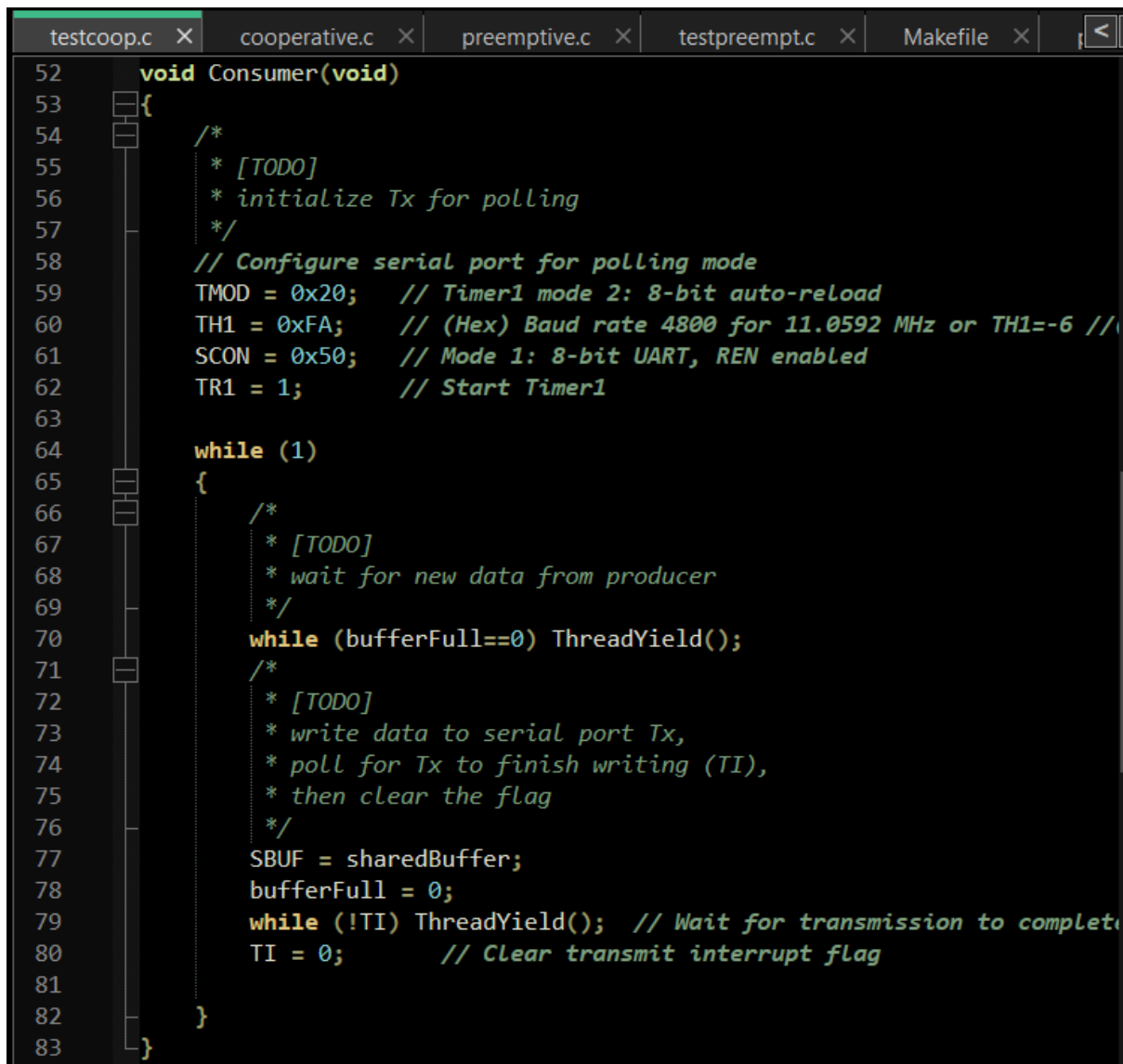
Table 3: Status of variables at respective memory during Producer is running.

Observing 3EH and 3FH as from Figure 10 indicating address of "sharedBuffer" and "currentChar" we can see in Table 2 that 3EH gets updated with value from 3FH and which is hex values of character "A" to "Z".



## Consumer in Run

When Consumer is running the SBUF receives the character from “sharedBuffer” and can be displayed on UART Receiver as shown in Figure 13 (a) and 13 (b) with code snippet in Fig. 12.



```
52 void Consumer(void)
53 {
54     /*
55      * [TODO]
56      * initialize Tx for polling
57      */
58     // Configure serial port for polling mode
59     TMOD = 0x20; // Timer1 mode 2: 8-bit auto-reload
60     TH1 = 0xFA; // (Hex) Baud rate 4800 for 11.0592 MHz or TH1=-6 //
61     SCON = 0x50; // Mode 1: 8-bit UART, REN enabled
62     TR1 = 1; // Start Timer1
63
64     while (1)
65     {
66         /*
67          * [TODO]
68          * wait for new data from producer
69          */
70         while (bufferFull==0) ThreadYield();
71         /*
72          * [TODO]
73          * write data to serial port Tx,
74          * poll for Tx to finish writing (TI),
75          * then clear the flag
76          */
77         SBUF = sharedBuffer;
78         bufferFull = 0;
79         while (!TI) ThreadYield(); // Wait for transmission to complete
80         TI = 0; // Clear transmit interrupt flag
81     }
82 }
83 }
```

Fig. 12: Consumer code snippet

In Figure 13 (a), Consumer running with SBUF having W/O 0x42H (i.e. “B”) which will be the next character to displayed in UART receiver along with previous character (i.e. “A”).

And in Figure 13 (b), Consumer running with SBUF having W/O 0x49H (i.e. “I”) which will be the next character to displayed in UART receiver along with previous character (i.e. “ABCDEFGH”).

0x3DH (i.e. “bufferFull”) is switched to 0 during consumer in execution as per Consumer code snippet in Fig. 12, it can be seen in Fig. 13(a & b).

System Clock (MHz) 11.0592 | 100 | Update Freq.

SBUF: R/O 0x00, W/O 0x42

TH0 0x00, TL0 0x00

R7 0x00, B 0x00

R6 0x02, ACC 0x01

R5 0x00, PSW 0x01

R4 0x03, IP 0x00

R3 0x00, IE 0x00

R2 0x00, PCON 0x00

R1 0x31, DPH 0x00

R0 0x30, DPL 0x01

SP 0x3F

TMOD 0x20

TCON 0xC0

pins bits: TH1 0xFA, TL1 0xFA

PC 0x004D

PSW 0 0 0 0 0 0 0 1

Modify RAM: addr 0x00, value 0x00

Data Memory:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	31	00	00	03	00	02	00	31	32	00	00	03	00	04
10	31	30	00	00	00	00	08	01	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	46	56	00	00	00	01	03	41	08	01	00	00	00	42	43
40	53	00	01	00	01	00	01	00	00	00	00	00	00	00	00
50	13	00	01	00	00	00	09	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 ©2005-2024 James Rogers | Remove All Breakpoints

Executed 0x004A: MOV 3DH, #00H | Time: 2ms

0022 | SJMP 08H

0024 | MOV R5, 3FH

0026 | INC R5

0027 | MOV A, R5

0028 | MOV R6, A

0029 | RLC A

002A | SUBB A, 0E0H

002C | MOV 3FH, R6

002E | LCALL 0117H

0031 | SJMP 0D9H

0033 | MOV 89H, #20H

0036 | MOV 8DH, #0FAH

0039 | MOV 98H, #50H

003C | SETB 8EH

003E | MOV A, 3DH

0040 | JNZ 05H

0042 | LCALL 0117H

0045 | SJMP 0F7H

0047 | MOV 99H, 3EH

004A\* | MOV 3DH, #00H

004D | JB 99H, 05H

DI i LD

1 2 3 AND Gate Disabled

4 5 6 Key Bounce Disabled

U No Parity 8-bit UART @ 4800 Baud

A

Rx Res

Fig. 13 (a): Screenshot (1) of EdSim51 while Consumer is running.

System Clock (MHz) 11.0592 | 100 | Update Freq.

SBUF: R/O 0x00, W/O 0x49

TH0 0x00, TL0 0x00

R7 0x00, B 0x00

R6 0x02, ACC 0x01

R5 0x00, PSW 0x01

R4 0x03, IP 0x00

R3 0x00, IE 0x00

R2 0x00, PCON 0x00

R1 0x31, DPH 0x00

R0 0x30, DPL 0x01

SP 0x3F

TMOD 0x20

TCON 0xC0

pins bits: TH1 0xFA, TL1 0xFA

PC 0x0050

PSW 0 0 0 0 0 0 0 1

Modify RAM: addr 0x00, value 0x00

Data Memory:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	31	00	00	03	00	02	00	31	32	00	00	03	00	04
10	31	30	00	00	00	00	08	01	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
30	46	56	00	00	00	01	03	41	08	01	00	00	00	49	4A
40	53	00	01	00	01	00	01	00	00	00	00	00	00	00	00
50	13	00	01	00	00	00	09	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 ©2005-2024 James Rogers | Remove All Breakpoints

Executed 0x004D: JB 99H, 05H | Time: 17ms

0024 | MOV R5, 3FH

0026 | INC R5

0027 | MOV A, R5

0028 | MOV R6, A

0029 | RLC A

002A | SUBB A, 0E0H

002C | MOV 3FH, R6

002E | LCALL 0117H

0031 | SJMP 0D9H

0033 | MOV 89H, #20H

0036 | MOV 8DH, #0FAH

0039 | MOV 98H, #50H

003C | SETB 8EH

003E | MOV A, 3DH

0040 | JNZ 05H

0042 | LCALL 0117H

0045 | SJMP 0F7H

0047 | MOV 99H, 3EH

004A\* | MOV 3DH, #00H

004D | JB 99H, 05H

0050 | LCALL 0117H

DI i LD

1 2 3 AND Gate Disabled

4 5 6 Key Bounce Disabled

U No Parity 8-bit UART @ 4800 Baud

ABCDEFGH

Rx Res

Fig. 13 (b): Screenshot (2) of EdSim51 while Consumer is running.