

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.

Creating and Compiling Makefile

Using same Makefile from CP2 for CP3 since the file names mentioned in both are the same hence it's compatible. 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.

```

/cygdrive/d/PhD/NTHU/OS/2024/Project/cp3/test_1

Snehit@LAPTOP-V8N83JAP ~
$ cd "D:\PhD\NTHU\OS\2024\Project\cp3\test_1"

Snehit@LAPTOP-V8N83JAP /cygdrive/d/PhD/NTHU/OS/2024/Project/cp3/test_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/cp3/test_1
$ make
sdcc -c testpreempt.c
sdcc -c preemptive.c
sdcc -o testpreempt.hex testpreempt.rel preemptive.rel

Snehit@LAPTOP-V8N83JAP /cygdrive/d/PhD/NTHU/OS/2024/Project/cp3/test_1
$

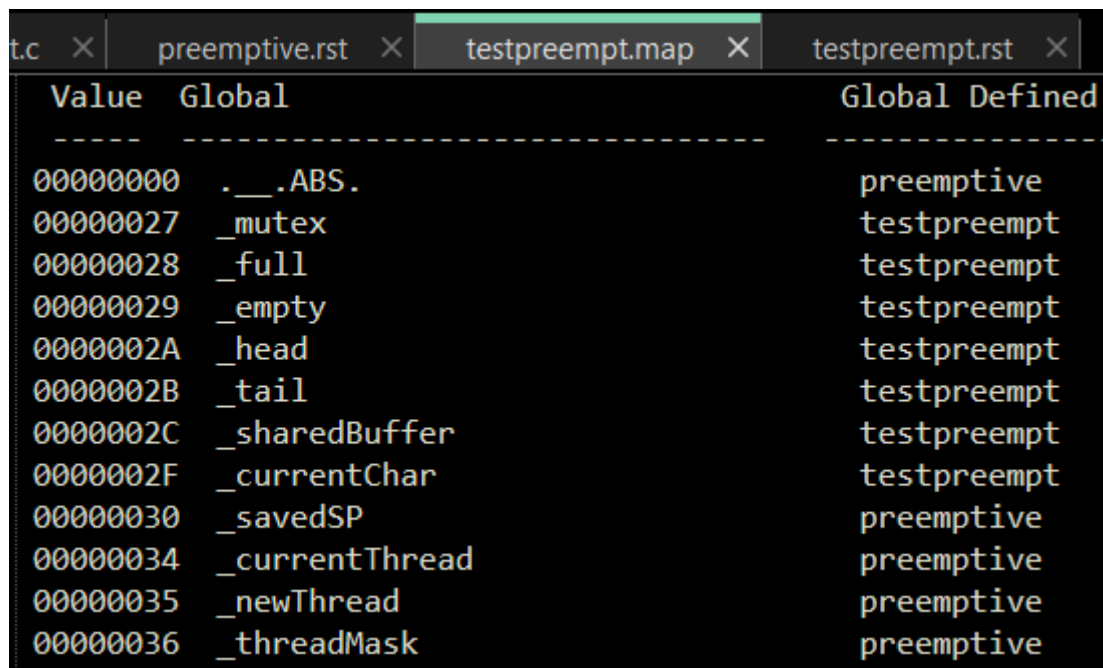
```

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

After \$ make clean	After \$ make

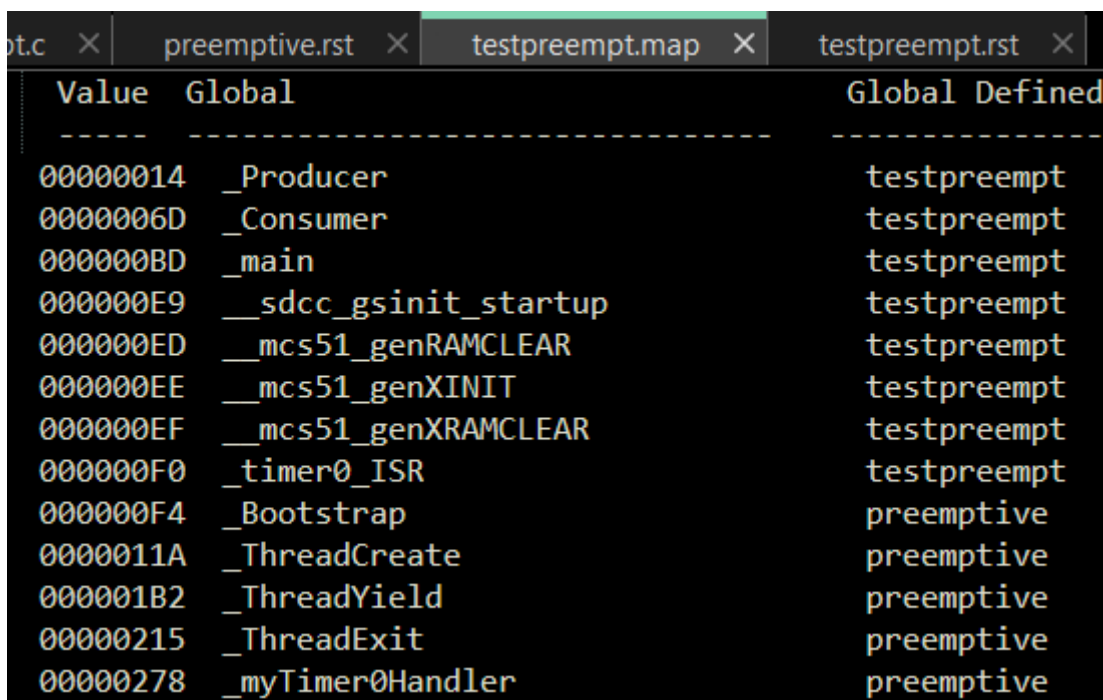
Table 2: results of Makefile compilation

Mapping of variables and functions:



Value	Global	Global Defined
00000000	___.ABS.	preemptive
00000027	_mutex	testpreempt
00000028	_full	testpreempt
00000029	_empty	testpreempt
0000002A	_head	testpreempt
0000002B	_tail	testpreempt
0000002C	_sharedBuffer	testpreempt
0000002F	_currentChar	testpreempt
00000030	_savedSP	preemptive
00000034	_currentThread	preemptive
00000035	_newThread	preemptive
00000036	_threadMask	preemptive

Fig. 3: Mapping of variables to respective memory locations.



Value	Global	Global Defined
00000014	_Producer	testpreempt
0000006D	_Consumer	testpreempt
000000BD	_main	testpreempt
000000E9	__sdcc_gsinit_startup	testpreempt
000000ED	__mcs51_genRAMCLEAR	testpreempt
000000EE	__mcs51_genXINIT	testpreempt
000000EF	__mcs51_genXRAMCLEAR	testpreempt
000000F0	_timer0_ISR	testpreempt
000000F4	_Bootstrap	preemptive
0000011A	_ThreadCreate	preemptive
000001B2	_ThreadYield	preemptive
00000215	_ThreadExit	preemptive
00000278	_myTimer0Handler	preemptive

Fig. 4: Mapping of functions to respective memory locations.

Producer in run:

The code snippet of Producer is as given in Fig. 5. Figure 6 shows the status of Semaphore and variable status during Producer in run. A breakpoint is added at address 004DH which corresponds SemaphoreSignal(mutex) in Producer.

```

preemptive.c x preemptive.h x testpreempt.c x preemptive.h x test3threads.c
20 void Producer(void)
21 {
22     currentChar = 'A';
23     while (1)
24     {
25         SemaphoreWait(empty);
26         __critical{
27             SemaphoreWait(mutex);
28             sharedBuffer[head] = currentChar;
29             head = (head==BUFFER_SIZE-1) ? 0 : head+1;
30             SemaphoreSignal(mutex);
31         }
32         SemaphoreSignal(full);
33         currentChar = (currentChar == 'Z') ? 'A' : currentChar + 1;
34     }
35 }

```

Fig. 5: Producer code snippet

It was said to declare 3-deep char buffer, Hence, we have variable sharedBuffer is of length 3. Variable head is used as index to assign value in sharedBuffer, e.g. from Table 3, head=01 (Fig. 6 (a)) is the index for next Character to be assigned at sharedBuffer i.e. sharedBuffer[1] (Fig. 6 (b)). And similarly for onward head values, once it reaches maximum index (i.e. 2) it'll reset to 0 and cycle repeats in the same way.

The screenshot displays the Proteus ISIS simulation environment for an 8051 microcontroller. The top panel shows the system clock at 11.0592 MHz and the update frequency set to 1000. The register window on the left shows the PSW register at 00000100. The data memory window shows the shared buffer at addresses 0000, 0001, and 0002, with values 00, 02, and 41 respectively. The instruction list on the right shows the execution of the INC 27H instruction at address 004D.

Addr	Value
00	30
01	31
02	00
03	00
04	00
05	00
06	00
07	00
08	00
09	00
0A	00
0B	00
0C	00
0D	00
0E	00
0F	00
10	31
11	30
12	00
13	00
14	00
15	00
16	00
17	00
18	00
19	00
1A	00
1B	00
1C	00
1D	00
1E	00
1F	00
20	0F
21	00
22	00
23	00
24	00
25	00
26	00
27	00
28	00
29	00
2A	00
2B	00
2C	00
2D	00
2E	00
2F	00
30	46
31	56
32	00
33	00
34	00
35	00
36	00
37	00
38	00
39	00
3A	00
3B	00
3C	00
3D	00
3E	00
3F	00
40	78
41	00
42	00
43	00
44	00
45	00
46	00
47	00
48	00
49	00
4A	00
4B	00
4C	00
4D	00
4E	00
4F	00

(a)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	31	00	00	01	01	00	01	2D	01	00	00	00	02	02	00
10	31	30	00	00	00	00	08	01	00	00	00	00	00	00	00	00
20	0F	00	00	00	00	00	00	00	01	01	02	00	41	42	00	42
30	46	56	00	00	01	01	03	00	01	01	00	00	00	00	00	00
40	78	00	00	00	01	00	80	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

(b)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	31	00	00	01	01	00	01	2E	01	00	00	00	43	00	00
10	31	30	00	00	00	00	08	01	00	00	00	00	00	00	00	00
20	0F	00	00	00	00	00	00	00	02	00	00	00	41	42	43	43
30	46	56	00	00	01	01	03	00	01	01	00	00	00	00	00	00
40	78	00	00	00	01	00	80	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

(c)

Fig 6: Producer in run with respective semaphore and variable changes till it full (from a to c).

Variable	Memory location	Value in Fig. 5 (a)	Value in Fig. 5 (b)	Value in Fig. 5 (c)
Mutex	0x27	00	00	00
Full	0x28	00	01	02
empty	0x29	02	01	00
head	0x2A	01	02	00
tail	0x2B	00	00	00
sharedBuffer[0]	0x2C	41	41	41
sharedBuffer[1]	0x2D	00	42	42
sharedBuffer[2]	0x2E	00	00	43
currentChar	0x2F	41	42	43

Table 3: Semaphore and variables change during producer run from Fig. 5 (a-c).

currentChar is variable for loop through character A-Z, the value of which is going to be assigned to sharedBuffer as per head index. Produce remain in run till Semaphore full reached maximum index possible (or say, till all the sharedBuffer get assigned to new values). Hence notice from Fig. 6 (c) the semaphore full is at 02 (at maximum) and sharedBuffer gets assigned to values 41, 42, 43 (i.e. character “A”, “B” and “C”).

For convenience Table 3 is given indicating necessary values of variables at mentioned memory locations during producer run which is taken from Fig. 6 (a -c).

Consumer in run:

After Producer completes its run with sharedBuffer filled with new set of characters, the Consumer is designed to transfer those characters to SBUF to UART. Code snippet of Consumer is given in Fig. 7.

Breakpoint at 00B9H is added which corresponds to the SemaphoreSignal(empty) during consumer is running. Which means before reaching this Breakpoint, the SBUF is updated with the character from sharedBuffer at index decided by tail from previous value. Consider Fig. 8 (a) previous value of tail is 0, sharedBuffer[tail] is 41H (i.e. “A”) which is assigned to SBUF at W/O, and it’ll be the character displayed in UART receiver (see Fig. 8 (b)).

Consumer will be in run till Semaphore empty reach maximum value (i.e. 2) since design buffer size is of length 3. Hence as empty gets updated in each cycle, the SBUF will gets assigned to characters from sharedBuffer with index from tail. Observe from 8 (a), (b) and (c), SBUF is assigned to 41, 42, and 43 in each cycle and this character will then transfer to UART receiver i.e. “A”, “B” and “C” respectively. Fig. 9 is after consumer is executed, shows all the characters in transferred to SBUF.

```

preemptive.c × preemptive.h × testpreempt.c × preemptive.h × test3threads.c
38 void Consumer(void)
39 {
40     // Configure serial port for polling mode
41     TMOD |= 0x20; // Timer1 mode 2: 8-bit auto-reload
42     TH1 = 0xFA; // (Hex) Baud rate 4800 for 11.0592 MHz or TH1=-6
43     SCON = 0x50; // Mode 1: 8-bit UART, REN enabled
44     TR1 = 1; // Start Timer1
45
46     while (1)
47     {
48         SemaphoreWait(full);
49         __critical{
50             SemaphoreWait(mutex);
51             SBUF = sharedBuffer[tail];
52             while (!TI); // Wait for transmission to complete
53             TI = 0; // Clear transmit interrupt flag
54             tail = (tail==BUFFER_SIZE-1) ? 0 : tail+1;
55             SemaphoreSignal(mutex);
56         }
57         SemaphoreSignal(empty);
58     }
59 }

```

Fig. 7: Consumer code snippet

For convenience Table 4 is given indicating necessary values of variables at mentioned memory locations during consumer run which is taken from Fig. 8 (a -c).

Variable	Memory location	Value in Fig. 6 (a)	Value in Fig. 6 (b)	Value in Fig. 6 (c)
Mutex	0x27	01	01	01
Full	0x28	02	01	00
empty	0x29	00	01	02
head	0x2A	00	00	00
tail	0x2B	01	02	00
sharedBuffer	0x2C - 0x2F	41, 42, 43	41, 42, 43	41, 42, 43
SBUF	0x99	41	42	43

Table 4: Semaphore, variables and SBUF change during consumer run from Fig. 6 (a-c).

System Clock (MHz) 11.0592 1000 Update Freq.

SBUF

R/O W/O TH0 TL0 R7 0x00 B 0x00
0x00 0x41 0x3D 0x0A R6 0x01 ACC 0x00
RXD TXD R5 0x01 PSW 0x80
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 0x2C DPH 0x00
0xFF 0xFF P3 0xFA 0xFA 8051
0xFF 0xFF P2
0xFF 0xFF P1
0xFF 0xFF P0
PC 0x00B9 PSW 1 0 0 0 0 0 0 0

Data Memory

addr	0x00	0x30	value
0	0	1	2
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
A	0	0	0
B	0	0	0
C	0	0	0
D	0	0	0
E	0	0	0
F	0	0	0

Copyright ©2005-2024 James Rogers Remove All Breakpoints

DI i LD

1 2 3 AND Gate Disabled
4 5 6 Key Bounce Disabled
7 8 9

U No Parity 8-bit UART @ 4800 Baud
Rx Rx Rese

Time: 19ms 916us - Instructions: 11965

0095 | MOV R1, A
0096 | MOV 99H, @R1
0098 | JBC 99H, 02H
009B | SJMP 0FBH
009D | MOV A, #02H
009F | CJNE A, 2BH, 06H
00A2 | MOV R6, #00H
00A4 | MOV R7, #00H
00A6 | SJMP 09H
00A8 | MOV R5, 2BH
00AA | INC R5
00AB | MOV A, R5
00AC | MOV R6, A
00AD | RLC A
00AE | SUBB A, 0E0H
00B0 | MOV R7, A
00B1 | MOV 2BH, R6
00B3 | INC 27H
00B5 | MOV C, 01H
00B7 | MOV 0AFH, C
00B9* | INC 29H

Fig 8 (a): Consumer in run with SBUF W/O at 0x41

System Clock (MHz) 11.0592 1000 Update Freq.

SBUF

R/O W/O TH0 TL0 R7 0x00 B 0x00
0x00 0x42 0x79 0x0C R6 0x02 ACC 0x00
RXD TXD R5 0x02 PSW 0x80
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 0x2D DPH 0x00
0xFF 0xFF P3 0xFA 0xFC 8051
0xFF 0xFF P2
0xFF 0xFF P1
0xFF 0xFF P0
PC 0x00B9 PSW 1 0 0 0 0 0 0 0

Data Memory

addr	0x00	0x30	value
0	0	1	2
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
A	0	0	0
B	0	0	0
C	0	0	0
D	0	0	0
E	0	0	0
F	0	0	0

Copyright ©2005-2024 James Rogers Remove All Breakpoints

DI i LD

1 2 3 AND Gate Disabled
4 5 6 Key Bounce Disabled
7 8 9

U No Parity 8-bit UART @ 4800 Baud
Rx Rx Rese

Time: 22ms 2us - Instructions: 12935

0095 | MOV R1, A
0096 | MOV 99H, @R1
0098 | JBC 99H, 02H
009B | SJMP 0FBH
009D | MOV A, #02H
009F | CJNE A, 2BH, 06H
00A2 | MOV R6, #00H
00A4 | MOV R7, #00H
00A6 | SJMP 09H
00A8 | MOV R5, 2BH
00AA | INC R5
00AB | MOV A, R5
00AC | MOV R6, A
00AD | RLC A
00AE | SUBB A, 0E0H
00B0 | MOV R7, A
00B1 | MOV 2BH, R6
00B3 | INC 27H
00B5 | MOV C, 01H
00B7 | MOV 0AFH, C
00B9* | INC 29H

Fig 8 (b): Consumer in run with SBUF W/O at 0x42 and UART display A (i.e. 0x41)

System Clock (MHz) 11.0592 1000 Update Freq.

SBUF

R/O W/O TH0 TL0 R7 0x00 B 0x00
0x00 0x43 0xB5 0x06 R6 0x00 ACC 0x02
RXD TXD 1 1 TMOD 0x20 R5 0x02 PSW 0x81
SCON 0x50 TCON 0xD0 R4 0x00 IP 0x00
R3 0x00 IE 0x82
R2 0x00 PCON 0x00
R1 0x2E DPH 0x00
R0 0x30 DPL 0x01
SP 0x3F

pins bits TH1 TL1
0xFF 0xFF P3 0xFA 0xFC
0xFF 0xFF P2
0xFF 0xFF P1
0xFF 0xFF P0

PC 0x00B9 PSW 1 0 0 0 0 0 0 0 1

Modify RAM
Data Memory addr 0x00 0x30 value

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	2E	00	00	00	00	00	00	00	31	01	00	00	00	44	44
10	31	30	00	00	00	00	08	01	00	00	00	00	00	00	00	00
20	0F	00	00	00	00	00	00	01	00	02	00	00	41	42	43	44
30	46	56	00	00	00	01	03	00	01	01	00	00	00	00	00	00
40	78	00	00	00	01	00	80	00	00	00	00	00	00	00	00	00
50	17	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 ©2005-2024 James Rogers Remove All Breakpoints

DI i LD

1 2 3 AND Gate Disabled
4 5 6 Key Bounce Disabled
7 8 9

U No Parity 8-bit UART @ 4800 Baud
Rx AB Rx R

Time: 24ms 79us - Instructions: 13899

```

0095| MOV R1,A
0096| MOV 99H,@R1
0098| JBC 99H,02H
009B| SJMP 0FBH
009D| MOV A,#02H
009F| CJNE A,2BH,06H
00A2| MOV R6,#00H
00A4| MOV R7,#00H
00A6| SJMP 09H
00A8| MOV R5,2BH
00AA| INC R5
00AB| MOV A,R5
00AC| MOV R6,A
00AD| RLC A
00AE| SUBB A,0E0H
00B0| MOV R7,A
00B1| MOV 2BH,R6
00B3| INC 27H
00B5| MOV C,01H
00B7| MOV 0AFH,C
00B9* INC 29H

```

Fig 8 (c): Consumer in run with SBUF W/O at 0x43 and UART display AB (i.e. 0x41 and 0x42)

System Clock (MHz) 11.0592 1000 Update Freq.

SBUF

R/O W/O TH0 TL0 R7 0x00 B 0x00
0x00 0x43 0x04 0x0B R6 0x01 ACC 0x00
RXD TXD 1 1 TMOD 0x20 R5 0x01 PSW 0x08
SCON 0x50 TCON 0xD0 R4 0x00 IP 0x00
R3 0x00 IE 0x02
R2 0x00 PCON 0x00
R1 0x01 DPH 0x00
R0 0x2C DPL 0x00
SP 0x4F

pins bits TH1 TL1
0xFF 0xFF P3 0xFA 0xFD
0xFF 0xFF P2
0xFF 0xFF P1
0xFF 0xFF P0

PC 0x004D PSW 0 0 0 0 0 1 0 0 0

Modify RAM
Data Memory addr 0x00 0x30 value

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	31	00	00	00	03	00	00	2C	01	00	00	00	01	01	00
10	31	30	00	00	00	00	08	01	00	00	00	00	00	00	00	00
20	0F	00	00	00	00	00	00	00	02	01	00	44	42	43	44	
30	46	56	00	00	01	01	03	00	01	00	00	00	00	00	00	00
40	78	00	00	00	01	00	80	00	00	00	00	00	00	00	00	00
50	17	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 ©2005-2024 James Rogers Remove All Breakpoints

DI i LD

1 2 3 AND Gate Disabled
4 5 6 Key Bounce Disabled
7 8 9

U No Parity 8-bit UART @ 4800 Baud
Rx ABC Rx R

Time: 26ms 827us - Instructions: 15587

```

004D* INC 27H
004F| MOV C,00H
0051| MOV 0AFH,C
0053| INC 28H
0055| MOV A,#5AH
0057| CJNE A,2FH,06H
005A| MOV R6,#41H
005C| MOV R7,#00H
005E| SJMP 09H
0060| MOV R5,2FH
0062| INC R5
0063| MOV A,R5
0064| MOV R6,A
0065| RLC A
0066| SUBB A,0E0H
0068| MOV R7,A
0069| MOV 2FH,R6
006B| SJMP 0AAH
006D* ORL 89H,#20H
0070| MOV 8DH,#0FAH
0073| MOV 98H,#50H

```

Fig 9: Consumer after execution with UART display ABC (i.e. 0x41, 0x42 and 0x43)

Onward the cycle is repeated with new characters assigned to sharedBuffer by Producer and Consumer forward it to SBUF and displayed on UART receiver which is shown in Figure 10.

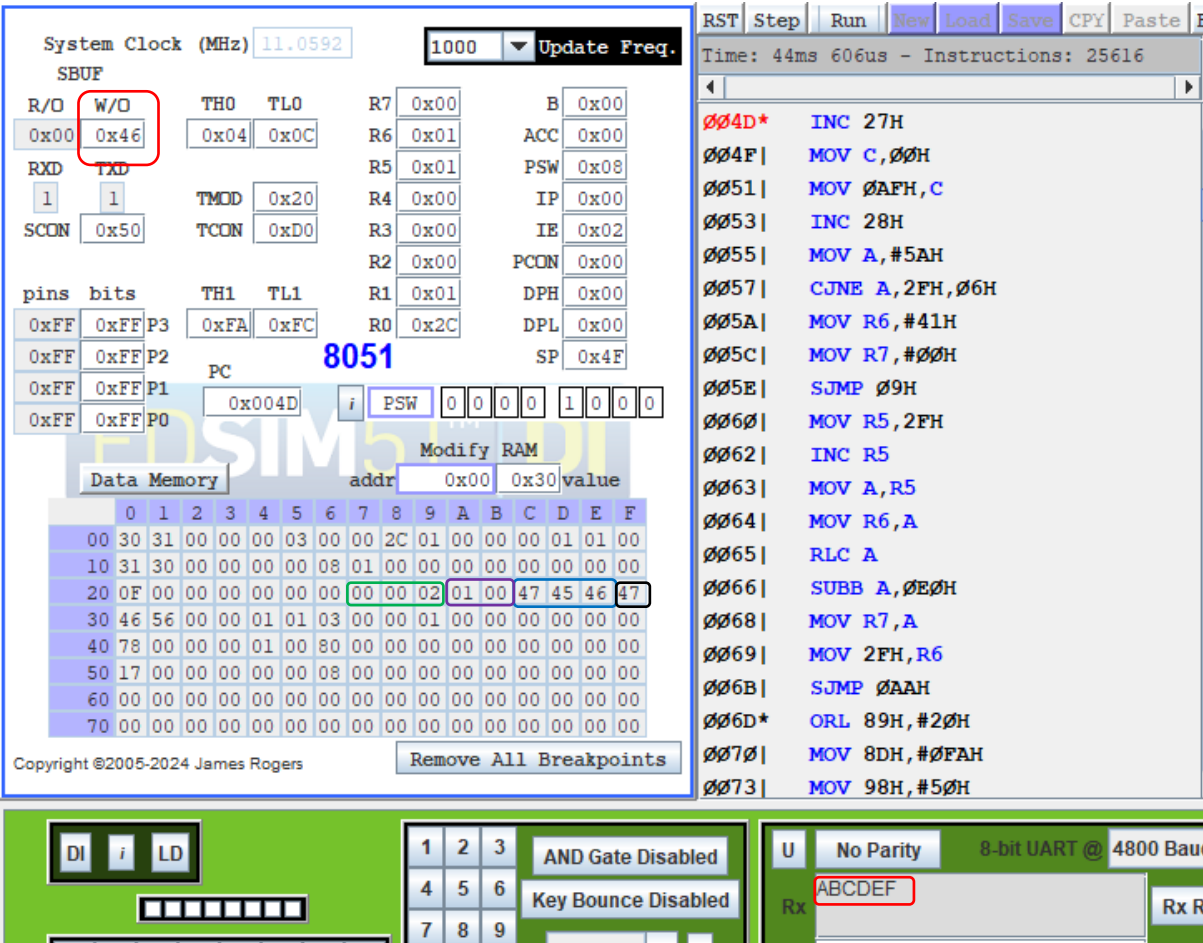


Fig 10: Sample showing the execution of producer for updated sharedBuffer and execution of consumer with latest characters on SBUF and UART