111006203 許裕華

Code Explanation

Preemptive.h:

Most of the function, I follow the provided google docs for implementations

- **CNAME(s):** This macro adds an underscore prefix to the variable s. This can help to avoid naming conflicts and is likely used to generate unique variable names for semaphores.
- **LABEL_NAME(l):** This macro appends a dollar sign ($) to the label l to create a unique label for assembly code. This is important when generating multiple labels.
- **SemaphoreCreate(s, n):**
  - The statement *&s = n sets the semaphore s to the value n
- **SemaphoreWait:** This macro is a way to generate a unique label for each call to SemaphoreWaitBody.
  - I add __COUNTER__ , the __COUNTER__ is a preprocessor feature that generates a unique integer value each time it is invoked. This ensures that each semaphore wait operation gets a unique label in assembly code.
- **SemaphoreWaitBody:** This macro implements the wait operation on a semaphore.
  - **LABEL_NAME(label):** Defines a label for assembly code to facilitate the loop that checks the semaphore value.
  - **mov a, CNAME(s):** Loads the semaphore value into the accumulator (ACC).
  - **jz LABEL_NAME(label):** If the semaphore is zero, it jumps back to the label, effectively looping until the semaphore value becomes non-zero.
  - **jb ACC.7, LABEL_NAME(label**): Checks if the accumulator's most significant bit (bit 7) is set. If true, it means the semaphore is non-positive (i.e., <= 0), and it jumps back to the label.
  - **dec CNAME(s):** Decreases the semaphore value, which allows the thread to proceed if the semaphore was positive.
- **SemaphoreSignal:** I have done nothing but add “inc CNAME(s)” to increment the value of the semaphore

testpreempt.c:

I implemented this code similar to the previous one so I would like to explain only the difference parts.

Global varaibles:

- **head and tail (0x3A and 0x3B):** Pointers for the producer and consumer to track where to add and remove items in the bounded buffer.
- **nextChar (0x3C):** (same as the previous checkpoint’s)

- **mutex (0x3D):** A semaphore used to ensure that only one thread accesses the buffer at a time (mutex lock).
- **full (0x3E):** A semaphore that counts the number of filled slots in the buffer. It helps ensure that the consumer only consumes when there are items in the buffer.
- **empty (0x3F):** A semaphore that counts the number of empty slots in the buffer. It ensures that the producer only produces when there is space in the buffer.
- **BoundedBuffer[3] (0x20):** The bounded buffer, 3-deep char buffer, with a maximum size of 3, initialized with spaces (' ').

Producer and Consumer:

- I have followed the format from the google docs by adding semaphores to each function:
- **For producer:**
  SemaphoreWait(empty);
  SemaphoreWait(mutex);
  *my code from the 2nd checkpoint but using tail to track the current position of just produced characters, this ensures my circular buffer works correctly
  SemaphoreSignal(mutex);
  SemaphoreSignal(full);
- **For consumer:**
  SemaphoreWait(full);
  SemaphoreWait(mutex);
  *my code from the 2nd checkpoint but using head to track the current position of just consumed characters, this ensures my circular buffer works correctly
  SemaphoreSignal(mutex);
  SemaphoreSignal(empty);

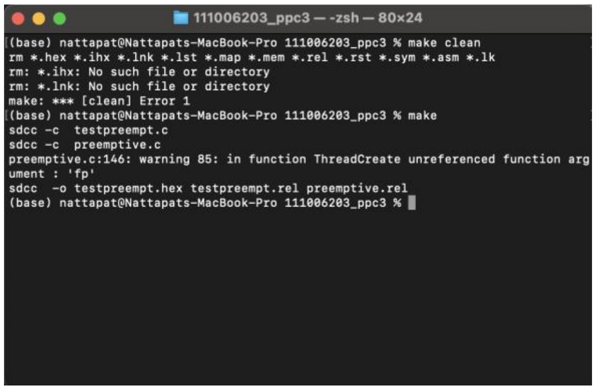
For main function:
I have initialized all the variables as follows:

- head and tail as 0
- SemaphoreCreate -> mutex as 1 (this ensures only one thread can access the buffer at a time)
- SemaphoreCreate -> full as 0 (indicate buffer as empty)
- SemaphoreCreate -> empty as 3 (buffer has 3 empty slots)
- Then bounded buffer as empty = ‘ ’ for all spaces

Preemptive.c:

To be honest nothing has changed from the checkpoint 2 one

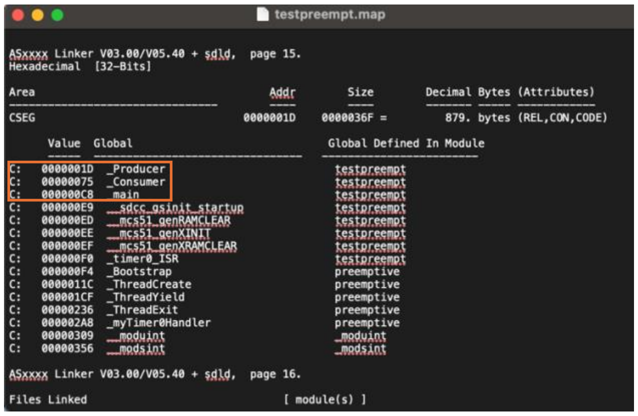
Screenshots for compilation



```
(base) nattapat@Nattapats-MacBook-Pro 111006203_ppc3 % 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
(base) nattapat@Nattapats-MacBook-Pro 111006203_ppc3 % make
sdcc -c  testpreempt.c
sdcc -c  preemptive.c
preemptive.c:146: warning 85: in function ThreadCreate unreferenced function arg
ument : 'fp'
sdcc  -o testpreempt.hex testpreempt.rel preemptive.rel
(base) nattapat@Nattapats-MacBook-Pro 111006203_ppc3 %
```

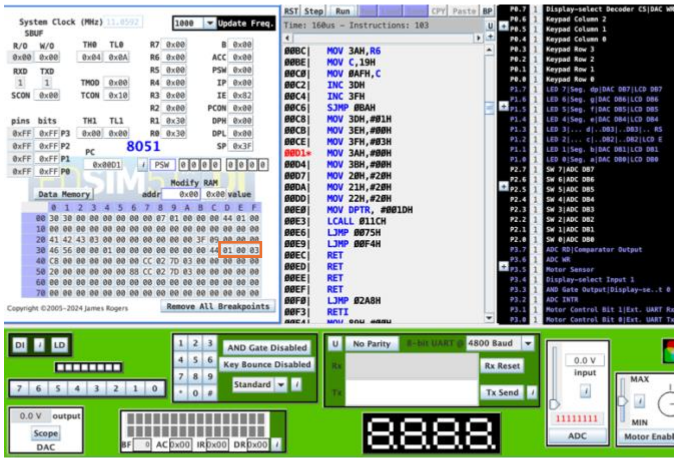
Figure 1 Screenshot for compilation

Screenshots and explanation



```
ASxxxx Linker V03.00/V05.40 + sdld,  page 15.
Hexadecimal  [32-Bits]

Area                                    Addr        Size        Decimal Bytes (Attributes)
--------------------------------        ----        ----        ------- ----- ------------
CSEG                                0000001D    0000036F =         879. bytes (REL,CON,CODE)

      Value  Global                              Global Defined In Module
      -----  --------------------------------   ------------------------
C:   0000001D  _Producer                          testpreempt
C:   00000075  _Consumer                          testpreempt
C:   000000C8  _main                              testpreempt
C:   000000E9  __sdcc_gsinit_startup              testpreempt
C:   000000ED  __mcs51_genRAMCLEAR                testpreempt
C:   000000EE  __mcs51_genXINIT                   testpreempt
C:   000000EF  __mcs51_genXRAMCLEAR               testpreempt
C:   000000F0  _timer0_ISR                        testpreempt
C:   000000F4  _Bootstrap                         preemptive
C:   0000011C  _ThreadCreate                      preemptive
C:   000001CF  _ThreadYield                       preemptive
C:   00000236  _ThreadExit                        preemptive
C:   000002A8  _myTimer0Handler                   preemptive
C:   00000309  __moduint                          _moduint
C:   00000356  __modsint                          _modsint

ASxxxx Linker V03.00/V05.40 + sdld,  page 16.

Files Linked                              [ module(s) ]
```

Figure 2 testcoop.map (will be used for explanation in the following section)

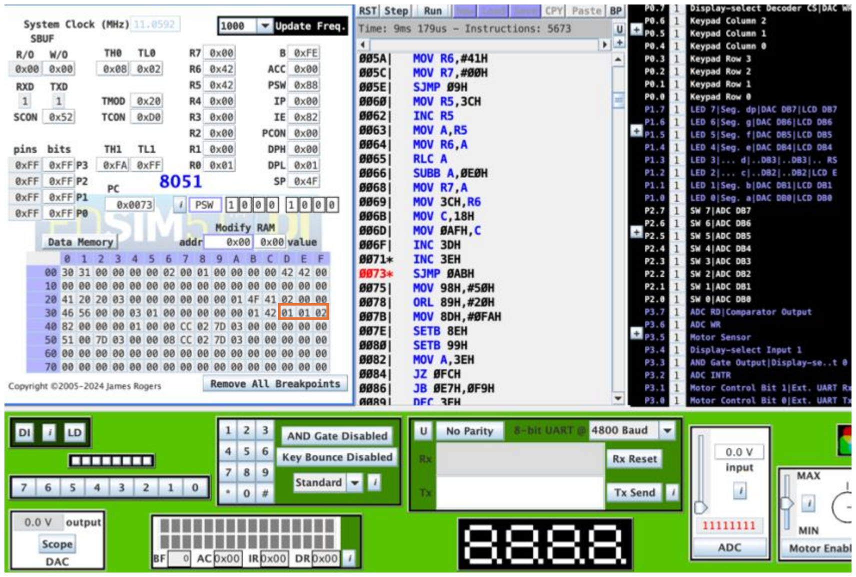


```
Time: 160us - Instructions: 103
00BC|   MOV 3AH,R6
00BE|   MOV C,19H
00C0|   MOV 0AFH,C
00C2|   INC 3DH
00C4|   INC 3FH
00C6|   SJMP 0BAH
00C8|   MOV 3DH,#01H
00CB|   MOV 3EH,#00H
00CE|   MOV 3FH,#03H
00D1*   MOV 3AH,#00H
00D4|   MOV 3BH,#00H
00D7|   MOV 20H,#20H
00DA|   MOV 21H,#20H
00DD|   MOV 22H,#20H
00E0|   MOV DPTR, #001DH
00E3|   LCALL 011CH
00E6|   LJMP 0075H
00E9|   LJMP 00F4H
00EC|   RET
00ED|   RET
00EE|   RET
00EF|   RET
00F0|   LJMP 02A8H
00F3|   RETI
```

Figure 3 initialize semaphores

As figure 3 shown, the semaphores are initialized in the main function as 0x3D (mutex) = 1, 0x3E (full) = 0, and 0x3F (empty) = 3.

Producer is running and show semaphore changes



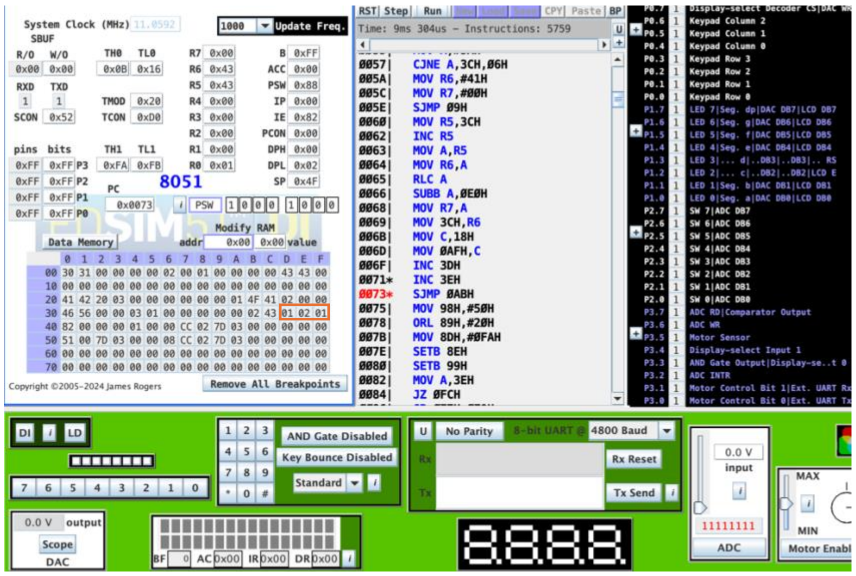
```
R7 0x00   B    0xFE
R6 0x42   ACC  0x00
R5 0x42   PSW  0x88
R4 0x00   IP   0x00
R3 0x00   IE   0x82
R2 0x00   PCON 0x00
R1 0x00   DPH  0x00
R0 0x01   DPL  0x01
          SP   0x4F
PC 0x0073
```

```
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 30 31 00 00 00 00 02 00 01 00 00 00 00 42 42 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 41 20 20 03 00 00 00 00 00 00 01 4F 41 02 00 00
30 46 56 00 00 03 01 00 00 00 00 00 01 42 01 01 02
40 82 00 00 00 01 00 00 CC 02 7D 03 00 00 00 00 00
50 51 00 7D 03 00 00 08 CC 02 7D 03 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
```

```
Time: 9ms 179us - Instructions: 5673
005A|   MOV R6,#41H
005C|   MOV R7,#00H
005E|   SJMP 09H
0060|   MOV R5,3CH
0062|   INC R5
0063|   MOV A,R5
0064|   MOV R6,A
0065|   RLC A
0066|   SUBB A,0E0H
0068|   MOV R7,A
0069|   MOV 3CH,R6
006B|   MOV C,18H
006D|   MOV 0AFH,C
006F|   INC 3DH
0071*   INC 3EH
0073*   SJMP 0ABH
0075|   MOV 98H,#50H
0078|   ORL 89H,#20H
007B|   MOV 8DH,#0FAH
007E|   SETB 8EH
0080|   SETB 99H
0082|   MOV A,3EH
0084|   JZ 0FCH
0086|   JB 0E7H,0F9H
```

Figure 4 produce the first character

The firgure 4 shows when the producer writes the first character into the buffer, then the sephamore full will increase by 1 (0x3E (full) = 1) and sephamore empty will decrease by 1 (0x3F (empty) = 2).



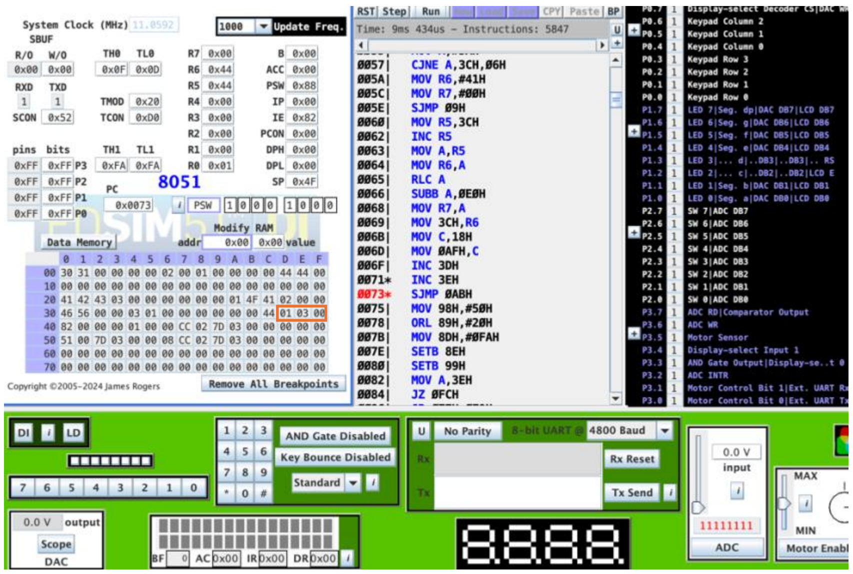
```
R7 0x00   B    0xFF
R6 0x43   ACC  0x00
R5 0x43   PSW  0x88
R4 0x00   IP   0x00
R3 0x00   IE   0x82
R2 0x00   PCON 0x00
R1 0x00   DPH  0x00
R0 0x01   DPL  0x02
          SP   0x4F
PC 0x0073
```

```
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 30 31 00 00 00 00 02 00 01 00 00 00 00 43 43 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 41 42 20 03 00 00 00 00 00 00 01 4F 41 02 00 00
30 46 56 00 00 03 01 00 00 00 00 00 02 43 01 02 01
40 82 00 00 00 01 00 00 CC 02 7D 03 00 00 00 00 00
50 51 00 7D 03 00 00 08 CC 02 7D 03 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
```

```
Time: 9ms 304us - Instructions: 5759
0057|   CJNE A,3CH,06H
005A|   MOV R6,#41H
005C|   MOV R7,#00H
005E|   SJMP 09H
0060|   MOV R5,3CH
0062|   INC R5
0063|   MOV A,R5
0064|   MOV R6,A
0065|   RLC A
0066|   SUBB A,0E0H
0068|   MOV R7,A
0069|   MOV 3CH,R6
006B|   MOV C,18H
006D|   MOV 0AFH,C
006F|   INC 3DH
0071*   INC 3EH
0073*   SJMP 0ABH
0075|   MOV 98H,#50H
0078|   ORL 89H,#20H
007B|   MOV 8DH,#0FAH
007E|   SETB 8EH
0080|   SETB 99H
0082|   MOV A,3EH
0084|   JZ 0FCH
```

Figure 5 produce the second character

The firgure 5 shows when the producer writes the second character into the buffer, then the sephamore full will increase by 1 (0x3E (full) = 2) and sephamore empty will decrease by 1 (0x3F (empty) = 1).

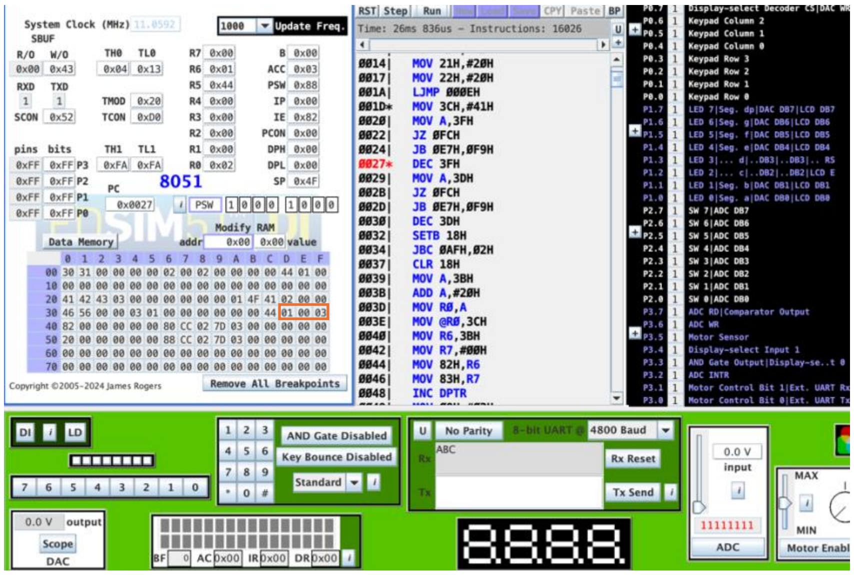


```
Time: 9ms 434us - Instructions: 5847

0057|   CJNE A,3CH,06H
005A|   MOV R6,#41H
005C|   MOV R7,#00H
005E|   SJMP 09H
0060|   MOV R5,3CH
0062|   INC R5
0063|   MOV A,R5
0064|   MOV R6,A
0065|   RLC A
0066|   SUBB A,0E0H
0068|   MOV R7,A
0069|   MOV 3CH,R6
006B|   MOV C,18H
006D|   MOV 0AFH,C
006F|   INC 3DH
0071*   INC 3EH
0073*   SJMP 0ABH
0075|   MOV 98H,#50H
0078|   ORL 89H,#20H
007B|   MOV 8DH,#0FAH
007E|   SETB 8EH
0080|   SETB 99H
0082|   MOV A,3EH
0084|   JZ 0FCH
```

```
SBUF R/O 0x00  W/O 0x00    TH0 0x0F  TL0 0x0D    TMOD 0x20  TCON 0xD0   SCON 0x52
R7 0x00  R6 0x44  R5 0x44  R4 0x00  R3 0x00  R2 0x00  R1 0x00  R0 0x01
B 0x00  ACC 0x00  PSW 0x88  IP 0x00  IE 0x82  PCON 0x00  DPH 0x00  DPL 0x00  SP 0x4F
TH1 0xFA  TL1 0xFA
P3 0xFF 0xFF  P2 0xFF 0xFF  P1 0xFF 0xFF  P0 0xFF 0xFF
PC 0x0073   PSW 1 0 0 0 1 0 0 0

Data Memory
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 30 31 00 00 00 00 02 00 01 00 00 00 00 44 44 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 41 42 43 03 00 00 00 00 00 00 01 4F 41 02 00 00
30 46 56 00 00 03 01 00 00 00 00 00 00 44 01 03 00
40 82 00 00 00 01 00 00 CC 02 7D 03 00 00 00 00 00
50 51 00 7D 03 00 00 08 CC 02 7D 03 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
```

Figure 6 produce the third character

The firgure 6 shows when the producer writes the third character into the buffer, then the sephamore full will increase by 1 (0x3E (full) = 3) and sephamore empty will decrease by 1 (0x3F (empty) = 0). Indicating the buffer is now full.



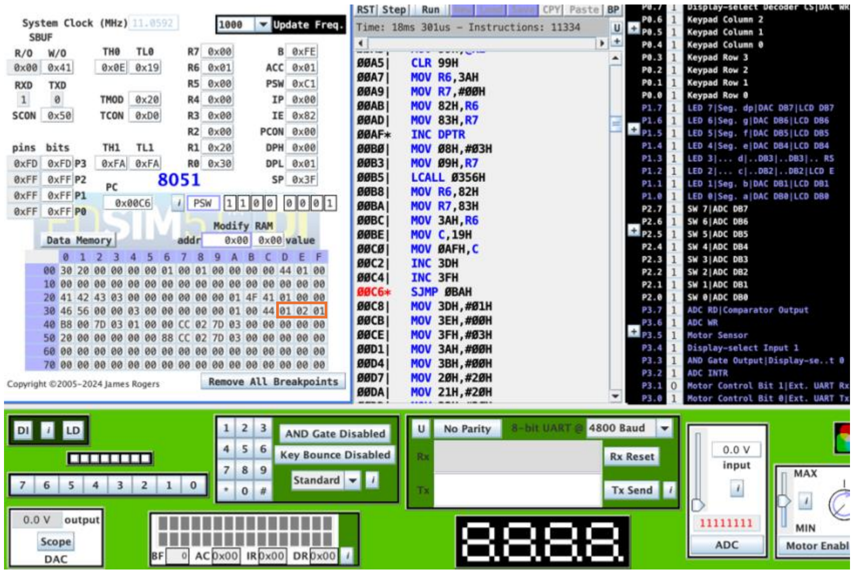
```
Time: 26ms 836us - Instructions: 16026

0014|   MOV 21H,#20H
0017|   MOV 22H,#20H
001A|   LJMP 000EH
001D*   MOV 3CH,#41H
0020|   MOV A,3FH
0022|   JZ 0FCH
0024|   JB 0E7H,0F9H
0027*   DEC 3FH
0029|   MOV A,3DH
002B|   JZ 0FCH
002D|   JB 0E7H,0F9H
0030|   DEC 3DH
0032|   SETB 18H
0034|   JBC 0AFH,02H
0037|   CLR 18H
0039|   MOV A,3BH
003B|   ADD A,#20H
003D|   MOV R0,A
003E|   MOV @R0,3CH
0040|   MOV R6,3BH
0042|   MOV R7,#00H
0044|   MOV 82H,R6
0046|   MOV 83H,R7
0048|   INC DPTR
```

```
SBUF R/O 0x00  W/O 0x43    TH0 0x04  TL0 0x13    TMOD 0x20  TCON 0xD0   SCON 0x52
R7 0x00  R6 0x01  R5 0x44  R4 0x00  R3 0x00  R2 0x00  R1 0x00  R0 0x02
B 0x00  ACC 0x03  PSW 0x88  IP 0x00  IE 0x82  PCON 0x00  DPH 0x00  DPL 0x00  SP 0x4F
TH1 0xFA  TL1 0xFA
P3 0xFF 0xFF  P2 0xFF 0xFF  P1 0xFF 0xFF  P0 0xFF 0xFF
PC 0x0027   PSW 1 0 0 0 1 0 0 0

Data Memory
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 30 31 00 00 00 00 02 00 02 00 00 00 00 44 01 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 41 42 43 03 00 00 00 00 00 00 01 4F 41 02 00 00
30 46 56 00 00 03 01 00 00 00 00 00 00 44 01 00 03
40 82 00 00 00 00 00 80 CC 02 7D 03 00 00 00 00 00
50 20 00 00 00 00 00 88 CC 02 7D 03 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

Rx: ABC
```

Figure 7 before the next round of writing the nextChar in the buffer

As figure 7 shown, the semaphores will be reset back to the original state (as shown figure 3) after the consumer is done consuming all the elements in the buffer.

Consumer is running and show semaphore changes



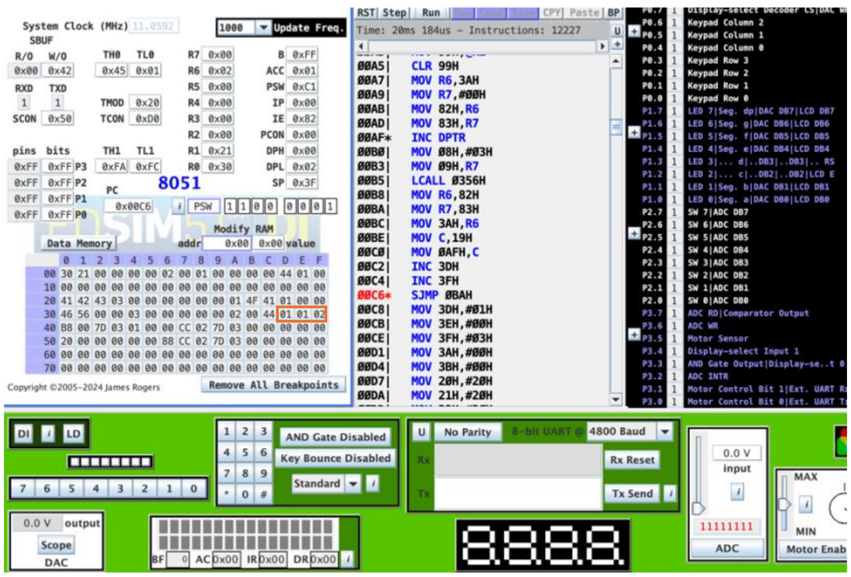
```
System Clock (MHz) 11.0592        1000  Update Freq.
SBUF
R/O   W/O      TH0   TL0     R7 0x00      B    0xFE
0x00  0x41     0x0E  0x19    R6 0x01      ACC  0x01
RXD   TXD                    R5 0x00      PSW  0xC1
1     0        TMOD  0x20    R4 0x00      IP   0x00
SCON  0x50     TCON  0xD0    R3 0x00      IE   0x82
                             R2 0x00      PCON 0x00
pins  bits     TH1   TL1     R1 0x20      DPH  0x00
0xFD  0xFD P3  0xFA  0xFA    R0 0x30      DPL  0x01
0xFF  0xFF P2           8051              SP   0x3F
0xFF  0xFF P1  PC
0xFF  0xFF P0  0x00C6        PSW 1 1 0 0  0 0 0 1
                              Modify RAM
Data Memory              addr 0x00  0x00 value

    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 30 20 00 00 00 00 01 00 01 00 00 00 00 44 01 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 41 42 43 03 00 00 00 00 00 00 01 4F 41 01 00 00
30 46 56 00 00 03 00 00 00 00 00 01 00 44 01 02 01
40 B8 00 7D 03 01 00 00 CC 02 7D 03 00 00 00 00 00
50 20 00 00 00 00 00 88 CC 02 7D 03 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
```

```
Time: 18ms 301us - Instructions: 11334
00A5|   CLR 99H
00A7|   MOV R6,3AH
00A9|   MOV R7,#00H
00AB|   MOV 82H,R6
00AD|   MOV 83H,R7
00AF*   INC DPTR
00B0|   MOV 08H,#03H
00B3|   MOV 09H,R7
00B5|   LCALL 0356H
00B8|   MOV R6,82H
00BA|   MOV R7,83H
00BC|   MOV 3AH,R6
00BE|   MOV C,19H
00C0|   MOV 0AFH,C
00C2|   INC 3DH
00C4|   INC 3FH
00C6*   SJMP 0BAH
00C8|   MOV 3DH,#01H
00CB|   MOV 3EH,#00H
00CE|   MOV 3FH,#03H
00D1|   MOV 3AH,#00H
00D4|   MOV 3BH,#00H
00D7|   MOV 20H,#20H
00DA|   MOV 21H,#20H
```

Figure 8 consume the first character

After the producer finish writing characters in the buffer. The semaphores indicate that the buffer is currecntly full: 0x3E (full) = 3 and 0x3F (empty) = 0.

The firgure 8 shows when the consumer consumes the first character from the buffer, then the sephamore full will decrease by 1 (0x3E (full) = 2) and sephamore empty will increase by 1 (0x3F (empty) = 1).

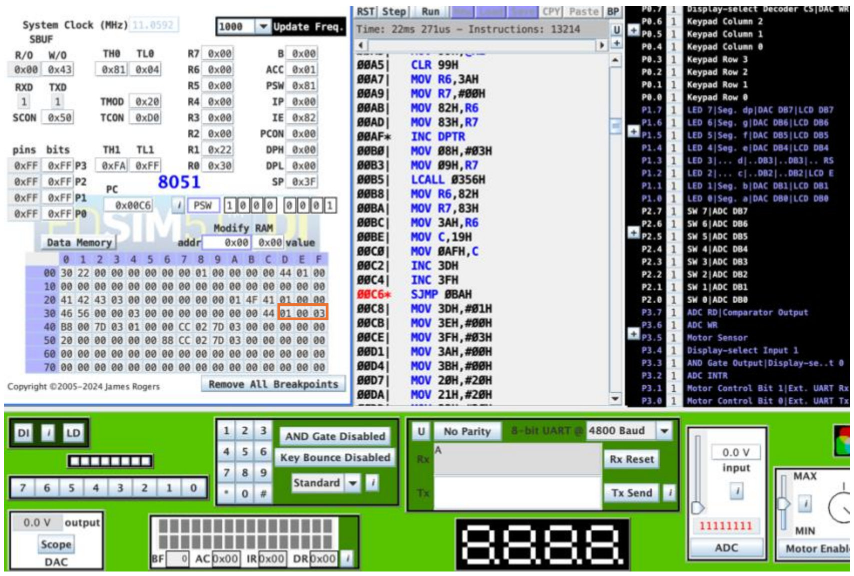


```
System Clock (MHz) 11.0592          1000  Update Freq.
SBUF
R/O   W/O      TH0   TL0     R7 0x00      B    0xFF
0x00  0x42     0x45  0x01    R6 0x02      ACC  0x01
RXD   TXD                    R5 0x00      PSW  0xC1
1     1        TMOD  0x20    R4 0x00      IP   0x00
SCON  0x50     TCON  0xD0    R3 0x00      IE   0x82
                             R2 0x00      PCON 0x00
pins  bits     TH1   TL1     R1 0x21      DPH  0x00
0xFF  0xFF P3  0xFA  0xFC    R0 0x30      DPL  0x02
0xFF  0xFF P2                8051         SP   0x3F
0xFF  0xFF P1     PC
0xFF  0xFF P0     0x00C6        PSW 1 1 0 0  0 0 0 1

Data Memory            Modify RAM addr 0x00  0x00 value
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 30 21 00 00 00 00 02 00 01 00 00 00 00 44 01 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 41 42 43 03 00 00 00 00 00 00 01 4F 41 01 00 00
30 46 56 00 00 03 00 00 00 00 00 02 00 44 01 01 02
40 B8 00 7D 03 01 00 00 CC 02 7D 03 00 00 00 00 00
50 20 00 00 00 00 00 88 CC 02 7D 03 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
```

```
Time: 20ms 184us - Instructions: 12227
00A5|   CLR 99H
00A7|   MOV R6,3AH
00A9|   MOV R7,#00H
00AB|   MOV 82H,R6
00AD|   MOV 83H,R7
00AF*   INC DPTR
00B0|   MOV 08H,#03H
00B3|   MOV 09H,R7
00B5|   LCALL 0356H
00B8|   MOV R6,82H
00BA|   MOV R7,83H
00BC|   MOV 3AH,R6
00BE|   MOV C,19H
00C0|   MOV 0AFH,C
00C2|   INC 3DH
00C4|   INC 3FH
00C6*   SJMP 0BAH
00C8|   MOV 3DH,#01H
00CB|   MOV 3EH,#00H
00CE|   MOV 3FH,#03H
00D1|   MOV 3AH,#00H
00D4|   MOV 3BH,#00H
00D7|   MOV 20H,#20H
00DA|   MOV 21H,#20H
```

Figure 9 consume the second character

The firgure 9 shows when the consumer consumes the second character from the buffer, then the sephamore full will decrease by 1 (0x3E (full) = 1) and sephamore empty will increase by 1 (0x3F (empty) = 2).



```
System Clock (MHz) 11.0592          1000  Update Freq.
SBUF
R/O   W/O      TH0   TL0     R7 0x00      B    0x00
0x00  0x43     0x81  0x04    R6 0x00      ACC  0x01
RXD   TXD                    R5 0x00      PSW  0x81
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 0x22      DPH  0x00
0xFF  0xFF P3  0xFA  0xFF    R0 0x30      DPL  0x00
0xFF  0xFF P2                8051         SP   0x3F
0xFF  0xFF P1     PC
0xFF  0xFF P0     0x00C6        PSW 1 0 0 0  0 0 0 1

Data Memory            Modify RAM addr 0x00  0x00 value
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 30 22 00 00 00 00 00 00 01 00 00 00 00 44 01 00
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20 41 42 43 03 00 00 00 00 00 00 01 4F 41 01 00 00
30 46 56 00 00 03 00 00 00 00 00 00 00 44 01 00 03
40 B8 00 7D 03 01 00 00 CC 02 7D 03 00 00 00 00 00
50 20 00 00 00 00 00 88 CC 02 7D 03 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
```

```
Time: 22ms 271us - Instructions: 13214
00A5|   CLR 99H
00A7|   MOV R6,3AH
00A9|   MOV R7,#00H
00AB|   MOV 82H,R6
00AD|   MOV 83H,R7
00AF*   INC DPTR
00B0|   MOV 08H,#03H
00B3|   MOV 09H,R7
00B5|   LCALL 0356H
00B8|   MOV R6,82H
00BA|   MOV R7,83H
00BC|   MOV 3AH,R6
00BE|   MOV C,19H
00C0|   MOV 0AFH,C
00C2|   INC 3DH
00C4|   INC 3FH
00C6*   SJMP 0BAH
00C8|   MOV 3DH,#01H
00CB|   MOV 3EH,#00H
00CE|   MOV 3FH,#03H
00D1|   MOV 3AH,#00H
00D4|   MOV 3BH,#00H
00D7|   MOV 20H,#20H
00DA|   MOV 21H,#20H
```

```
Rx: A
```

Figure 10 consume the third character

The firgure 10 shows when the consumer consumes the third character from the buffer, then the sephamore full will decrease by 1 (0x3E (full) = 0) and sephamore empty will increase by 1 (0x3F (empty) = 3). Indicating the buffer is empty.