Code explanation

Test3threads.c:

I implemented this code similar to the testpreempt.c so I would like to explain only the difference parts.

Global varaibles:

- **isConsumer (0x25) and producerNum (0x26):**
  - These two new varibales are used to switching between 2 producers

Producer and Consumer:

- So, I have changed from one producer to 2 producers (Producer1 and Producer2). I used to code from the last checkpoint and implement Producer2 by changing ‘A’ to ‘Z’ to ‘0’ to ‘9’.
- Consumer remains unchanged

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

- isConsumer = 0 and producerNum as 1.

Preemptive.c:

I added if else condition to toggle between 2 producers:

- **if (isConsumer)**: If the current thread is the **consumer**, it sets currThread to 0, indicating the consumer thread.
- **else**: If the current thread is not the consumer, the function switches between two **producer threads** (identified by producerNum). It sets currThread to the value of producerNum and alternates producerNum between 1 and 2 (thus switching between two producer threads in a round-robin manner).
- **isConsumer = !isConsumer;**: After handling the consumer, the function toggles the isConsumer flag to switch between consumer and producer threads on the next context switch.

So to be concluded, I assign one thread per consumer (thread 0) and producers (thread 1 and 2).

Screenshots for compilation

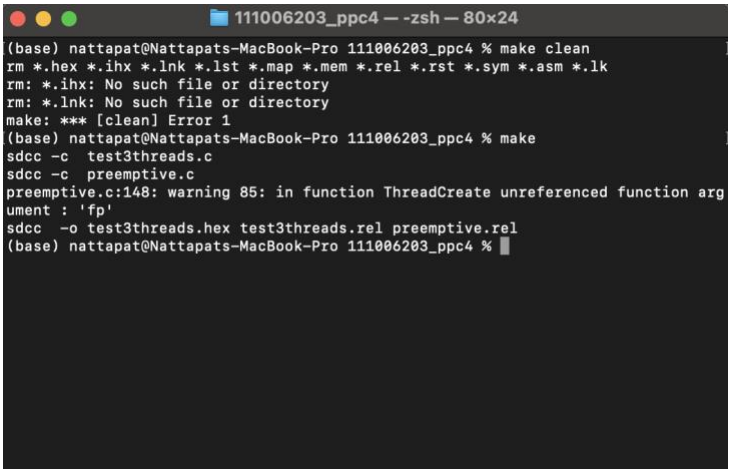

Figure 1 Screenshot for compilation

Screenshots and explanation

Producer1 running and show semaphore changes.

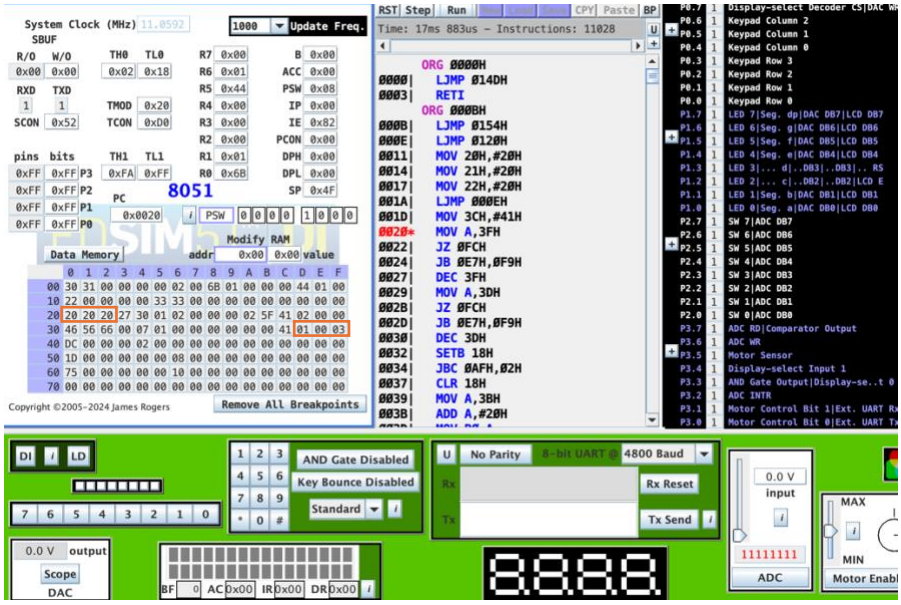

Figure 2 Initializing semaphores Producer 1

As figure 2 shown, buffer is in address: 0x20, 0x21 and 0x22. The semaphores are initialized in the main function as 0x3D (mutex) = 1, 0x3E (full) = 0, and 0x3F (empty) = 3. Indicating the buffer is empty.

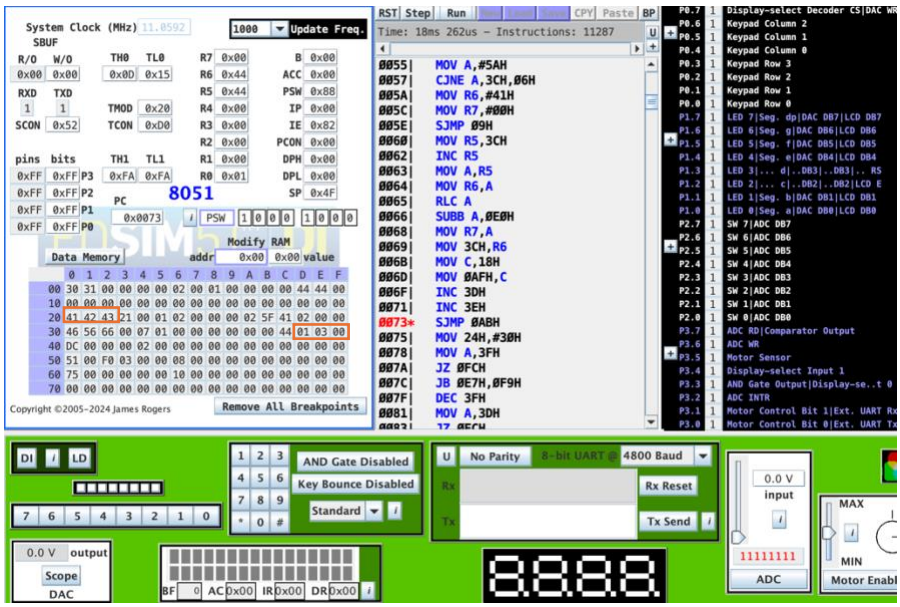

Figure 3 Producer 1 running

As figure 3 shown, after the producer 1 finish writing characters to buffer (buffer = full), the addresses of the buffer will hold a value of ‘A’ (address 0x20), ‘B’(address 0x21), and ‘C’(address 0x22).

And the value of the semaphores will be change from (0x3E (full) = 0) and (0x3F (empty) = 3) to (0x3E (full) = 3) and (0x3F (empty) = 0), indicating the the buffer is full.

Producer2 running and show semaphore changes.

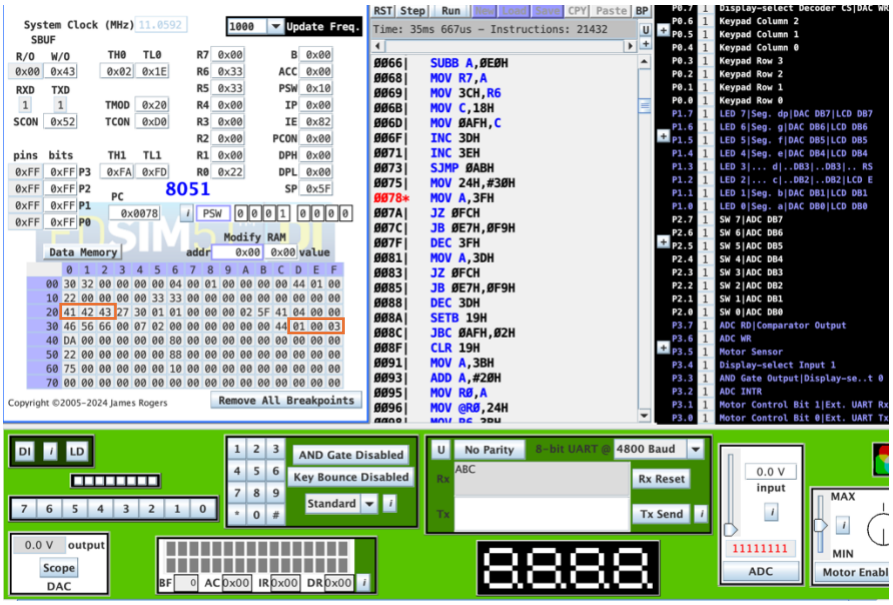

Figure 4 Initializing semaphores Producer 2

As figure 4 shown, buffer is in address: 0x20, 0x21 and 0x22. The semaphores are initialized in the main function as 0x3D (mutex) = 1, 0x3E (full) = 0, and 0x3F (empty) = 3. Indicating the buffer is empty.

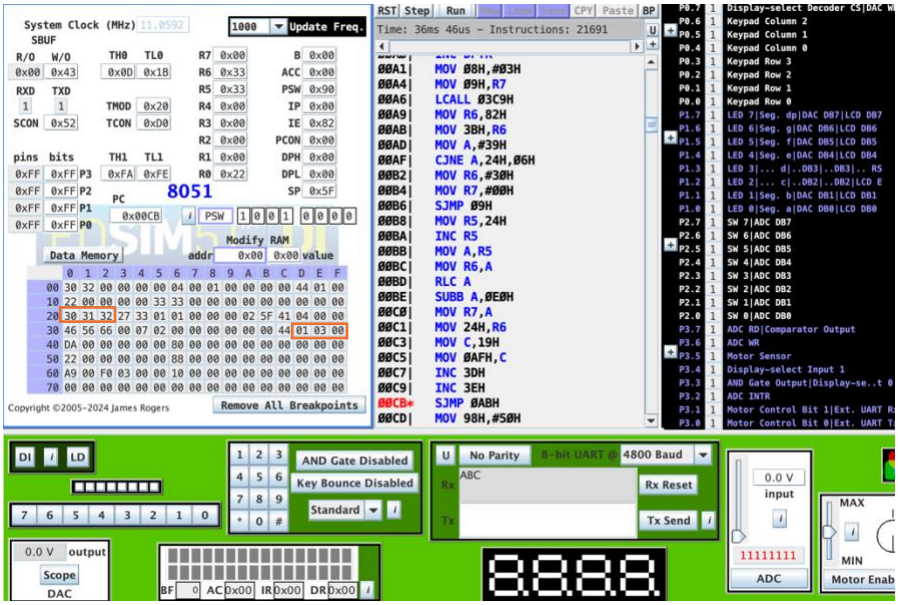

Figure 5 Producer 2 running

As figure 5 shown, after the producer 2 finish writing characters to buffer (buffer = full), the addresses of the buffer will hold a value of ‘0’ (address 0x20), ‘1’(address 0x21), and ‘2’(address 0x22).

And the value of the semaphores will be change from (0x3E (full) = 0) and (0x3F (empty) = 3) to (0x3E (full) = 3) and (0x3F (empty) = 0), indicating the the buffer is full.

Consumer running and show semaphore changes.

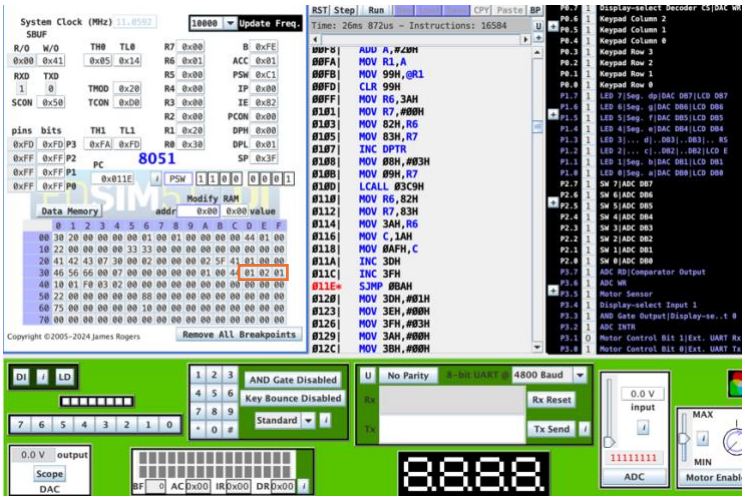

Figure 6 consume the first character

After one of 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 6 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).

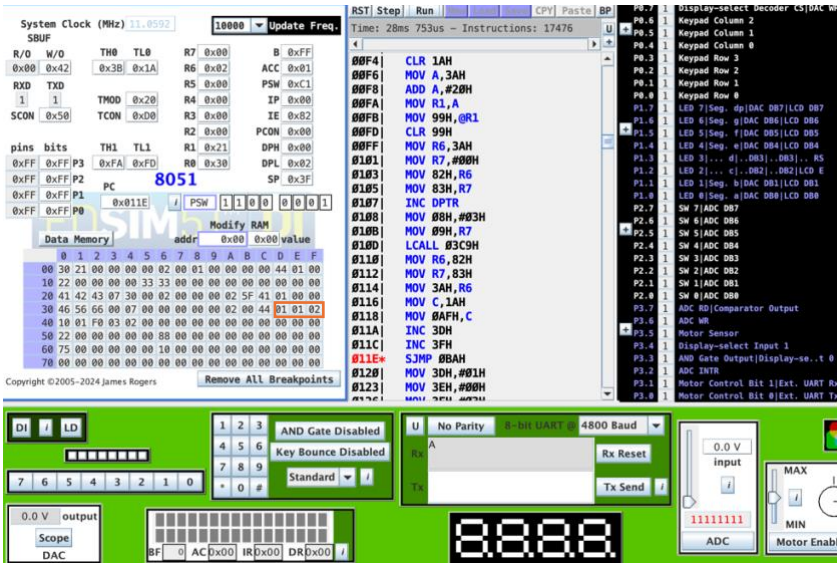

Figure 7 consume the second character

The firgure 7 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).

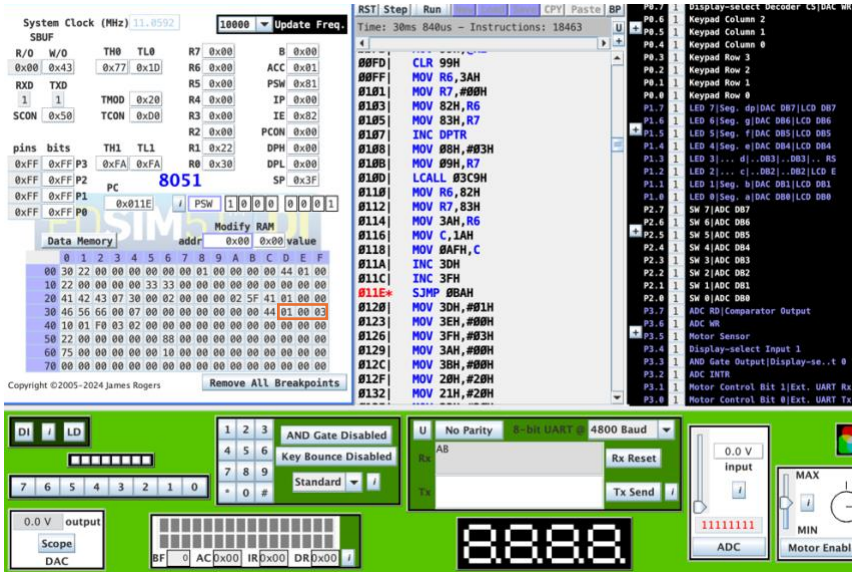

Figure 8 consume the third character

The firgure 8 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.

Show and explain UART output to show the unfair version, if any, and the fair version.

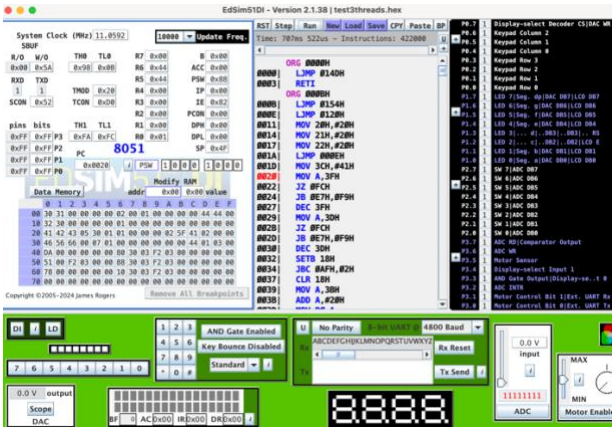

Figure 9 UART output unfair version

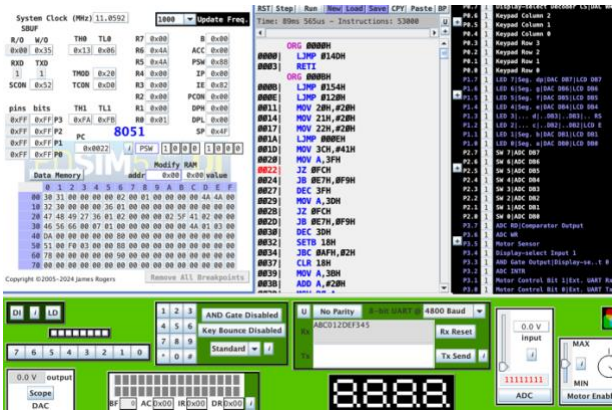

Figure 10 UART output fair version

As shown in figure 9, in the unfair version, I have implemented the round-robin scheduling policy with two producers, however it results in prioritizing the Letter Producer before the Number Producer causes the Letter Producer to fill the buffer.

However in the figure 10, when it switches to the Number Producer (between 1 and 2), it gets blocked due to the empty semaphore. By implementing a fair version (Producer1 -> Consumer, Producer2 -> Consumer), this issue is resolved, ensuring fairness in the program.