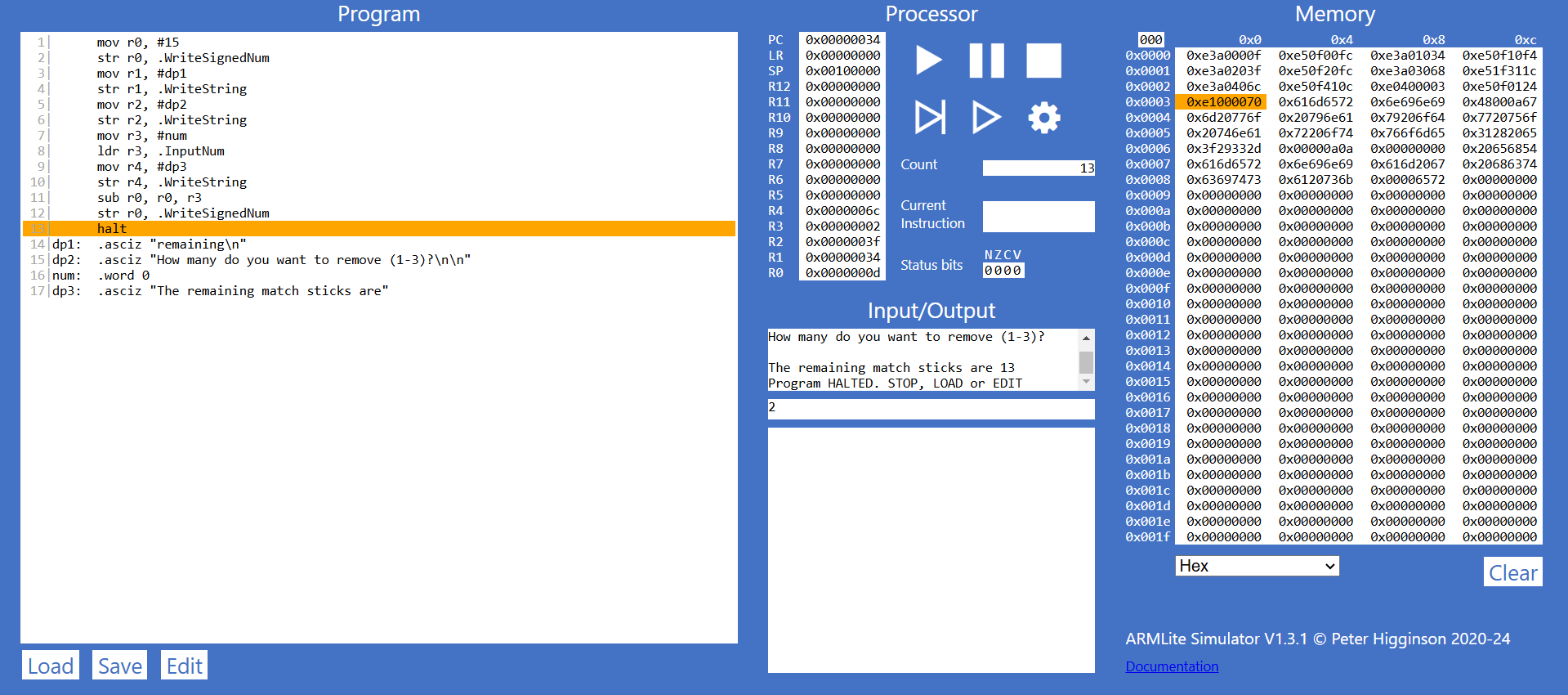
**Faculty of Science, Engineering and Technology**



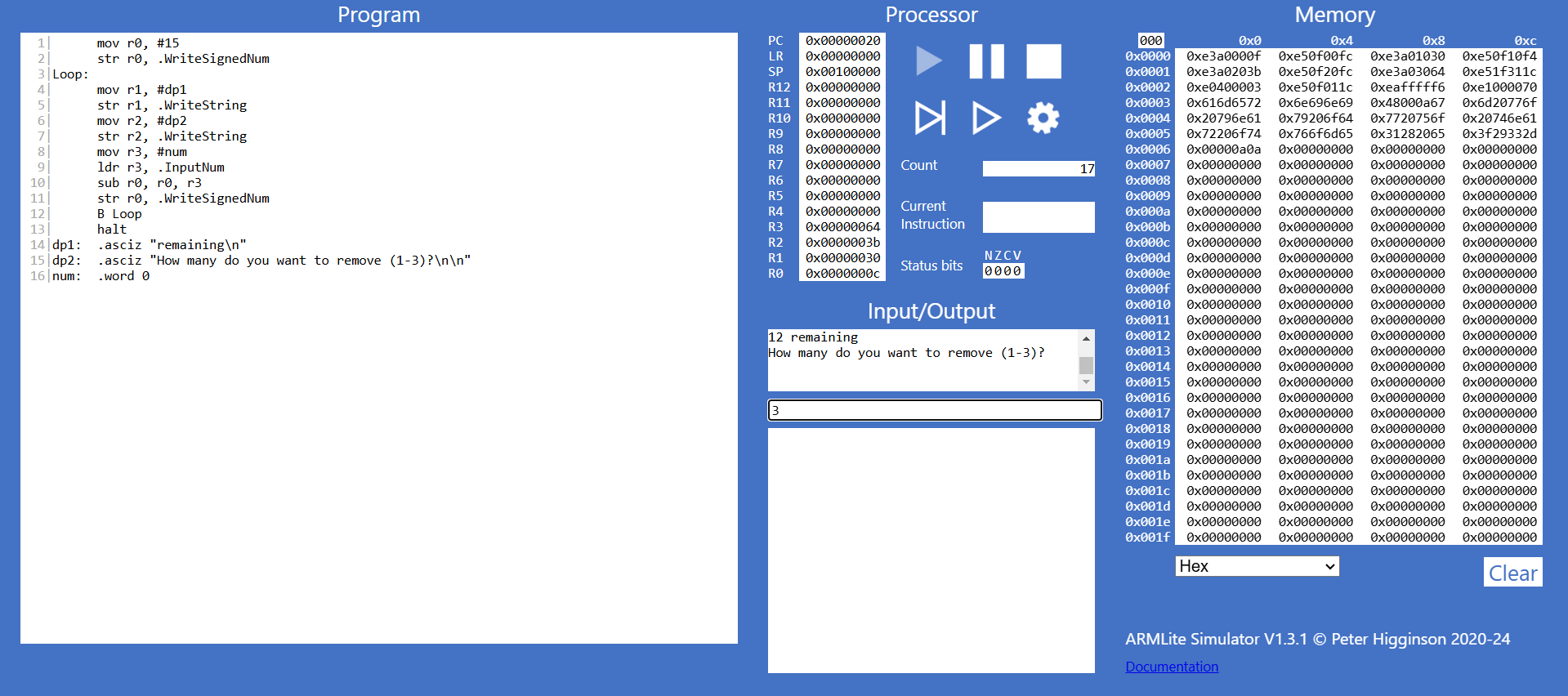
**Computer Systems**

***Week 8***

***Part 8.1***



***Part 8.2***

***8.2.1***

==> When you enter a number that causes the matchsticks to go negative, the following can happen:

- Negative Value: If you subtract a larger number than the remaining matchsticks, the value will become negative (e.g., 2 - 3 = -1).

- Program Behavior: If there’s no check, the program will continue running and display the negative number, which can confuse players. The game-over condition may not trigger correctly if it only checks for matchsticks <= 0.

- Register Values: In low-level programming, if the matchsticks value is stored in a register, it may lead to unexpected behavior with negative values or underflow in unsigned types.

***8.2.2(a)***

==> Condition: The user input must be less than 1 or greater than 3.

This means that if the user enters a number that is either less than 1 or greater than 3, the program should continue to prompt them for a valid input until they provide a number within the acceptable range (1 to 3).

***8.2.2(b)***

==> 2 intructions:

- CMP: Compare the user input with 1.

- BLT: Branch if the input is less than 1.

Or

- CMP: Compare the user input with 3.

- BGT: Branch if greater than 3.

***8.2.2(c)***

==> If the loop is to repeat due to invalid user input, the N (Negative) bit would be set to 1 if the input is less than 1. For the condition of being greater than 3, the loop would not set a specific bit indicating a repeat, but the Z (Zero) bit would be set to 0.

In summary, the N (Negative) bit indicates that the input is invalid when it is less than 1.

***8.2.2(d)***

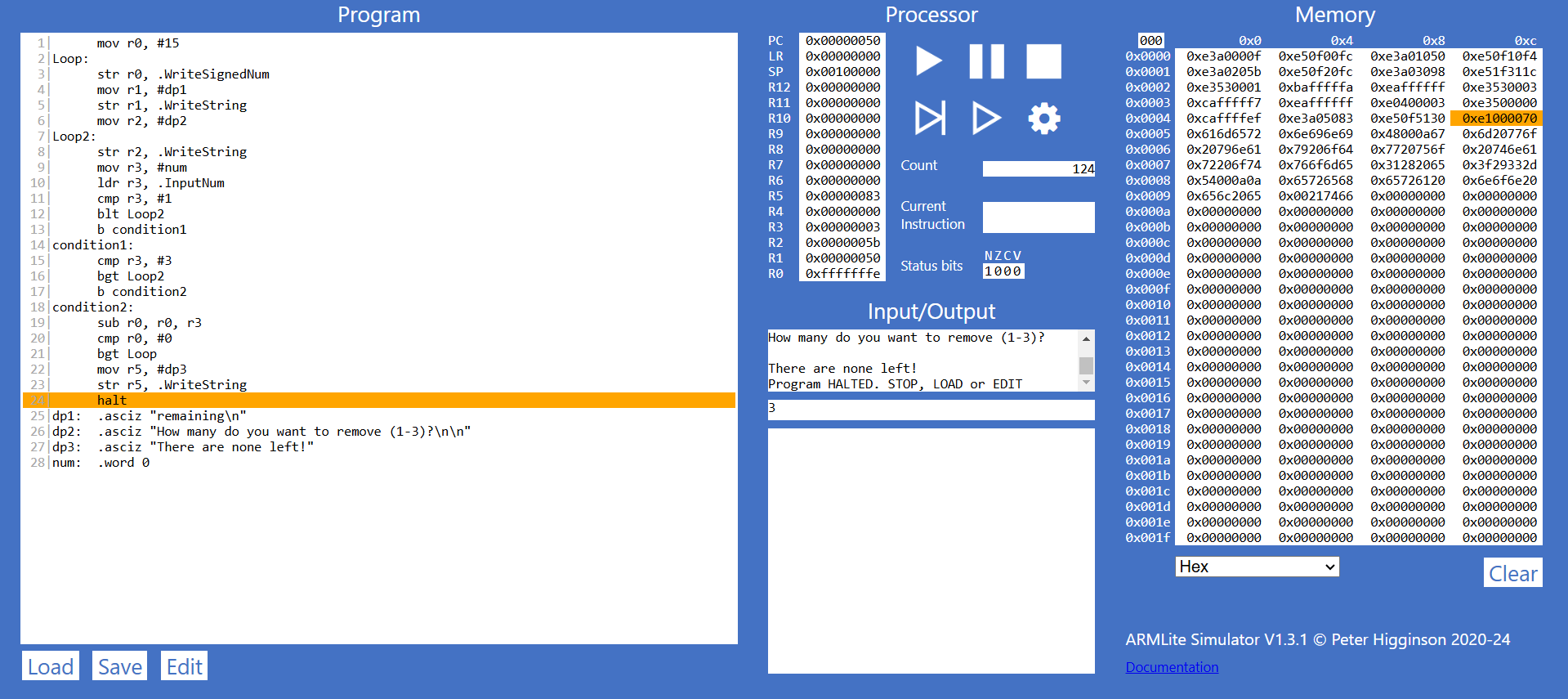
==> Modifications Needed:

- Input Handling: Ensure the program correctly reads user input.

- Comparison Logic: Add comparisons to check if the input is less than 1 or greater than 3.

- Loop Structure: Implement a loop that continues to prompt for input until valid input (1 to 3) is received.

- Status Bit Checks: Use status bits to determine if the loop should repeat based on the comparisons.



***Part 8.3***

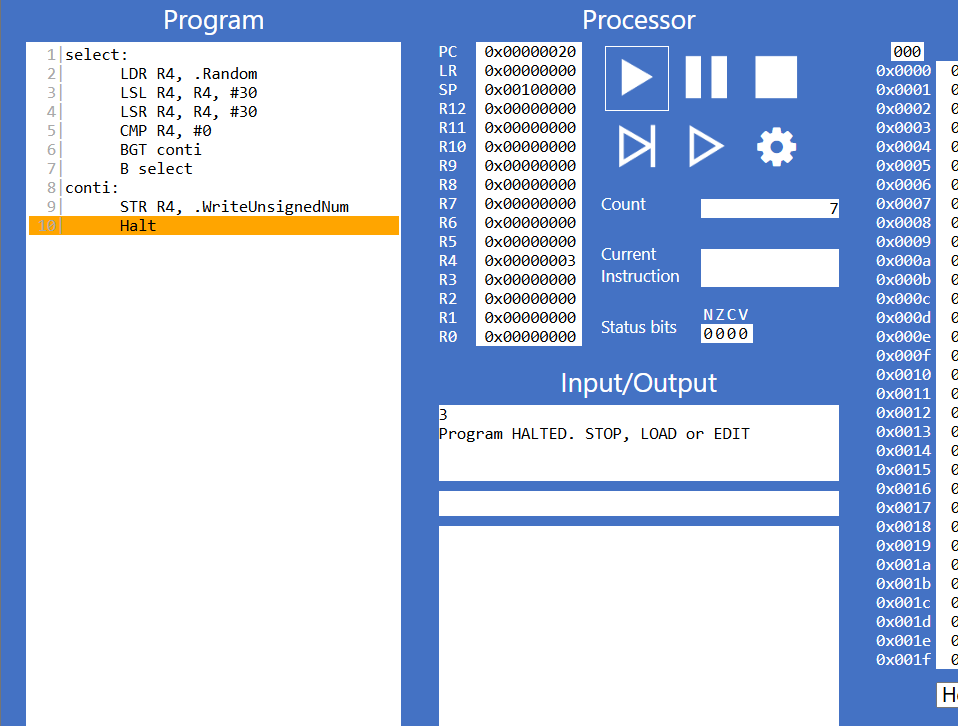
***8.3.1(a)***

==>To isolate the least significant 2 bits of a 32-bit pattern in a register (let's say the pattern is stored in register R2), you can use a bitwise AND operation with a mask that has the least significant 2 bits set to 1 and all other bits set to 0. The mask for this operation is 0b00000000000000000000000000000011 or 0x3.

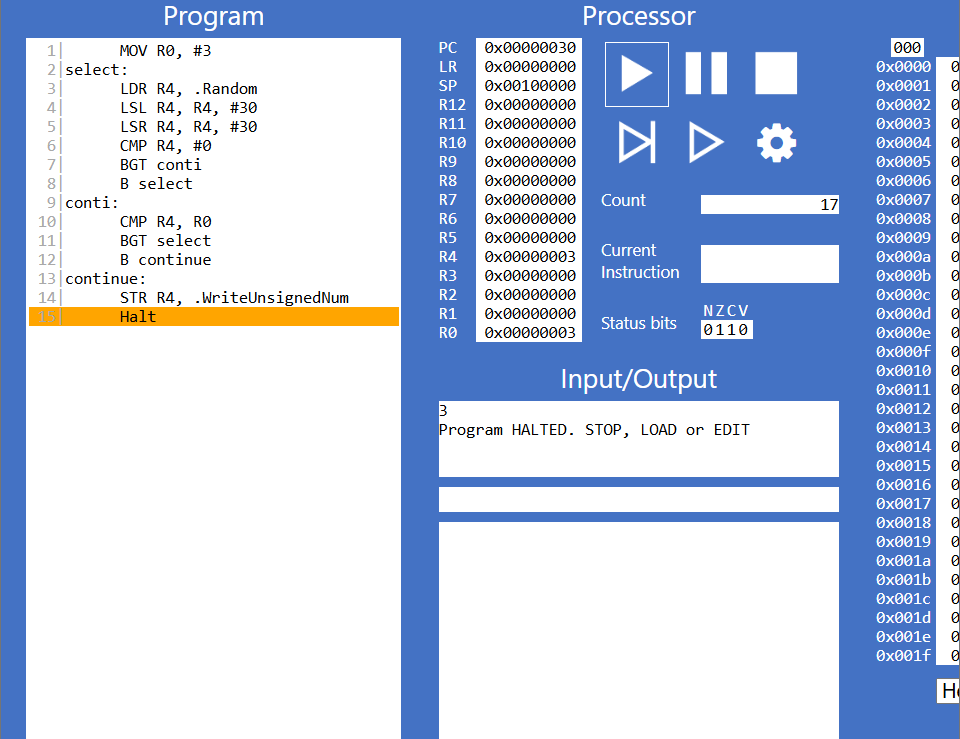
LSL R4, R4, #30

LSR R4, R4, #30

***8.3.1(b)***



***8.3.2***



***Part 8.4***

