

8.1.1/

```

1|  MOV R0, #15
2|  STR R0, .WriteUnsignedNum
3|  MOV R0, #msg1
4|  STR R0, .WriteString
5|  HALT
6|msg1: .ASCIZ "remaining\n"

```

The screenshot shows the ARM simulator interface. On the left, the 'Program' window displays the assembly code for 8.1.1, with line 5 (HALT) highlighted. On the right, the 'Processor' window shows the state of the processor registers (PC, LR, SP, R12-R15) and the status bits (NZCV). The 'Input/Output' window shows the output '15 remaining' and the message 'Program HALTED. STOP, LOAD or EDIT'.

8.1.2/

```

1|  MOV R0, #15
2|  STR R0, .WriteUnsignedNum
3|  MOV R0, #msg1
4|  STR R0, .WriteString
5|  MOV R0, #msg2
6|  STR R0, .WriteString
7|  LDR R1, .InputNum
8|  HALT
9|msg1: .ASCIZ "remaining\n"
10|msg2: .ASCIZ "How many do you want to remove (1-3)?"

```

The screenshot shows the ARM simulator interface. On the left, the 'Program' window displays the assembly code for 8.1.2, with line 8 (HALT) highlighted. On the right, the 'Processor' window shows the state of the processor registers (PC, LR, SP, R12-R15) and the status bits (NZCV). The 'Input/Output' window shows the output '15 remaining' and the message 'How many do you want to remove (1-3)?' followed by 'Modifying Program Area' and the input '2'.

8.1.3/

```

1|  MOV R0, #15
2|  STR R0, .WriteUnsignedNum
3|  MOV R1, #msg1
4|  STR R1, .WriteString
5|  MOV R1, #msg2
6|  STR R1, .WriteString
7|  LDR R2, .InputNum
8|  SUB R0, R0, R2
9|  STR R0, .WriteUnsignedNum
10| MOV R1, #msg1
11| STR R1, .WriteString
12| HALT
13|msg1: .ASCIZ "remaining\n"
14|msg2: .ASCIZ "How many do you want to remove
(1-3)?\n"

```

The screenshot displays a debugger interface with three main panels: Program, Processor, and Memory.

- Program Panel:** Shows assembly code starting at address 0x00000000. The code includes instructions for moving values into registers, storing them to memory, and performing arithmetic. The instruction at address 0x00000009, `HALT`, is highlighted in orange. Below the code, two messages are defined: `msg1: .ASCIZ "remaining\n"` and `msg2: .ASCIZ "How many do you want to remove (1-3)?\n"`.
- Processor Panel:** Shows the state of the processor registers. The PC (Program Counter) register is at address 0x00000009 and contains the value 0x00000000. Other registers like LR, SP, R12, R11, R10, R9, R8, R7, R6, R5, R4, R3, R2, R1, and R0 are also listed with their addresses and values.
- Memory Panel:** Shows the memory contents. The address 0x00000009 is highlighted, and the value stored at this address is 0x00000000. The panel also shows the status bits (NZCV) and the input/output window.

8.2.1/

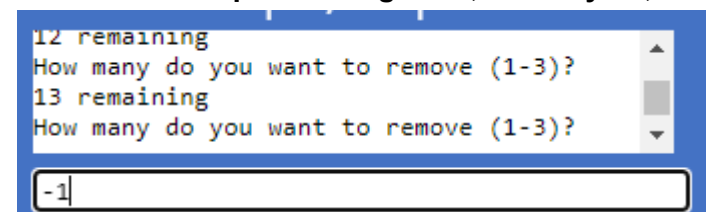
```

1|  MOV R0, #15
2|  STR R0, .WriteUnsignedNum
3|  MOV R1, #msg1
4|  STR R1, .WriteString
5| Loop:
6|  MOV R1, #msg2
7|  STR R1, .WriteString
8|  LDR R2, .InputNum
9|  SUB R0, R0, R2
10| STR R0, .WriteUnsignedNum
11| MOV R1, #msg1
12| STR R1, .WriteString
13| B Loop
14| HALT
15| msg1: .ASCIIZ "remaining\n"
16| msg2: .ASCIIZ "How many do you want to remove (1-3)?\n"

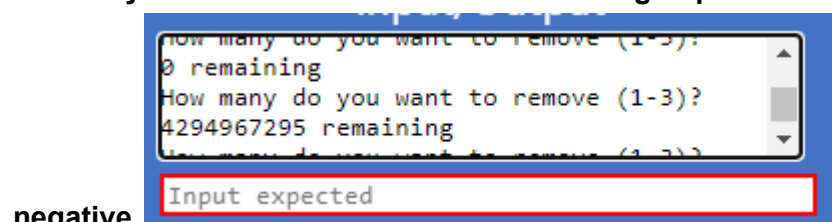
```

8.2.1/ What happens if you enter a number that takes the number of matchsticks remaining beyond 0 (i.e., into negative values) ?

If the number input is a negative , let's say -1 ; then the number of matchsticks would go up , eg. -1 into the program would bring it up



While if you take all the values then it would go up to the maximum number in 32-bit unsigned integer because it cannot display



negative

8.2.2/

(a) What is the condition that needs to be satisfied in order for this loop to occur ? Write this as a comparison using an inequality (ie., less than, greater than, less than or equal, greater than or equal)

$0 < R2 < 4$

(b) What two ARM assembly instructions could be used to create a branch that only occurs under this condition ?

Two assembly instructions could be used to create a branch that only occurs under the condition (BGT & BLT) , (BGT: Z clear , N and V being the same) , (BLT : where N and V are different).

(c) Based on the instructions you outlined in 8.2.2(b), what status bit would be set to 1 if the loop was to repeat ?

If the first condition is not met ($R2 > 0$) where R2 is negative then N = 1 ;

If the first condition is not met ($R2 > 0$) & $R2 = 0$, then Z = 1 ;

If the second condition is not met ($R2 < 4$) and ($R2 > 4$) then C = 1;

If the second condition is not met ($R2 > 0$) and ($R2 = 4$) then both Z and C = 1.

(d)

The image shows two side-by-side screenshots of an ARM simulator interface. Both screenshots display the same registers: R1 at 0x0000008e and R0 at 0x0000000c. The 'Status bits' section shows NZCV. In the left screenshot, NZCV is 0100. In the right screenshot, NZCV is 0110. Below the registers is an 'Input/Output' section with a text area containing the text: '12 remaining', 'How many do you want to remove (1-3)?', and 'Please input a valid number!'. At the bottom of each screenshot is an input field labeled 'Input expected'.

```

1|  MOV R0, #15
2|  STR R0, .WriteUnsignedNum
3|  MOV R1, #msg1
4|  STR R1, .WriteString
5| Loop:
6|  MOV R1, #msg2
7|  STR R1, .WriteString
8|  LDR R2, .InputNum
9| start:
10|  CMP R2, #0
11|  BGT else1
12|  B invalid1
13| else1:
14|  CMP R2, #4
15|  BLT cont
16|  B invalid1
17| invalid1:
18|  MOV R1, #msg3
19|  STR R1, .WriteString
20|  LDR R2, .InputNum
21|  B start
22| cont:
23|  SUB R0, R0, R2
24|  STR R0, .WriteUnsignedNum
25|  MOV R1, #msg1
26|  STR R1, .WriteString
27|  B Loop
28|  HALT
29| msg1: .ASCIZ "remaining\n"
30| msg2: .ASCIZ "How many do you want to remove
(1-3)?\n"
31| msg3: .ASCIZ "Please input a valid number!\n"

```

```

1|  MOV R0, #15
2|  STR R0, .WriteUnsignedNum
3|  MOV R1, #msg1
4|  STR R1, .WriteString
5| Loop:
6|  MOV R1, #msg2
7|  STR R1, .WriteString
8|  LDR R2, .InputNum
9| start:
10|  CMP R2, #0
11|  BGT else1
12|  B invalid1
13| else1:
14|  CMP R2, #4
15|  BLT cont
16|  B invalid1
17| invalid1:
18|  MOV R1, #msg3
19|  STR R1, .WriteString
20|  LDR R2, .InputNum
21|  B start
22| cont:
23|  SUB R0, R0, R2
24|  STR R0, .WriteUnsignedNum
25|  MOV R1, #msg1
26|  STR R1, .WriteString
27|  B Loop
28|  HALT
29| msg1: .ASCIZ "remaining\n"
30| msg2: .ASCIZ "How many do you want to
31| msg3: .ASCIZ "Please input a valid nu

```

PC	0x0000001c
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000000
R6	0x00000000
R5	0x00000000
R4	0x00000000
R3	0x00000000
R2	0x00000000
R1	0x00000067
R0	0x0000000f

▶ || ◻

⏮ ⏭ ⚙

Count

Current Instruction

Status bits **NZCV**

Input/Output

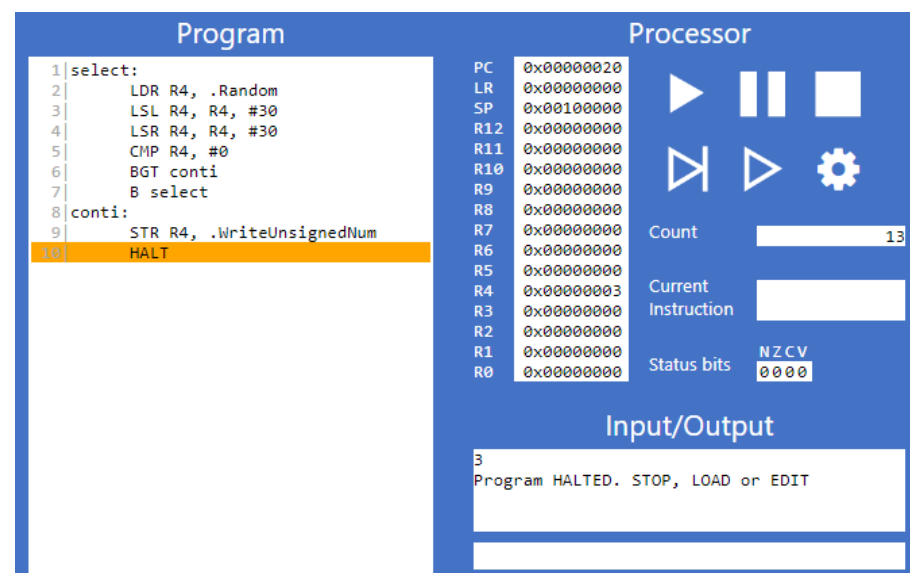
15 remaining
How many do you want to remove (1-3)?

8.3.1/

(a) What bit-wise operation can we perform on the register holding the 32 bit pattern to set all bits in the register to zero except the least significant 2 bits ? Write this as a single line of code.

```
3| LSL R4, R4, #30
4| LSR R4, R4, #30
```

(b) Using a label named "select:" Write the code needed to repeatedly sample a random number (from .Random) until the value is in the range 1-3. For now, just write this as a separate program and test it.

<pre>1 select: 2 LDR R4, .Random 3 LSL R4, R4, #30 4 LSR R4, R4, #30 5 CMP R4, #0 6 BGT conti 7 B select 8 conti: 9 STR R4, .WriteUnsignedNum 10 HALT</pre>	 <p>The screenshot shows a program editor on the left and a processor simulator on the right. The program editor displays the assembly code from the previous block, with line 10 (HALT) highlighted. The processor simulator shows the state of the processor, including registers (PC, LR, SP, R12-R0), status bits (NZCV), and input/output. The program has halted, and the output shows "Program HALTED. STOP, LOAD or EDIT".</p>
---	--

8.3.2/

1|
2|
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

MOV R0, #3
select:
LDR R4, .Random
LSL R4, R4, #30
LSR R4, R4, #30
CMP R4, #0
BGT conti
B select
conti:
CMP R4, R0
BGT select
B continue
continue:
STR R4, .WriteUnsignedNum
HALT

Program

1|
2|
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|

MOV R0, #3
select:
LDR R4, .Random
LSL R4, R4, #30
LSR R4, R4, #30
CMP R4, #0
BGT conti
B select
conti:
CMP R4, R0
BGT select
B continue
continue:
STR R4, .WriteUnsignedNum
HALT

Processor

PC
LR
SP
R12
R11
R10
R9
R8
R7
R6
R5
R4
R3
R2
R1
R0

0x00000030
0x00000000
0x00100000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000000
0x00000001
0x00000000
0x00000000
0x00000000
0x00000003

▶
⏸
◻

⏮
⏭
⚙

Count
17

Current
Instruction

Status bits
NZCV
1000

Input/Output

1
Program HALTED. STOP, LOAD or EDIT

8.4.1/

```

1|  MOV R0, #15
2|  STR R0, .WriteUnsignedNum
3|  MOV R1, #msg1
4|  STR R1, .WriteString
5| Loop:
6|  MOV R1, #msg2
7|  STR R1, .WriteString
8|  LDR R2, .InputNum
9| start:
10|  CMP R2, #0
11|  BGT else1      // checking if R2 > 0
12|  B invalid1
13| else1:
14|  CMP R2, #4
15|  BLT cont1      // checking if R2 < 4
16|  B invalid1
17| invalid1:
18|  MOV R1, #msg3
19|  STR R1, .WriteString
20|  LDR R2, .InputNum
21|  B start
22| cont1:
23|  CMP R0, R2
24|  BLT invalid1
25|  B cont2
26| cont2:
27|  SUB R0, R0, R2
28|  STR R0, .WriteUnsignedNum
29|  MOV R1, #msg1
30|  STR R1, .WriteString
31|  MOV R1, #msg5
32|  STR R1, .WriteString
33|  B select
34| select:

```

Program

```

1|  MOV R0, #15
2|  STR R0, .WriteUnsignedNum
3|  MOV R1, #msg1
4|  STR R1, .WriteString
5| Loop:
6|  MOV R1, #msg2
7|  STR R1, .WriteString
8|  LDR R2, .InputNum
9| start:
10|  CMP R2, #0
11|  BGT else1      // checking
12|  B invalid1
13| else1:
14|  CMP R2, #4
15|  BLT cont1      // checking
16|  B invalid1
17| invalid1:
18|  MOV R1, #msg3
19|  STR R1, .WriteString
20|  LDR R2, .InputNum
21|  B start
22| cont1:
23|  CMP R0, R2
24|  BLT invalid1
25|  B cont2
26| cont2:
27|  SUB R0, R0, R2
28|  STR R0, .WriteUnsignedNum
29|  MOV R1, #msg1
30|  STR R1, .WriteString
31|  MOV R1, #msg5
32|  STR R1, .WriteString
33|  B select
34| select:
35|  LDR R4, .Random
36|  LSL R4, R4, #30
37|  LSR R4, R4, #30
38|  CMP R4, #0
39|  BGT cont1

```

Processor

PC	0x0000001c
LR	0x00000000
SP	0x00100000
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000000
R6	0x00000000
R5	0x00000000
R4	0x00000003
R3	0x00000000
R2	0x00000001
R1	0x000000bb
R0	0x00000001

Count: 161668046

Current Instruction:

Status bits: NZCV 1000

Input/Output

How many do you want to remove (1-3)?

0 remaining

It's computer's turn!

1


```

35| LDR R4, .Random
36| LSL R4, R4, #30
37| LSR R4, R4, #30
38| CMP R4, #0
39| BGT conti
40| B select
41| conti:
42| CMP R4, R0
43| BGT select
44| B continue
45| continue:
46| SUB R0, R0, R4
47| STR R0, .WriteUnsignedNum
48| MOV R1, #msg1
49| STR R1, .WriteString
50| MOV R1, #msg4
51| STR R1, .WriteString
52| B Loop
53| HALT
54| msg1: .ASCIZ "remaining\n"
55| msg2: .ASCIZ "How many do you want to remove (1-3)?\n"
56| msg3: .ASCIZ "Please input a valid number!\n"
57| msg4: .ASCIZ "It's your turn!\n"
58| msg5: .ASCIZ "It's computer's turn!\n"

```

The screenshot displays the ARMLite Simulator V1.2.4 interface. The main window is divided into several sections:

- Program:** A list of assembly instructions with line numbers. The current instruction is at line 35: `LDR R4, .Random`.
- Processor:** A table showing the state of various processor registers (PC, LR, SP, R12, R11, R10, R9, R8, R7, R6, R5, R4, R3, R2, R1, R0) and their values in hexadecimal.
- Count:** A display showing the value 64908461.
- Current Instruction:** A display showing the current instruction being executed.
- Status bits:** A display showing the status bits NZCV and the value 1000.
- Input/Output:** A section for user input and output. It shows a prompt "How many do you want to remove (1-3)?" and the user's input "2". It also shows the output "It's computer's turn!".
- Hex:** A section for displaying memory addresses and their corresponding hexadecimal values.

The ARMLite Simulator V1.2.4 logo and a link to the documentation are visible in the bottom right corner.