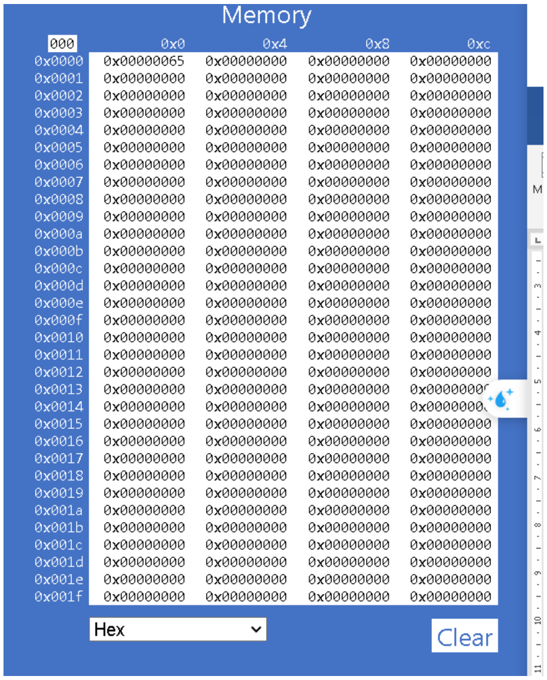
Lab 7

Student Name: Lau Ngoc Quyen

Student ID: 104198996

7.1.1 What value is displayed? Why?

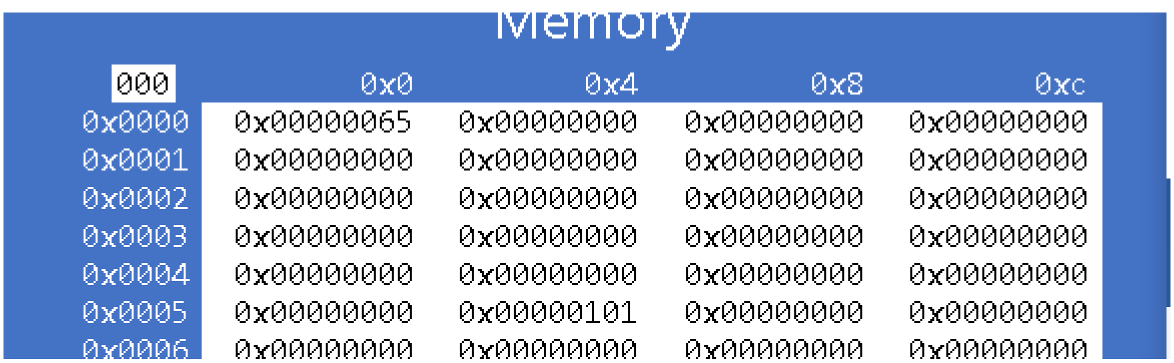
When enter 101 into a memory the value displayed will likely be 0x00000065. This is because 101 in decimal is equivalent to 65 in hexadecimal. The ARMLite simulator stores values in hexadecimal format.



000	0x0	0x4	0x8	0xc
0x0000	0x00000065	0x00000000	0x00000000	0x00000000
0x0001	0x00000000	0x00000000	0x00000000	0x00000000
0x0002	0x00000000	0x00000000	0x00000000	0x00000000
0x0003	0x00000000	0x00000000	0x00000000	0x00000000
0x0004	0x00000000	0x00000000	0x00000000	0x00000000
0x0005	0x00000000	0x00000000	0x00000000	0x00000000
0x0006	0x00000000	0x00000000	0x00000000	0x00000000
0x0007	0x00000000	0x00000000	0x00000000	0x00000000
0x0008	0x00000000	0x00000000	0x00000000	0x00000000
0x0009	0x00000000	0x00000000	0x00000000	0x00000000
0x000a	0x00000000	0x00000000	0x00000000	0x00000000
0x000b	0x00000000	0x00000000	0x00000000	0x00000000
0x000c	0x00000000	0x00000000	0x00000000	0x00000000
0x000d	0x00000000	0x00000000	0x00000000	0x00000000
0x000e	0x00000000	0x00000000	0x00000000	0x00000000
0x000f	0x00000000	0x00000000	0x00000000	0x00000000
0x0010	0x00000000	0x00000000	0x00000000	0x00000000
0x0011	0x00000000	0x00000000	0x00000000	0x00000000
0x0012	0x00000000	0x00000000	0x00000000	0x00000000
0x0013	0x00000000	0x00000000	0x00000000	0x0000000
0x0014	0x00000000	0x00000000	0x00000000	0x0000000
0x0015	0x00000000	0x00000000	0x00000000	0x00000000
0x0016	0x00000000	0x00000000	0x00000000	0x00000000
0x0017	0x00000000	0x00000000	0x00000000	0x00000000
0x0018	0x00000000	0x00000000	0x00000000	0x00000000
0x0019	0x00000000	0x00000000	0x00000000	0x00000000
0x001a	0x00000000	0x00000000	0x00000000	0x00000000
0x001b	0x00000000	0x00000000	0x00000000	0x00000000
0x001c	0x00000000	0x00000000	0x00000000	0x00000000
0x001d	0x00000000	0x00000000	0x00000000	0x00000000
0x001e	0x00000000	0x00000000	0x00000000	0x00000000
0x001f	0x00000000	0x00000000	0x00000000	0x00000000

7.1.2 What value is displayed, and why?

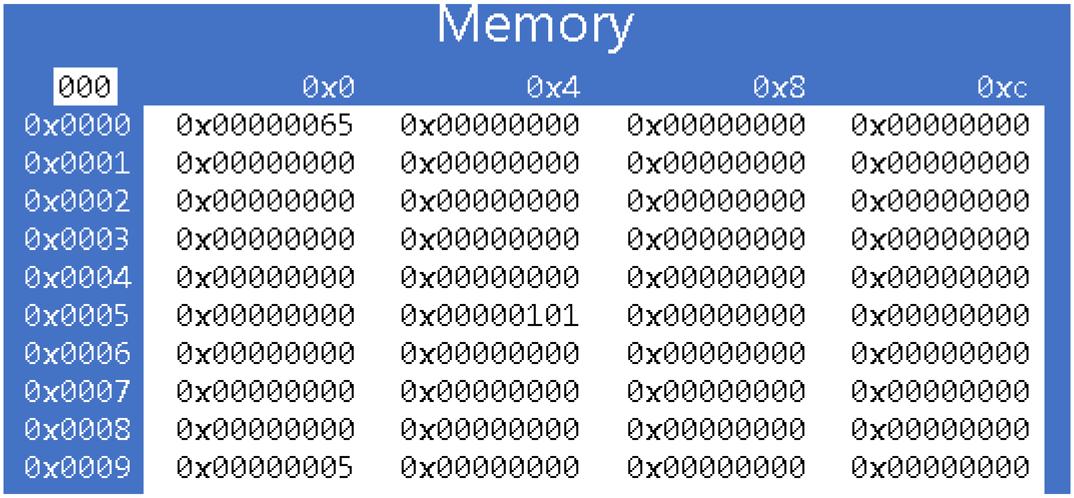
When enter 0x101 into a memory word, the value displayed will be 0x00000101. This is because 0x101 is already in hexadecimal format.



000	0x0	0x4	0x8	0xc
0x0000	0x00000065	0x00000000	0x00000000	0x00000000
0x0001	0x00000000	0x00000000	0x00000000	0x00000000
0x0002	0x00000000	0x00000000	0x00000000	0x00000000
0x0003	0x00000000	0x00000000	0x00000000	0x00000000
0x0004	0x00000000	0x00000000	0x00000000	0x00000000
0x0005	0x00000000	0x00000101	0x00000000	0x00000000
0x0006	0x00000000	0x00000000	0x00000000	0x00000000

7.1.3 What value is displayed, and why?

When enter 0b101 into a memory word, the value displayed will likely be 0x00000005. This is because 0b101 is in binary format, which is equivalent to 5 in decimal and 0x00000005 in hexadecimal.

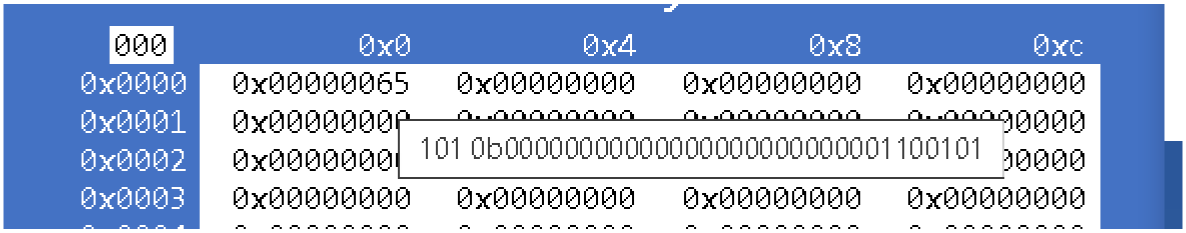


Memory

000	0x0	0x4	0x8	0xc
0x0000	0x00000065	0x00000000	0x00000000	0x00000000
0x0001	0x00000000	0x00000000	0x00000000	0x00000000
0x0002	0x00000000	0x00000000	0x00000000	0x00000000
0x0003	0x00000000	0x00000000	0x00000000	0x00000000
0x0004	0x00000000	0x00000000	0x00000000	0x00000000
0x0005	0x00000000	0x00000101	0x00000000	0x00000000
0x0006	0x00000000	0x00000000	0x00000000	0x00000000
0x0007	0x00000000	0x00000000	0x00000000	0x00000000
0x0008	0x00000000	0x00000000	0x00000000	0x00000000
0x0009	0x00000005	0x00000000	0x00000000	0x00000000

7.1.4 What does the tooltip tell you?

The tooltip likely tells the decimal value of the memory word that are hovering over. This can be helpful for understanding the value in a more familiar format.



000	0x0	0x4	0x8	0xc
0x0000	0x00000065	0x00000000	0x00000000	0x00000000
0x0001	0x0000000			00000
0x0002	0x0000000			00000
0x0003	0x00000000	0x00000000	0x00000000	0x00000000

101 0b00000000000000000000000001100101

Does changing the representation of the data in memory also change the representation of the row and column-headers? Should it?

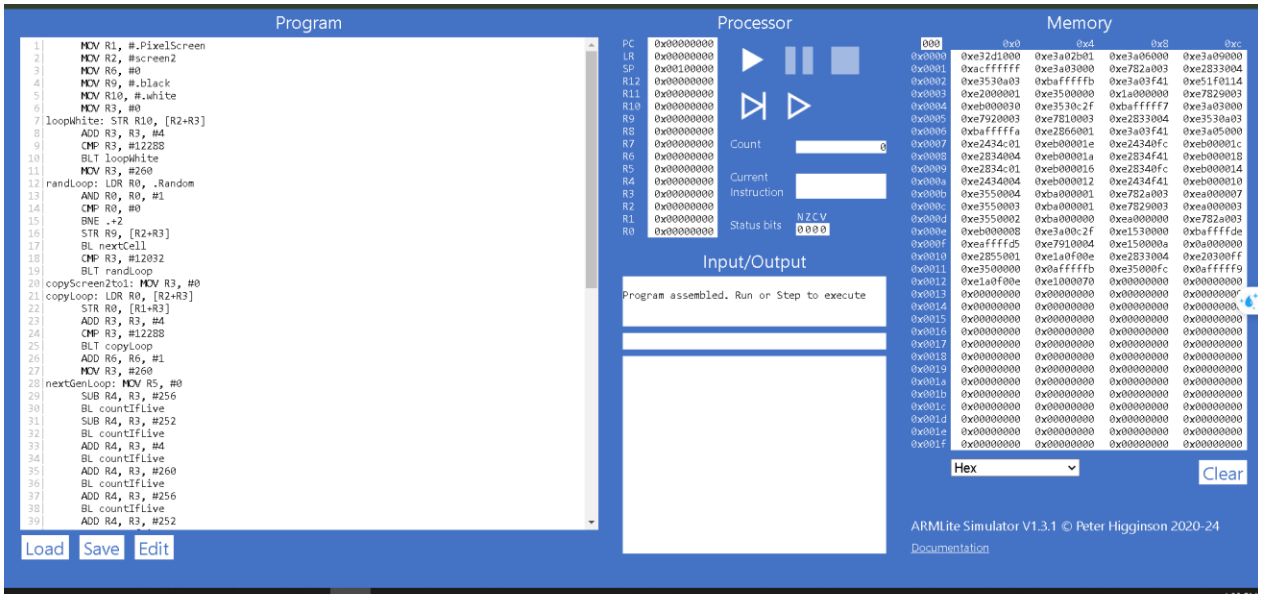
Changing the representation of the data in memory does not change the representation of the row and column-headers. This is because the row and column-headers represent the memory addresses, which are always displayed in hexadecimal format. It would not be useful to change the representation of the memory addresses, as this would make it difficult to identify specific locations in memory.

Question 7.2.1:

Why are the column headers in the ARMLite simulator increasing by 0x4?

The column header memory address offsets are in multiples of 0x4 because each memory word is 32 bits long, which is equivalent to 4 bytes.

7.3.1 Take a screen shot of the simulator in full and add it to your submission document



```
        MOV R1, #.PixelScreen
        MOV R2, #screen2
        MOV R6, #0
        MOV R9, #.black
        MOV R10, #.white
        MOV R3, #0
loopWhite: STR R10, [R2+R3]
        ADD R3, R3, #4
        CMP R3, #12288
        BLT loopWhite
        MOV R3, #260
randLoop: LDR R0, .Random
        AND R0, R0, #1
        CMP R0, #0
        BNE .+2
        STR R9, [R2+R3]
        BL nextCell
        CMP R3, #12032
        BLT randLoop
copyScreen2to1: MOV R3, #0
copyLoop: LDR R0, [R2+R3]
        STR R0, [R1+R3]
        ADD R3, R3, #4
        CMP R3, #12288
        BLT copyLoop
        ADD R6, R6, #1
        MOV R3, #260
nextGenLoop: MOV R5, #0
        SUB R4, R3, #256
        BL countIfLive
        SUB R4, R3, #252
        BL countIfLive
        ADD R4, R3, #4
        BL countIfLive
        ADD R4, R3, #260
        BL countIfLive
        ADD R4, R3, #256
        BL countIfLive
        ADD R4, R3, #252
```

```
PC   0x00000000
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   0x00000000
R0   0x00000000

Count                0
Current Instruction
Status bits  NZCV
             0000
```

```
Program assembled. Run or Step to execute
```

```
000            0x0         0x4         0x8         0xc
0x0000  0xe32d1000  0xe3a02b01  0xe3a06000  0xe3a09000
0x0001  0xacffffff  0xe3a03000  0xe782a003  0xe2833004
0x0002  0xe3530a03  0xbafffffb  0xe3a03f41  0xe51f0114
0x0003  0xe2000001  0xe3500000  0x1a000000  0xe7829003
0x0004  0xeb000030  0xe3530c2f  0xbafffff7  0xe3a03000
0x0005  0xe7920003  0xe7810003  0xe2833004  0xe3530a03
0x0006  0xbafffffa  0xe2866001  0xe3a03f41  0xe3a05000
0x0007  0xe2434c01  0xeb00001e  0xe24340fc  0xeb00001c
0x0008  0xe2834004  0xeb00001a  0xe2834f41  0xeb000018
0x0009  0xe2834c01  0xeb000016  0xe28340fc  0xeb000014
0x000a  0xe2434004  0xeb000012  0xe2434f41  0xeb000010
0x000b  0xe3550004  0xba000001  0xe782a003  0xea000007
0x000c  0xe3550003  0xba000001  0xe7829003  0xea000003
0x000d  0xe3550002  0xba000000  0xea000000  0xe782a003
0x000e  0xeb000008  0xe3a00c2f  0xe1530000  0xbaffffde
0x000f  0xeaffffd5  0xe7910004  0xe150000a  0x0a000000
0x0010  0xe2855001  0xe1a0f00e  0xe2833004  0xe20300ff
0x0011  0xe3500000  0x0afffffb  0xe35000fc  0x0afffff9
0x0012  0xe1a0f00e  0xe1000070  0x00000000  0x00000000
0x0013  0x00000000  0x00000000  0x00000000  0x00000000
0x0014  0x00000000  0x00000000  0x00000000  0x00000000
0x0015  0x00000000  0x00000000  0x00000000  0x00000000
0x0016  0x00000000  0x00000000  0x00000000  0x00000000
0x0017  0x00000000  0x00000000  0x00000000  0x00000000
0x0018  0x00000000  0x00000000  0x00000000  0x00000000
0x0019  0x00000000  0x00000000  0x00000000  0x00000000
0x001a  0x00000000  0x00000000  0x00000000  0x00000000
0x001b  0x00000000  0x00000000  0x00000000  0x00000000
0x001c  0x00000000  0x00000000  0x00000000  0x00000000
0x001d  0x00000000  0x00000000  0x00000000  0x00000000
0x001e  0x00000000  0x00000000  0x00000000  0x00000000
0x001f  0x00000000  0x00000000  0x00000000  0x00000000
```

ARMLite Simulator V1.3.1 © Peter Higginson 2020-24

7.3.2 Based on what we've learnt about assemblers and Von Neuman architectures, explain what you think just happened.

When submitted the assembly code, the assembler translated it into machine code. This machine code is then loaded into memory. The ARMLite simulator executes this machine code, which causes the memory contents to change.

7.3.3 Based on what we have learnt about memory addressing in ARMlite, and your response to 7.3.2, what do you think this value represents?

The value represents the memory address where the instruction is stored. This is because the ARMlite simulator uses a Von Neuman architecture, where instructions and data are stored in the same memory space.

7.4.1 What do you think the highlighting in both windows signifies?

The highlighting in both windows signifies the current line of code being executed and the corresponding memory address. This is helpful for debugging, as it allows you to see which part of the code is being executed and what data is being accessed.

7.4.2 What do you think happens when you click the button circled in red?

The button circled in red is the step-by-step button. When you click it, the simulator executes one instruction at a time. This is useful for debugging, as it allows you to carefully examine the state of the program after each instruction.

7.4.3 Has the processor paused just before or just after executing the line with the breakpoint?

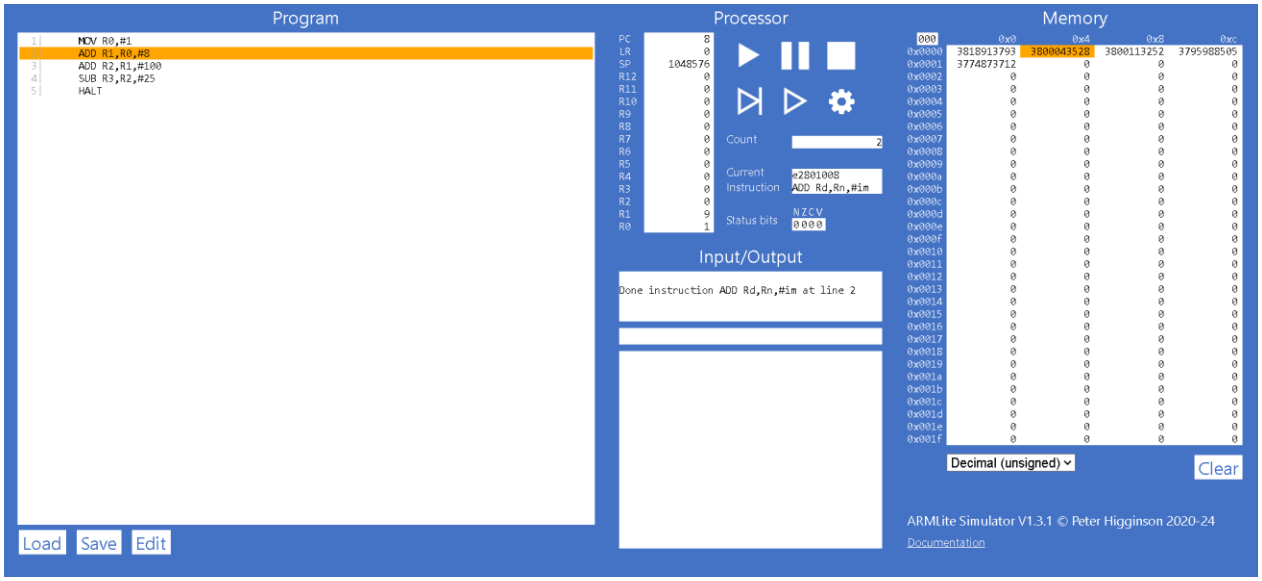
The processor has paused just before executing the line with the breakpoint. This is because breakpoints are set at the beginning of instructions.

Task 7.5.1

ADD R1, R0, #8

My expectation is R0 will remain unchanged and R1 = R0 + 8 and the result will store in R1.

Task 7.5.2



```
1  MOV R0,#1
2  ADD R1,R0,#8
3  ADD R2,R1,#100
4  SUB R3,R2,#25
5  HALT
```

PC	8
LR	0
SP	1048576
R12	0
R11	0
R10	0
R9	0
R8	0
R7	0
R6	0
R5	0
R4	0
R3	0
R2	0
R1	9
R0	1

Count 2

Current Instruction e2801008 ADD Rd,Rn,#im

Status bits N Z C V 0 0 0 0

```
Done instruction ADD Rd,Rn,#im at line 2
```

000	0x0	0x4	0x8	0xc
0x0000	3818913793	3800043528	3800113252	3795988505
0x0001	3774873712	0	0	0
0x0002	0	0	0	0
0x0003	0	0	0	0
0x0004	0	0	0	0
0x0005	0	0	0	0
0x0006	0	0	0	0
0x0007	0	0	0	0
0x0008	0	0	0	0
0x0009	0	0	0	0
0x000a	0	0	0	0
0x000b	0	0	0	0
0x000c	0	0	0	0
0x000d	0	0	0	0
0x000e	0	0	0	0
0x000f	0	0	0	0
0x0010	0	0	0	0
0x0011	0	0	0	0
0x0012	0	0	0	0
0x0013	0	0	0	0
0x0014	0	0	0	0
0x0015	0	0	0	0
0x0016	0	0	0	0
0x0017	0	0	0	0
0x0018	0	0	0	0
0x0019	0	0	0	0
0x001a	0	0	0	0
0x001b	0	0	0	0
0x001c	0	0	0	0
0x001d	0	0	0	0
0x001e	0	0	0	0
0x001f	0	0	0	0

Decimal (unsigned)

ARMLite Simulator V1.3.1 © Peter Higginson 2020-24

Task 7.5.3

MOV R0, #300

SUB R1, R0, #21

ADD R2, R1, #5

MOV R3, #64

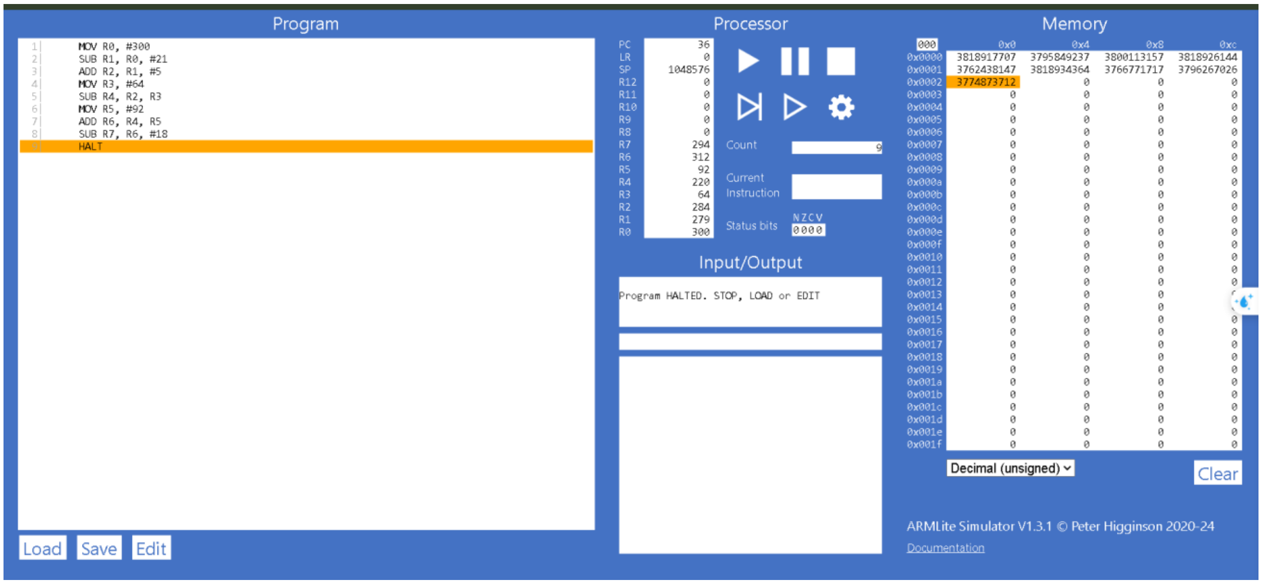
SUB R4, R2, R3

MOV R5, #92

ADD R6, R4, R5

SUB R7, R6, #18

HALT



```
MOV R0, #300
SUB R1, R0, #21
ADD R2, R1, #5
MOV R3, #64
SUB R4, R2, R3
MOV R5, #92
ADD R6, R4, R5
SUB R7, R6, #18
HALT
```

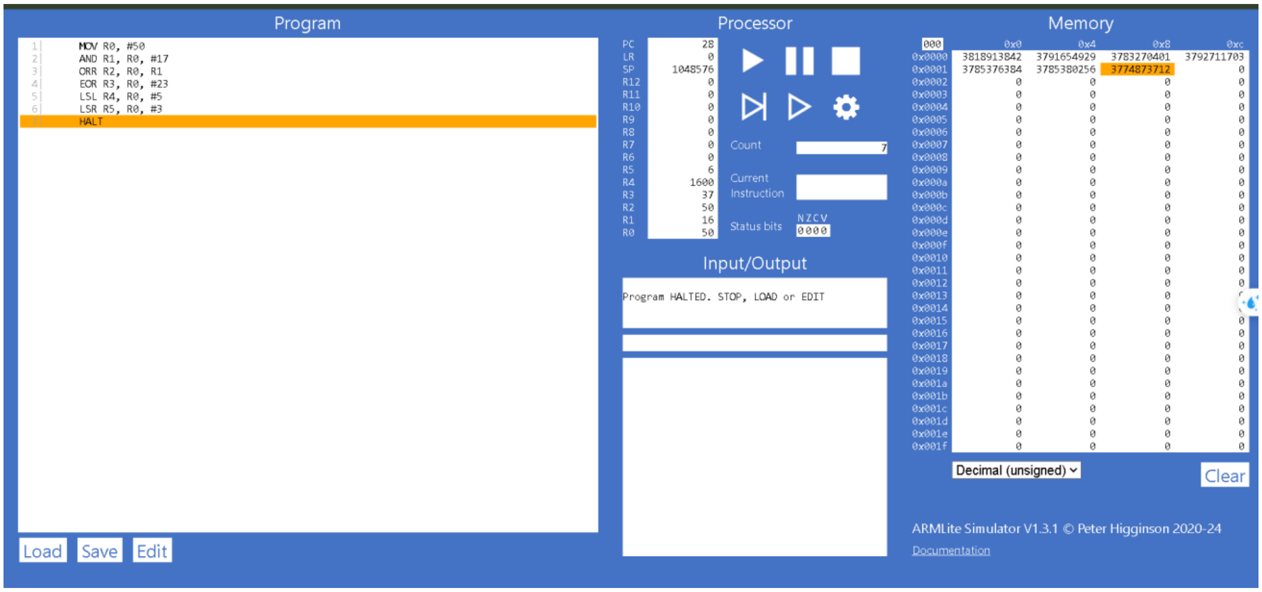
```
PC         36
LR          0
SP    1048576
R12         0
R11         0
R10         0
R9          0
R8          0
R7        294
R6        312
R5         92
R4        220
R3         64
R2        284
R1        279
R0        300

Count                  9
Current Instruction
Status bits   N Z C V
              0 0 0 0
```

```
Program HALTED. STOP, LOAD or EDIT
```

```
000            0x0         0x4         0x8         0xc
0x0000  3818917707  3795849237  3800113157  3818926144
0x0001  3762438147  3818934364  3766771717  3796267026
0x0002  3774873712           0           0           0
0x0003           0           0           0           0
0x0004           0           0           0           0
0x0005           0           0           0           0
0x0006           0           0           0           0
0x0007           0           0           0           0
0x0008           0           0           0           0
0x0009           0           0           0           0
0x000a           0           0           0           0
0x000b           0           0           0           0
0x000c           0           0           0           0
0x000d           0           0           0           0
0x000e           0           0           0           0
0x000f           0           0           0           0
0x0010           0           0           0           0
0x0011           0           0           0           0
0x0012           0           0           0           0
0x0013           0           0           0           0
0x0014           0           0           0           0
0x0015           0           0           0           0
0x0016           0           0           0           0
0x0017           0           0           0           0
0x0018           0           0           0           0
0x0019           0           0           0           0
0x001a           0           0           0           0
0x001b           0           0           0           0
0x001c           0           0           0           0
0x001d           0           0           0           0
0x001e           0           0           0           0
0x001f           0           0           0           0
```

Task 7.5.4

```
MOV R0, #50
AND R1, R0, #17
ORR R2, R0, R1
EOR R3, R0, #23
LSL R4, R0, #5
LSR R5, R0, #3
HALT
```



```
MOV R0, #50
AND R1, R0, #17
ORR R2, R0, R1
EOR R3, R0, #23
LSL R4, R0, #5
LSR R5, R0, #3
HALT
```

```
PC         28
LR          0
SP    1048576
R12         0
R11         0
R10         0
R9          0
R8          0
R7          0
R6          0
R5          6
R4       1600
R3         37
R2         50
R1         16
R0         50
```

Count: 7

Current Instruction:

Status bits: NZCV 0000

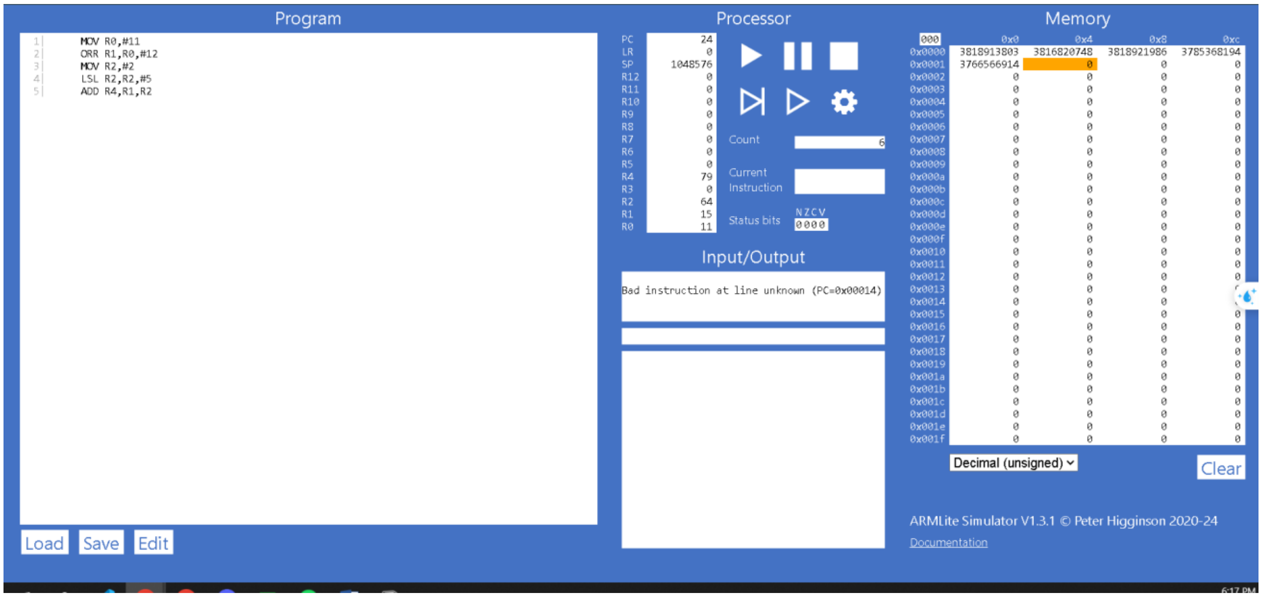
```
Program HALTED. STOP, LOAD or EDIT
```

```
000             0x0        0x4        0x8        0xc
0x0000   3818913842 3791654929 3783270401 3792711703
0x0001   3785376384 3785380256 3774873712          0
0x0002            0          0          0          0
0x0003            0          0          0          0
0x0004            0          0          0          0
0x0005            0          0          0          0
0x0006            0          0          0          0
0x0007            0          0          0          0
0x0008            0          0          0          0
0x0009            0          0          0          0
0x000a            0          0          0          0
0x000b            0          0          0          0
0x000c            0          0          0          0
0x000d            0          0          0          0
0x000e            0          0          0          0
0x000f            0          0          0          0
0x0010            0          0          0          0
0x0011            0          0          0          0
0x0012            0          0          0          0
0x0013            0          0          0
0x0014            0          0          0
0x0015            0          0          0          0
0x0016            0          0          0          0
0x0017            0          0          0          0
0x0018            0          0          0          0
0x0019            0          0          0          0
0x001a            0          0          0          0
0x001b            0          0          0          0
0x001c            0          0          0          0
0x001d            0          0          0          0
0x001e            0          0          0          0
0x001f            0          0          0          0
```

Instruction	Decimal value of the destination register	Binary value of the destination register
MOV R0, #50	50	0011 0010
AND R1, R0, #17	16	0001 0000
ORR R2, R0, R1	62	0011 1110
EOR R3, R0, #23	27	0001 1011
LSL R4, R0, #5	320	1010 0000
LSR R5, R0, #3	6	0000 0110

Task 7.5.5

```
MOV R0,#11
ORR R1,R0,#12
MOV R2,#2
LSL R2,R2,#5
ADD R4,R1,R2
```



```
MOV R0,#11
ORR R1,R0,#12
MOV R2,#2
LSL R2,R2,#5
ADD R4,R1,R2
```

PC	24
LR	0
SP	1048576
R12	0
R11	0
R10	0
R9	0
R8	0
R7	0
R6	0
R5	0
R4	79
R3	0
R2	64
R1	15
R0	11

Count: 6

Current Instruction:

Status bits NZCV 0000

```
Bad instruction at line unknown (PC=0x00014)
```

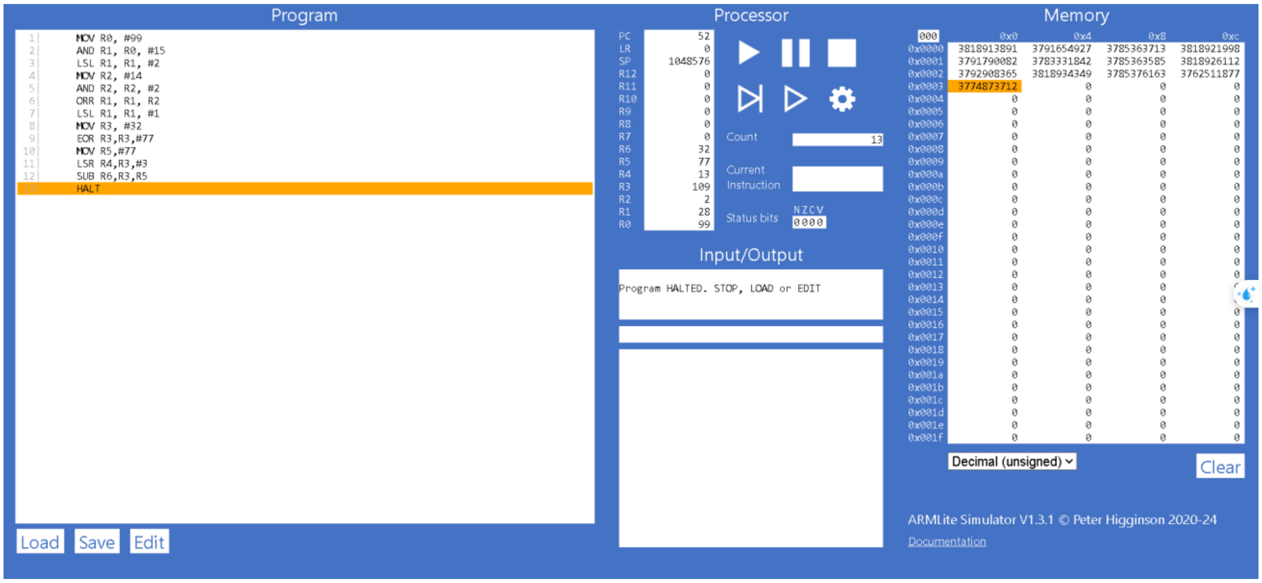
000	0x0	0x4	0x8	0xc
0x0000	3818913803	3816820748	3818921986	3785368194
0x0001	3766566914	0	0	0
0x0002	0	0	0	0
0x0003	0	0	0	0
0x0004	0	0	0	0
0x0005	0	0	0	0
0x0006	0	0	0	0
0x0007	0	0	0	0
0x0008	0	0	0	0
0x0009	0	0	0	0
0x000a	0	0	0	0
0x000b	0	0	0	0
0x000c	0	0	0	0
0x000d	0	0	0	0
0x000e	0	0	0	0
0x000f	0	0	0	0
0x0010	0	0	0	0
0x0011	0	0	0	0
0x0012	0	0	0	0
0x0013	0	0	0	0
0x0014	0	0	0	0
0x0015	0	0	0	0
0x0016	0	0	0	0
0x0017	0	0	0	0
0x0018	0	0	0	0
0x0019	0	0	0	0
0x001a	0	0	0	0
0x001b	0	0	0	0
0x001c	0	0	0	0
0x001d	0	0	0	0
0x001e	0	0	0	0
0x001f	0	0	0	0

Decimal (unsigned)

ARMLite Simulator V1.3.1 © Peter Higginson 2020-24

Task 7.5.6

```
MOV R0, #99
AND R1, R0, #15
LSL R1, R1, #2
MOV R2, #14
AND R2, R2, #2
ORR R1, R1, R2
LSL R1, R1, #1
MOV R3, #32
EOR R3,R3,#77
MOV R5,#77
LSR R4,R3,#3
SUB R6,R3,R5
HALT
```



```
        MOV R0, #99
        AND R1, R0, #15
        LSL R1, R1, #2
        MOV R2, #14
        AND R2, R2, #2
        ORR R1, R1, R2
        LSL R1, R1, #1
        MOV R3, #32
        EOR R3,R3,#77
        MOV R5,#77
        LSR R4,R3,#3
        SUB R6,R3,R5
        HALT
```

PC	52
LR	0
SP	1048576
R12	0
R11	0
R10	0
R9	0
R8	0
R7	0
R6	32
R5	77
R4	13
R3	109
R2	2
R1	28
R0	99

Count: 13

Current Instruction:

Status bits: N Z C V 0 0 0 0

```
Program HALTED. STOP, LOAD or EDIT
```

000	0x0	0x4	0x8	0xc
0x0000	3818913891	3791654927	3785363713	3818921998
0x0001	3791790082	3783331842	3785363585	3818926112
0x0002	3792908365	3818934349	3785376163	3762511877
0x0003	3774873712	0	0	0
0x0004	0	0	0	0
0x0005	0	0	0	0
0x0006	0	0	0	0
0x0007	0	0	0	0
0x0008	0	0	0	0
0x0009	0	0	0	0
0x000a	0	0	0	0
0x000b	0	0	0	0
0x000c	0	0	0	0
0x000d	0	0	0	0
0x000e	0	0	0	0
0x000f	0	0	0	0
0x0010	0	0	0	0
0x0011	0	0	0	0
0x0012	0	0	0	0
0x0013	0	0	0	[illegible]
0x0014	0	0	0	[illegible]
0x0015	0	0	0	0
0x0016	0	0	0	0
0x0017	0	0	0	0
0x0018	0	0	0	0
0x0019	0	0	0	0
0x001a	0	0	0	0
0x001b	0	0	0	0
0x001c	0	0	0	0
0x001d	0	0	0	0
0x001e	0	0	0	0
0x001f	0	0	0	0

Task 7.6.1

The result shown in **R1** is negative due to the behavior of the **LSL** (Logical Shift Left) operation. Shifting a positive number left by 18 bits results in a large number, exceeding the range of a signed 32-bit integer, which causes it to wrap around and become negative.

Task 7.6.3

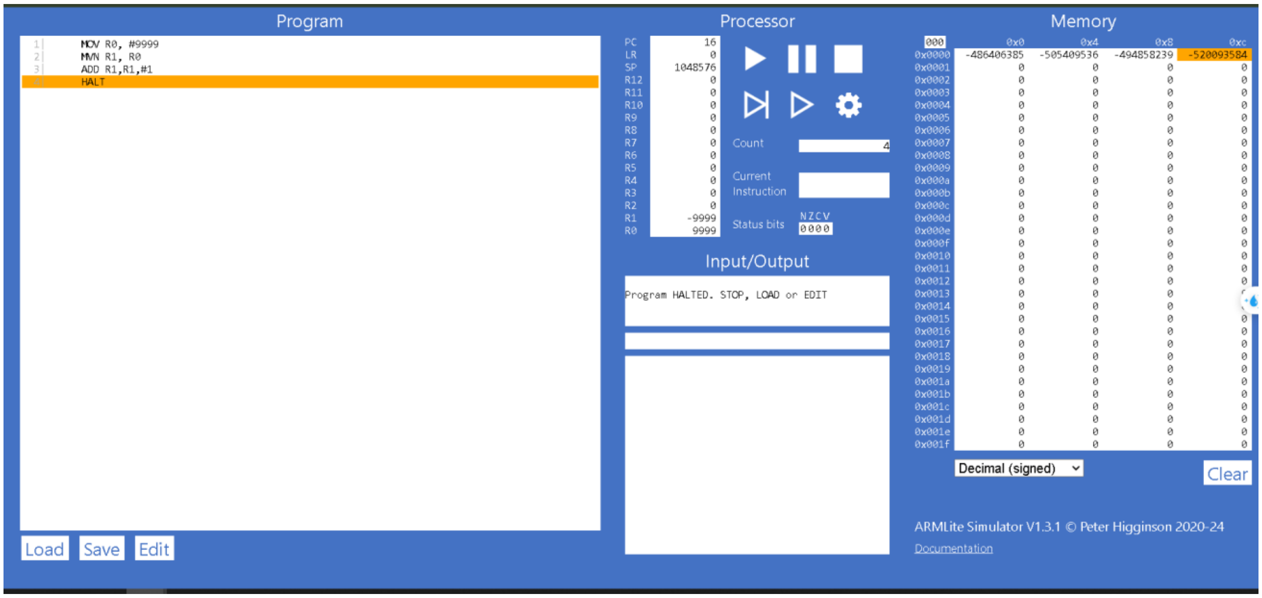
The binary representations are:

- **1**: 0000 0000 0000 0000 0000 0000 0000 0001
- **-1**: 1111 1111 1111 1111 1111 1111 1111 1111
- **2**: 0000 0000 0000 0000 0000 0000 0000 0010
- **-2**: 1111 1111 1111 1111 1111 1111 1111 1110

Pattern Observed

The pattern shows that the negative representation of a number is obtained by inverting its bits and adding **1**, confirming the two's complement system used for signed integers.

Task 7.6.4



```
MOV R0, #9999
MVN R1, R0
ADD R1,R1,#1
HALT
```

PC	16
LR	0
SP	1048576
R12	0
R11	0
R10	0
R9	0
R8	0
R7	0
R6	0
R5	0
R4	0
R3	0
R2	0
R1	-9999
R0	9999

Count 4

Current Instruction

Status bits NZCV 0000

```
Program HALTED. STOP, LOAD or EDIT
```

000	0x0	0x4	0x8	0xc
0x0000	-486406385	-505409536	-494858239	-520093584
0x0001	0	0	0	0
0x0002	0	0	0	0
0x0003	0	0	0	0
0x0004	0	0	0	0
0x0005	0	0	0	0
0x0006	0	0	0	0
0x0007	0	0	0	0
0x0008	0	0	0	0
0x0009	0	0	0	0
0x000a	0	0	0	0
0x000b	0	0	0	0
0x000c	0	0	0	0
0x000d	0	0	0	0
0x000e	0	0	0	0
0x000f	0	0	0	0
0x0010	0	0	0	0
0x0011	0	0	0	0
0x0012	0	0	0	0
0x0013	0	0	0	0
0x0014	0	0	0	0
0x0015	0	0	0	0
0x0016	0	0	0	0
0x0017	0	0	0	0
0x0018	0	0	0	0
0x0019	0	0	0	0
0x001a	0	0	0	0
0x001b	0	0	0	0
0x001c	0	0	0	0
0x001d	0	0	0	0
0x001e	0	0	0	0
0x001f	0	0	0	0

Decimal (signed)

ARMLite Simulator V1.3.1 © Peter Higginson 2020-24