COS-10004 (Computer Systems)

Name: Dave Nguyen (Nguyen Quang Anh).

ID: 104697710.

Q7.1.1:

The displayed value is 0x000000065. This is not shown as 101 since there is no prefix when typing 101. This implies that the ARM simulator interprets the input as a decimal integer and translates it to hexadecimal as 0x00000065.

Q7.1.2:

The displayed value is 0x00000101. This number was presented as 0x0000101 because the ARM simulator recognises it as a hexadecimal value with the 0x prefix.

Q7.1.3;

The tooltip presents the displayed values in several forms.

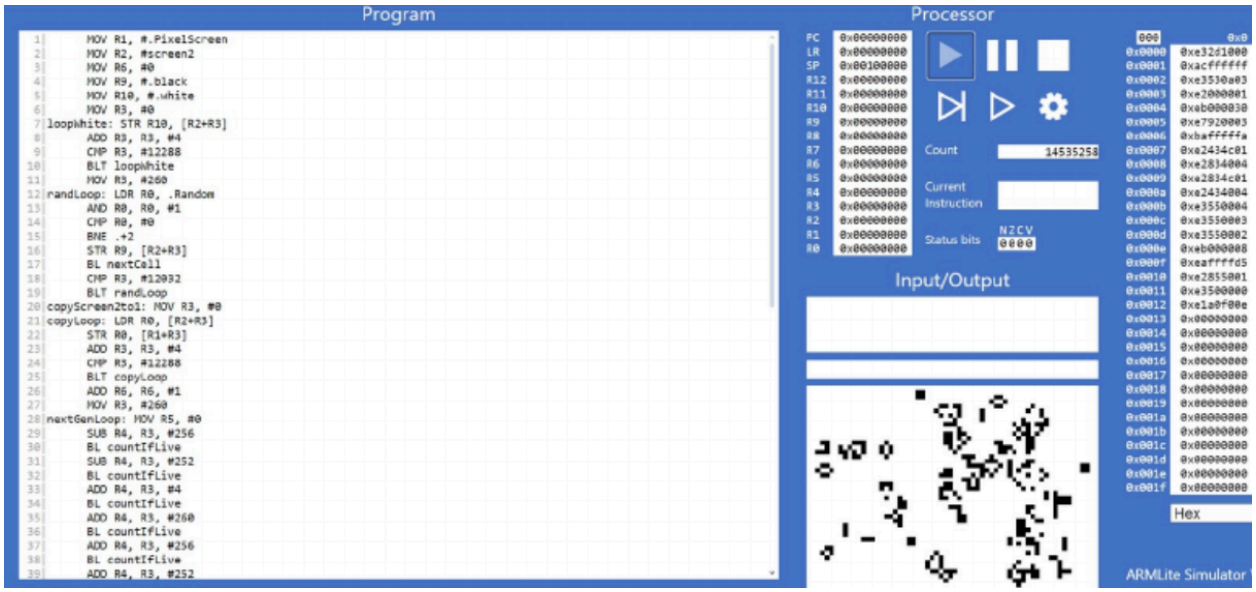
Q7.1.4:

No, since the row and column headers reflect where the data is stored (memory addresses), which are always given in hexadecimal for consistency and are separate from the data representation.

Q7.2.1:

Because 32-bit words require four bytes to control, the memory address offsets for each column header increase by multiples of 0x4. This guarantees that each address goes to the beginning of the following 32-bit word in memory.

Q7.3.1:



Q7.3.2:

The ARM simulator turned it into machine code that the processor could comprehend, and this machine code was put into memory using the Von Neumann architecture. In this design, the code was translated to binary and stored in memory at precise memory locations.

Q7.3.3:

- This number indicates the memory address at which the machine code is stored.
- When submitted, the ARMs simulator removes these blank lines.
- ARMs simulator also removes these additional spaces.
- ARMs simulator keeps these comments but doesn't display them as the representation of data.
- ARMs simulator keeps these comments but doesn't change the line numbers.
- The overall number of instructions transformed into machine code remains constant since blank lines, additional spaces, and comments are not considered as instructions.
- When the comma is removed from the first line of code, mistakes arise, and the ARMs simulator highlights this problem by coloring the code lines that contain errors in RED. The rationale is that a comma is one of the criteria that allows the ARM emulator to appropriately interpret and translate the command. Without an appropriate comma, the program will not be understood correctly.

Q7.4.1:

Orange highlighting occurs in both the Program and Memory panes, showing the current directive that is running and where it is kept in memory.

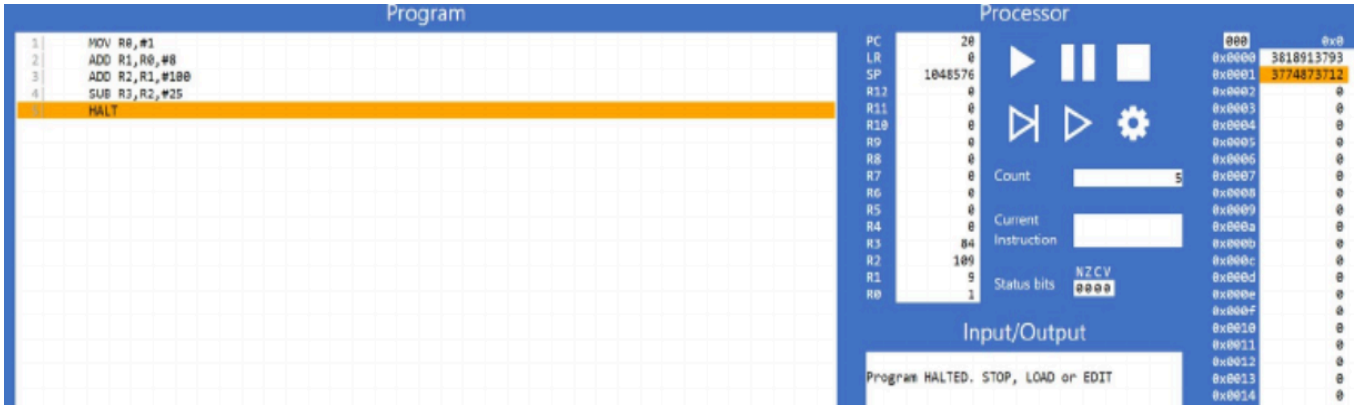
Q7.4.2:

Clicking the button will perform the command each time and then halt once completed.

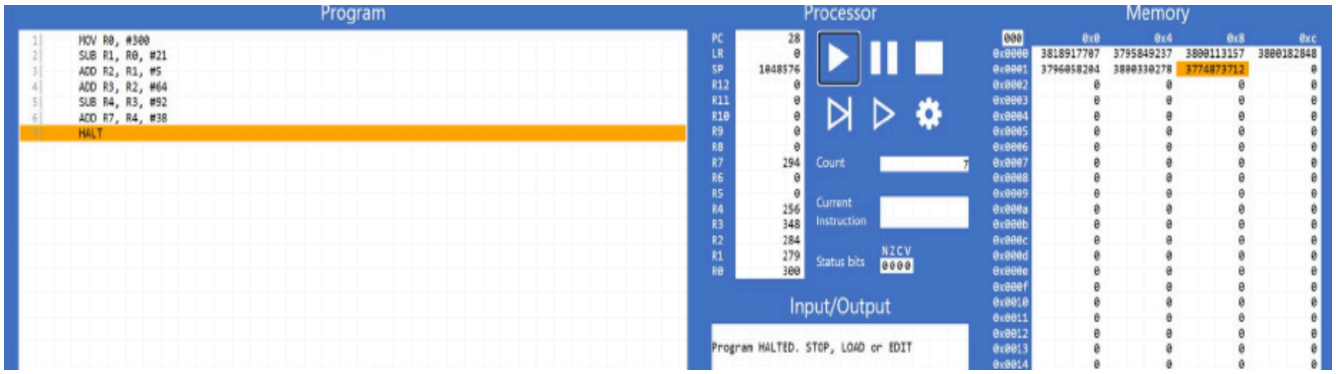
Q7.4.3:

It pauses just before executing the line with the breakpoint.

Q7.5.2:



Q7.5.3:

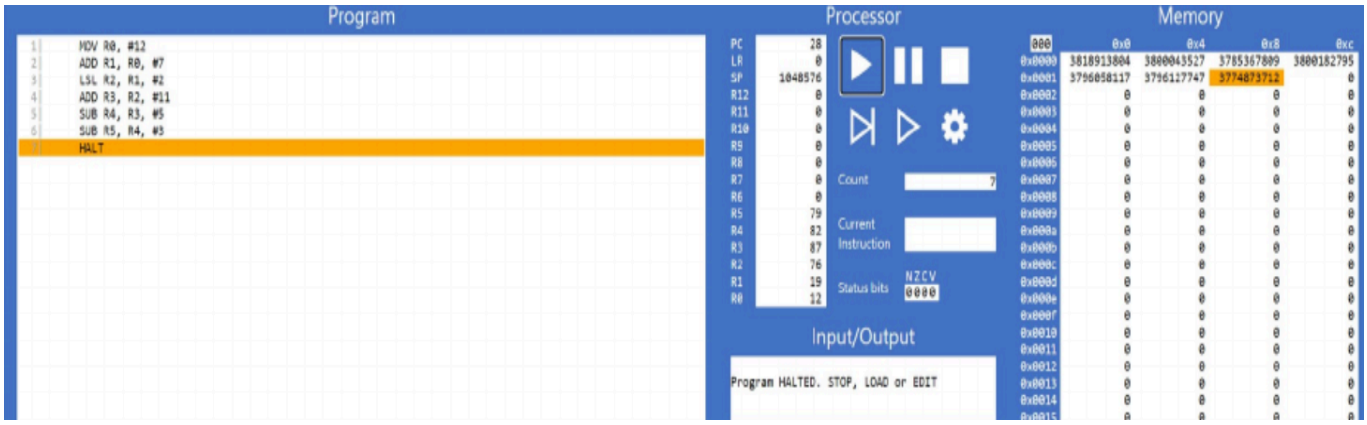


Q7.5.4:

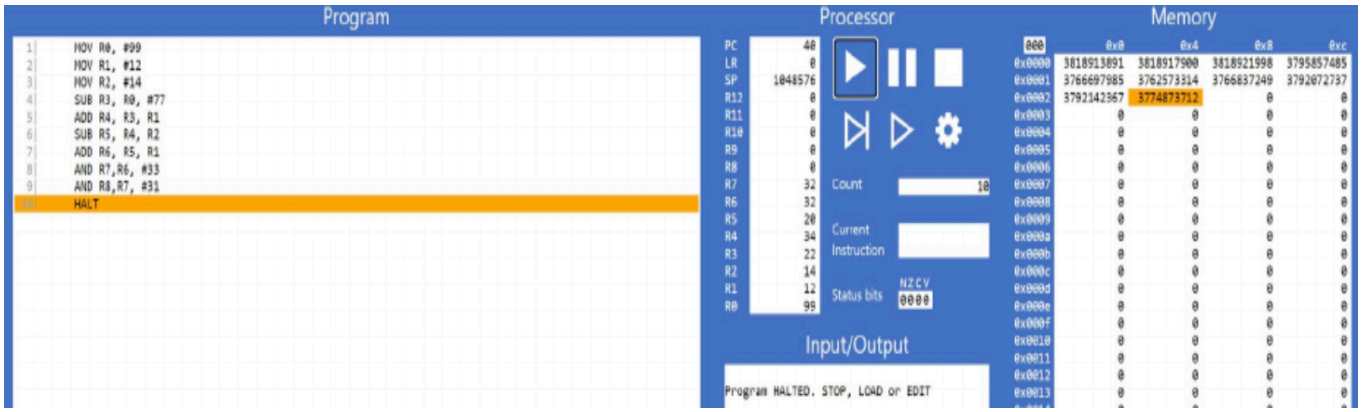
Instruction:	The decimal value of the destination registers after executing this instruction.	The binary value of the destination registers after executing this instruction.
MOV R0, #10.	10.	00000000 00000000 00000000 00001111
AND R1, R0, #2.	2.	00000000 00000000 00000000 00000010
ORR R2, R1, #3.	3.	00000000 00000000 00000000 00000011

EOR R3, R2, #21.	22.	00000000 00000000 00000000 00010110
LSL R4, R3, #1.	44.	00000000 00000000 00000000 00101100
LSR R5, R4, #1.	22.	00000000 00000000 00000000 00010110

Q7.5.5:



Q7.5.6:



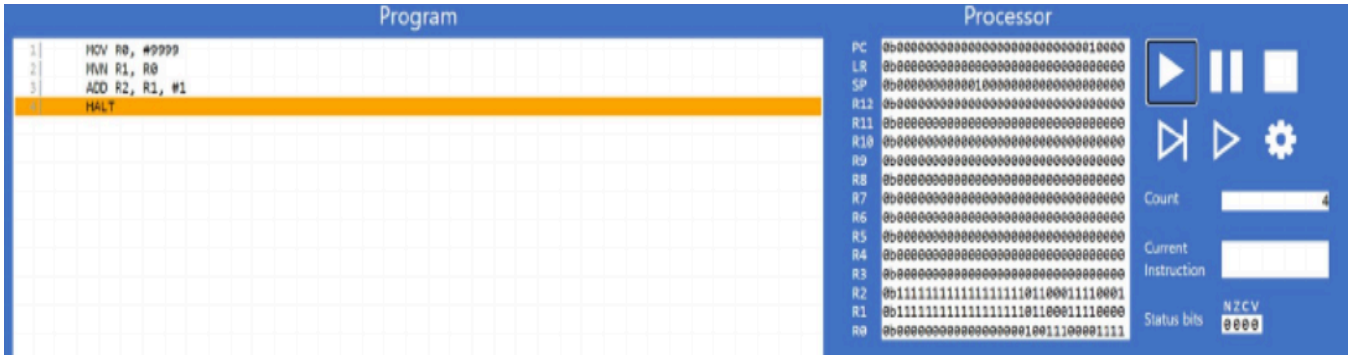
Q7.6.1:

The result reported in R1 is a negative decimal number since the most significant bit after the shift is 1 (10011100001111000000000000000000), signifying a negative integer.

Q7.6.3:

1. 1111 1111 1111 1111 1111 1111 1111 1111
2. 1111 1111 1111 1111 1111 1111 1111 1110

Q7.6.4:


Program
1 MOV R0, #9999
2 MVN R1, R0
3 ADD R2, R1, #1
4 HALT
Processor
PC
LR
SP
R12
R11
R10
R9
R8
R7
R6
R5
R4
R3
R2
R1
R0
Count
Current Instruction
Status bits
NZCV
0000