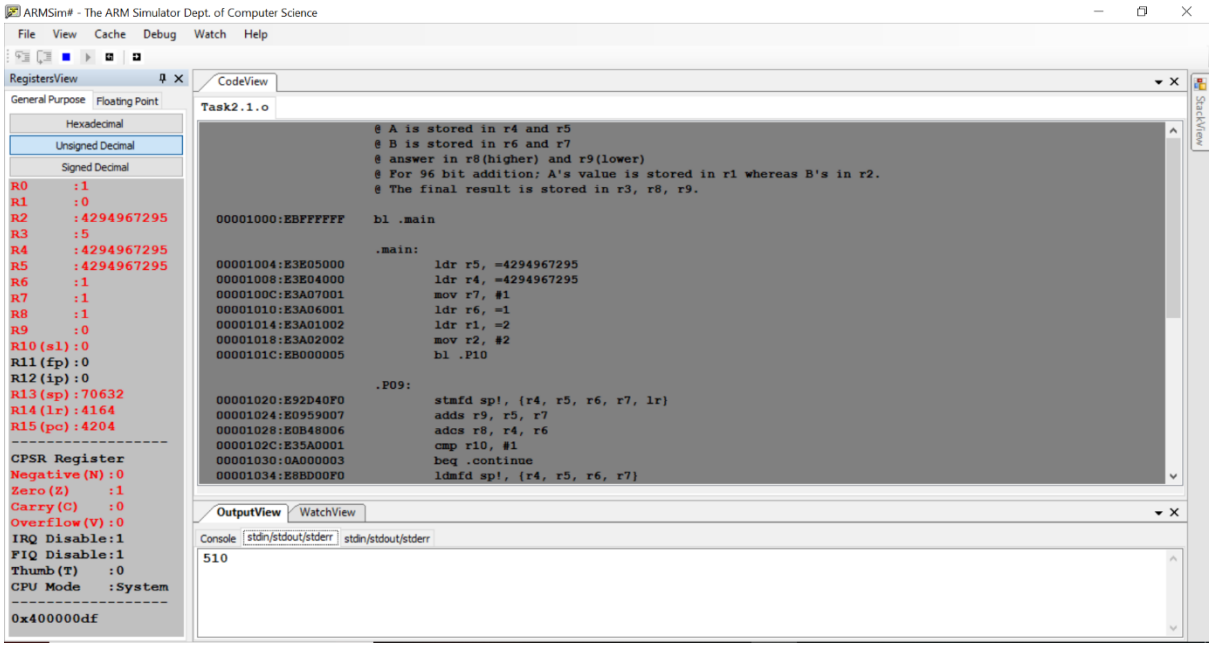
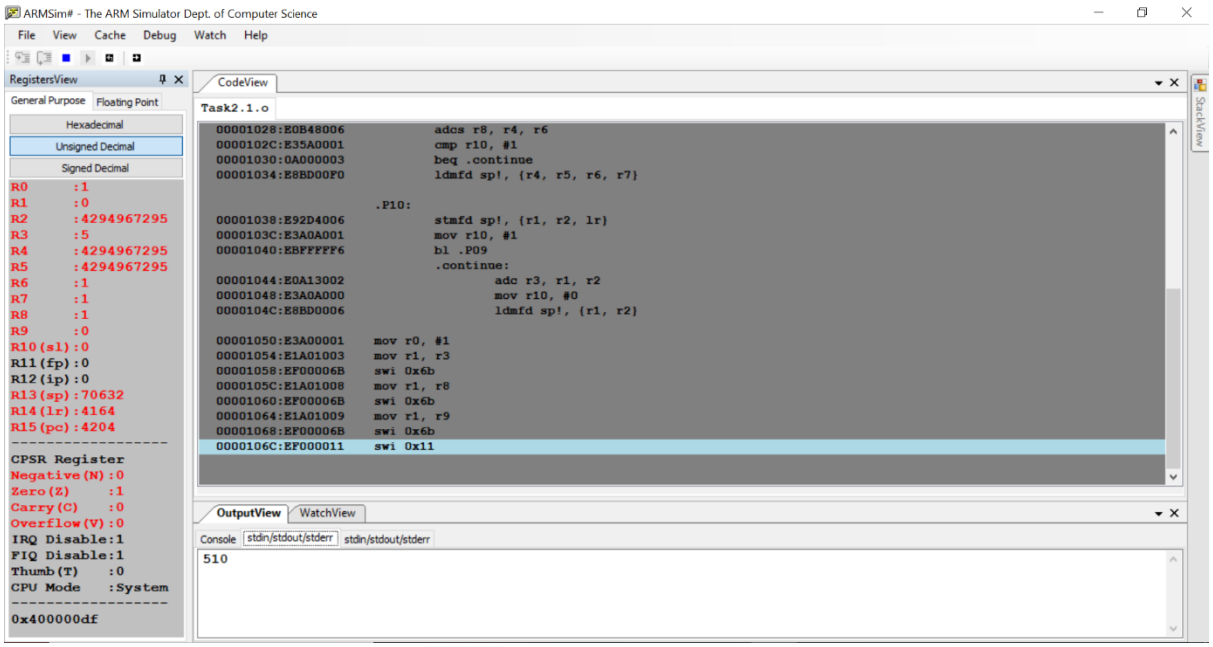
CSE 112 Lab 2

2.1)





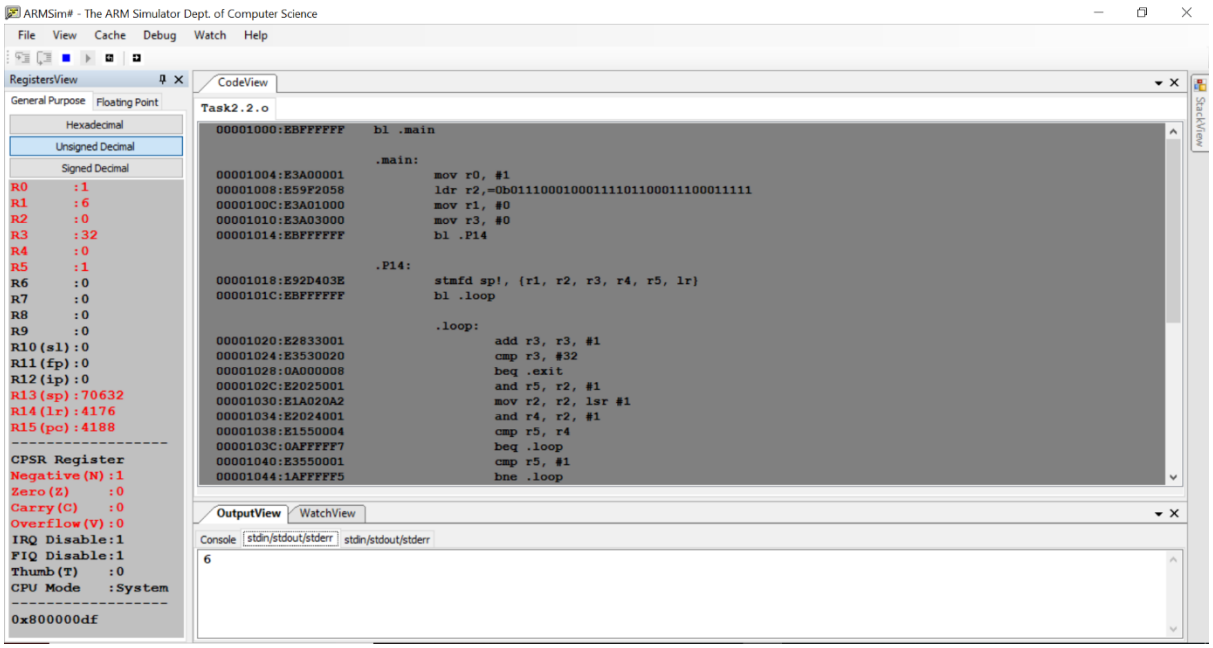
Link to the code: https://drive.google.com/drive/folders/115r_DHRfdI0UkSOsbTN2DeQpgx0VnWdd?usp=sharing (named Task2.1)

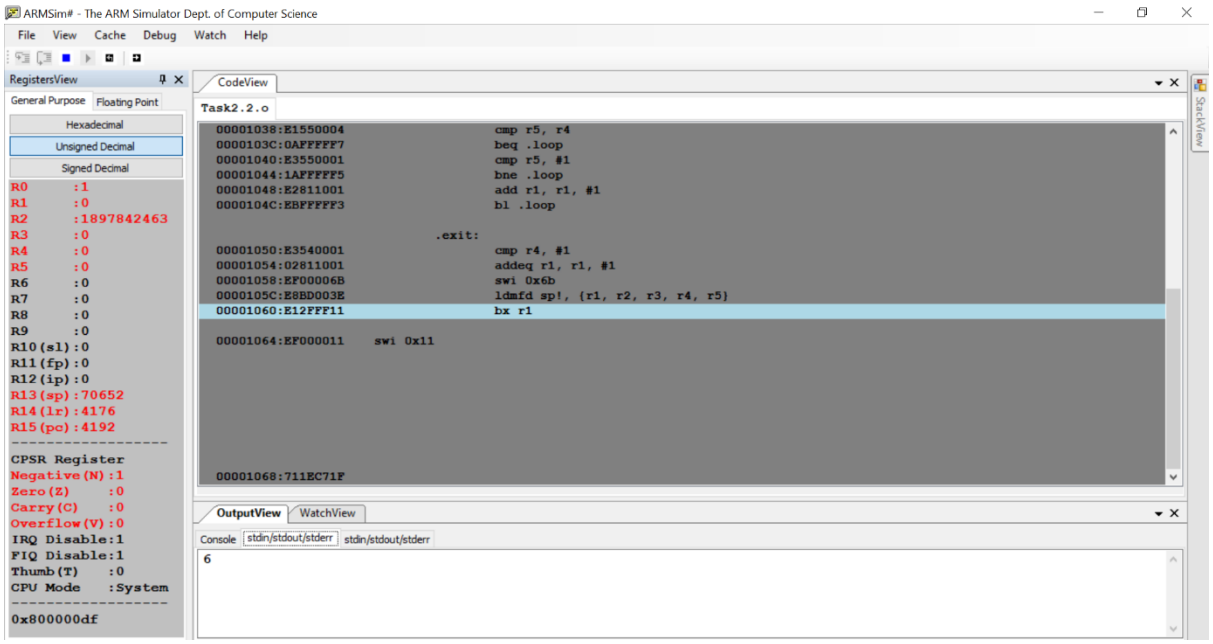
Explanation to the code:

- The code starts by branching to the main label.

- It has two procedures named P09 and P10.
- The procedure P09 has conditional statements which are executed when P10 is called else only 64 bits are added in P09.
- When P10 is called 1 is moved into r10 so that the conditional statements in P09 can be executed.
- Now, the program branches to P09 where it first adds the rightmost 32 bits and set the carry flag (if any).
- It adds the next 32 bits number with the carry and sets carry again (if any).
- Since P09 is for addition of 64 bits, the program then jumps to the remaining statements of P10 mentioned under label .continue where it adds the leftmost 32 bits number with the previous carry flag (if any).
- It moves to r10 a value of 0 so that the initial conditions can be met and P09 can be called individually if required.
- For printing swi0x6b command has been used. The value to be printed is first moved in r1 so that printing command can be used and then the decimal value of each result register is printed.

2.2)





Link to the code:
https://drive.google.com/open?id=115r_DHRfdI0UkSOsbTN2DeQpgx0VnWdd (named Task2.2)

Explanation to the code:

- The code starts by branching to the main label.
- In main label: 1 is moved to r0 so as to help in printing using swi0x6b command. The input is loaded into r2 in binary format. R1 and r3 are initialised with 0 where r1 will be storing the number of sequences of 1 whereas r3 will be storing the bits shifted in the input. Now, the procedure P14 is called.
- In the procedure, all the registers to be used are pushed into the stack and the loop starts.
    - 1 is added to r3 at each iteration.
    - R3 is compared with 32 in every iteration so that it does not lead to infinite loop.
    - R5 is storing the current bit of r2 by and operation between r2 and 1. Similarly, r4 is storing the next bit of r2.
    - R5 and r4 are then compared. If they are equal it leads to next iteration otherwise the current bit is checked. If it equals 1, r1 is incremented by 1 else the next iteration takes place.

The crux of the loop is that compares the current and next bit. If both are unequal, it sees if the current bit is 1 then the number of sequences is incremented by 1 else the next iteration takes place.

- After all the bits are considered it leads to exit label, where the MSB of the input is considered as this is what is left unexamined in the loop.
- It prints the value stored in r1.
- P14 then pops all what was stored in the stack.
- The programme ends.

Yes. I tried to use as much conditional statements as I could. As it can be seen in the explanation of the code also that conditionals statements made the working of the code possible. I have used three conditional statements. First, to check whether I have traversed all the 32 bits. Second, to check if the current and the next bit are same or not. If yes, it leads to another conditional statement where it is checked if the current bit is 1 or 0. Any other solution of this question would also depend more or less on the same logic and hence this much number of conditional statements are enough to make the code working.