ASSIGNMENT 2 -REPORT

Name : Nguyen Quoc Thang

ID number : 104193360

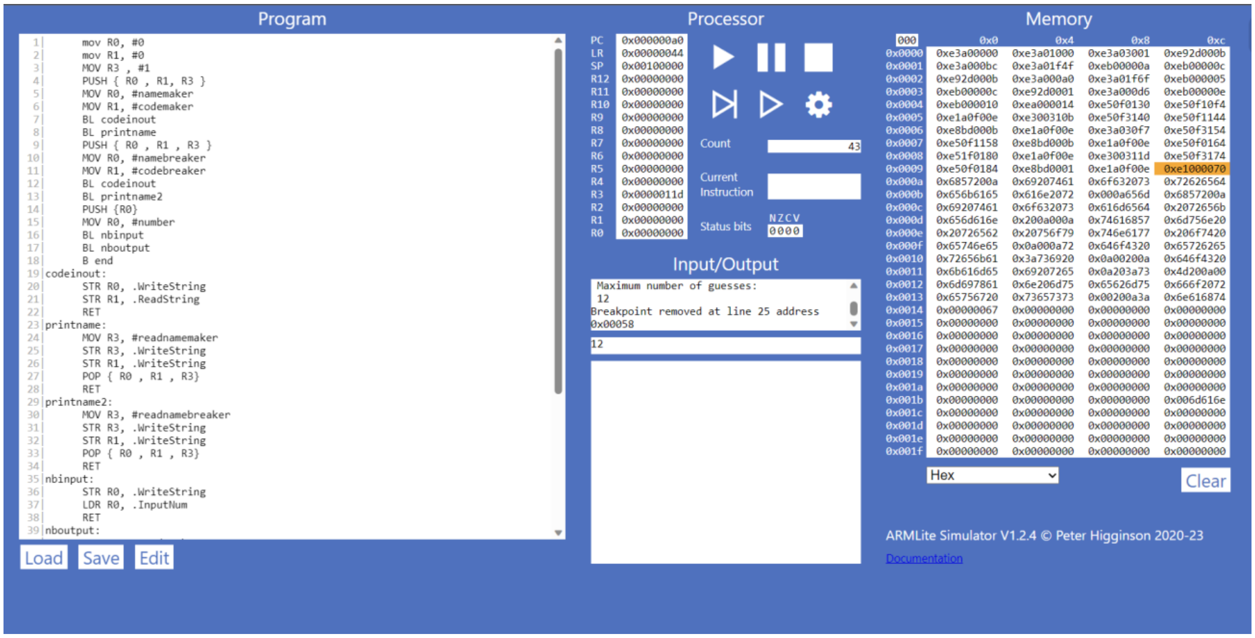
Unit code : COS10004 – Computer System

AIM

- This project asks to create a game called Mastermind for 2 players, in which codemakers are the ones who write secret codes using 4 wall letters corresponding to 4 different colors in 6 characters corresponding to 6 different colors and the codebreaker has to decode that secret.

Stage 1 :

- Function codeinout : I use it to print out the question string about name breaker and name maker and ask user to enter name of codebreaker and codemaker.
- Function printname : I use it to print the string 'Codemaker is ' and print the name of codemaker to the output .
- Function printname1 : I use it to print the string 'Codebreaker is ' and print the name of codebreaker to the output .
- Function nbinput : I use it to print out string to require the user enter the maximum number of guesses and ask user to a number.
- Function nboutput : I use it to print the string 'Maximum number of guesses ' and print out the number which is you enter.
- **- .WriteString** : I use to print a string of characters to the output. For example, in the code I use to print out 5 sentences 'What is codebreaker name' , 'What is codebreaker name' , 'Codebreaker is' , 'codemaker is' and 'Maximum number of guesses'
- **.ReadString :** I use it to ask the user to enter a string of characters from input
- **.**InputNum : I use it to ask the user to enter a number.



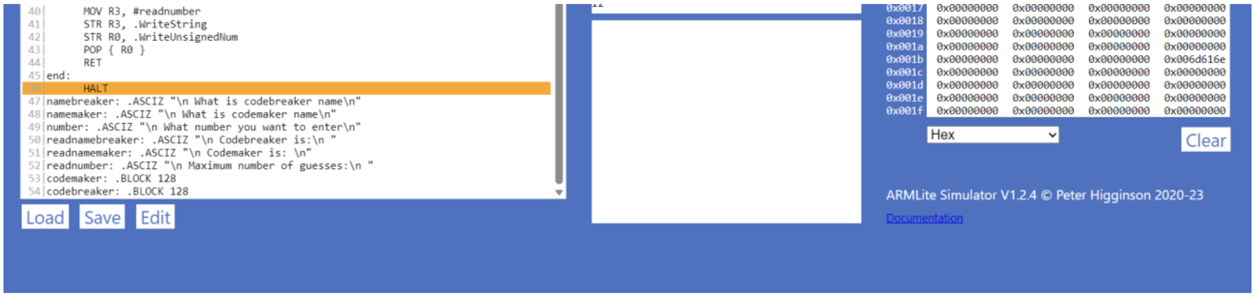
```
        mov R0, #0
        mov R1, #0
        MOV R3 , #1
        PUSH { R0 , R1, R3 }
        MOV R0, #namemaker
        MOV R1, #codemaker
        BL codeinout
        BL printname
        PUSH { R0 , R1 , R3 }
        MOV R0, #namebreaker
        MOV R1, #codebreaker
        BL codeinout
        BL printname2
        PUSH {R0}
        MOV R0, #number
        BL nbinput
        BL nboutput
        B end
codeinout:
        STR R0, .WriteString
        STR R1, .ReadString
        RET
printname:
        MOV R3, #readnamemaker
        STR R3, .WriteString
        STR R1, .WriteString
        POP { R0 , R1 , R3}
        RET
printname2:
        MOV R3, #readnamebreaker
        STR R3, .WriteString
        STR R1, .WriteString
        POP { R0 , R1 , R3}
        RET
nbinput:
        STR R0, .WriteString
        LDR R0, .InputNum
        RET
nboutput:
```

```
PC   0x000000a0
LR   0x00000044
SP   0x00100000
R12  0x00000000
R11  0x00000000
R10  0x00000000
R9   0x00000000
R8   0x00000000
R7   0x00000000
R6   0x00000000
R5   0x00000000
R4   0x00000000
R3   0x0000011d
R2   0x00000000
R1   0x00000000
R0   0x00000000

Count                 43
Current Instruction
Status bits  NZCV
             0000
```

```
 Maximum number of guesses:
 12
Breakpoint removed at line 25 address
0x00058

12
```

```
000         0x0         0x4         0x8         0xc
0x0000  0xe3a00000  0xe3a01000  0xe3a03001  0xe92d000b
0x0001  0xe3a000bc  0xe3a01f4f  0xeb00000a  0xeb00000c
0x0002  0xe92d000b  0xe3a000a0  0xe3a01f6f  0xeb000005
0x0003  0xeb00000c  0xe92d0001  0xe3a000d6  0xeb00000e
0x0004  0xeb000010  0xea000014  0xe50f0130  0xe50f10f4
0x0005  0xe1a0f00e  0xe300310b  0xe50f3140  0xe50f1144
0x0006  0xe8bd000b  0xe1a0f00e  0xe3a030f7  0xe50f3154
0x0007  0xe50f1158  0xe8bd000b  0xe1a0f00e  0xe50f0164
0x0008  0xe51f0180  0xe1a0f00e  0xe300311d  0xe50f3174
0x0009  0xe50f0184  0xe8bd0001  0xe1a0f00e  0xe1000070
0x000a  0x6857200a  0x69207461  0x6f632073  0x72626564
0x000b  0x656b6165  0x616e2072  0x000a656d  0x6857200a
0x000c  0x69207461  0x6f632073  0x616d6564  0x2072656b
0x000d  0x656d616e  0x200a000a  0x74616857  0x6d756e20
0x000e  0x20726562  0x20756f79  0x746e6177  0x206f7420
0x000f  0x65746e65  0x0a000a72  0x646f4320  0x65726265
0x0010  0x72656b61  0x3a736920  0x0a00200a  0x646f4320
0x0011  0x6b616d65  0x69207265  0x0a203a73  0x4d200a00
0x0012  0x6d697861  0x6e206d75  0x65626d75  0x666f2072
0x0013  0x65756720  0x73657373  0x00200a3a  0x6e616874
0x0014  0x00000067  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  0x006d616e
0x001c  0x00000000  0x00000000  0x00000000  0x00000000
0x001d  0x00000000  0x00000000  0x00000000  0x00000000
0x001e  0x00000000  0x00000000  0x00000000  0x00000000
0x001f  0x00000000  0x00000000  0x00000000  0x00000000
```

ARMLite Simulator V1.2.4 © Peter Higginson 2020-23



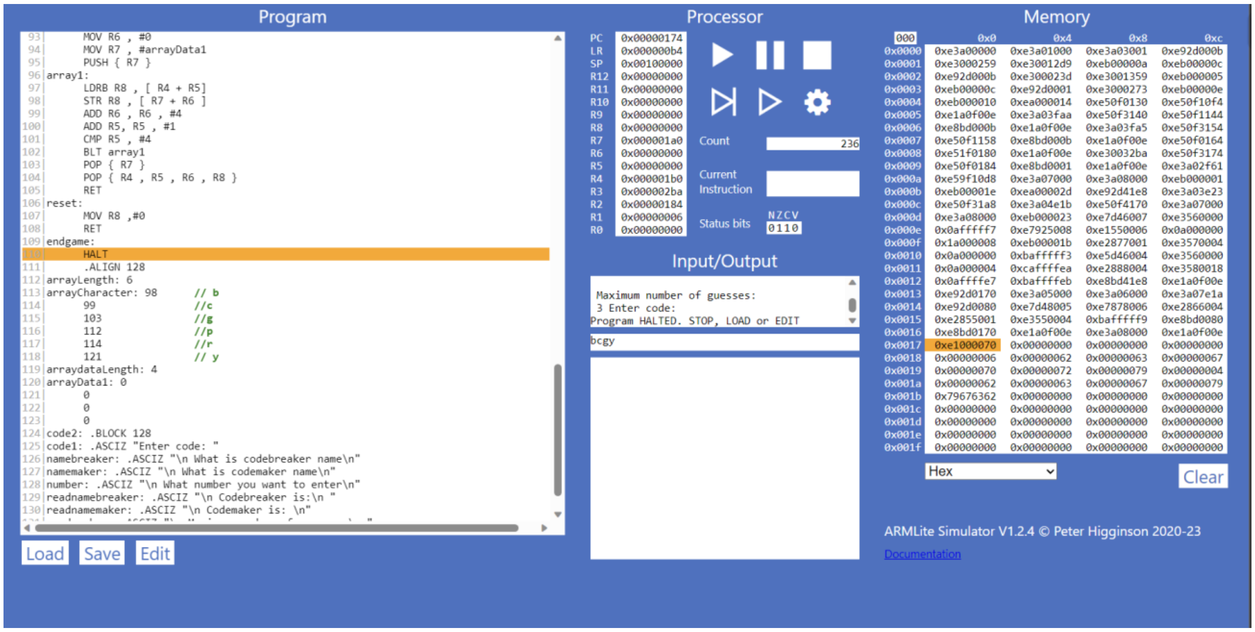
```
        MOV R3, #readnumber
        STR R3, .WriteString
        STR R0, .WriteUnsignedNum
        POP { R0 }
        RET
end:
        HALT
namebreaker: .ASCIZ "\n What is codebreaker name\n"
namemaker: .ASCIZ "\n What is codemaker name\n"
number: .ASCIZ "\n What number you want to enter\n"
readnamebreaker: .ASCIZ "\n Codebreaker is:\n "
readnamemaker: .ASCIZ "\n Codemaker is: \n"
readnumber: .ASCIZ "\n Maximum number of guesses:\n "
codemaker: .BLOCK 128
codebreaker: .BLOCK 128
```

```
0x0017  0x00000000  0x00000000  0x00000000  0x00000000
0x0018  0x00000000  0x00000000  0x00000000  0x00000000
0x0019  0x00000000  0x00000000  0x00000000  0x00000000
0x001a  0x00000000  0x00000000  0x00000000  0x00000000
0x001b  0x00000000  0x00000000  0x00000000  0x006d616e
0x001c  0x00000000  0x00000000  0x00000000  0x00000000
0x001d  0x00000000  0x00000000  0x00000000  0x00000000
0x001e  0x00000000  0x00000000  0x00000000  0x00000000
0x001f  0x00000000  0x00000000  0x00000000  0x00000000
```

ARMLite Simulator V1.2.4 © Peter Higginson 2020-23

Stage 2 :

In stage 2, I use the getcode function to input a code and check whether the entered code is correct or not.I use the WriteString function to prompt the user to enter the code.I use the reset function to reset R8 to 0 each time the condition is checked again from the beginning.I assign 6 values corresponding to 6 characters of colors into an array named arrayData1.I use the **loop2** label to compare each element in the array with each byte stored in the memory address that holds it. If they are not equal, it will jump to the jump label to increase 1 byte, and then it continues to compare. When they match, it will increase 4 bytes to get the next value of the array. After checking all 4 characters, it moves to the **character5** label to check the 5th character. If there is no 5th character and the entered code matches, it will return to the instruction after the getcode function, which is the pusharray function to save the code into the array.



```
        MOV R6 , #0
        MOV R7 , #arrayData1
        PUSH { R7 }
array1:
        LDRB R8 , [ R4 + R5]
        STR R8 , [ R7 + R6 ]
        ADD R6 , R6 , #4
        ADD R5, R5 , #1
        CMP R5 , #4
        BLT array1
        POP { R7 }
        POP { R4 , R5 , R6 , R8 }
        RET
reset:
        MOV R8 ,#0
        RET
endgame:
        HALT
        .ALIGN 128
arrayLength: 6
arrayCharacter: 98      // b
        99              //c
        103             //g
        112             //p
        114             //r
        121             // y
arraydataLength: 4
arrayData1: 0
        0
        0
        0
code2: .BLOCK 128
code1: .ASCIZ "Enter code: "
namebreaker: .ASCIZ "\n What is codebreaker name\n"
namemaker: .ASCIZ "\n What is codemaker name\n"
number: .ASCIZ "\n What number you want to enter\n"
readnamebreaker: .ASCIZ "\n Codebreaker is:\n "
readnamemaker: .ASCIZ "\n Codemaker is: \n"
```

```
PC   0x00000174
LR   0x000000b4
SP   0x00100000
R12  0x00000000
R11  0x00000000
R10  0x00000000
R9   0x00000000
R8   0x00000000
R7   0x000001a0
R6   0x00000000
R5   0x00000000
R4   0x000001b0
R3   0x000002ba
R2   0x00000184
R1   0x00000006
R0   0x00000000

Count 236
Status bits NZCV 0110
```

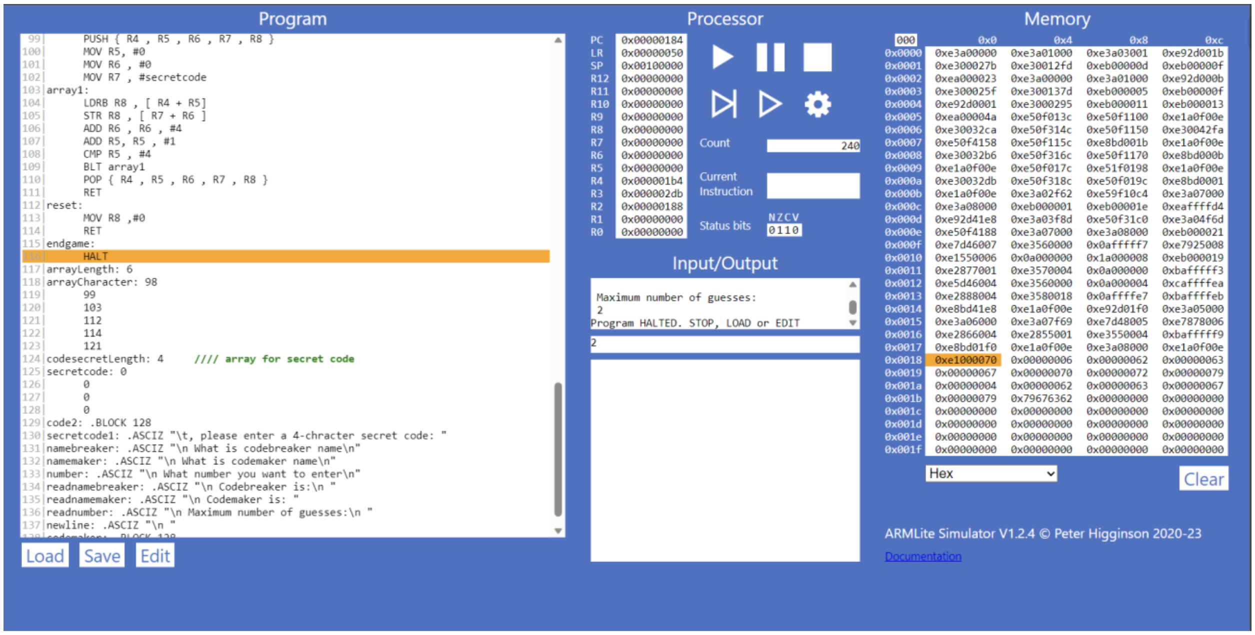
```
 Maximum number of guesses:
 3 Enter code:
Program HALTED. STOP, LOAD or EDIT
```

```
bcgy
```

```
   000         0x0         0x4         0x8         0xc
0x0000  0xe3a00000  0xe3a01000  0xe3a03001  0xe92d000b
0x0001  0xe3000259  0xe30012d9  0xeb00000a  0xeb00000c
0x0002  0xe92d000b  0xe300023d  0xe3001359  0xeb000005
0x0003  0xeb00000c  0xe92d0001  0xe3000273  0xeb00000e
0x0004  0xeb000010  0xea000014  0xe50f0130  0xe50f10f4
0x0005  0xe1a0f00e  0xe3a03faa  0xe50f3140  0xe50f1144
0x0006  0xe8bd000b  0xe1a0f00e  0xe3a03fa5  0xe50f3154
0x0007  0xe50f1158  0xe8bd000b  0xe1a0f00e  0xe50f0164
0x0008  0xe51f0180  0xe1a0f00e  0xe30032ba  0xe50f3174
0x0009  0xe50f0184  0xe8bd0001  0xe1a0f00e  0xe3a02f61
0x000a  0xe59f10d8  0xe3a07000  0xe3a08000  0xeb000001
0x000b  0xeb00001e  0xea00002d  0xe92d41e8  0xe3a03e23
0x000c  0xe50f31a8  0xe3a04e1b  0xe50f4170  0xe3a07000
0x000d  0xe3a08000  0xeb000023  0xe7d46007  0xe3560000
0x000e  0x0afffff7  0xe7925008  0xe1550006  0x0a000000
0x000f  0x1a000008  0xeb00001b  0xe2877001  0xe3570004
0x0010  0x0a000000  0xbafffff3  0xe5d46004  0xe3560000
0x0011  0x0a000004  0xcaffffea  0xe2888004  0xe3580018
0x0012  0x0affffe7  0xbaffffeb  0xe8bd41e8  0xe1a0f00e
0x0013  0xe92d0170  0xe3a05000  0xe3a06000  0xe3a07e1a
0x0014  0xe92d0080  0xe7d48005  0xe7878006  0xe2866004
0x0015  0xe2855001  0xe3550004  0xbafffff9  0xe8bd0080
0x0016  0xe8bd0170  0xe1a0f00e  0xe3a08000  0xe1a0f00e
0x0017  0xe1000070  0x00000000  0x00000000  0x00000000
0x0018  0x00000006  0x00000062  0x00000063  0x00000067
0x0019  0x00000070  0x00000072  0x00000079  0x00000004
0x001a  0x00000062  0x00000063  0x00000067  0x00000079
0x001b  0x79676362  0x00000000  0x00000000  0x00000000
0x001c  0x00000000  0x00000000  0x00000000  0x00000000
0x001d  0x00000000  0x00000000  0x00000000  0x00000000
0x001e  0x00000000  0x00000000  0x00000000  0x00000000
0x001f  0x00000000  0x00000000  0x00000000  0x00000000
```

Stage 3:

In stage 3 I keep some order number same as in stage 2 but i enter secret code right after enter code maker name then i still check condition of entered secretcode, when it is correct i saved it in a secretcod array , then I just enter the codebreaker'name and the maximum number.



```
        PUSH { R4 , R5 , R6 , R7 , R8 }
        MOV R5, #0
        MOV R6 , #0
        MOV R7 , #secretcode
array1:
        LDRB R8 , [ R4 + R5]
        STR R8 , [ R7 + R6 ]
        ADD R6 , R6 , #4
        ADD R5, R5 , #1
        CMP R5 , #4
        BLT array1
        POP { R4 , R5 , R6 , R7 , R8 }
        RET
reset:
        MOV R8 ,#0
        RET
endgame:
        HALT
arrayLength: 6
arrayCharacter: 98
        99
        103
        112
        114
        121
codesecretLength: 4     //// array for secret code
secretcode: 0
        0
        0
        0
code2: .BLOCK 128
secretcode1: .ASCIZ "\t, please enter a 4-chracter secret code: "
namebreaker: .ASCIZ "\n What is codebreaker name\n"
namemaker: .ASCIZ "\n What is codemaker name\n"
number: .ASCIZ "\n What number you want to enter\n"
readnamebreaker: .ASCIZ "\n Codebreaker is:\n "
readnamemaker: .ASCIZ "\n Codemaker is: "
readnumber: .ASCIZ "\n Maximum number of guesses:\n "
newline: .ASCIZ "\n "
```

```
PC   0x00000184
LR   0x00000050
SP   0x00100000
R12  0x00000000
R11  0x00000000
R10  0x00000000
R9   0x00000000
R8   0x00000000
R7   0x00000000
R6   0x00000000
R5   0x00000000
R4   0x000001b4
R3   0x000002db
R2   0x00000188
R1   0x00000000
R0   0x00000000

Count 240
Status bits NZCV 0110
```

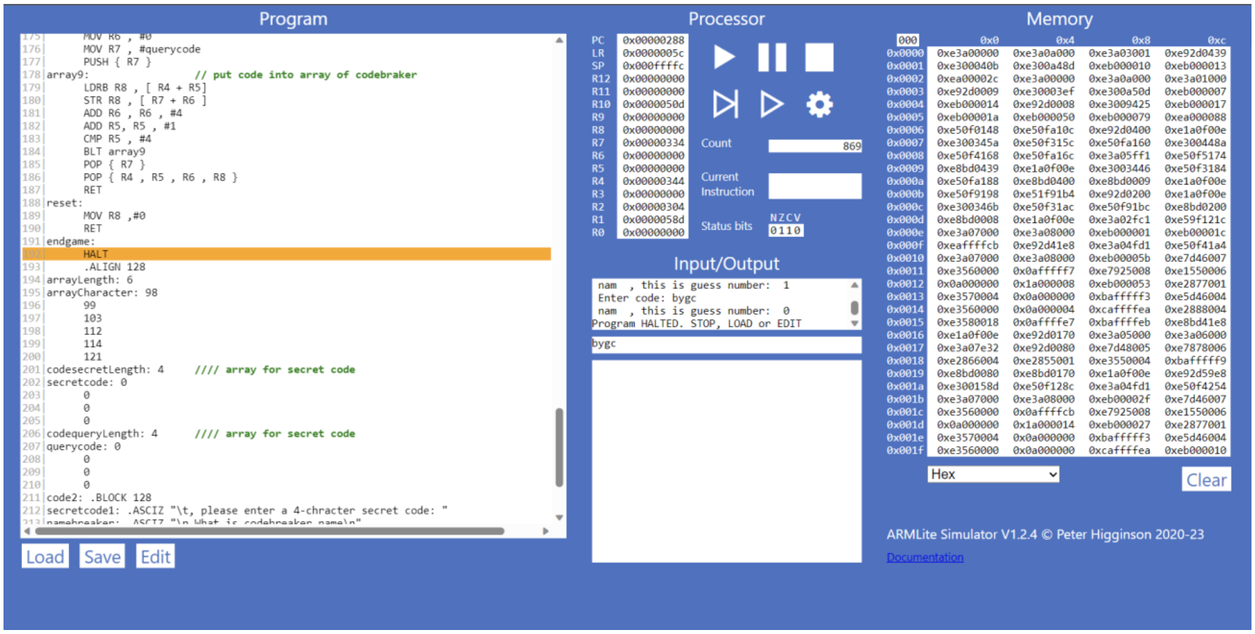
```
 Maximum number of guesses:
 2
Program HALTED. STOP, LOAD or EDIT
```

```
2
```

```
   000         0x0         0x4         0x8         0xc
0x0000  0xe3a00000  0xe3a01000  0xe3a03001  0xe92d001b
0x0001  0xe300027b  0xe30012fd  0xeb00000d  0xeb00000f
0x0002  0xea000023  0xe3a00000  0xe3a01000  0xe92d000b
0x0003  0xe300025f  0xe300137d  0xeb000005  0xeb00000f
0x0004  0xe92d0001  0xe3000295  0xeb000011  0xeb000013
0x0005  0xea00004a  0xe50f013c  0xe50f1100  0xe1a0f00e
0x0006  0xe30032ca  0xe50f314c  0xe50f1150  0xe30042fa
0x0007  0xe50f4158  0xe50f115c  0xe8bd001b  0xe1a0f00e
0x0008  0xe30032b6  0xe50f316c  0xe50f1170  0xe8bd000b
0x0009  0xe1a0f00e  0xe50f017c  0xe51f0198  0xe1a0f00e
0x000a  0xe30032db  0xe50f318c  0xe50f019c  0xe8bd0001
0x000b  0xe1a0f00e  0xe3a02f62  0xe59f10c4  0xe3a07000
0x000c  0xe3a08000  0xeb000001  0xeb00001e  0xeaffffd4
0x000d  0xe92d41e8  0xe3a03f8d  0xe50f31c0  0xe3a04f6d
0x000e  0xe50f4188  0xe3a07000  0xe3a08000  0xeb000021
0x000f  0xe7d46007  0xe3560000  0x0afffff7  0xe7925008
0x0010  0xe1550006  0x0a000000  0x1a000008  0xeb000019
0x0011  0xe2877001  0xe3570004  0x0a000000  0xbafffff3
0x0012  0xe5d46004  0xe3560000  0x0a000004  0xcaffffea
0x0013  0xe2888004  0xe3580018  0x0affffe7  0xbaffffeb
0x0014  0xe8bd41e8  0xe1a0f00e  0xe92d01f0  0xe3a05000
0x0015  0xe3a06000  0xe3a07f69  0xe7d48005  0xe7878006
0x0016  0xe2866004  0xe2855001  0xe3550004  0xbafffff9
0x0017  0xe8bd01f0  0xe1a0f00e  0xe3a08000  0xe1a0f00e
0x0018  0xe1000070  0x00000006  0x00000062  0x00000063
0x0019  0x00000067  0x00000070  0x00000072  0x00000079
0x001a  0x00000004  0x00000062  0x00000063  0x00000067
0x001b  0x00000079  0x79676362  0x00000000  0x00000000
0x001c  0x00000000  0x00000000  0x00000000  0x00000000
0x001d  0x00000000  0x00000000  0x00000000  0x00000000
0x001e  0x00000000  0x00000000  0x00000000  0x00000000
0x001f  0x00000000  0x00000000  0x00000000  0x00000000
```

Stage 4 :

In stage 4 I use getcode1 function to do codebreaker code filling process and calculate the number of guesses left after I finish filling in ' Maximum number of guesses ', codebreaker will be asked to entercode , after filling code the paragraph that code will also be checked according to the condition of the problem if the code is correct it will be stored querycode array that is done by the query function then the number of guesses will be subtracted by 1 if the input code is valid , label loopsub does that task until it is zero then it will terminate the program.



```
        MOV R6 , #0
        MOV R7 , #querycode
        PUSH { R7 }
array9:                     // put code into array of codebraker
        LDRB R8 , [ R4 + R5]
        STR R8 , [ R7 + R6 ]
        ADD R6 , R6 , #4
        ADD R5, R5 , #1
        CMP R5 , #4
        BLT array9
        POP { R7 }
        POP { R4 , R5 , R6 , R8 }
        RET
reset:
        MOV R8 ,#0
        RET
endgame:
        HALT
        .ALIGN 128
arrayLength: 6
arrayCharacter: 98
        99
        103
        112
        114
        121
codesecretLength: 4     //// array for secret code
secretcode: 0
        0
        0
        0
codequeryLength: 4      //// array for secret code
querycode: 0
        0
        0
        0
code2: .BLOCK 128
secretcode1: .ASCIZ "\t, please enter a 4-chracter secret code: "
namebreaker: .ASCIZ "\n What is codebreaker name\n"
```

```
PC   0x00000288
LR   0x0000005c
SP   0x000ffffc
R12  0x00000000
R11  0x00000000
R10  0x0000050d
R9   0x00000000
R8   0x00000000
R7   0x00000334
R6   0x00000000
R5   0x00000000
R4   0x00000344
R3   0x00000000
R2   0x00000304
R1   0x0000058d
R0   0x00000000
```

Count 869

Current Instruction

Status bits NZCV 0110

```
 nam  , this is guess number:  1
 Enter code: bygc
 nam  , this is guess number:  0
Program HALTED. STOP, LOAD or EDIT
```

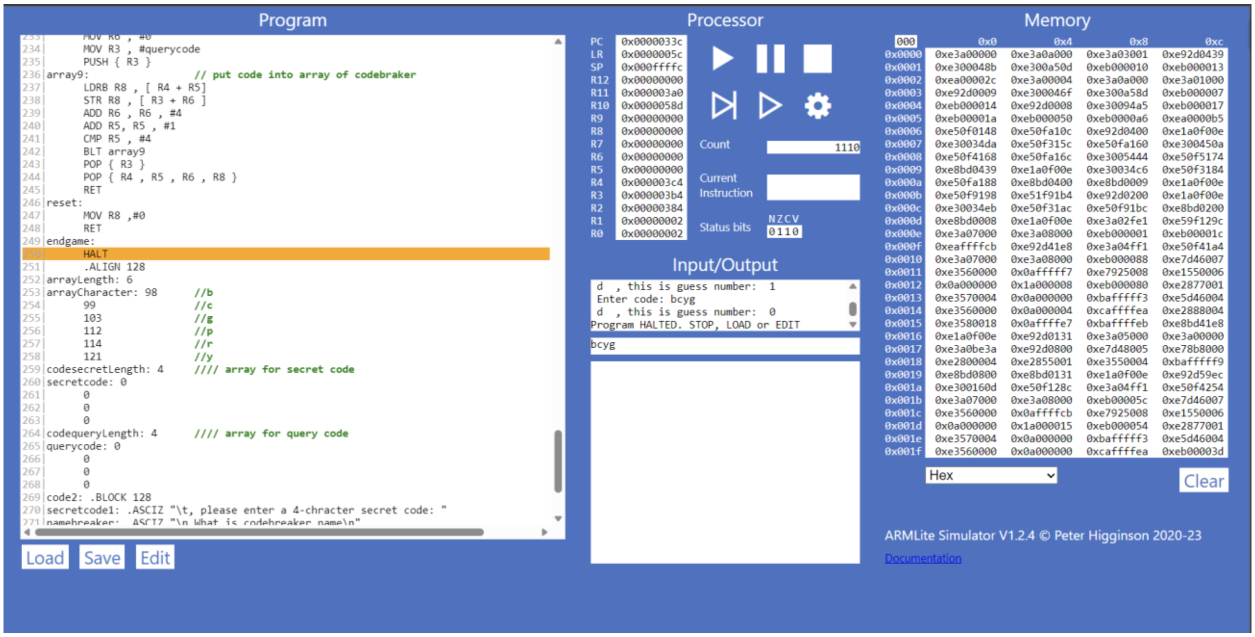
bygc

```
000            0x0         0x4         0x8         0xc
0x0000  0xe3a00000  0xe3a0a000  0xe3a03001  0xe92d0439
0x0001  0xe300040b  0xe300a48d  0xeb000010  0xeb000013
0x0002  0xea00002c  0xe3a00000  0xe3a0a000  0xe3a01000
0x0003  0xe92d0009  0xe30003ef  0xe300a50d  0xeb000007
0x0004  0xeb000014  0xe92d0008  0xe3009425  0xeb000017
0x0005  0xeb00001a  0xeb000050  0xeb000079  0xea000088
0x0006  0xe50f0148  0xe50fa10c  0xe92d0400  0xe1a0f00e
0x0007  0xe300345a  0xe50f315c  0xe50fa160  0xe300448a
0x0008  0xe50f4168  0xe50fa16c  0xe3a05ff1  0xe50f5174
0x0009  0xe8bd0439  0xe1a0f00e  0xe3003446  0xe50f3184
0x000a  0xe50fa188  0xe8bd0400  0xe8bd0009  0xe1a0f00e
0x000b  0xe50f9198  0xe51f91b4  0xe92d0200  0xe1a0f00e
0x000c  0xe300346b  0xe50f31ac  0xe50f91bc  0xe8bd0200
0x000d  0xe8bd0008  0xe1a0f00e  0xe3a02fc1  0xe59f121c
0x000e  0xe3a07000  0xe3a08000  0xeb000001  0xeb00001c
0x000f  0xeaffffcb  0xe92d41e8  0xe3a04fd1  0xe50f41a4
0x0010  0xe3a07000  0xe3a08000  0xeb00005b  0xe7d46007
0x0011  0xe3560000  0x0afffff7  0xe7925008  0xe1550006
0x0012  0x0a000000  0x1a000008  0xeb000053  0xe2877001
0x0013  0xe3570004  0x0a000000  0xbafffff3  0xe5d46004
0x0014  0xe3560000  0x0a000004  0xcaffffea  0xe2888004
0x0015  0xe3580018  0x0affffe7  0xbaffffeb  0xe8bd41e8
0x0016  0xe1a0f00e  0xe92d0170  0xe3a05000  0xe3a06000
0x0017  0xe3a07e32  0xe92d0080  0xe7d48005  0xe7878006
0x0018  0xe2866004  0xe2855001  0xe3550004  0xbafffff9
0x0019  0xe8bd0080  0xe8bd0170  0xe1a0f00e  0xe92d59e8
0x001a  0xe300158d  0xe50f128c  0xe3a04fd1  0xe50f4254
0x001b  0xe3a07000  0xe3a08000  0xeb00002f  0xe7d46007
0x001c  0xe3560000  0x0affffcb  0xe7925008  0xe1550006
0x001d  0x0a000000  0x1a000014  0xeb000027  0xe2877001
0x001e  0xe3570004  0x0a000000  0xbafffff3  0xe5d46004
0x001f  0xe3560000  0x0a000000  0xcaffffea  0xeb000010
```

ARMLite Simulator V1.2.4 © Peter Higginson 2020-23

Stage 5a:

In stage 5a, I use comparecodes funtion to check if the secret code and query code are the same according to each case that the problem provides. I use the compare loop to contain the condition of checking how many peg queries directly match its corresponding peg in the secret code of case 1, in this condition I will first get the position of each element in the querycode array and in the secretcode array to compare if any elements overlap I will +4 at the same time to the addresses of the 2 arrays to move the next element in the array and +1 for R0 (Case 1 count) only to get the next element until it reaches the position of the last element in the array. Then I do a condition check for case 2, I also take the position of each element in the querycode array and in the secretcode array for comparison but here when the two elements are equal I do not add +4 bytes at the same time to get the next address of the 2 arrays but I will +4 for each depending on the condition in the code. This means that I use the loop to compare all the elements in the secretcode array with an element in the querycode array until the end of the loop, then I will +4 to get the next element of the querycode and I start the loop again. That allows me to know how many elements in the querycode array coincide with the parts in the secretcode array including both same and different locations and save it to R1 then I subtract it from R0 This will allow me to count the number of query handles that match at least one other one in the secret code but are not in the correct location and then save the link input fruit for R1 (Case 2 count).



```
        MOV R3 , #querycode
        PUSH { R3 }
array9:                 // put code into array of codebraker
        LDRB R8 , [ R4 + R5]
        STR R8 , [ R3 + R6 ]
        ADD R6 , R6 , #4
        ADD R5, R5 , #1
        CMP R5 , #4
        BLT array9
        POP { R3 }
        POP { R4 , R5 , R6 , R8 }
        RET
reset:
        MOV R8 ,#0
        RET
endgame:
        HALT
        .ALIGN 128
arrayLength: 6
arrayCharacter: 98      //b
        99              //c
        103             //g
        112             //p
        114             //r
        121             //y
codesecretLength: 4     //// array for secret code
secretcode: 0
        0
        0
        0
codequeryLength: 4      //// array for query code
querycode: 0
        0
        0
        0
code2: .BLOCK 128
secretcode1: .ASCIZ "\t, please enter a 4-chracter secret code: "
namebreaker: .ASCIZ "\n What is codebreaker name\n"
```

```
PC   0x0000033c
LR   0x0000005c
SP   0x000ffffc
R12  0x00000000
R11  0x000003a0
R10  0x0000058d
R9   0x00000000
R8   0x00000000
R7   0x00000000
R6   0x00000000
R5   0x00000000
R4   0x000003c4
R3   0x000003b4
R2   0x00000384
R1   0x00000002
R0   0x00000002
```

Count 1110

Current Instruction

Status bits NZCV 0110

```
 d  , this is guess number:  1
 Enter code: bcyg
 d  , this is guess number:  0
Program HALTED. STOP, LOAD or EDIT
```

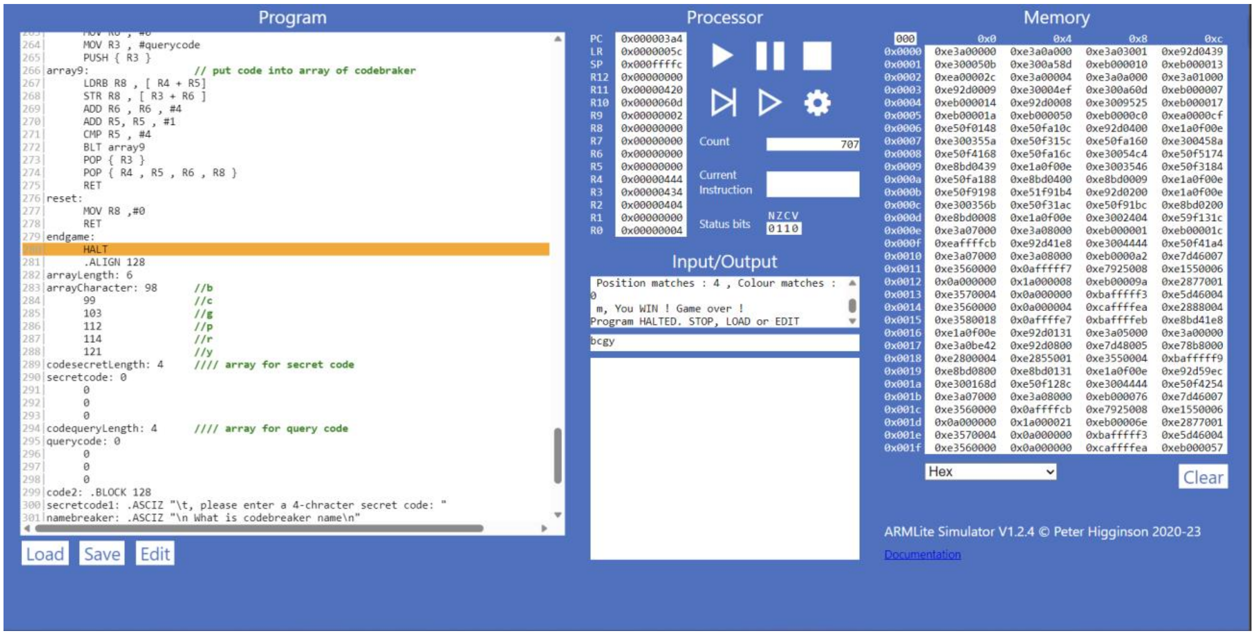
bcyg

```
000       0x0         0x4         0x8         0xc
0x0000  0xe3a00000  0xe3a0a000  0xe3a03001  0xe92d0439
0x0001  0xe300048b  0xe300a50d  0xeb000010  0xeb000013
0x0002  0xea00002c  0xe3a00004  0xe3a0a000  0xe3a01000
0x0003  0xe92d0009  0xe300046f  0xe300a58d  0xeb000007
0x0004  0xeb000014  0xe92d0008  0xe30094a5  0xeb000017
0x0005  0xeb00001a  0xeb000050  0xeb0000a6  0xea0000b5
0x0006  0xe50f0148  0xe50fa10c  0xe92d0400  0xe1a0f00e
0x0007  0xe30034da  0xe50f315c  0xe50fa160  0xe300450a
0x0008  0xe50f4168  0xe50fa16c  0xe3005444  0xe50f5174
0x0009  0xe8bd0439  0xe1a0f00e  0xe30034c6  0xe50f3184
0x000a  0xe50fa188  0xe8bd0400  0xe8bd0009  0xe1a0f00e
0x000b  0xe50f9198  0xe51f91b4  0xe92d0200  0xe1a0f00e
0x000c  0xe30034eb  0xe50f31ac  0xe50f91bc  0xe8bd0200
0x000d  0xe8bd0008  0xe1a0f00e  0xe3a02fe1  0xe59f129c
0x000e  0xe3a07000  0xe3a08000  0xeb000001  0xeb00001c
0x000f  0xeaffffcb  0xe92d41e8  0xe3a04ff1  0xe50f41a4
0x0010  0xe3a07000  0xe3a08000  0xeb000088  0xe7d46007
0x0011  0xe3560000  0x0afffff7  0xe7925008  0xe1550006
0x0012  0x0a000000  0x1a000008  0xeb000080  0xe2877001
0x0013  0xe3570004  0x0a000000  0xbafffff3  0xe5d46004
0x0014  0xe3560000  0x0a000004  0xcaffffea  0xe2888004
0x0015  0xe3580018  0x0affffe7  0xbaffffeb  0xe8bd41e8
0x0016  0xe1a0f00e  0xe92d0131  0xe3a05000  0xe3a00000
0x0017  0xe3a0be3a  0xe92d0800  0xe7d48005  0xe78b8000
0x0018  0xe2800004  0xe2855001  0xe3550004  0xbafffff9
0x0019  0xe8bd0800  0xe8bd0131  0xe1a0f00e  0xe92d59ec
0x001a  0xe300160d  0xe50f128c  0xe3a04ff1  0xe50f4254
0x001b  0xe3a07000  0xe3a08000  0xeb00005c  0xe7d46007
0x001c  0xe3560000  0x0affffcb  0xe7925008  0xe1550006
0x001d  0x0a000000  0x1a000015  0xeb000054  0xe2877001
0x001e  0xe3570004  0x0a000000  0xbafffff3  0xe5d46004
0x001f  0xe3560000  0x0a000000  0xcaffffea  0xeb00003d
```

ARMLite Simulator V1.2.4 © Peter Higginson 2020-23

Stage 5b:

In stage 5B I print out you win and you lose and the game over output. If Case 1 counts 4, it will print ' You win ' and print 'Game over! 'The end of the show and if the number of guesses returns does not mean that there is no case Case 1 count equal 4 then it will print 'You LOSE! ' and print out 'Game Over!' at the end of the show.



```
        MOV R3 , #querycode
        PUSH { R3 }
array9:                 // put code into array of codebraker
        LDRB R8 , [ R4 + R5]
        STR R8 , [ R3 + R6 ]
        ADD R6 , R6 , #4
        ADD R5, R5 , #1
        CMP R5 , #4
        BLT array9
        POP { R3 }
        POP { R4 , R5 , R6 , R8 }
        RET
reset:
        MOV R8 ,#0
        RET
endgame:
        HALT
        .ALIGN 128
arrayLength: 6
arrayCharacter: 98      //b
        99              //c
        103             //g
        112             //p
        114             //r
        121             //y
codesecretLength: 4     //// array for secret code
secretcode: 0
        0
        0
        0
codequeryLength: 4      //// array for query code
querycode: 0
        0
        0
        0
code2: .BLOCK 128
secretcode1: .ASCIZ "\t, please enter a 4-chracter secret code: "
namebreaker: .ASCIZ "\n What is codebreaker name\n"
```

```
PC   0x000003a4
LR   0x0000005c
SP   0x000ffffc
R12  0x00000000
R11  0x00000420
R10  0x0000060d
R9   0x00000002
R8   0x00000000
R7   0x00000000
R6   0x00000000
R5   0x00000000
R4   0x00000444
R3   0x00000434
R2   0x00000404
R1   0x00000000
R0   0x00000004
```

Count 707

Current Instruction

Status bits NZCV 0110

```
 Position matches : 4 , Colour matches :
0
 m, You WIN ! Game over !
Program HALTED. STOP, LOAD or EDIT
```

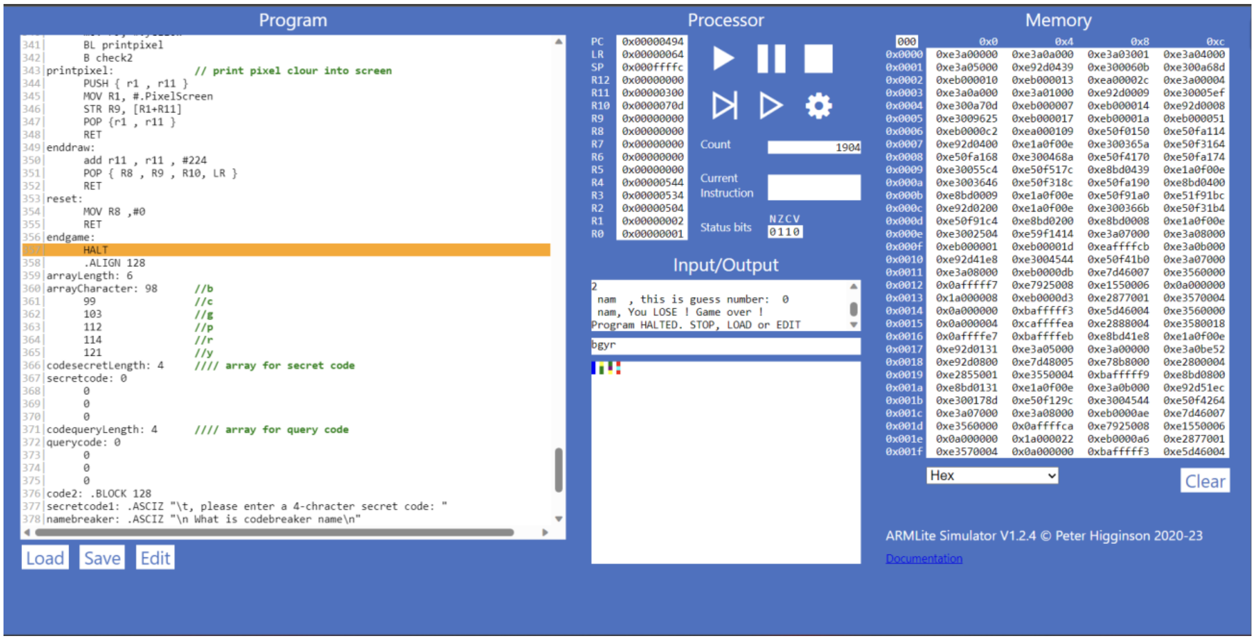
bcgy

```
000       0x0         0x4         0x8         0xc
0x0000  0xe3a00000  0xe3a0a000  0xe3a03001  0xe92d0439
0x0001  0xe300050b  0xe300a58d  0xeb000010  0xeb000013
0x0002  0xea00002c  0xe3a00004  0xe3a0a000  0xe3a01000
0x0003  0xe92d0009  0xe30004ef  0xe300a60d  0xeb000007
0x0004  0xeb000014  0xe92d0008  0xe3009525  0xeb000017
0x0005  0xeb00001a  0xeb000050  0xeb0000c0  0xea0000cf
0x0006  0xe50f0148  0xe50fa10c  0xe92d0400  0xe1a0f00e
0x0007  0xe300355a  0xe50f315c  0xe50fa160  0xe300458a
0x0008  0xe50f4168  0xe50fa16c  0xe30054c4  0xe50f5174
0x0009  0xe8bd0439  0xe1a0f00e  0xe3003546  0xe50f3184
0x000a  0xe50fa188  0xe8bd0400  0xe8bd0009  0xe1a0f00e
0x000b  0xe50f9198  0xe51f91b4  0xe92d0200  0xe1a0f00e
0x000c  0xe300356b  0xe50f31ac  0xe50f91bc  0xe8bd0200
0x000d  0xe8bd0008  0xe1a0f00e  0xe3002404  0xe59f131c
0x000e  0xe3a07000  0xe3a08000  0xeb000001  0xeb00001c
0x000f  0xeaffffcb  0xe92d41e8  0xe3004444  0xe50f41a4
0x0010  0xe3a07000  0xe3a08000  0xeb0000a2  0xe7d46007
0x0011  0xe3560000  0x0afffff7  0xe7925008  0xe1550006
0x0012  0x0a000000  0x1a000008  0xeb00009a  0xe2877001
0x0013  0xe3570004  0x0a000000  0xbafffff3  0xe5d46004
0x0014  0xe3560000  0x0a000004  0xcaffffea  0xe2888004
0x0015  0xe3580018  0x0affffe7  0xbaffffeb  0xe8bd41e8
0x0016  0xe1a0f00e  0xe92d0131  0xe3a05000  0xe3a00000
0x0017  0xe3a0be42  0xe92d0800  0xe7d48005  0xe78b8000
0x0018  0xe2800004  0xe2855001  0xe3550004  0xbafffff9
0x0019  0xe8bd0800  0xe8bd0131  0xe1a0f00e  0xe92d59ec
0x001a  0xe300168d  0xe50f128c  0xe3004444  0xe50f4254
0x001b  0xe3a07000  0xe3a08000  0xeb000076  0xe7d46007
0x001c  0xe3560000  0x0affffcb  0xe7925008  0xe1550006
0x001d  0x0a000000  0x1a000021  0xeb00006e  0xe2877001
0x001e  0xe3570004  0x0a000000  0xbafffff3  0xe5d46004
0x001f  0xe3560000  0x0a000000  0xcaffffea  0xeb000057
```

ARMLite Simulator V1.2.4 © Peter Higginson 2020-23

Stage 6 :

In stage 6 I printed a horizontal line of 4 pixels each representing a color in the corresponding query code. I use the drawpixel function to check if each character in the querycode is equal to the value of one of the 6 characters corresponding to 6 colors or not, checkpixel will be responsible for doing that if it is assigned a color corresponding to that character and then will switch to drawpixel to print that color to the output screen.



```
      BL printpixel
      B check2
printpixel:             // print pixel clour into screen
      PUSH { r1 , r11 }
      MOV R1, #.PixelScreen
      STR R9, [R1+R11]
      POP {r1 , r11 }
      RET
enddraw:
      add r11 , r11 , #224
      POP { R8 , R9 , R10, LR }
      RET
reset:
      MOV R8 ,#0
      RET
endgame:
      HALT
      .ALIGN 128
arrayLength: 6
arrayCharacter: 98      //b
      99                //c
      103               //g
      112               //p
      114               //r
      121               //y
codesecretLength: 4     //// array for secret code
secretcode: 0
      0
      0
      0
codequeryLength: 4      //// array for query code
querycode: 0
      0
      0
      0
code2: .BLOCK 128
secretcode1: .ASCIZ "\t, please enter a 4-chracter secret code: "
namebreaker: .ASCIZ "\n What is codebreaker name\n"
```

```
PC   0x00000494
LR   0x00000064
SP   0x000ffffc
R12  0x00000000
R11  0x00000300
R10  0x0000070d
R9   0x00000000
R8   0x00000000
R7   0x00000000
R6   0x00000000
R5   0x00000000
R4   0x00000544
R3   0x00000534
R2   0x00000504
R1   0x00000002
R0   0x00000001
```

Count 1904

Current Instruction

Status bits NZCV 0110

```
2
 nam  , this is guess number:  0
 nam, You LOSE ! Game over !
Program HALTED. STOP, LOAD or EDIT
```

```
bgyr
```

```
000           0x0         0x4         0x8         0xc
0x0000  0xe3a00000  0xe3a0a000  0xe3a03001  0xe3a04000
0x0001  0xe3a05000  0xe92d0439  0xe300060b  0xe300a68d
0x0002  0xeb000010  0xeb000013  0xea00002c  0xe3a00004
0x0003  0xe3a0a000  0xe3a01000  0xe92d0009  0xe30005ef
0x0004  0xe300a70d  0xeb000007  0xeb000014  0xe92d0008
0x0005  0xe3009625  0xeb000017  0xeb00001a  0xeb000051
0x0006  0xeb0000c2  0xea000109  0xe50f0150  0xe50fa114
0x0007  0xe92d0400  0xe1a0f00e  0xe300365a  0xe50f3164
0x0008  0xe50fa168  0xe300468a  0xe50f4170  0xe50fa174
0x0009  0xe30055c4  0xe50f517c  0xe8bd0439  0xe1a0f00e
0x000a  0xe3003646  0xe50f318c  0xe50fa190  0xe8bd0400
0x000b  0xe8bd0009  0xe1a0f00e  0xe50f91a0  0xe51f91bc
0x000c  0xe92d0200  0xe1a0f00e  0xe300366b  0xe50f31b4
0x000d  0xe50f91c4  0xe8bd0200  0xe8bd0008  0xe1a0f00e
0x000e  0xe3002504  0xe59f1414  0xe3a07000  0xe3a08000
0x000f  0xeb000001  0xeb00001d  0xeaffffcb  0xe3a0b000
0x0010  0xe92d41e8  0xe3004544  0xe50f41b0  0xe3a07000
0x0011  0xe3a08000  0xeb0000db  0xe7d46007  0xe3560000
0x0012  0x0afffff7  0xe7925008  0xe1550006  0x0a000000
0x0013  0x1a000008  0xeb0000d3  0xe2877001  0xe3570004
0x0014  0x0a000000  0xbafffff3  0xe5d46004  0xe3560000
0x0015  0x0a000004  0xcaffffea  0xe2888004  0xe3580018
0x0016  0x0affffe7  0xbaffffeb  0xe8bd41e8  0xe1a0f00e
0x0017  0xe92d0131  0xe3a05000  0xe3a00000  0xe3a0be52
0x0018  0xe92d0800  0xe7d48005  0xe78b8000  0xe2800004
0x0019  0xe2855001  0xe3550004  0xbafffff9  0xe8bd0800
0x001a  0xe8bd0131  0xe1a0f00e  0xe3a0b000  0xe92d51ec
0x001b  0xe300178d  0xe50f129c  0xe3004544  0xe50f4264
0x001c  0xe3a07000  0xe3a08000  0xeb0000ae  0xe7d46007
0x001d  0xe3560000  0x0affffca  0xe7925008  0xe1550006
0x001e  0x0a000000  0x1a000022  0xeb0000a6  0xe2877001
0x001f  0xe3570004  0x0a000000  0xbafffff3  0xe5d46004
```

ARMLite Simulator V1.2.4 © Peter Higginson 2020-23

Assumption I have made

- I tried to do this assignment without any assumption .

Problem

- My program seems to work as instruction, I have tested the test run I have not come across an error. Maybe there are still some problems that will exist, I think there is nothing perfect in real life