2022

ASSIGNMENT 2

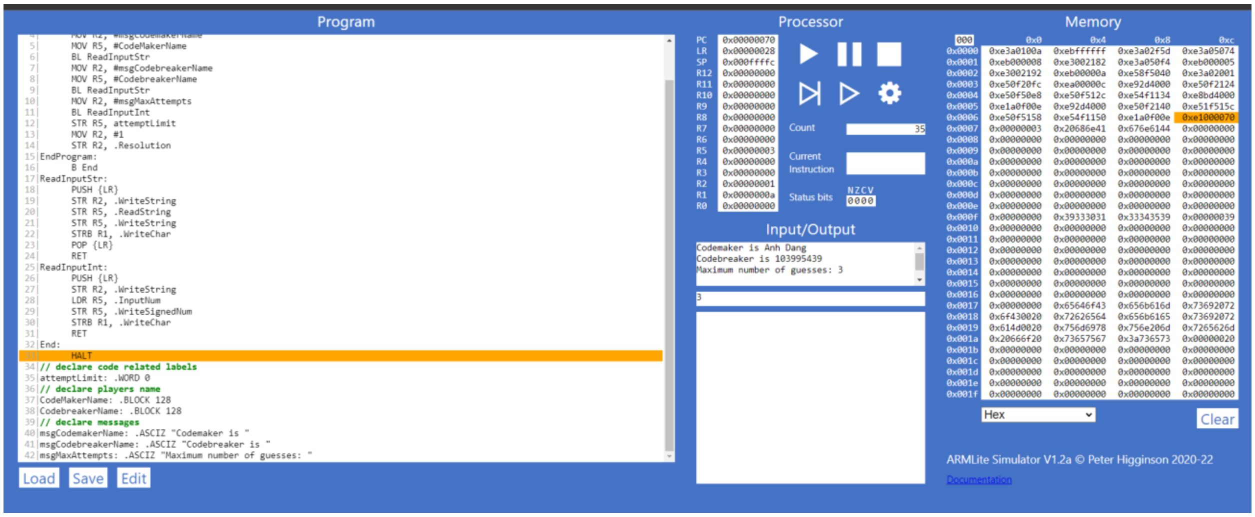
REPORT

TRAN DUC ANH DANG | 103995439

Aim:

The objective of this project is to recreate a working Mastermind game in which the Codebreaker must decipher a secret code created by the Codemaker using a series of four different coloured pegs. Each visitor's feedback helps to limit the code's potential outcomes. The player who cracks his opponent's secret code with the fewest number of guesses wins.

Stage 1:



```
        MOV R2, #msgCodemakerName
        MOV R5, #CodeMakerName
        BL ReadInputStr
        MOV R2, #msgCodebreakerName
        MOV R5, #CodebreakerName
        BL ReadInputStr
        MOV R2, #msgMaxAttempts
        BL ReadInputInt
        STR R5, attemptLimit
        MOV R2, #1
        STR R2, .Resolution
EndProgram:
        B End
ReadInputStr:
        PUSH {LR}
        STR R2, .WriteString
        STR R5, .ReadString
        STR R5, .WriteString
        STRB R1, .WriteChar
        POP {LR}
        RET
ReadInputInt:
        PUSH {LR}
        STR R2, .WriteString
        LDR R5, .InputNum
        STR R5, .WriteSignedNum
        STRB R1, .WriteChar
        RET
End:
        HALT
// declare code related labels
attemptLimit: .WORD 0
// declare players name
CodeMakerName: .BLOCK 128
CodebreakerName: .BLOCK 128
// declare messages
msgCodemakerName: .ASCIZ "Codemaker is "
msgCodebreakerName: .ASCIZ "Codebreaker is "
msgMaxAttempts: .ASCIZ "Maximum number of guesses: "
```

```
PC   0x00000070
LR   0x00000028
SP   0x000ffffc
R12  0x00000000
R11  0x00000000
R10  0x00000000
R9   0x00000000
R8   0x00000000
R7   0x00000000
R6   0x00000000
R5   0x00000003
R4   0x00000000
R3   0x00000000
R2   0x00000001
R1   0x0000000a
R0   0x00000000
```

Count: 35

Current Instruction:

Status bits: NZCV 0000

```
Codemaker is Anh Dang
Codebreaker is 103995439
Maximum number of guesses: 3
```

```
3
```

```
000          0x0         0x4         0x8         0xc
0x0000  0xe3a0100a  0xebffffff  0xe3a02f5d  0xe3a05074
0x0001  0xeb000008  0xe3002182  0xe3a050f4  0xeb000005
0x0002  0xe3002192  0xeb00000a  0xe58f5040  0xe3a02001
0x0003  0xe50f20fc  0xea00000c  0xe92d4000  0xe50f2124
0x0004  0xe50f50e8  0xe50f512c  0xe54f1134  0xe8bd4000
0x0005  0xe1a0f00e  0xe92d4000  0xe50f2140  0xe51f515c
0x0006  0xe50f5158  0xe54f1150  0xe1a0f00e  0xe1000070
0x0007  0x00000003  0x20686e41  0x676e6144  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  0x39333031  0x33343539  0x00000039
0x0010  0x00000000  0x00000000  0x00000000  0x00000000
0x0011  0x00000000  0x00000000  0x00000000  0x00000000
0x0012  0x00000000  0x00000000  0x00000000  0x00000000
0x0013  0x00000000  0x00000000  0x00000000  0x00000000
0x0014  0x00000000  0x00000000  0x00000000  0x00000000
0x0015  0x00000000  0x00000000  0x00000000  0x00000000
0x0016  0x00000000  0x00000000  0x00000000  0x00000000
0x0017  0x00000000  0x65646f43  0x656b616d  0x73692072
0x0018  0x6f430020  0x72626564  0x656b6165  0x73692072
0x0019  0x614d0020  0x756d6978  0x756e206d  0x7265626d
0x001a  0x20666f20  0x73657567  0x3a736573  0x00000020
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
```

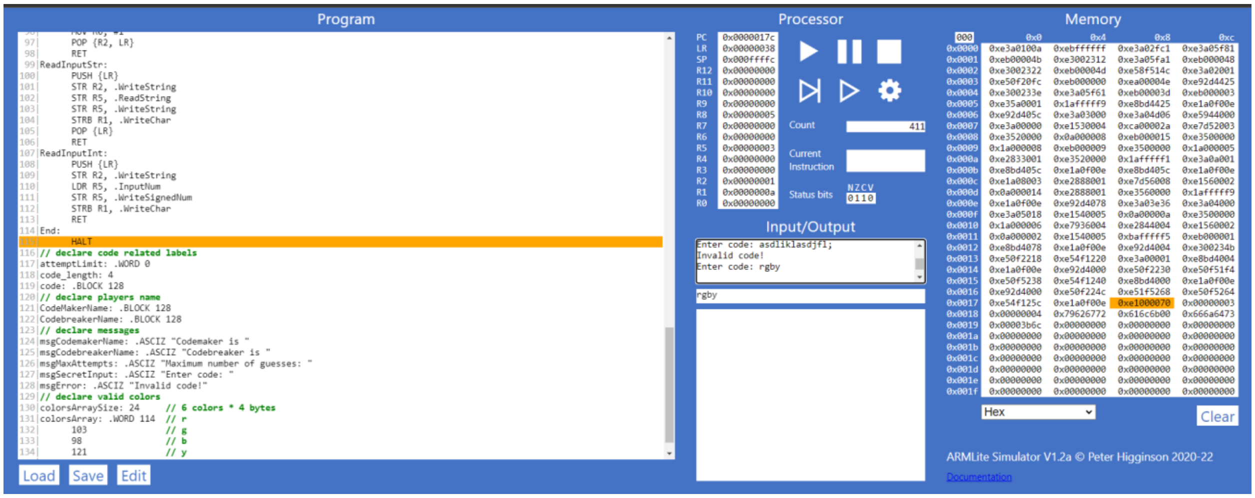
In Stage 1, I have created a few extra functions such as Program, ReadInputStr, ReadInputInt

Functions:

- Program: This is the main function where I called at the start to run the code
- ReadInputStr: This is the function where I use to read string input such as Codemaker and Codebreaker name
- ReadInputInt: This is the function I use to read integer input such as the maximum guesses that the player could have

All the functions have worked as required in the instruction and I haven’t yet found any issues at this stage.

Stage 2:



```
        POP {R2, LR}
        RET
ReadInputStr:
        PUSH {LR}
        STR R2, .WriteString
        STR R5, .ReadString
        STR R5, .WriteString
        STRB R1, .WriteChar
        POP {LR}
        RET
ReadInputInt:
        PUSH {LR}
        STR R2, .WriteString
        LDR R5, .InputNum
        STR R5, .WriteSignedNum
        STRB R1, .WriteChar
        RET
End:
        HALT
// declare code related labels
attemptLimit: .WORD 0
code_length: 4
code: .BLOCK 128
// declare players name
CodeMakerName: .BLOCK 128
CodebreakerName: .BLOCK 128
// declare messages
msgCodemakerName: .ASCIZ "Codemaker is "
msgCodebreakerName: .ASCIZ "Codebreaker is "
msgMaxAttempts: .ASCIZ "Maximum number of guesses: "
msgSecretInput: .ASCIZ "Enter code: "
msgError: .ASCIZ "Invalid code!"
// declare valid colors
colorsArraySize: 24     // 6 colors * 4 bytes
colorsArray: .WORD 114  // r
        103             // g
        98              // b
        121             // y
```

```
PC   0x0000017c
LR   0x00000038
SP   0x000ffffc
R12  0x00000000
R11  0x00000000
R10  0x00000000
R9   0x00000000
R8   0x00000005
R7   0x00000000
R6   0x00000000
R5   0x00000003
R4   0x00000000
R3   0x00000000
R2   0x00000001
R1   0x0000000a
R0   0x00000000

Count               411
Current Instruction
Status bits  NZCV
             0110
```

```
Enter code: asdliklasdjfl;
Invalid code!
Enter code: rgby
```

```
rgby
```

```
000          0x0        0x4        0x8        0xc
0x0000  0xe3a0100a 0xebffffff 0xe3a02fc1 0xe3a05f81
0x0001  0xeb00004b 0xe3002312 0xe3a05fa1 0xeb000048
0x0002  0xe3002322 0xeb00004d 0xe58f514c 0xe3a02001
0x0003  0xe50f20fc 0xeb000000 0xea00004e 0xe92d4425
0x0004  0xe300233e 0xe3a05f61 0xeb00003d 0xeb000003
0x0005  0xe35a0001 0x1afffff9 0xe8bd4425 0xe1a0f00e
0x0006  0xe92d405c 0xe3a03000 0xe3a04d06 0xe5944000
0x0007  0xe3a00000 0xe1530004 0xca00002a 0xe7d52003
0x0008  0xe3520000 0x0a000008 0xeb000015 0xe3500000
0x0009  0x1a000008 0xeb000009 0xe3500000 0x1a000005
0x000a  0xe2833001 0xe3520000 0x1afffff1 0xe3a0a001
0x000b  0xe8bd405c 0xe1a0f00e 0xe8bd405c 0xe1a0f00e
0x000c  0xe1a08003 0xe2888001 0xe7d56008 0xe1560002
0x000d  0x0a000014 0xe2888001 0xe3560000 0x1afffff9
0x000e  0xe1a0f00e 0xe92d4078 0xe3a03e36 0xe3a04000
0x000f  0xe3a05018 0xe1540005 0x0a00000a 0xe3500000
0x0010  0x1a000006 0xe7936004 0xe2844004 0xe1560002
0x0011  0x0a000002 0xe1540005 0xbafffff5 0xeb000001
0x0012  0xe8bd4078 0xe1a0f00e 0xe92d4004 0xe300234b
0x0013  0xe50f2218 0xe54f1220 0xe3a00001 0xe8bd4004
0x0014  0xe1a0f00e 0xe92d4000 0xe50f2230 0xe50f51f4
0x0015  0xe50f5238 0xe54f1240 0xe8bd4000 0xe1a0f00e
0x0016  0xe92d4000 0xe50f224c 0xe51f5268 0xe50f5264
0x0017  0xe54f125c 0xe1a0f00e 0xe1000070 0x00000003
0x0018  0x00000004 0x79626772 0x616c6b00 0x666a6473
0x0019  0x00003b6c 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
```

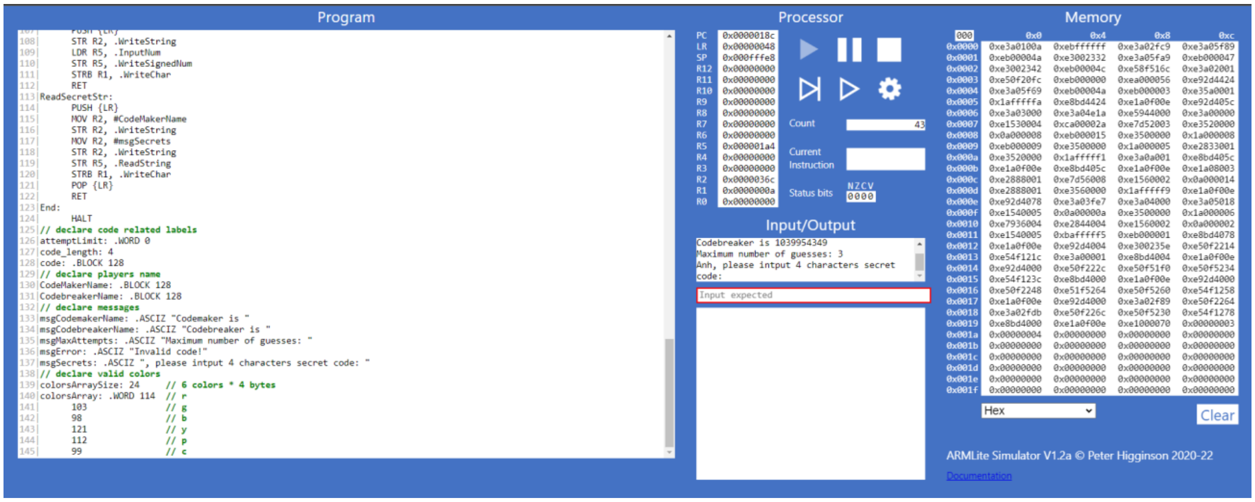
In this stage, I have added functions called SecretCode, CodeValidation, ReadCharValidation, ColorValidation, and CodeErrors.

At functions:

- SecretCode: This function uses function ReadInputStr to read the input and uses CodeValidation to validate whether the input is valid.
- CodeValidation: This function will double check whether it is matching the valid length (4) and the validate the character whether they are valid color
- ReadCharValidation: This function will double check whether same character has been input example (rrrr) will be invalid and (rgby) will be the valid input. Therefore if it is invalid, this will throw in a CodeErrors.
- CodeErrors: This uses to print out the error message.

All the functions have worked as required in the instruction and I haven't yet found any issues at this stage.

Stage 3:



```
        STR R2, .WriteString
        LDR R5, .InputNum
        STR R5, .WriteSignedNum
        STRB R1, .WriteChar
        RET
ReadSecretStr:
        PUSH {LR}
        MOV R2, #CodeMakerName
        STR R2, .WriteString
        MOV R2, #msgSecrets
        STR R2, .WriteString
        STR R5, .ReadString
        STRB R1, .WriteChar
        POP {LR}
        RET
End:
        HALT
// declare code related labels
attemptLimit: .WORD 0
code_length: 4
code: .BLOCK 128
// declare players name
CodeMakerName: .BLOCK 128
CodebreakerName: .BLOCK 128
// declare messages
msgCodemakerName: .ASCIZ "Codemaker is "
msgCodebreakerName: .ASCIZ "Codebreaker is "
msgMaxAttempts: .ASCIZ "Maximum number of guesses: "
msgError: .ASCIZ "Invalid code!"
msgSecrets: .ASCIZ ", please intput 4 characters secret code: "
// declare valid colors
colorsArraySize: 24     // 6 colors * 4 bytes
colorsArray: .WORD 114  // r
        103             // g
        98              // b
        121             // y
        112             // p
        99              // c
```

```
PC   0x0000018c
LR   0x00000048
SP   0x000fffe8
R12  0x00000000
R11  0x00000000
R10  0x00000000
R9   0x00000000
R8   0x00000000
R7   0x00000000
R6   0x00000000
R5   0x000001a4
R4   0x00000000
R3   0x00000000
R2   0x0000036c
R1   0x0000000a
R0   0x00000000

Count               43
Current Instruction
Status bits  NZCV
             0000
```

```
Codebreaker is 1039954349
Maximum number of guesses: 3
Anh, please intput 4 characters secret
code:
```

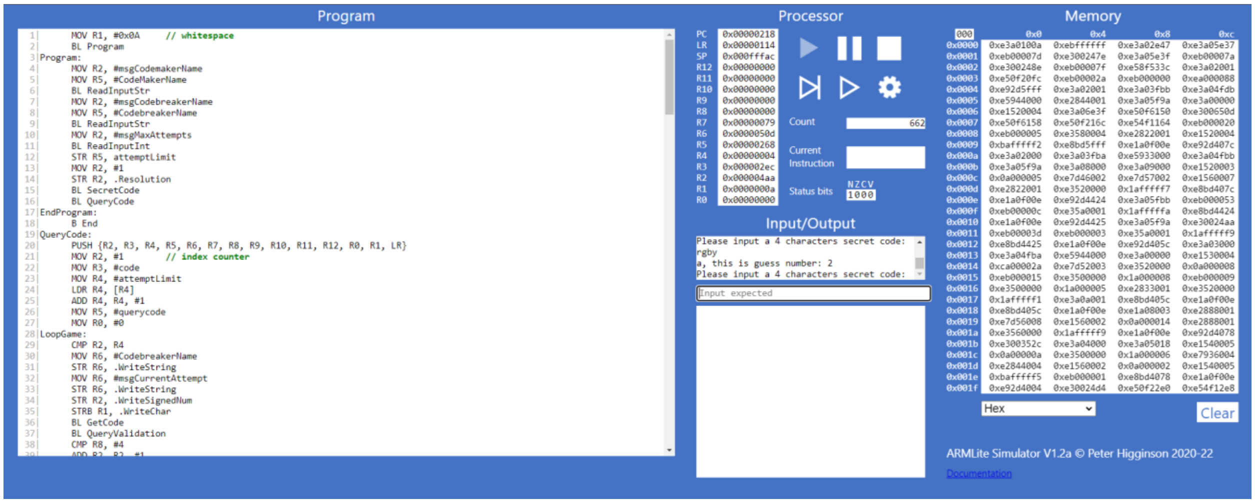
```
Input expected
```

```
000          0x0        0x4        0x8        0xc
0x0000  0xe3a0100a 0xebffffff 0xe3a02fc9 0xe3a05f89
0x0001  0xeb00004a 0xe3002332 0xe3a05fa9 0xeb000047
0x0002  0xe3002342 0xeb00004c 0xe58f516c 0xe3a02001
0x0003  0xe50f20fc 0xeb000000 0xea000056 0xe92d4424
0x0004  0xe3a05f69 0xeb00004a 0xeb000003 0xe35a0001
0x0005  0x1afffffa 0xe8bd4424 0xe1a0f00e 0xe92d405c
0x0006  0xe3a03000 0xe3a04e1a 0xe5944000 0xe3a00000
0x0007  0xe1530004 0xca00002a 0xe7d52003 0xe3520000
0x0008  0x0a000008 0xeb000015 0xe3500000 0x1a000008
0x0009  0xeb000009 0xe3500000 0x1a000005 0xe2833001
0x000a  0xe3520000 0x1afffff1 0xe3a0a001 0xe8bd405c
0x000b  0xe1a0f00e 0xe8bd405c 0xe1a0f00e 0xe1a08003
0x000c  0xe2888001 0xe7d56008 0xe1560002 0x0a000014
0x000d  0xe2888001 0xe3560000 0x1afffff9 0xe1a0f00e
0x000e  0xe92d4078 0xe3a03fe7 0xe3a04000 0xe3a05018
0x000f  0xe1540005 0x0a00000a 0xe3500000 0x1a000006
0x0010  0xe7936004 0xe2844004 0xe1560002 0x0a000002
0x0011  0xe1540005 0xbafffff5 0xeb000001 0xe8bd4078
0x0012  0xe1a0f00e 0xe92d4004 0xe300235e 0xe50f2214
0x0013  0xe54f121c 0xe3a00001 0xe8bd4004 0xe1a0f00e
0x0014  0xe92d4000 0xe50f222c 0xe50f51f0 0xe50f5234
0x0015  0xe54f123c 0xe8bd4000 0xe1a0f00e 0xe92d4000
0x0016  0xe50f2248 0xe51f5264 0xe50f5260 0xe54f1258
0x0017  0xe1a0f00e 0xe92d4000 0xe3a02f89 0xe50f2264
0x0018  0xe3a02fdb 0xe50f226c 0xe50f5230 0xe54f1278
0x0019  0xe8bd4000 0xe1a0f00e 0xe1000070 0x00000003
0x001a  0x00000004 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
```

In this stage, I added a function call ReadSecretStr. This is the function where it will get a CodeMakerName and add Message down the back. Therefore I have the result: “<codemaker name> , please intput 4 characters secret code: “ as required.

All the functions have worked as required in the instruction and I haven’t yet found any issues at this stage.

Stage 4:



Program

```
        MOV R1, #0x0A     // whitespace
        BL Program
Program:
        MOV R2, #msgCodemakerName
        MOV R5, #CodeMakerName
        BL ReadInputStr
        MOV R2, #msgCodebreakerName
        MOV R5, #CodebreakerName
        BL ReadInputStr
        MOV R2, #msgMaxAttempts
        BL ReadInputInt
        STR R5, attemptLimit
        MOV R2, #1
        STR R2, .Resolution
        BL SecretCode
        BL QueryCode
EndProgram:
        B End
QueryCode:
        PUSH {R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R0, R1, LR}
        MOV R2, #1        // index counter
        MOV R3, #code
        MOV R4, #attemptLimit
        LDR R4, [R4]
        ADD R4, R4, #1
        MOV R5, #querycode
        MOV R0, #0
LoopGame:
        CMP R2, R4
        MOV R6, #CodebreakerName
        STR R6, .WriteString
        MOV R6, #msgCurrentAttempt
        STR R6, .WriteString
        STR R2, .WriteSignedNum
        STRB R1, .WriteChar
        BL GetCode
        BL QueryValidation
        CMP R8, #4
        ADD R2, R2, #1
```

Processor

```
PC   0x00000218
LR   0x00000114
SP   0x000fffac
R12  0x00000000
R11  0x00000000
R10  0x00000000
R9   0x00000000
R8   0x00000000
R7   0x00000079
R6   0x0000050d
R5   0x00000268
R4   0x00000004
R3   0x000002ec
R2   0x000004aa
R1   0x0000000a
R0   0x00000000
```

Count 662

Current Instruction

Status bits NZCV 1000

Input/Output

```
Please input a 4 characters secret code:
rgby
a, this is guess number: 2
Please input a 4 characters secret code:
```

Input expected

Memory

```
000         0x0         0x4         0x8         0xc
0x0000  0xe3a0100a  0xebffffff  0xe3a02e47  0xe3a05e37
0x0001  0xeb00007d  0xe300247e  0xe3a05e3f  0xeb00007a
0x0002  0xe300248e  0xeb00007f  0xe58f533c  0xe3a02001
0x0003  0xe50f20fc  0xeb00002a  0xeb000000  0xea000088
0x0004  0xe92d5fff  0xe3a02001  0xe3a03fbb  0xe3a04fdb
0x0005  0xe5944000  0xe2844001  0xe3a05f9a  0xe3a00000
0x0006  0xe1520004  0xe3a06e3f  0xe50f6150  0xe300650d
0x0007  0xe50f6158  0xe50f216c  0xe54f1164  0xeb000020
0x0008  0xeb000005  0xe3580004  0xe2822001  0xe1520004
0x0009  0xbafffff2  0xe8bd5fff  0xe1a0f00e  0xe92d407c
0x000a  0xe3a02000  0xe3a03fba  0xe5933000  0xe3a04fbb
0x000b  0xe3a05f9a  0xe3a08000  0xe3a09000  0xe1520003
0x000c  0x0a000005  0xe7d46002  0xe7d57002  0xe1560007
0x000d  0xe2822001  0xe3520000  0x1afffff7  0xe8bd407c
0x000e  0xe1a0f00e  0xe92d4424  0xe3a05fbb  0xeb000053
0x000f  0xeb00000c  0xe35a0001  0x1afffffa  0xe8bd4424
0x0010  0xe1a0f00e  0xe92d4425  0xe3a05f9a  0xe30024aa
0x0011  0xeb00003d  0xeb000003  0xe35a0001  0x1afffff9
0x0012  0xe8bd4425  0xe1a0f00e  0xe92d405c  0xe3a03000
0x0013  0xe3a04fba  0xe5944000  0xe3a00000  0xe1530004
0x0014  0xca00002a  0xe7d52003  0xe3520000  0x0a000008
0x0015  0xeb000015  0xe3500000  0x1a000008  0xeb000009
0x0016  0xe3500000  0x1a000005  0xe2833001  0xe3520000
0x0017  0x1afffff1  0xe3a0a001  0xe8bd405c  0xe1a0f00e
0x0018  0xe8bd405c  0xe1a0f00e  0xe1a08003  0xe2888001
0x0019  0xe7d56008  0xe1560002  0x0a000014  0xe2888001
0x001a  0xe3560000  0x1afffff9  0xe1a0f00e  0xe92d4078
0x001b  0xe300352c  0xe3a04000  0xe3a05018  0xe1540005
0x001c  0x0a00000a  0xe3500000  0x1a000006  0xe7936004
0x001d  0xe2844004  0xe1560002  0x0a000002  0xe1540005
0x001e  0xbafffff5  0xeb000001  0xe8bd4078  0xe1a0f00e
0x001f  0xe92d4004  0xe30024d4  0xe50f22e0  0xe54f12e8
```

Hex

Clear

ARMLite Simulator V1.2a © Peter Higginson 2020-22

Documentation

At this stage, I have added 3 functions such as QueryCode, QueryValidation and GetCode

Functions:

- QueryCode: This function has the responsible to compare the number of guess with how maximum guess that it has. And will loop until both of the index and maximum guesses are the same. Furthermore this also uses QueryValidation to make sure they are matching the requirement from its input from GetCode. (Max length input = 4 numbers, and can only enter r,g,b,y,p,c)
- QueryValidation: This function will double check whether it is matching the valid length (4) and the validate the character whether they are valid color.
- GetCode: This function will mostlikely getting the secret codes from the users input but will also double check double check whether it is matching the valid length (4) and the validate the character whether they are valid color by using the function CodeValidation.

All the functions have worked as required in the instruction and I haven’t yet found any issues at this stage as it prints out like the requirement.

Stage 5a:

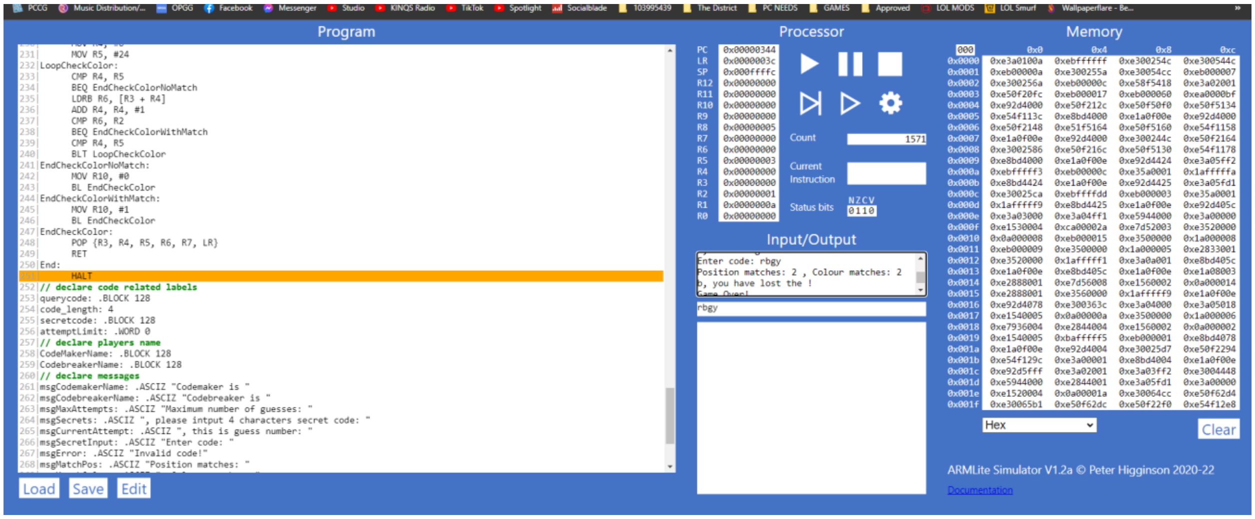
At this stage, I have added a function called CheckColor and rename QueryCode => CompareCodes (this is required by the instruction)

- CompareCodes still works like QueryCode but having more QueryValidation to make sure:
  - Case 1: the query peg is a direct match with its corresponding peg in the secret code

- Case 2: the query peg is a match with at least one other peg in the secret code but not in the correct position
- Case 3: the peg has no match in the secret code

These cases will uses a function call CheckColor to see whether they match or not match. If they matched, Case 1 will generated a count of 4 and Case 2 has a count of 0

Stage 5b:



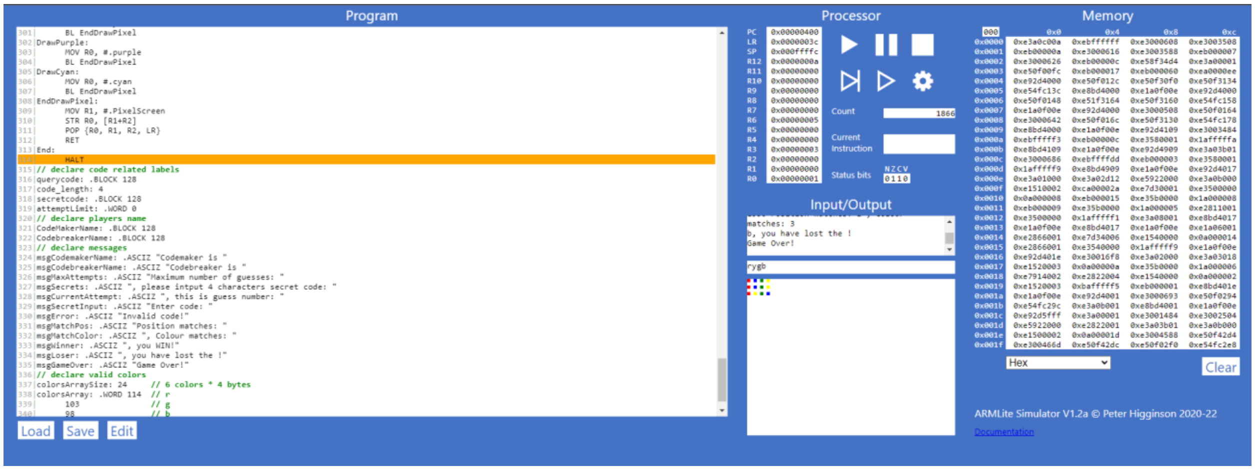
```
        MOV R5, #24
LoopCheckColor:
        CMP R4, R5
        BEQ EndCheckColorNoMatch
        LDRB R6, [R3 + R4]
        ADD R4, R4, #1
        CMP R6, R2
        BEQ EndCheckColorWithMatch
        CMP R4, R5
        BLT LoopCheckColor
EndCheckColorNoMatch:
        MOV R10, #0
        BL EndCheckColor
EndCheckColorWithMatch:
        MOV R10, #1
        BL EndCheckColor
EndCheckColor:
        POP {R3, R4, R5, R6, R7, LR}
        RET
End:
        HALT
// declare code related labels
querycode: .BLOCK 128
code_length: 4
secretcode: .BLOCK 128
attemptLimit: .WORD 0
// declare players name
CodeMakerName: .BLOCK 128
CodebreakerName: .BLOCK 128
// declare messages
msgCodemakerName: .ASCIZ "Codemaker is "
msgCodebreakerName: .ASCIZ "Codebreaker is "
msgMaxAttempts: .ASCIZ "Maximum number of guesses: "
msgSecrets: .ASCIZ ", please intput 4 characters secret code: "
msgCurrentAttempt: .ASCIZ ", this is guess number: "
msgSecretInput: .ASCIZ "Enter code: "
msgError: .ASCIZ "Invalid code!"
msgMatchPos: .ASCIZ "Position matches: "
```

```
Enter code: rbgy
Position matches: 2 , Colour matches: 2
b, you have lost the !
Game Over!
```

```
rbgy
```

In this stage, I just added a Win, Lose, and GameOver function as well as printing out the Position matches: <X>, Colour matches: <Y> and this seems to worked as required in the instruction.

Stage 6:



```
        BL EndDrawPixel
DrawPurple:
        MOV R0, #.purple
        BL EndDrawPixel
DrawCyan:
        MOV R0, #.cyan
        BL EndDrawPixel
EndDrawPixel:
        MOV R1, #.PixelScreen
        STR R0, [R1+R2]
        POP {R0, R1, R2, LR}
        RET
End:
        HALT
// declare code related labels
querycode: .BLOCK 128
code_length: 4
secretcode: .BLOCK 128
attemptLimit: .WORD 0
// declare players name
CodeMakerName: .BLOCK 128
CodebreakerName: .BLOCK 128
// declare messages
msgCodemakerName: .ASCIZ "Codemaker is "
msgCodebreakerName: .ASCIZ "Codebreaker is "
msgMaxAttempts: .ASCIZ "Maximum number of guesses: "
msgSecrets: .ASCIZ ", please intput 4 characters secret code: "
msgCurrentAttempt: .ASCIZ ", this is guess number: "
msgSecretInput: .ASCIZ "Enter code: "
msgError: .ASCIZ "Invalid code!"
msgMatchPos: .ASCIZ "Position matches: "
msgMatchColor: .ASCIZ ", Colour matches: "
msgWinner: .ASCIZ ", you WIN!"
msgLoser: .ASCIZ ", you have lost the !"
msgGameOver: .ASCIZ "Game Over!"
// declare valid colors
colorsArraySize: 24     // 6 colors * 4 bytes
colorsArray: .WORD 114  // r
        103             // g
        98              // b
```

```
matches: 3
b, you have lost the !
Game Over!
```

```
rygb
```

In this stage, I added a function call DrawCode and DrawPixel

Functions:

- DrawCode: This function will loop until each character in querycode is finished
- DrawPixel: This function has a job to compare then converted each character in querycode to matches its color array then start drawing the color itself.

Problems:

The program seems to works as it is required in the instruction, no issues has been found but I still believe they’re still exist as coding never been perfect in real life therefore there’s a possible chance that bugs can still shows up.

Assumptions:

I have made an effort to construct the program during this project without making any assumptions. The preceding courses covered all of the rationale behind this