Summary of work

I separated the operations for the program into these few steps,

1. Initialize the address for S0
2. Loads S0
3. Initialize the address for S1
4. Loads S1
5. Create a copy of S0
6. Compare S0 and S1
7. Prints out “Y” or “N”
8. Checks if S0 needed to be reinitialized.

Step 1 is completed with the branch INIS0.
R12 is added as a counter for step 5.

Step 2 is completed with the branch INS0.
Each character is stored in different memory incrementally.
Counter (R12) would increment every loop.
INS0 compares the character inputted with 0x0D, which represents carriage return in ASCII.
When the character inputted is equal to 0x0D, the loop would then exited.

Step 3 is completed with the branch INIS1.

Step 4 is completed with the branch INS1.
INS1 operates in the same way as INS0, with the only difference being the absence of a counter.

Step 5 is completed with the branch COPYS0 and COPYS0L.
COPYS0 initialize the address for which the copy of S0 is stored.
S0 would be copied from the first character.
When the address of the copied character is equal to the address of the last character, the loop is then exited.

Name: YI XIANG TAN
Student ID:18315028

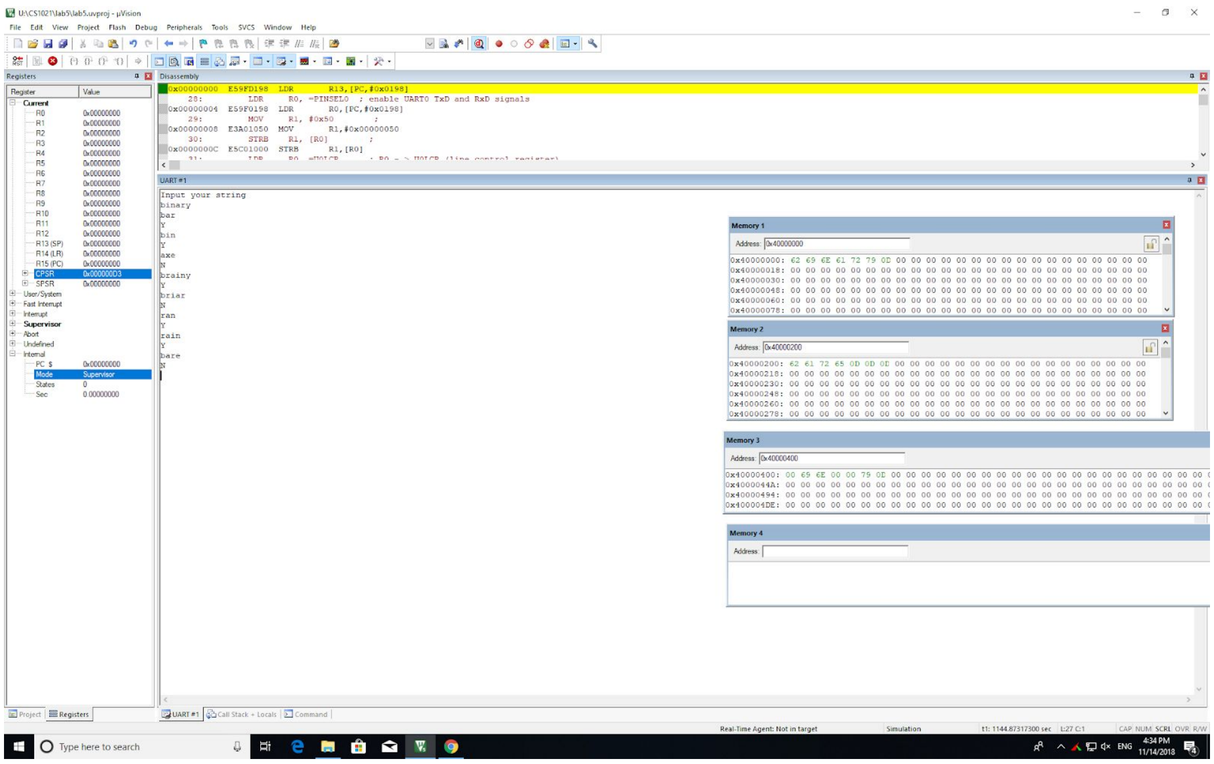
Step 6 is completed with the branch COMPAREL and NEXTC.
R0 will load the first character of S1, while R8 will load the first character of S0.
If the character in R0 is not equal to the character in R1, R8 will then load the next character of S0.
When the characters are equal, the character in memory will be replaced with 0x0.
The loop will exit when either S1 or S0 hits 0x0D.
If S1 hits 0x0D, the loop would branch to the branch that leads to a return of “Y”.
If S0 hits 0x0D, the loop would branch to the branch that leads to a return of “N”.

Step 7 is completed with the branch YES,NO and PRINT.
Branch YES and NO were branched by the branch COMPAREL.
They would represent the reply of “Y” and “N” respectively.
The declared string of “Y” and “N” would be loaded respectively into R0.
They would then branch to the branch PRINT which would print the loaded string to UART.

Step 8 is completed with the branch ISS1EMPTY.
The branch compares the inputted character with 0x0D.
(Enter would be entered if the user would like to reinitialize S0)
If the character is equal to 0x0D, the branch would branch to the branch INIS0 which would reinitialize S0.
If the character is not equal to 0x0D, the branch would branch to the branch INS1, and S0 would be reinitialized.

CS1021 Lab 5 Report

Proof of work



U:\CS1021\lab5\lab5.uvproj - µVision

Registers

Register	Value
Current	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000000
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000000
Mode	Supervisor
States	0
Sec	0.00000000

Disassembly

```
0x00000000  E59FD198  LDR       R13,[PC,#0x0198]
    28:         LDR    R0, =PINSEL0  ; enable UART0 TxD and RxD signals
0x00000004  E59F0198  LDR       R0,[PC,#0x0198]
    29:         MOV    R1, #0x50          ;
0x00000008  E3A01050  MOV       R1,#0x00000050
    30:         STRB   R1, [R0]           ;
0x0000000C  E5C01000  STRB      R1,[R0]
```

UART #1

```
Input your string
binary
bar
Y
bin
Y
axe
N
brainy
Y
briar
N
ran
Y
rain
Y
bare
N
```

Memory 1 — Address: 0x40000000

```
0x40000000: 62 69 6E 61 72 79 0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000018: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000048: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000078: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Memory 2 — Address: 0x40000200

```
0x40000200: 62 61 72 65 0D 0D 0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000218: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000230: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000248: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000260: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000278: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Memory 3 — Address: 0x40000400

```
0x40000400: 00 69 6E 00 00 79 0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000044A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000494: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400004DE: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Memory 4 — Address:

Real-Time Agent: Not in target   Simulation   t1: 1144.87317300 sec   L:27 C:1

Possible approach to extra mile

-Uppercase and lowercase

Checks the character inputted.

Convert all into uppercase.

We could see that a-z has ASCII value of 97-122, where every uppercase and lowercase character has a difference of 32.

(E.g a-97, A-65)

```
If (character > 97 and character < 122)
        Character -= 32
```

-Spaces containing string

When the character inputted is equal to ASCII for space, do not store it.

Name: YI XIANG TAN

Student ID:18315028