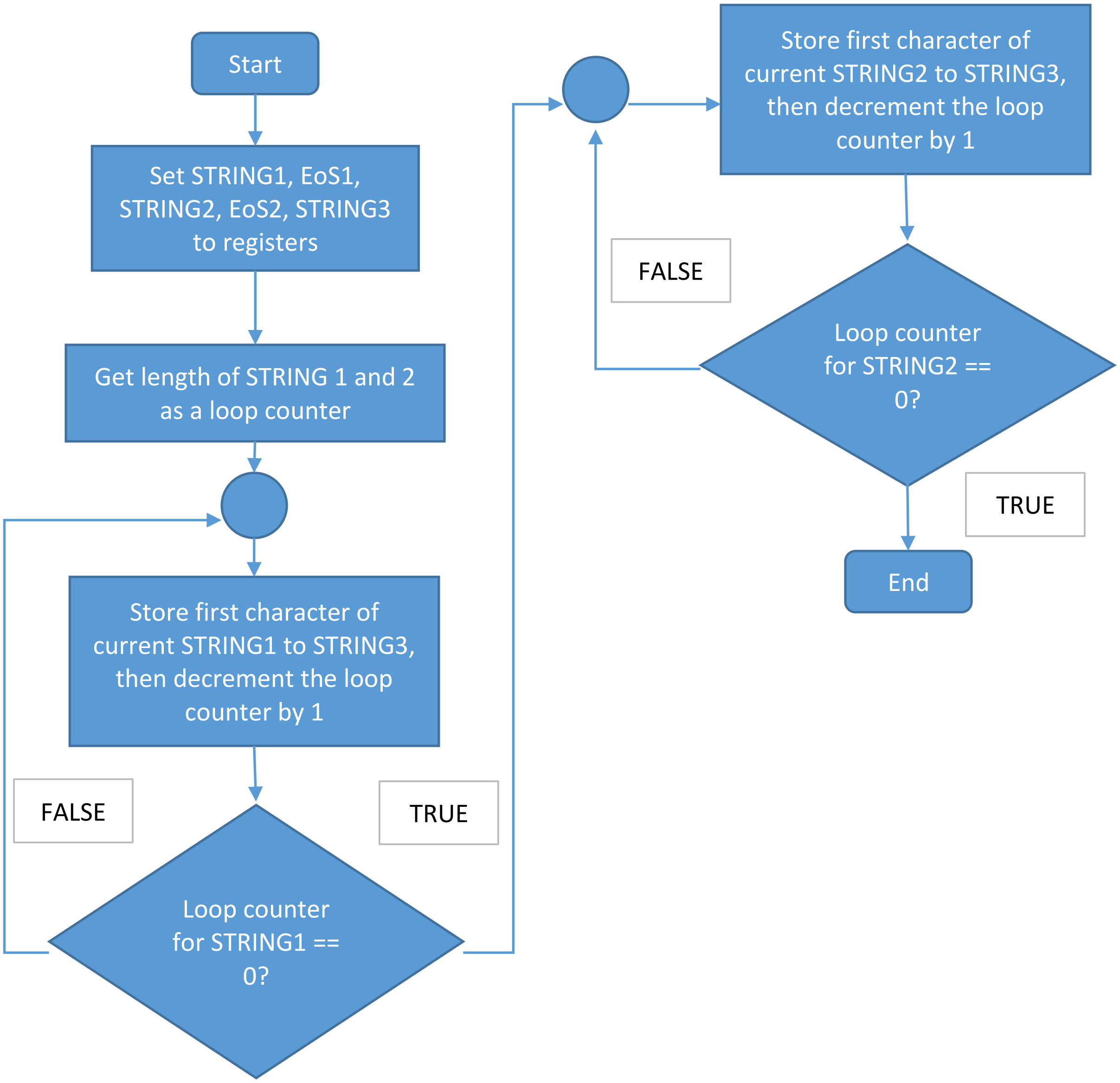
Assignment 4 Question 1 Flow Chart


Start
Set STRING1, EoS1, STRING2, EoS2, STRING3 to registers
Get length of STRING 1 and 2 as a loop counter
Store first character of current STRING1 to STRING3, then decrement the loop counter by 1
FALSE
TRUE
Loop counter for STRING1 == 0?
Store first character of current STRING2 to STRING3, then decrement the loop counter by 1
FALSE
Loop counter for STRING2 == 0?
TRUE
End

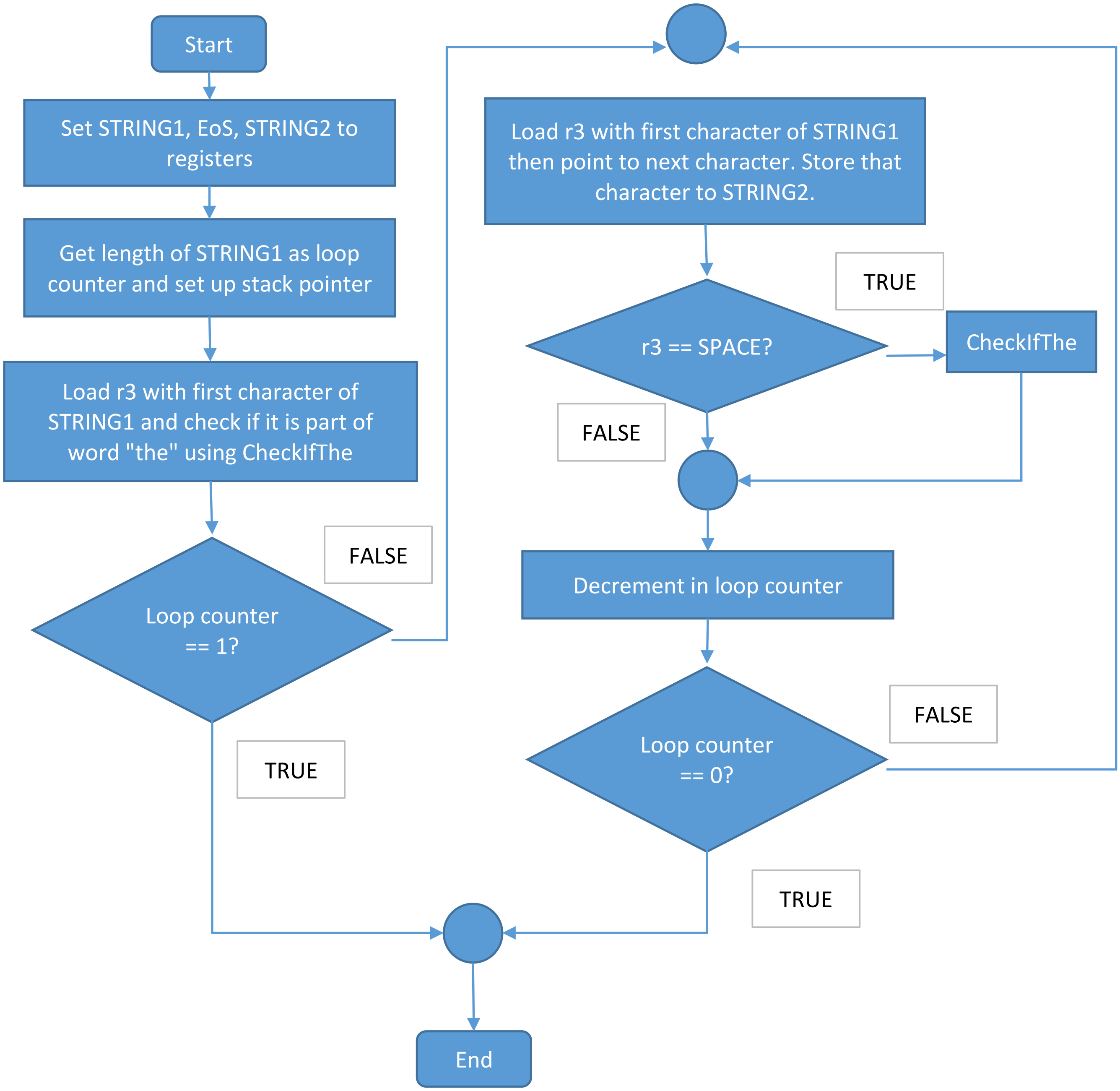
Assignment 4 Question 1 Code

```
                AREA Concatenate, CODE, READONLY
                ENTRY

                ADR r0, STRING1         ; Set r0 to point to STRING1
                ADR r1, EoS1            ; Set r1 to point to EoS1
                ADR r2, STRING2         ; Set r2 to point to STRING2
                ADR r3, EoS2            ; Set r3 to point to EoS2
                ADR r4, STRING3         ; Set r4 to point to STRING3, which will store the
                                        ; result
                SUB r1, r0              ; Get the length of STRING1, which will be used
                                        ; as loop counter for ConcatFirst
                SUB r3, r2              ; Get the length of STRING2, which will be used
                                        ; as loop counter for ConcatSecond
                ADD r3, #1              ; Add 1 to r3 to handle the null termination

ConcatFirst     LDRB r5, [r0], #1       ; Load r5 with the first ASCII character of
                                        ; current STRING1, then point to next character
                STRB r5, [r4], #1       ; Store r5 into STRING3, then point to next
                                        ; location to store a character
                SUBS r1, #1             ; Decrement the loop counter
                BNE ConcatFirst         ; Loop again, continue until r1 equals zero

ConcatSecond    LDRB r5, [r2], #1       ; Load r5 with the first ASCII character of
                                        ; current STRING2, then point to next character
                STRB r5, [r4], #1       ; Store r5 into STRING3, then point to next
                                        ; location to store the character
                SUBS r3, #1             ; Decrement the loop counter
                BNE ConcatSecond        ; Loop again, continue until r1 equals zero
```

```
EndProg         B EndProg                   ; Infinite loop


STRING1         DCB "This is a test string1"    ; String1
EoS1            DCB 0x00                    ; End of string1
STRING2         DCB "This is a test string2"    ; String2
EoS2            DCB 0x00                    ; End of string2
STRING3         space 0xFF                  ; String3, where the results are stored in


                END
```

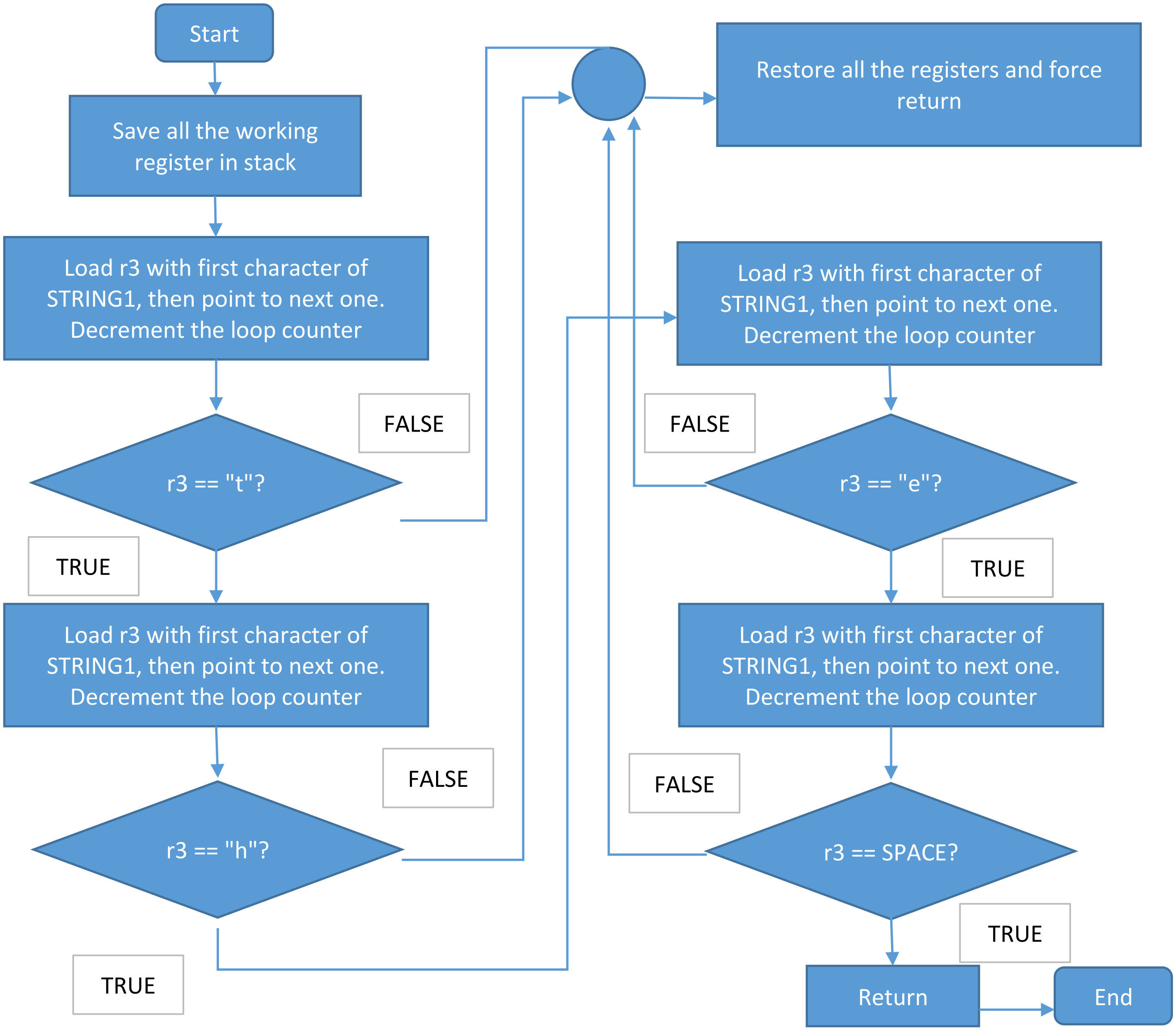
Assignment 4 Question 2 Flow Chart

For Main


Start
Set STRING1, EoS, STRING2 to registers
Get length of STRING1 as loop counter and set up stack pointer
Load r3 with first character of STRING1 and check if it is part of word "the" using CheckIfThe
Loop counter == 1?
FALSE
TRUE
Load r3 with first character of STRING1 then point to next character. Store that character to STRING2.
r3 == SPACE?
TRUE
CheckIfThe
FALSE
Decrement in loop counter
Loop counter == 0?
FALSE
TRUE
End

Assignment 4 Question 2 Flow Chart (Continue…)

For CheckIfThe

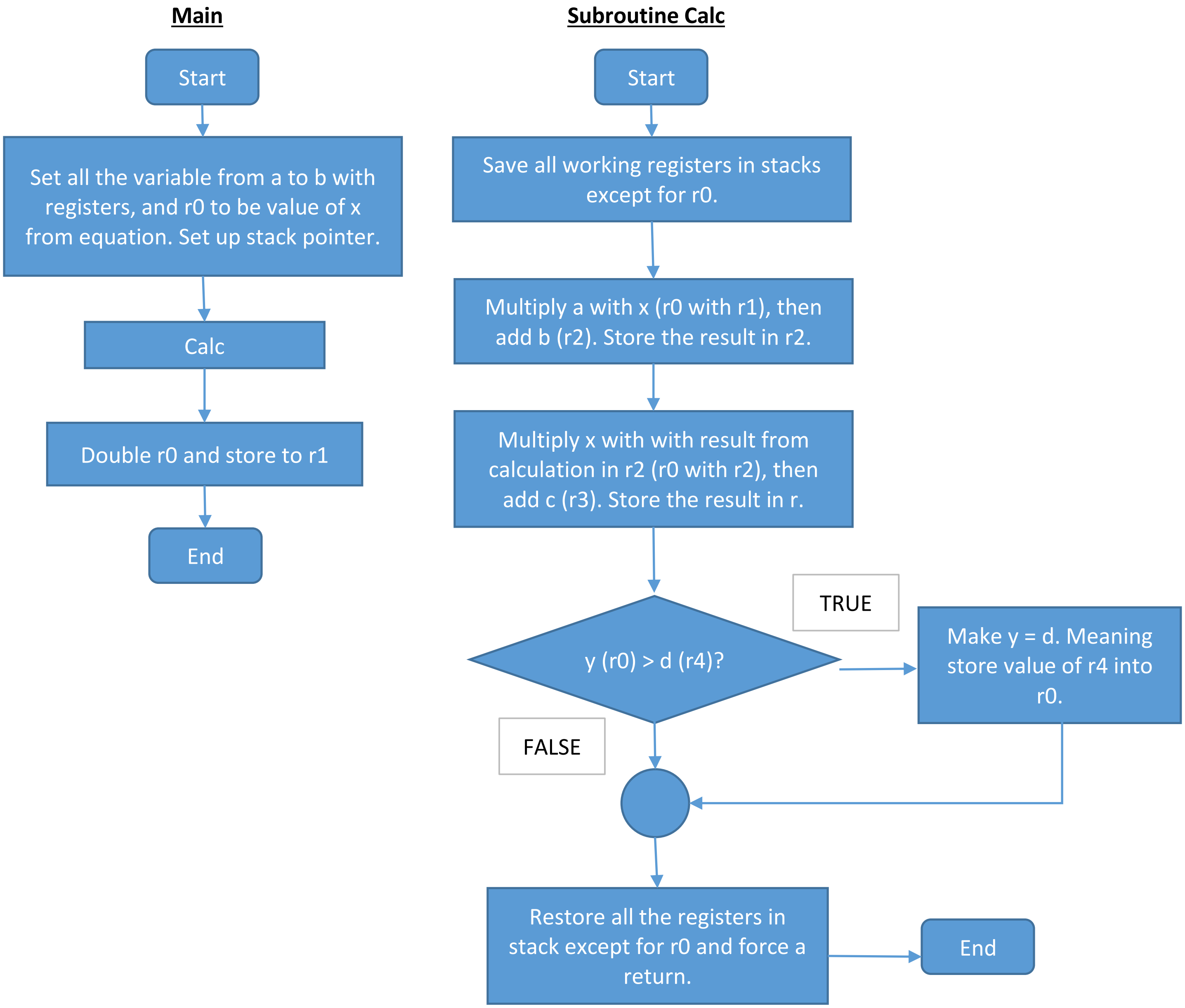

Start
Save all the working register in stack
Load r3 with first character of STRING1, then point to next one. Decrement the loop counter
r3 == "t"?
FALSE
TRUE
Load r3 with first character of STRING1, then point to next one. Decrement the loop counter
r3 == "h"?
FALSE
TRUE
Restore all the registers and force return
Load r3 with first character of STRING1, then point to next one. Decrement the loop counter
r3 == "e"?
FALSE
TRUE
Load r3 with first character of STRING1, then point to next one. Decrement the loop counter
r3 == SPACE?
FALSE
TRUE
Return
End

Assignment 4 Question 2 Code

```
            AREA Replace_THE, CODE, READONLY
            ENTRY

            ADR r0, STRING1                 ; Set r0 to point to STRING1
            ADR r1, EoS                     ; Set r1 to point to EoS
            ADR r2, STRING2                 ; Set r2 to point to STRING2
            SUB r1, r0                      ; Get the length of STRING1, which will be used
                                            ; as loop  counter for the index of STRING1
            ADD r1, #1                      ; Add 1 to r1 to handle the null termination
            ADR sp, stack                   ; Set up the stack pointer

CheckFirst  LDRB r3, [r0]                   ; Load r3 with the first ASCII character of
                                            ; current STRING1
            BL CheckIfThe                   ; Jump to subroutine CheckIfThe, where it
                                            ; checks if r3 is part of the character "the"
            CMP r1, #1                      ; Performs test to see if r1 is 1 or not
            STRBEQ r3, [r2], #1             ; IF TRUE, then store r3 into STRING2, then
                                            ; point to next location to store a character
            BEQ ENDPROG                     ; IF TRUE, then end program

CheckString LDRB r3, [r0], #1               ; Load r3 with the first ASCII character of
                                            ; current STRING1, then point to next character
            STRB r3, [r2], #1               ; Store r3 into STRING2, then point to next
                                            ; location to store a character
            CMP r3, #0x20                   ; Performs test to see if r3 is equal to 0x20
                                            ; which is SPACE in ASCII character
            BLEQ CheckIfThe                 ; IF TRUE, then jump to subroutine CheckIfThe
            SUBS r1, #1                     ; Decrement the loop counter
            BNE CheckString                 ; Loop again, continue until r1 equals zero

ENDPROG     B ENDPROG                       ; Infinite loop

CheckIfThe  STMFD sp!, {r0, r1, r3, lr}     ; Save the working registers and link register at
stack pointer

            LDRB r3, [r0], #1               ; Load r3 with the first ASCII character of

                                            ; current STRING1, then point to next character

            SUBS r1, #1                     ; Decrement the loop counter

            CMP r3, #0x74                   ; Performs test to see if r3 is equal to 0x74

                                            ; which is "t" in ASCII character

            LDMFDNE sp!, {r0, r1, r3, pc}   ; IF FALSE, restore registers and force a return

            LDRB r3, [r0], #1               ; Load r3 with the first ASCII character of

                                            ; current STRING1, then point to next character

            SUBS r1, #1                     ; Decrement the loop counter

            CMP r3, #0x68                   ; Performs test to see if r3 is equal to 0x68

                                            ; which is "h" in ASCII character

            LDMFDNE sp!, {r0, r1, r3, pc}   ; IF FALSE, restore registers and force a return

            LDRB r3, [r0], #1               ; Load r3 with the first ASCII character of

                                            ; current STRING1, then point to next character

            SUBS r1, #1                     ; Decrement the loop counter

            CMP r3, #0x65                   ; Performs test to see if r3 is equal to 0x65

                                            ; which is "e" in ASCII character

            LDMFDNE sp!, {r0, r1, r3, pc}   ; IF FALSE, restore registers and force a return

            LDRB r3, [r0]                   ; Load r3 with the first ASCII character of

                                            ; current STRING1

            CMP r3, #0x20                   ; Performs test to see if r3 is equal to 0x20

                                            ; which is SPACE in ASCII character

            CMPNE r3, #0x00                 ; IF FALSE, Performs test to see if r3 is equal to
```

```
                                                    ; 0x00 which is null in ASCII character
                LDMFDNE sp!, {r0, r1, r3, pc}       ; IF FALSE, restore registers and force a return
                MOV pc, lr                          ; ELSE, copy linked register into PC and return it

STRING1         DCB "and the man said they must go" ; String1
EoS             DCB 0x00                            ; End of string1
STRING2         space 0xFF                          ; String3, where the results are stored in
                SPACE 0x40                          ; Reserved room for stack to grow
stack           DCD 0x0                             ; Base of the stack

                END
```

Assignment 4 Question 3 Flow Chart



Main

Subroutine Calc

Start
Set all the variable from a to b with registers, and r0 to be value of x from equation. Set up stack pointer.
Calc
Double r0 and store to r1
End
Start
Save all working registers in stacks except for r0.
Multiply a with x (r0 with r1), then add b (r2). Store the result in r2.
Multiply x with with result from calculation in r2 (r0 with r2), then add c (r3). Store the result in r.
y (r0) > d (r4)?
TRUE
Make y = d. Meaning store value of r4 into r0.
FALSE
Restore all the registers in stack except for r0 and force a return.
End

Assignment 4 Question 3 Code

```
                AREA Calculation, CODE, READONLY
                ENTRY

                MOV r0, #2_00000011         ; Set r0 to be signed integer binary value, which will be x
                                            ; from equation y = a × x^2 + b × x + c. This will also be y
                LDR r1, Var_a               ; Set r1 to be variable a from equation
                                            ; y = a × x^2 + b × x + c
                LDR r2, Var_b               ; Set r2 to be variable b from equation
                                            ; y = a × x^2 + b × x + c
                LDR r3, Var_c               ; Set r3 to be variable c from equation
                                            ; y = a × x^2 + b × x + c
                LDR r4, Var_d               ; Set r4 to be variable d from equation
                                            ; y = a × x^2 + b × x + c
                ADR sp, stack               ; Set up stack pointer
                BL Calc                     ; Jump to Calc
                ADD r1, r0, r0              ; Double the value of r0 and store to r1

EndProg         B EndProg                   ; Infinite loop

Calc            STMFD sp!, {r1-r5, lr}      ; Save the working registers and link register at stack
                                            ; pointer
                MLA r2, r0, r1, r2          ; Multiply r0 and r1, and add it with r2. Store to value to
                                            ; r2. This will get the value (ax + b)
                MLA r0, r2, r0, r3          ; Multiply r2 and r0, and add it with r3. Store to value to
                                            ; r0. This will get the value x(ax + b) + c which will equal
                                            ; to a × x^2 + b × x + c
                CMP r0, r4                  ; Performs test to see if r0 is greater than r4
                MOVGT r0, r4                ; IF TRUE, then set r0 to be r4
                LDMFD sp!, {r1-r5, pc}          ; Restore registers and force a return

Var_a           DCD 5                           ; Variable a from equation y = a × x^2 + b × x + c
Var_b           DCD 6                           ; Variable b from equation y = a × x^2 + b × x + c
Var_c           DCD 7                           ; Variable c from equation y = a × x^2 + b × x + c
Var_d           DCD 50                          ; Variable d from equation y = a × x^2 + b × x + c
                SPACE 0x40                      ; Reserved room for stack to grow
stack           DCD 0x0                         ; Base of the stack

                END
```