PES UNIVERSITY

Department of Computer Science & Engineering

Microprocessor & Computer Architecture Lab

UE23CS251B

WEEK 5 submission

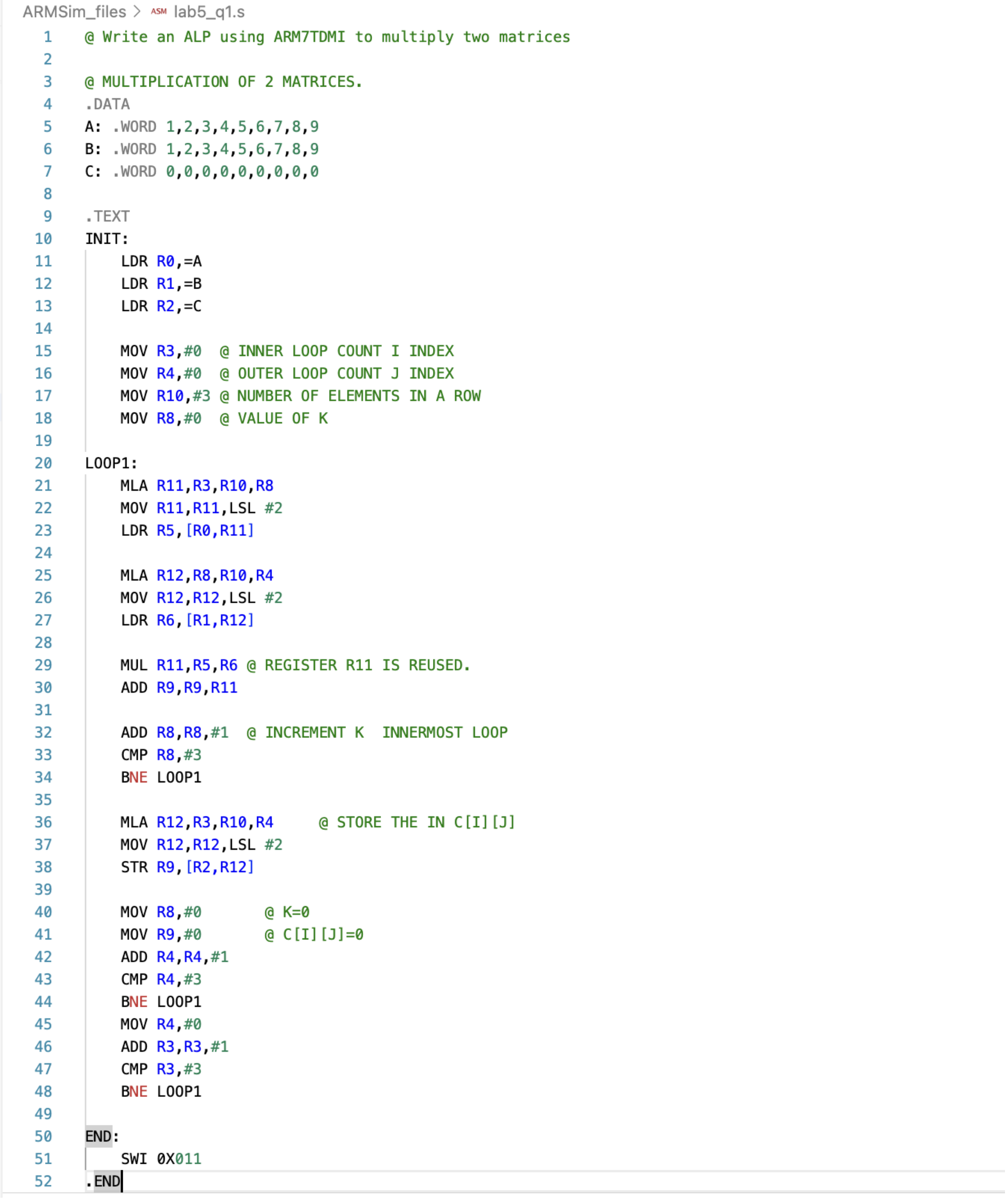
Name of the Student	Pranav Hemanth
SRN	PES1UG23CS433
Section	G
Department	CSE
Campus	RR

Department of Computer Science & Engineering
Microprocessor & Computer Architecture Lab

UE23CS251B

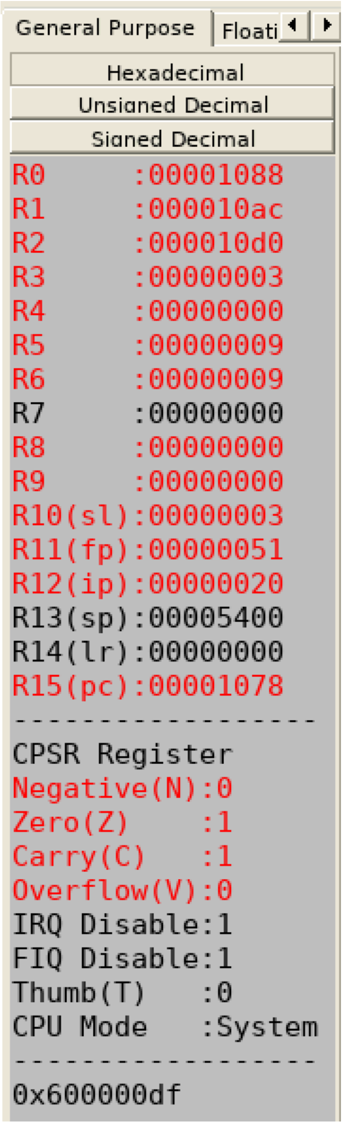
1 Write an ALP using ARM7TDMI to multiply two matrices.

Program screen shot:



```
@ Write an ALP using ARM7TDMI to multiply two matrices

@ MULTIPLICATION OF 2 MATRICES.
.DATA
A: .WORD 1,2,3,4,5,6,7,8,9
B: .WORD 1,2,3,4,5,6,7,8,9
C: .WORD 0,0,0,0,0,0,0,0,0

.TEXT
INIT:
    LDR R0,=A
    LDR R1,=B
    LDR R2,=C

    MOV R3,#0  @ INNER LOOP COUNT I INDEX
    MOV R4,#0  @ OUTER LOOP COUNT J INDEX
    MOV R10,#3 @ NUMBER OF ELEMENTS IN A ROW
    MOV R8,#0  @ VALUE OF K

LOOP1:
    MLA R11,R3,R10,R8
    MOV R11,R11,LSL #2
    LDR R5,[R0,R11]

    MLA R12,R8,R10,R4
    MOV R12,R12,LSL #2
    LDR R6,[R1,R12]

    MUL R11,R5,R6 @ REGISTER R11 IS REUSED.
    ADD R9,R9,R11

    ADD R8,R8,#1  @ INCREMENT K  INNERMOST LOOP
    CMP R8,#3
    BNE LOOP1

    MLA R12,R3,R10,R4     @ STORE THE IN C[I][J]
    MOV R12,R12,LSL #2
    STR R9,[R2,R12]

    MOV R8,#0       @ K=0
    MOV R9,#0       @ C[I][J]=0
    ADD R4,R4,#1
    CMP R4,#3
    BNE LOOP1
    MOV R4,#0
    ADD R3,R3,#1
    CMP R3,#3
    BNE LOOP1

END:
    SWI 0X011
.END
```

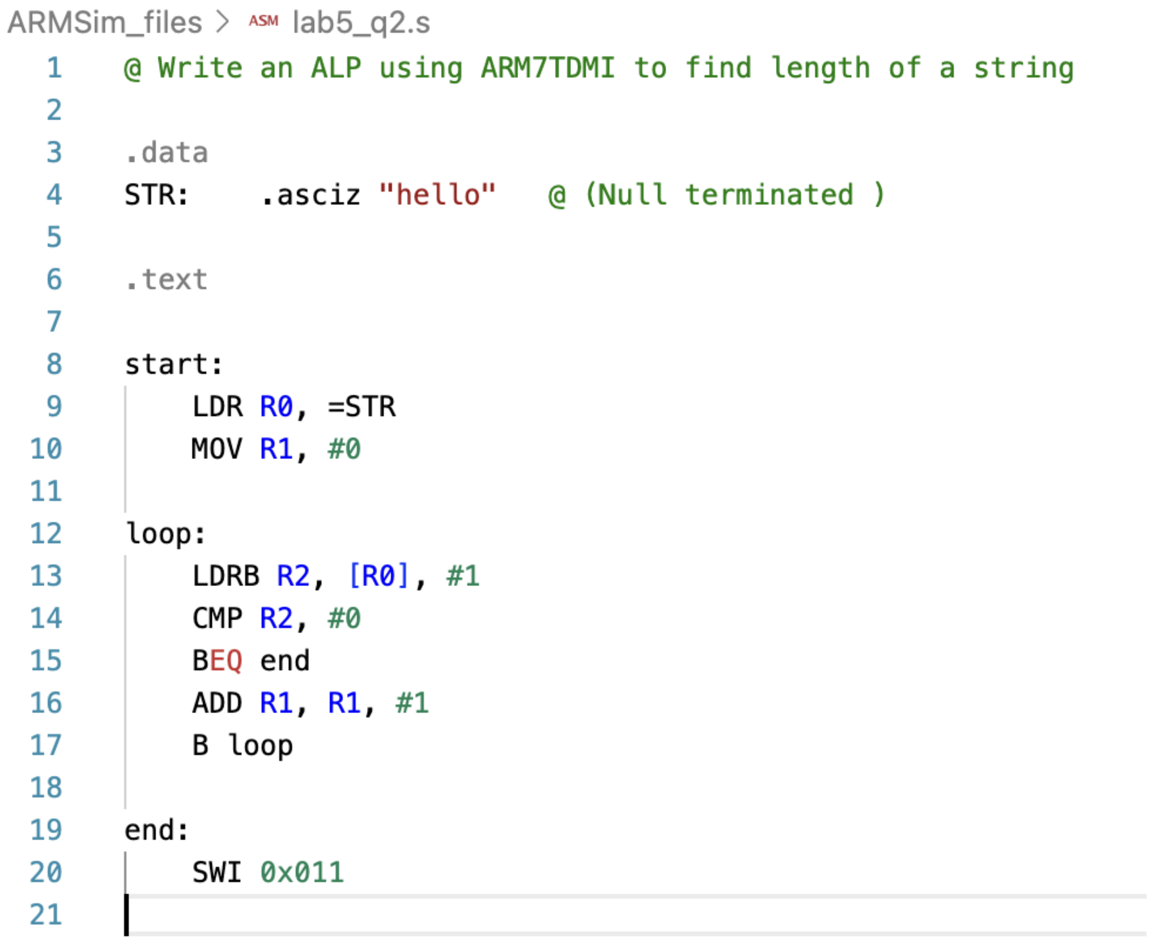
Screen shot of Register set output



```
R0       :00001088
R1       :000010ac
R2       :000010d0
R3       :00000003
R4       :00000000
R5       :00000009
R6       :00000009
R7       :00000000
R8       :00000000
R9       :00000000
R10(sl):00000003
R11(fp):00000051
R12(ip):00000020
R13(sp):00005400
R14(lr):00000000
R15(pc):00001078
------------------
CPSR Register
Negative(N):0
Zero(Z)    :1
Carry(C)   :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)   :0
CPU Mode   :System
------------------
0x600000df
```

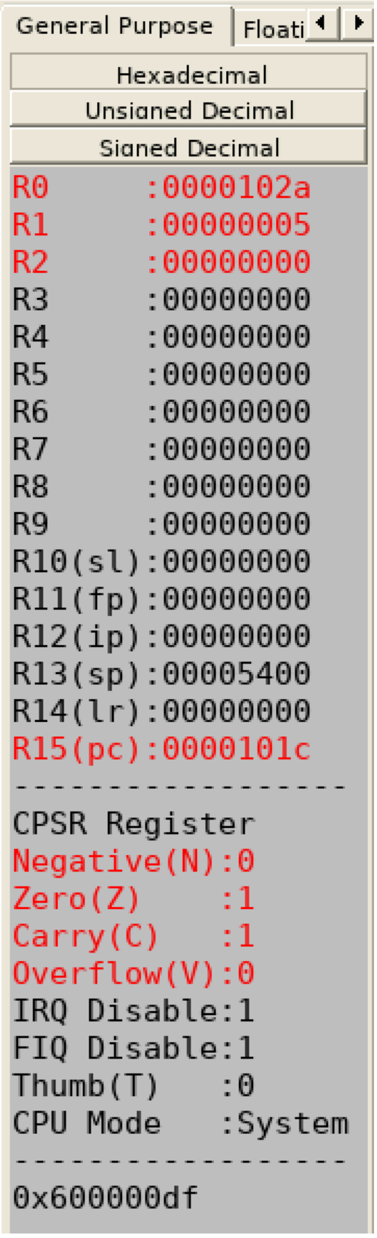
2 Write an ALP using ARM7TDMI to find length of a string

Program screen shot:

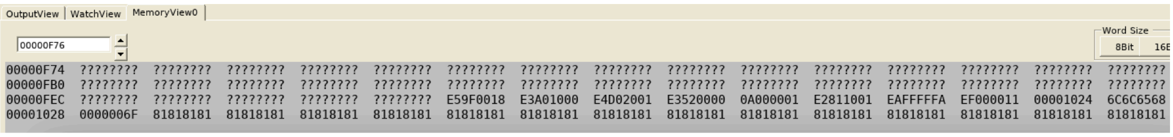


```
@ Write an ALP using ARM7TDMI to find length of a string

.data
STR:    .asciz "hello"   @ (Null terminated )

.text

start:
    LDR R0, =STR
    MOV R1, #0

loop:
    LDRB R2, [R0], #1
    CMP R2, #0
    BEQ end
    ADD R1, R1, #1
    B loop

end:
    SWI 0x011

```

Screen shot of Register set output and memory location:



```
R0       :0000102a
R1       :00000005
R2       :00000000
R3       :00000000
R4       :00000000
R5       :00000000
R6       :00000000
R7       :00000000
R8       :00000000
R9       :00000000
R10(sl):00000000
R11(fp):00000000
R12(ip):00000000
R13(sp):00005400
R14(lr):00000000
R15(pc):0000101c
------------------
CPSR Register
Negative(N):0
Zero(Z)    :1
Carry(C)   :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)   :0
CPU Mode   :System
------------------
0x600000df
```

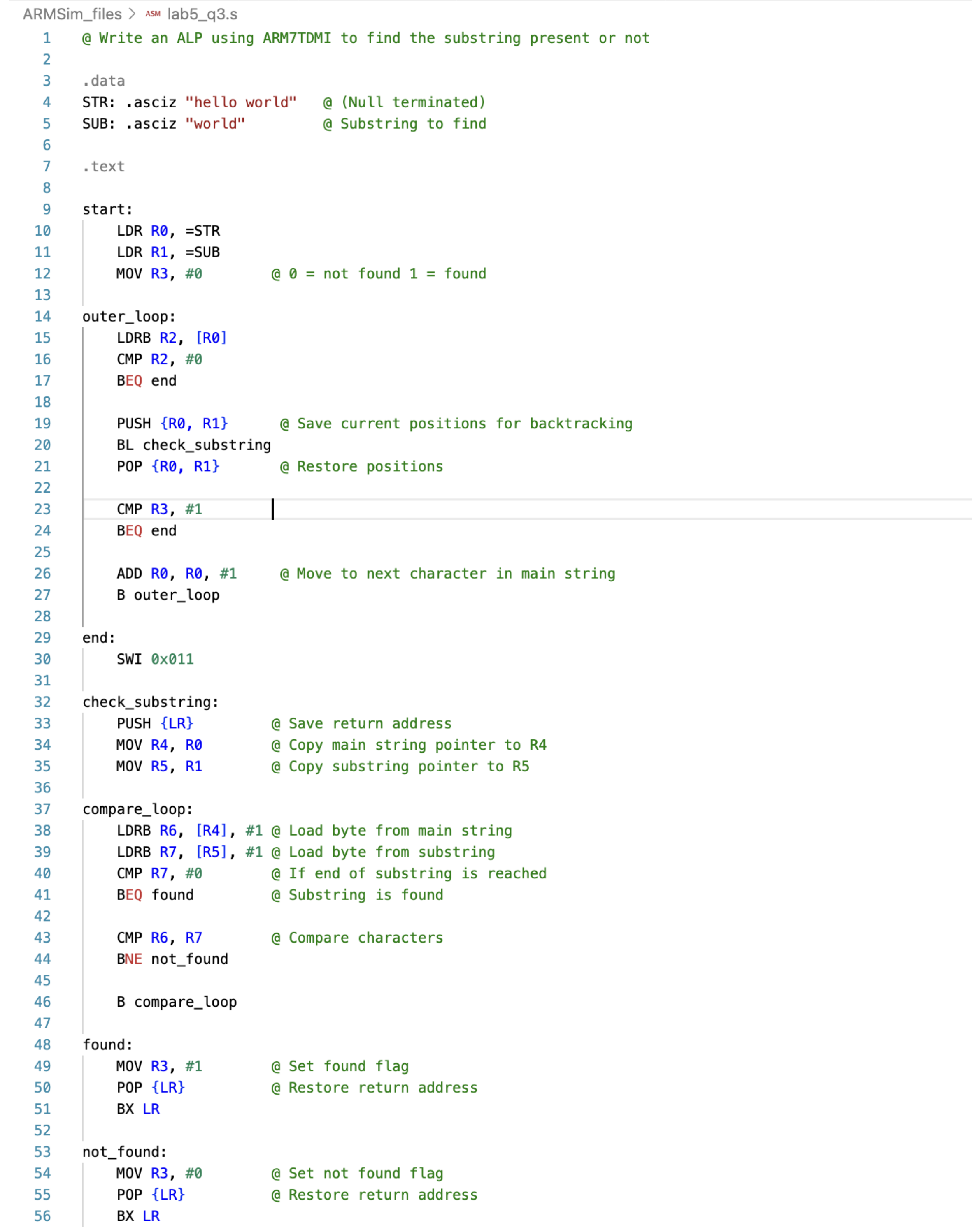


```
00000F76
00000F74  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????
00000FB0  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????  ????????
00000FEC  ????????  ????????  ????????  ????????  ????????  E59F0018  E3A01000  E4D02001  E3520000  0A000001  E2811001  EAFFFFFA  EF000011  00001024  6C6C6568
00001028  0000006F  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181
```

Assignments Questions

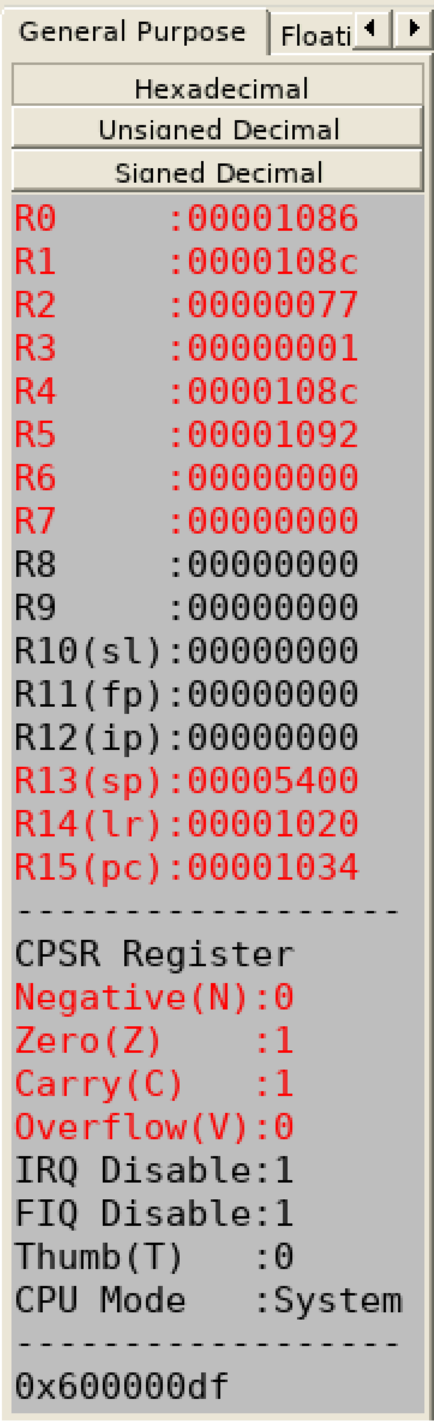
3 Write an ALP using ARM7TDMI to find the substring present or not

Program screen shot:

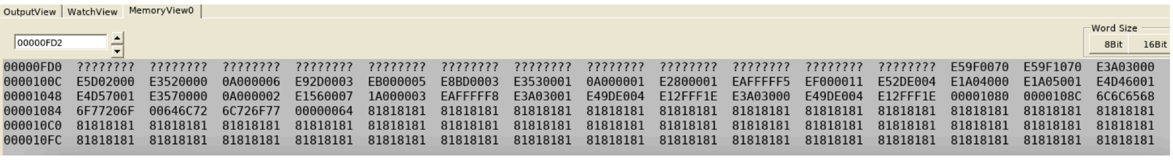


```
@ Write an ALP using ARM7TDMI to find the substring present or not

.data
STR: .asciz "hello world"   @ (Null terminated)
SUB: .asciz "world"         @ Substring to find

.text

start:
    LDR R0, =STR
    LDR R1, =SUB
    MOV R3, #0         @ 0 = not found 1 = found

outer_loop:
    LDRB R2, [R0]
    CMP R2, #0
    BEQ end

    PUSH {R0, R1}       @ Save current positions for backtracking
    BL check_substring
    POP {R0, R1}        @ Restore positions

    CMP R3, #1
    BEQ end

    ADD R0, R0, #1      @ Move to next character in main string
    B outer_loop

end:
    SWI 0x011

check_substring:
    PUSH {LR}          @ Save return address
    MOV R4, R0         @ Copy main string pointer to R4
    MOV R5, R1         @ Copy substring pointer to R5

compare_loop:
    LDRB R6, [R4], #1 @ Load byte from main string
    LDRB R7, [R5], #1 @ Load byte from substring
    CMP R7, #0         @ If end of substring is reached
    BEQ found          @ Substring is found

    CMP R6, R7         @ Compare characters
    BNE not_found

    B compare_loop

found:
    MOV R3, #1         @ Set found flag
    POP {LR}           @ Restore return address
    BX LR

not_found:
    MOV R3, #0         @ Set not found flag
    POP {LR}           @ Restore return address
    BX LR
```

Screen shot of Register set output and memory location:



```
R0        :00001086
R1        :0000108c
R2        :00000077
R3        :00000001
R4        :0000108c
R5        :00001092
R6        :00000000
R7        :00000000
R8        :00000000
R9        :00000000
R10(sl):00000000
R11(fp):00000000
R12(ip):00000000
R13(sp):00005400
R14(lr):00001020
R15(pc):00001034
------------------
CPSR Register
Negative(N):0
Zero(Z)    :1
Carry(C)   :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)   :0
CPU Mode   :System
------------------
0x600000df
```



00000FD2

```
00000FD0 ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? E59F0070 E59F1070 E3A03000
0000100C E5D02000 E3520000 0A000006 E92D0003 EB000005 E8BD0003 E3530001 0A000001 E2800001 EAFFFFF5 EF000011 E52DE004 E1A04000 E1A05001 E4D46001
00001048 E4D57001 E3570000 0A000002 E1560007 1A000003 EAFFFFF8 E3A03001 E49DE004 E12FFF1E E3A03000 E49DE004 E12FFF1E 00001080 0000108C 6C6C6568
00001084 6F77206F 00646C72 6C726F77 00000064 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
000010C0 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
000010FC 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
```

4

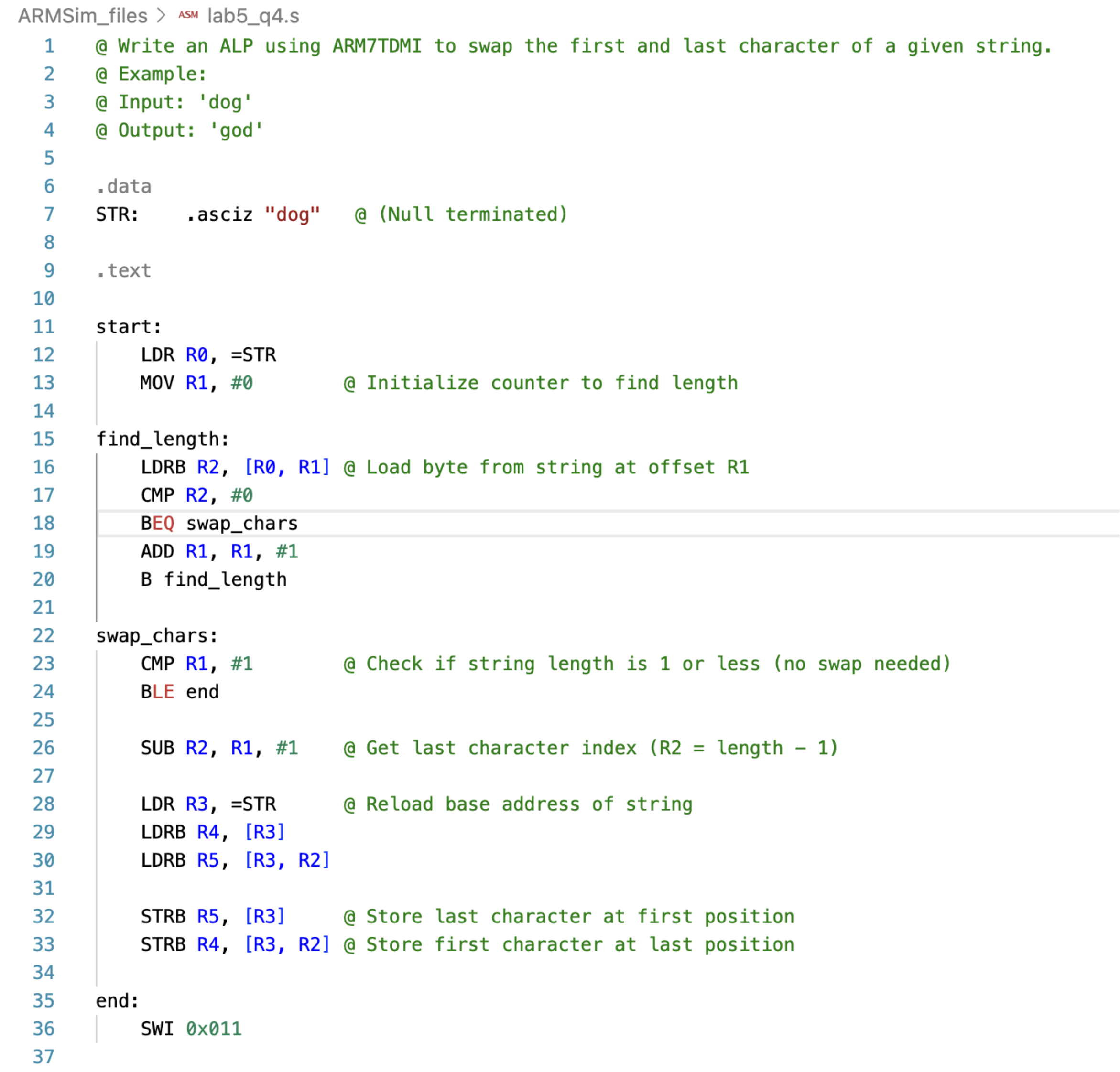
Write an ALP using ARM7TDMI to swap the first and last character of a given string.

Example:

Input: 'dog'

Output: 'god'

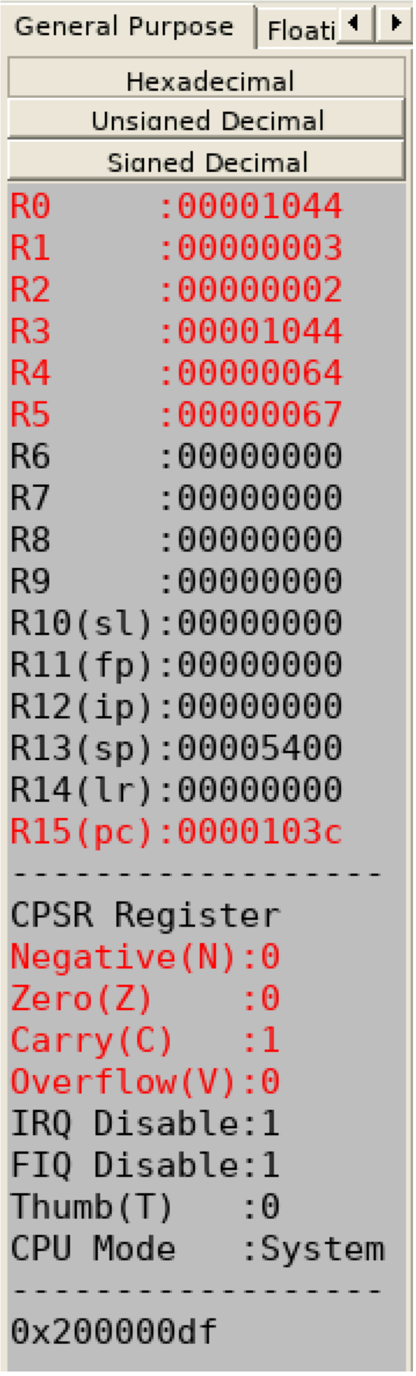
Program screen shot:



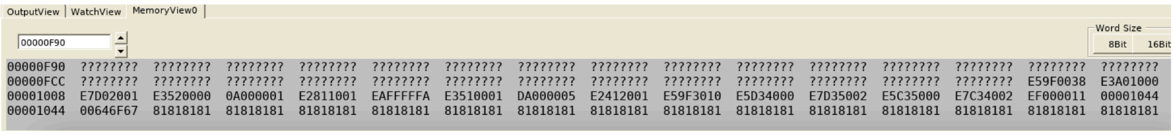
ARMSim_files > lab5_q4.s

```
@ Write an ALP using ARM7TDMI to swap the first and last character of a given string.
@ Example:
@ Input: 'dog'
@ Output: 'god'

.data
STR:    .asciz "dog"   @ (Null terminated)

.text

start:
    LDR R0, =STR
    MOV R1, #0         @ Initialize counter to find length

find_length:
    LDRB R2, [R0, R1] @ Load byte from string at offset R1
    CMP R2, #0
    BEQ swap_chars
    ADD R1, R1, #1
    B find_length

swap_chars:
    CMP R1, #1         @ Check if string length is 1 or less (no swap needed)
    BLE end

    SUB R2, R1, #1     @ Get last character index (R2 = length - 1)

    LDR R3, =STR       @ Reload base address of string
    LDRB R4, [R3]
    LDRB R5, [R3, R2]

    STRB R5, [R3]      @ Store last character at first position
    STRB R4, [R3, R2] @ Store first character at last position

end:
    SWI 0x011

```

Screen shot of Register set output and memory location:



```
R0       :00001044
R1       :00000003
R2       :00000002
R3       :00001044
R4       :00000064
R5       :00000067
R6       :00000000
R7       :00000000
R8       :00000000
R9       :00000000
R10(sl):00000000
R11(fp):00000000
R12(ip):00000000
R13(sp):00005400
R14(lr):00000000
R15(pc):0000103c
------------------
CPSR Register
Negative(N):0
Zero(Z)    :0
Carry(C)   :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)   :0
CPU Mode   :System
------------------
0x200000df
```



```
00000F90
00000F90 ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ????????
00000FCC ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? ???????? E59F0038 E3A01000
00001008 E7D02001 E3520000 0A000001 E2811001 EAFFFFFA E3510001 DA000005 E2412001 E59F3010 E5D34000 E7D35002 E5C35000 E7C34002 EF000011 00001044
00001044 00646F67 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
```