MPCA

ASSIGNMENT – 1

NAME: Nandan N **SEC:** F

SRN: PES1UG21CS361

A 1.1

Write a program in ARM7TDMI-ISA to search for an element in an array. Display appropriate messages on the standard output device. For Successful search display as “Successful Search” and if the search is unsuccessful, display as “Unsuccessful Search”. (Use Binary search Technique.)

CODE:

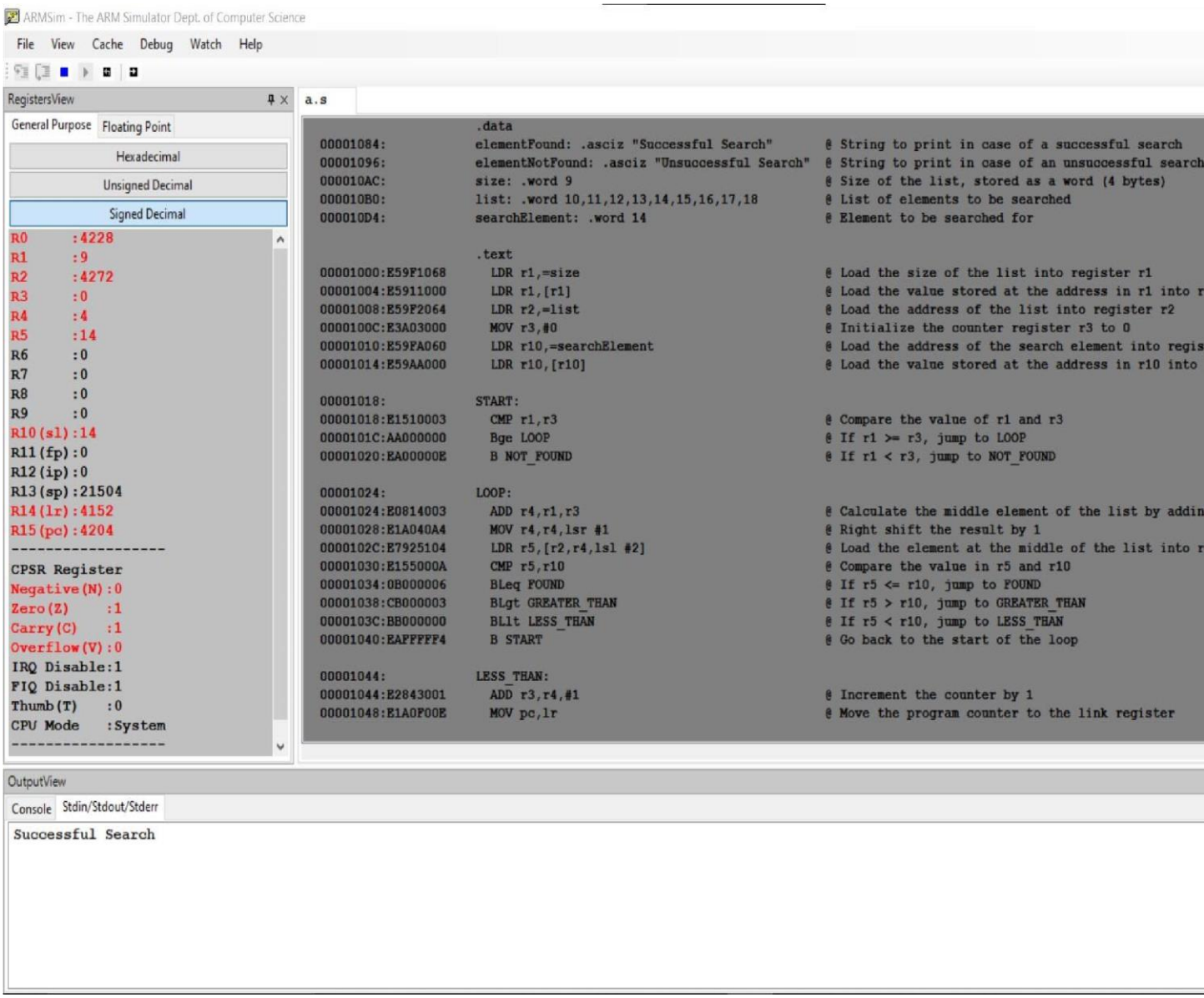
```
.data elementFound: .asciz "Successful Search"       @ String to print in case of a successful
search
elementNotFound: .asciz "Unsuccessful Search"  @ String to print in case of an unsuccessful
search
size: .word 9                          @ Size of the list, stored as a word (4 bytes)
list: .word 10,11,12,13,14,15,16,17,18        @ List of elements to be searched searchElement:
.word 14                  @ Element to be searched for

.text
  LDR r1,=size                    @ Load the size of the list into register r1
  LDR r1,[r1]                    @ Load the value stored at the address in r1 into r1
  LDR r2,=list                    @ Load the address of the list into register r2
  MOV r3,#0                     @ Initialize the counter register r3 to 0
```

```
  LDR r10,=searchElement                @ Load the address of the search element into register r10
  LDR r10,[r10]                         @ Load the value stored at the address in r10 into r10

START:
  CMP r1,r3                             @ Compare the value of r1 and r3
  Bge LOOP                              @ If r1 >= r3, jump to LOOP
  B NOT_FOUND                           @ If r1 < r3, jump to NOT_FOUND

LOOP:
  ADD r4,r1,r3                          @ Calculate the middle element of the list by adding r1 and r3
  MOV r4,r4,lsr #1                      @ Right shift the result by 1
  LDR r5,[r2,r4,lsl #2]                 @ Load the element at the middle of the list into register r5
  CMP r5,r10                            @ Compare the value in r5 and r10
  BLeq FOUND                            @ If r5 <= r10, jump to FOUND
  BLgt GREATER_THAN                     @ If r5 > r10, jump to GREATER_THAN
  BLlt LESS_THAN                        @ If r5 < r10, jump to LESS_THAN
  B START                               @ Go back to the start of the loop

LESS_THAN:
  ADD r3,r4,#1                          @ Increment the counter by 1
  MOV pc,lr                             @ Move the program counter to the link register

GREATER_THAN:                           @ Label for when the middle element is greater than the search element
  SUB r1,r4,#1                          @ Decrement the size of the list by 1
  MOV PC,LR                             @ Move the program counter to the link register
```

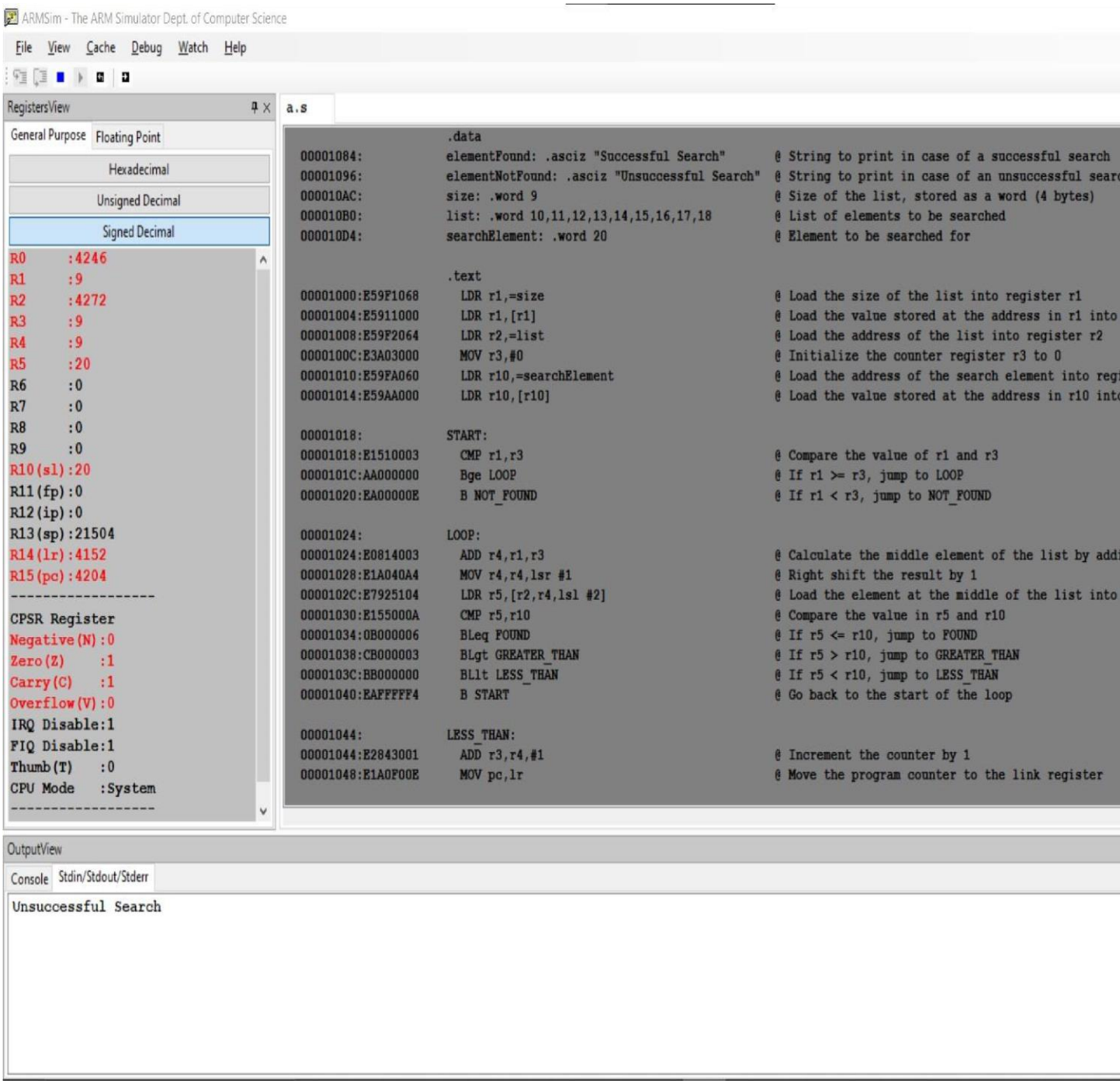
```
FOUND:
  LDR r0,=elementFound                    @ Load the address of the string "Successful Search" into
r0
  SWI 0x02                          @ Call the OS to display the string in r0
B EXIT

NOT_FOUND:
  LDR r0,=elementNotFound                    @ Load the address of the string "Unsuccessful Search"
into r0
  SWI 0x02                          @ Call the OS to display the string in r0
B EXIT

EXIT:
  SWI 0x11
```

OUTPUT SCREENSHOTS:

For successful search:



For Unsuccessful search:



A 1.2

Write a program in ARM7TDMI-ISA to find a sub string in a given main string.

Example 1: Main string: My name is Bond.

Character: ‘name’.

Expected Output: “String Present”

Example 2: Main string: My name is Bond.

Character: ‘James’.

Expected Output: “String Absent”

CODE:

```
.data exist: .asciz "String
Present." not_Exist: .asciz
"String Absent." string: .asciz
"My name is Bond" substr:
.asciz "James"

.text
  LDR r1,=exist              @ Load the address of the message "String Present." into register r1.
  LDR r2,=not_Exist          @ Load the address of the message "String Absent." into register r2.
  LDR r3,=string             @ Load the address of the string "My name is Bond" into register r3.
  LDR r4,=substr             @ Load the address of the string "possible" into register r4.
  LDRb r6,[r4],#1            @ Load the first byte of the substring into register r6 and increment the
address in r4.
```

LOOP:

LDRb r5,[r3],#1 @ Load the next byte of the string into register r5 and increment the address in r3.

CMP r5,r6 @ Compare the byte in r5 with the byte in r6.

STMFD R13!, {r3,r4} @ Store the values of r3 and r4 on the stack.

BLeq CHECK @ If the byte in r5 is less than or equal to the byte in r6, branch to the CHECK label.

CMP r5,#00 @ Compare the byte in r5 with the null terminator.

Beq NOT_FOUND @ If the byte in r5 is equal to the null terminator, branch to the NOT_FOUND label.

Bne LOOP @ If the byte in r5 is not equal to the byte in r6 or the null terminator, branch back to the LOOP label.

B EXIT @ Branch to the EXIT label.

CHECK:

LDMFD R13!, {r7,r8} @ Load the values of r3 and r4 from the stack.

WHILE:

LDRb r9,[r7],#1 @ Load the next byte of the string into register r9 and increment the address in r7.

LDRb r10,[r8],#1 @ Load the next byte of the substring into register r10 and increment the address in r8.

CMP r10,#00 @ Compare the byte in r10 with the null terminator.

Beq FOUND @ If the byte in r10 is equal to the null terminator, branch to the FOUND label.

CMP r10,r9 @ Compare the bytes in r10 and r9.

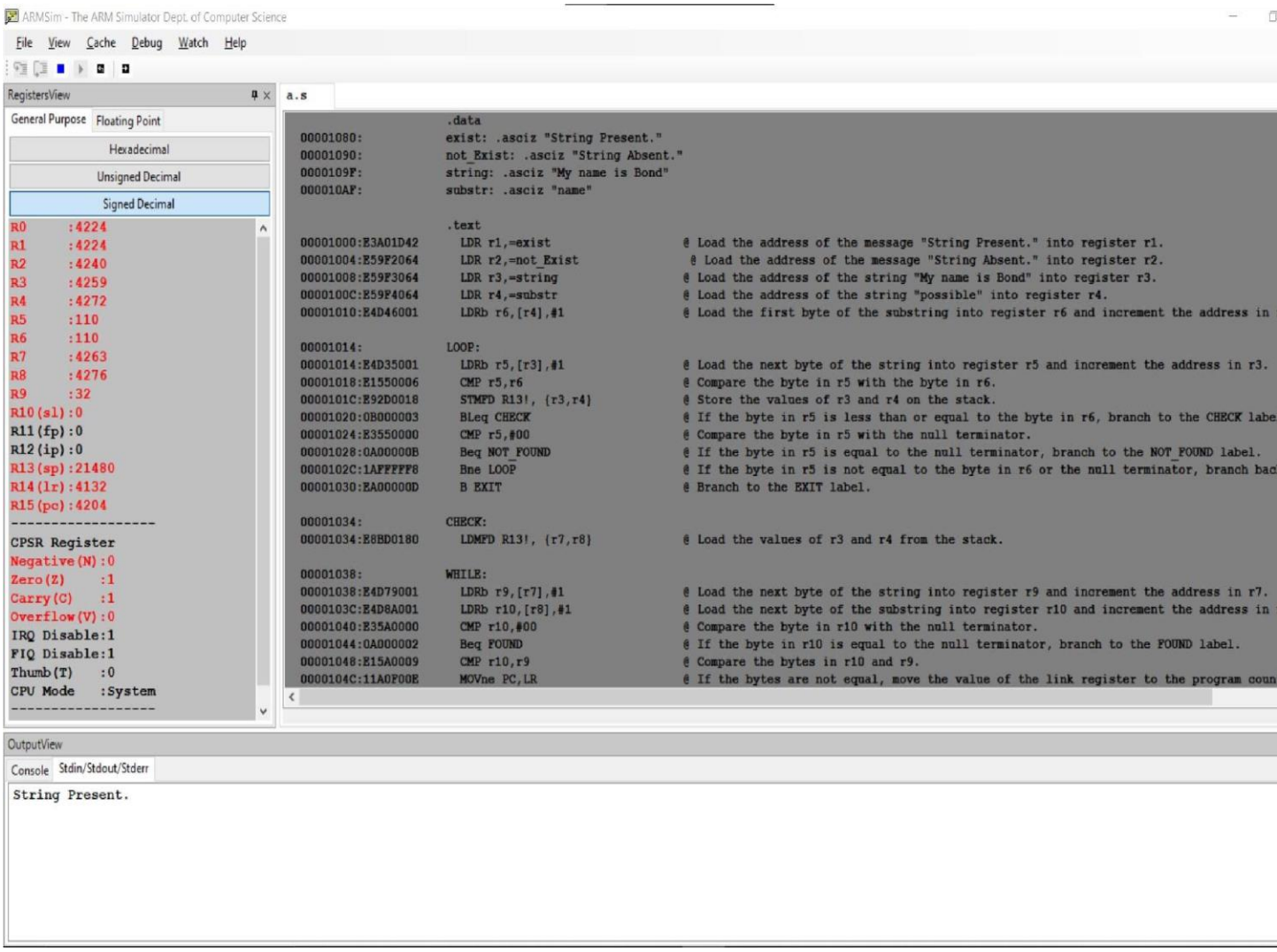
MOVne PC,LR @ If the bytes are not equal, move the value of the link register to the program counter.

B WHILE @ Branch back to the WHILE label.

```
FOUND:
  MOV r0,r1              @ Move the address of the message "String Present." into register r0.
  B OUTPUT               @ Branch to the OUTPUT label.

NOT_FOUND:
  MOV r0,r2              @ Move the address of the message "String Abesent." into register r0.
  B OUTPUT               @ Branch to the OUTPUT label.

OUTPUT:
SWI 0x02                 @ Call the OS to display the string in r0
B EXIT

EXIT:
SWI 0X11
```

OUTPUT SCREENSHOTS:

For string present:



For string absent: