

**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:

The screenshot displays the ARMSim - The ARM Simulator interface. The main window is divided into three panes: RegistersView, Assembly, and OutputView.

**RegistersView:** The 'General Purpose' register pane is active, showing the following values:

Register	Value
R0	:4228
R1	:9
R2	:4272
R3	:0
R4	:4
R5	:14
R6	:0
R7	:0
R8	:0
R9	:0
R10 (s1)	:14
R11 (fp)	:0
R12 (ip)	:0
R13 (sp)	:21504
R14 (lr)	:4152
R15 (pc)	:4204

Below the registers, the CPSR Register status is shown:

- Negative (N): 0
- Zero (Z): 1
- Carry (C): 1
- Overflow (V): 0
- IRQ Disable: 1
- FIQ Disable: 1
- Thumb (T): 0
- CPU Mode: System

**Assembly Pane:** The assembly code is displayed, showing a binary search algorithm. The code includes data definitions for search strings, list size, list elements, and search element, followed by the search logic in the .text section.

```
.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
00001000:E59F1068 LDR r1,=size @ Load the size of the list into register r1
00001004:E5911000 LDR r1,[r1] @ Load the value stored at the address in r1 into r1
00001008:E59F2064 LDR r2,=list @ Load the address of the list into register r2
0000100C:E3A03000 MOV r3,#0 @ Initialize the counter register r3 to 0
00001010:E59FA060 LDR r10,=searchElement @ Load the address of the search element into register r10
00001014:E59AA000 LDR r10,[r10] @ Load the value stored at the address in r10 into r10

00001018: START:
00001018:E1510003 CMP r1,r3 @ Compare the value of r1 and r3
0000101C:AA000000 Bge LOOP @ If r1 >= r3, jump to LOOP
00001020:EA00000E B NOT_FOUND @ If r1 < r3, jump to NOT_FOUND

00001024: LOOP:
00001024:E0814003 ADD r4,r1,r3 @ Calculate the middle element of the list by adding r1 and r3
00001028:E1A040A4 MOV r4,r4,lsr #1 @ Right shift the result by 1
0000102C:E7925104 LDR r5,[r2,r4,lsl #2] @ Load the element at the middle of the list into register r5
00001030:E155000A CMP r5,r10 @ Compare the value in r5 and r10
00001034:0B000006 BLEq FOUND @ If r5 <= r10, jump to FOUND
00001038:CB000003 BLgt GREATER_THAN @ If r5 > r10, jump to GREATER_THAN
0000103C:BB000000 BLlt LESS_THAN @ If r5 < r10, jump to LESS_THAN
00001040:EAF0FFFA B START @ Go back to the start of the loop

00001044: LESS_THAN:
00001044:E2843001 ADD r3,r4,#1 @ Increment the counter by 1
00001048:E1A0F00E MOV pc,lr @ Move the program counter to the link register
```

**OutputView:** The 'Console' tab is selected, showing the output: "Successful Search".

For Unsuccessful search:

ARMSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

Registers View

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 :4246

R1 :9

R2 :4272

R3 :9

R4 :9

R5 :20

R6 :0

R7 :0

R8 :0

R9 :0

R10 (s1) :20

R11 (fp) :0

R12 (ip) :0

R13 (sp) :21504

R14 (lr) :4152

R15 (pc) :4204

-----

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

-----

a.s

```
.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 20 @ Element to be searched for

.text
00001000:E59F1068 LDR r1,=size @ Load the size of the list into register r1
00001004:E5911000 LDR r1,[r1] @ Load the value stored at the address in r1 into
00001008:E59F2064 LDR r2,=list @ Load the address of the list into register r2
0000100C:E3A03000 MOV r3,#0 @ Initialize the counter register r3 to 0
00001010:E59FA060 LDR r10,=searchElement @ Load the address of the search element into register r10
00001014:E59AA000 LDR r10,[r10] @ Load the value stored at the address in r10 into register r10

00001018: START:
00001018:E1510003 CMP r1,r3 @ Compare the value of r1 and r3
0000101C:AA000000 Bge LOOP @ If r1 >= r3, jump to LOOP
00001020:EA00000E B NOT_FOUND @ If r1 < r3, jump to NOT_FOUND

00001024: LOOP:
00001024:E0814003 ADD r4,r1,r3 @ Calculate the middle element of the list by adding r1 and r3
00001028:E1A040A4 MOV r4,r4,lsr #1 @ Right shift the result by 1
0000102C:E7925104 LDR r5,[r2,r4,lsr #2] @ Load the element at the middle of the list into register r5
00001030:E155000A CMP r5,r10 @ Compare the value in r5 and r10
00001034:0B000006 BLEq FOUND @ If r5 <= r10, jump to FOUND
00001038:CB000003 BLgt GREATER_THAN @ If r5 > r10, jump to GREATER_THAN
0000103C:BB000000 BLlt LESS_THAN @ If r5 < r10, jump to LESS_THAN
00001040:EAF0FF4 B START @ Go back to the start of the loop

00001044: LESS_THAN:
00001044:E2843001 ADD r3,r4,#1 @ Increment the counter by 1
00001048:E1A0F00E MOV pc,lr @ Move the program counter to the link register
```

Output View

Console Stdin/Stdout/Stderr

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:

ARMSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 4224  
R1 : 4224  
R2 : 4240  
R3 : 4259  
R4 : 4272  
R5 : 110  
R6 : 110  
R7 : 4263  
R8 : 4276  
R9 : 32  
R10 (sl) : 0  
R11 (fp) : 0  
R12 (ip) : 0  
R13 (sp) : 21480  
R14 (lr) : 4132  
R15 (pc) : 4204

-----  
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

-----

a.s

```
.data
00001080:      exist: .asciz "String Present."
00001090:      not_Exist: .asciz "String Absent."
0000109F:      string: .asciz "My name is Bond"
000010AF:      substr: .asciz "name"

.text
00001000:E3A01D42      LDR r1,=exist           @ Load the address of the message "String Present." into register r1.
00001004:E59F2064      LDR r2,=not_Exist        @ Load the address of the message "String Absent." into register r2.
00001008:E59F3064      LDR r3,=string           @ Load the address of the string "My name is Bond" into register r3.
0000100C:E59F4064      LDR r4,=substr           @ Load the address of the string "possible" into register r4.
00001010:E4D46001      LDRb r6,[r4],#1          @ Load the first byte of the substring into register r6 and increment the address in

00001014:      LOOP:
00001014:E4D35001      LDRb r5,[r3],#1          @ Load the next byte of the string into register r5 and increment the address in r3.
00001018:E1550006      CMP r5,r6                @ Compare the byte in r5 with the byte in r6.
0000101C:E92D0018      STMPD R13!, {r3,r4}      @ Store the values of r3 and r4 on the stack.
00001020:0B000003      BLEQ CHECK               @ If the byte in r5 is less than or equal to the byte in r6, branch to the CHECK label.
00001024:E3550000      CMP r5,#00              @ Compare the byte in r5 with the null terminator.
00001028:0A00000B      BEQ NOT_FOUND            @ If the byte in r5 is equal to the null terminator, branch to the NOT_FOUND label.
0000102C:1AFFFFF8      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.
00001030:EA00000D      B EXIT                   @ Branch to the EXIT label.

00001034:      CHECK:
00001034:E8BD0180      LDMFD R13!, {r7,r8}      @ Load the values of r3 and r4 from the stack.

00001038:      WHILE:
00001038:E4D79001      LDRb r9,[r7],#1          @ Load the next byte of the string into register r9 and increment the address in r7.
0000103C:E4D8A001      LDRb r10,[r8],#1         @ Load the next byte of the substring into register r10 and increment the address in
00001040:E35A0000      CMP r10,#00              @ Compare the byte in r10 with the null terminator.
00001044:0A000002      BEQ FOUND                @ If the byte in r10 is equal to the null terminator, branch to the FOUND label.
00001048:E15A0009      CMP r10,r9               @ Compare the bytes in r10 and r9.
0000104C:11A0F00E      MOVne PC,LR              @ If the bytes are not equal, move the value of the link register to the program coun
```

OutputView

Console Stdin/Stdout/Stderr

String Present.

For string absent:

ARMSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 4240  
R1 : 4224  
R2 : 4240  
R3 : 4271  
R4 : 4272  
R5 : 0  
R6 : 74  
R7 : 0  
R8 : 0  
R9 : 0  
R10 (s1) : 0  
R11 (fp) : 0  
R12 (ip) : 0  
R13 (sp) : 21376  
R14 (lr) : 0  
R15 (pc) : 4204

-----  
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  
-----

a.s

```
.data
00001080:      exist: .asciz "String Present."
00001090:      not_Exist: .asciz "String Absent."
0000109F:      string: .asciz "My name is Bond"
000010AF:      substr: .asciz "James"

.text
00001000:E3A01D42      LDR r1,=exist           @ Load the address of the message "String Present." into register r1.
00001004:E59F2064      LDR r2,=not_Exist        @ Load the address of the message "String Absent." into register r2.
00001008:E59F3064      LDR r3,=string           @ Load the address of the string "My name is Bond" into register r3.
0000100C:E59F4064      LDR r4,=substr           @ Load the address of the string "possible" into register r4.
00001010:E4D46001      LDRb r6,[r4],#1          @ Load the first byte of the substring into register r6 and increment the address

00001014:      LOOP:
00001014:E4D35001      LDRb r5,[r3],#1          @ Load the next byte of the string into register r5 and increment the address
00001018:E1550006      CMP r5,r6                @ Compare the byte in r5 with the byte in r6.
0000101C:E92D0018      STMFd R13!, {r3,r4}      @ Store the values of r3 and r4 on the stack.
00001020:0B000003      BLEq CHECK               @ If the byte in r5 is less than or equal to the byte in r6, branch to the CHECK label.
00001024:E3550000      CMP r5,#00               @ Compare the byte in r5 with the null terminator.
00001028:0A00000B      Beq NOT_FOUND            @ If the byte in r5 is equal to the null terminator, branch to the NOT_FOUND label.
0000102C:1AFFFFF8      Bne LOOP                 @ If the byte in r5 is not equal to the byte in r6 or the null terminator, branch to the LOOP label.
00001030:EAD0000D      B EXIT                   @ Branch to the EXIT label.

00001034:      CHECK:
00001034:E8BD0180      LDMFD R13!, {r7,r8}      @ Load the values of r3 and r4 from the stack.

00001038:      WHILE:
00001038:E4D79001      LDRb r9,[r7],#1          @ Load the next byte of the string into register r9 and increment the address
0000103C:E4D8A001      LDRb r10,[r8],#1         @ Load the next byte of the substring into register r10 and increment the address
00001040:E35A0000      CMP r10,#00              @ Compare the byte in r10 with the null terminator.
00001044:0A000002      Beq FOUND                @ If the byte in r10 is equal to the null terminator, branch to the FOUND label.
00001048:E15A0009      CMP r10,r9                @ Compare the bytes in r10 and r9.
0000104C:11A0F00E      MOVne PC,LR              @ If the bytes are not equal, move the value of the link register to the program counter

EXIT:
```

OutputView

Console Stdin/Stdout/Stderr

String Absent.