Microprocessor and Computer Architecture
UE20CS252
Assignment

Date: 18/03/2022
Name: Vishwa Mehul Mehta
SRN: PES2UG20CS389
Section: F

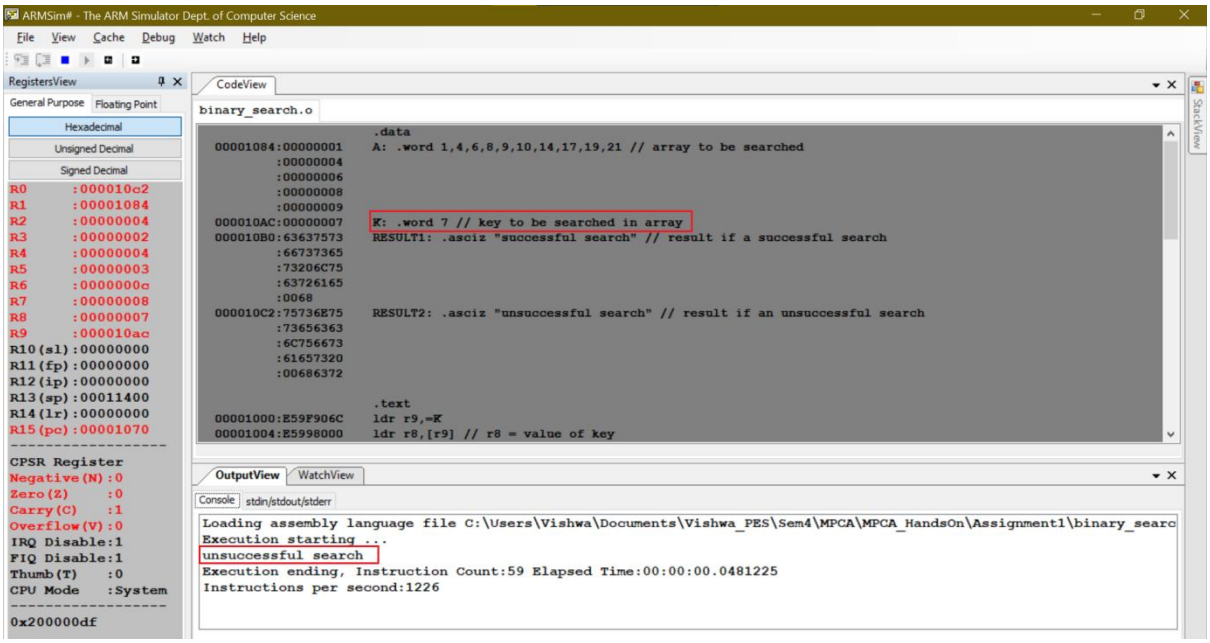
1. Write a program in ARM7TDMI-ISA to search for an element in an array. Display appropriate messages on the standard output device. For Successful search display as “Successful Search” and if the search is unsuccessful, display as “Unsuccessful Search”. Use Binary search Technique.

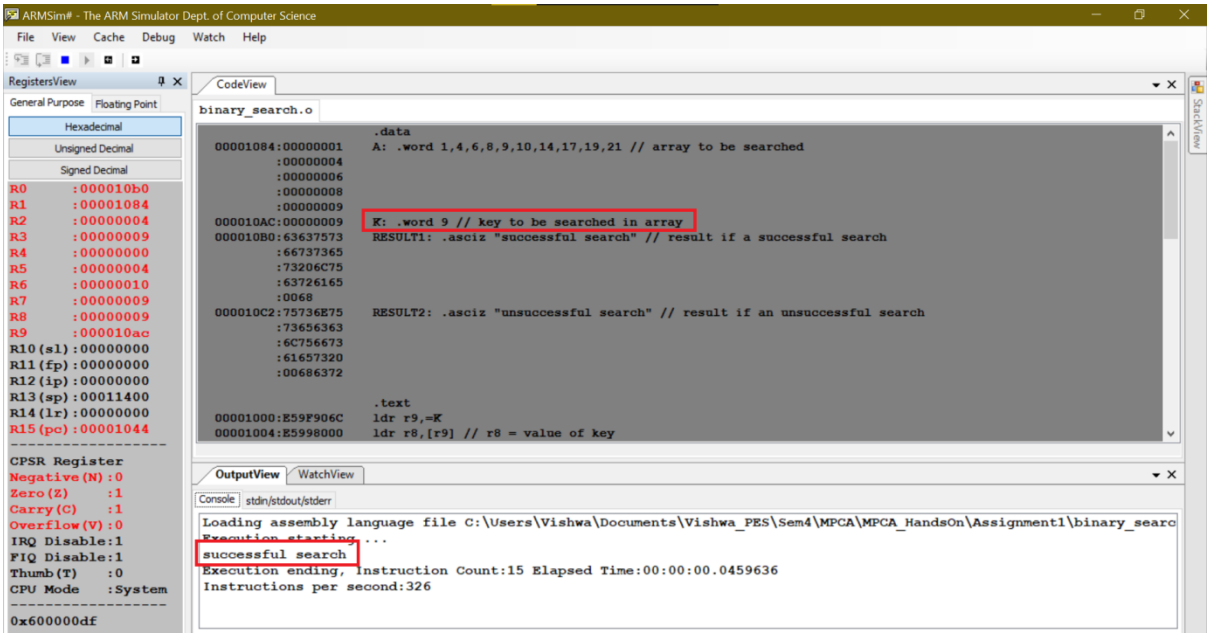
Code:

```
 1 .data
 2 A: .word 1,4,6,8,9,10,14,17,19,21 // array to be searched
 3 K: .word 7 // key to be searched in array
 4 RESULT1: .asciz "successful search" // result if a successful search
 5 RESULT2: .asciz "unsuccessful search" // result if an unsuccessful search
 6
 7 .text
 8 ldr r9,=K
 9 ldr r8,[r9] // r8 = value of key
10 ldr r1,=A // array pointer
11 mov r2,#4 // step
12 mov r3,#9 // initial high value
13 mov r4,#0 // initial low value
14 loop:
15     add r5,r3,r4
16     mov r5,r5,lsr #1 // r5 = mid value
17     mul r6,r5,r2 // position of mid from the start of array
18     ldr r7,[r1,r6] // r7 = array[mid]
19     cmp r7,r8 // compare array[mid] and key
20     beq success // key was found (array[mid] == key)
21     bgt l1 // array[mid] > key
22     b l2 // array[mid] < key
23 b fail // key not found
24
25 success:
26     ldr r0,=RESULT1
27     swi 0x02
"binary_search.s" 46L, 957C
```

```
27      swi 0x02
28      swi 0x11
29
30 l1:
31      mov r3,r5
32      sub r3,r3,#1 // high = mid - 1
33      cmp r4,r3
34      ble loop
35
36 l2:
37      mov r4,r5
38      add r4,r4,#1 // low = mid + 1
39      cmp r4,r3
40      ble loop
41
42 fail:
43      ldr r0,=RESULT2
44      swi 0x02
45      swi 0x11
46 .end
```

Output:





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

Example1: Main string : My name is Bond.

Character : ‘name’.

Expected Output : “String Present”

Example2: Main string : My name is Bond.

Character : ‘James’.

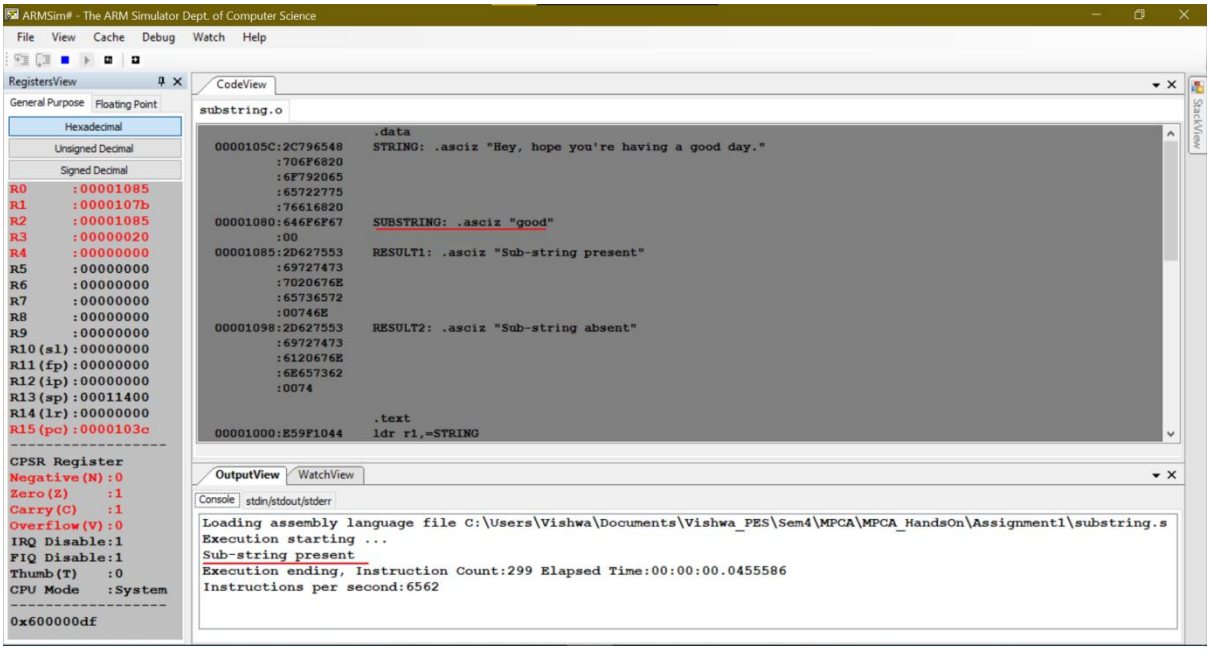
Expected Output : “String Absent”

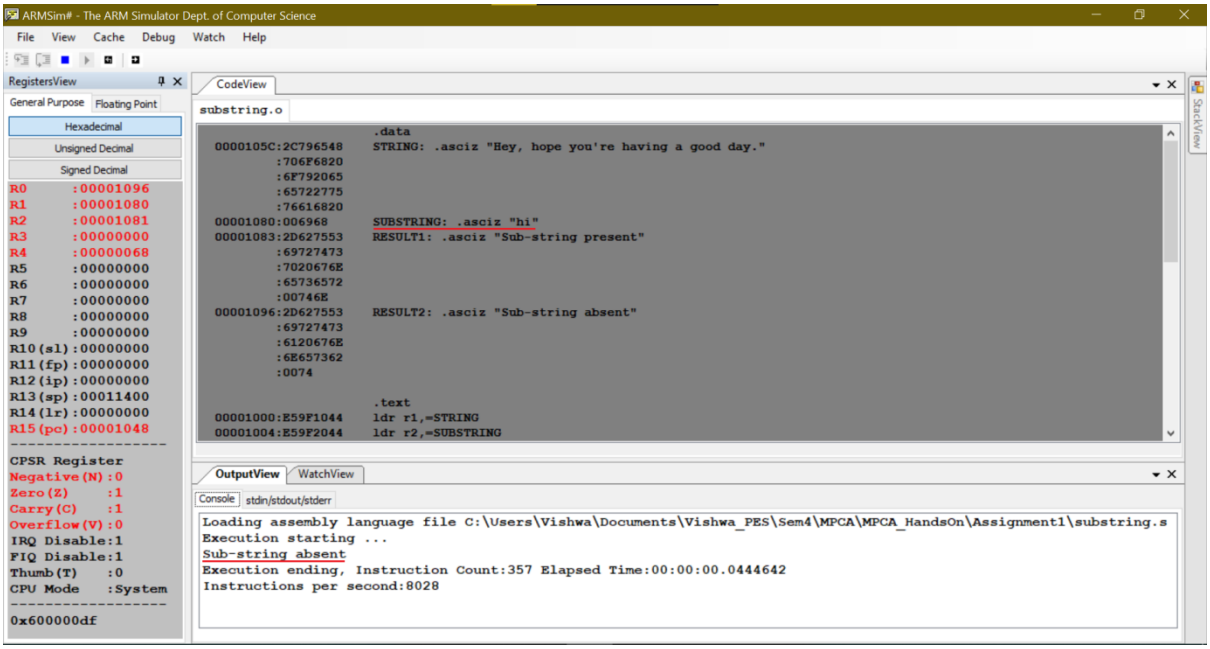
Code:

```
 1 .data
 2 STRING: .asciz "Hey, hope you're having a good day."
 3 SUBSTRING: .asciz "good"
 4 RESULT1: .asciz "Sub-string present"
 5 RESULT2: .asciz "Sub-string absent"
 6
 7 .text
 8 ldr r1,=STRING
 9 ldr r2,=SUBSTRING
10
11 loop:
12     ldrb r3,[r1],#1
13     ldrb r4,[r2],#1
14     cmp r4,#0 // check if substring has been completely traversed
15     beq success //substring present
16     cmp r3,#0 // check if string has been completely traversed
17     beq fail // substring not present
18     cmp r3,r4 // checks if string character and substring character are equal
19     beq loop // if equal continue
20     ldr r2,=SUBSTRING // else re-initialise substring
21     b loop
22 b fail
23
24 success:
25     ldr r0,=RESULT1
26     swi 0x02
27     swi 0x11
"substring.s" 34L, 700C
```

```
27      swi 0x11
28
29 fail:
30      ldr r0,=RESULT2
31      swi 0x02
32      swi 0x11
33
34 .end
```

Output:





3. Consider the following sequence of instructions in MIPS architecture.

LDR R1, [R2, #40]

ADD R2, R3, R3

ADD R1, R1, R2

STR R1, [R2, #20]

a. Find all dependencies in this instruction sequence.

b. Find all hazards in this instruction sequence for a five stage pipeline with and without data forwarding.

c. Find whether NOPs are required to be introduced inspite of data forwarding in this instruction sequence.

SRN : PES2UG20CS389
NAME : Vishwa Mehul Mehta

MPCA Assignment 1:

Q3.

a) Instructions 1 & 3 have RAW dependency
Instructions 2 & 3 have RAW dependency
Instructions 1 & 2 have WAR "
Instructions 3 & 4 have RAW and WAW dependency
Instructions 2 & 4 have WAW dependency.

b) Without data forwarding:

Structural hazard ID vs WB in instructions 1 and 4.
~~Data hazard can't be determined as there's a structural hazard.~~

~~c) NOPs aren't required~~
Structural hazard : IF vs MEM in instruction 1 & 4. (if no split cache)
Data hazards :-
Instr. 1 & 3, 1 & 4, 2 & 3, 2 & 4, 3 & 4
(w/o partition)

With data forwarding:
Instr. 1 & 3

c) 1 NOP required after 2nd instruction. ~~to push ID stage of 4th instr.~~

4. Consider the following sequence of instructions in MIPS architecture.

```
LDR  R1, [R6,#40]
BEQ  R2, R3, LABEL2   ; BRANCH TAKEN
ADD R1, R6, R4
```

```
LABEL2: BEQ R1,R2, LABEL1       ; BRANCH NOT TAKEN
        STR R2,[R4, #20]
        AND R1, R1, R4
```

a. Draw the pipeline execution diagram for this code, assuming there are no delay slots and that branches execute in the EX stage.

b. Repeat the exercise mentioned in a and draw the pipeline execution diagram for this code, assuming that delay slots are used by writing a “SAFE INSTRUCTION” in the delay slot.