PES University, Bangalore
(EstablishedunderKarnatakaActNo.16of2013)
B.Tech., 4th Semester, March 2022
UE20CS252: Microprocessor and Computer Architecture
Assignment -1
Last Date of Submission : 18th March 2022.

NAME : VISHWAS M
SRN : PES2UG20CS390
SEC : F

DATE : 18/03/2022

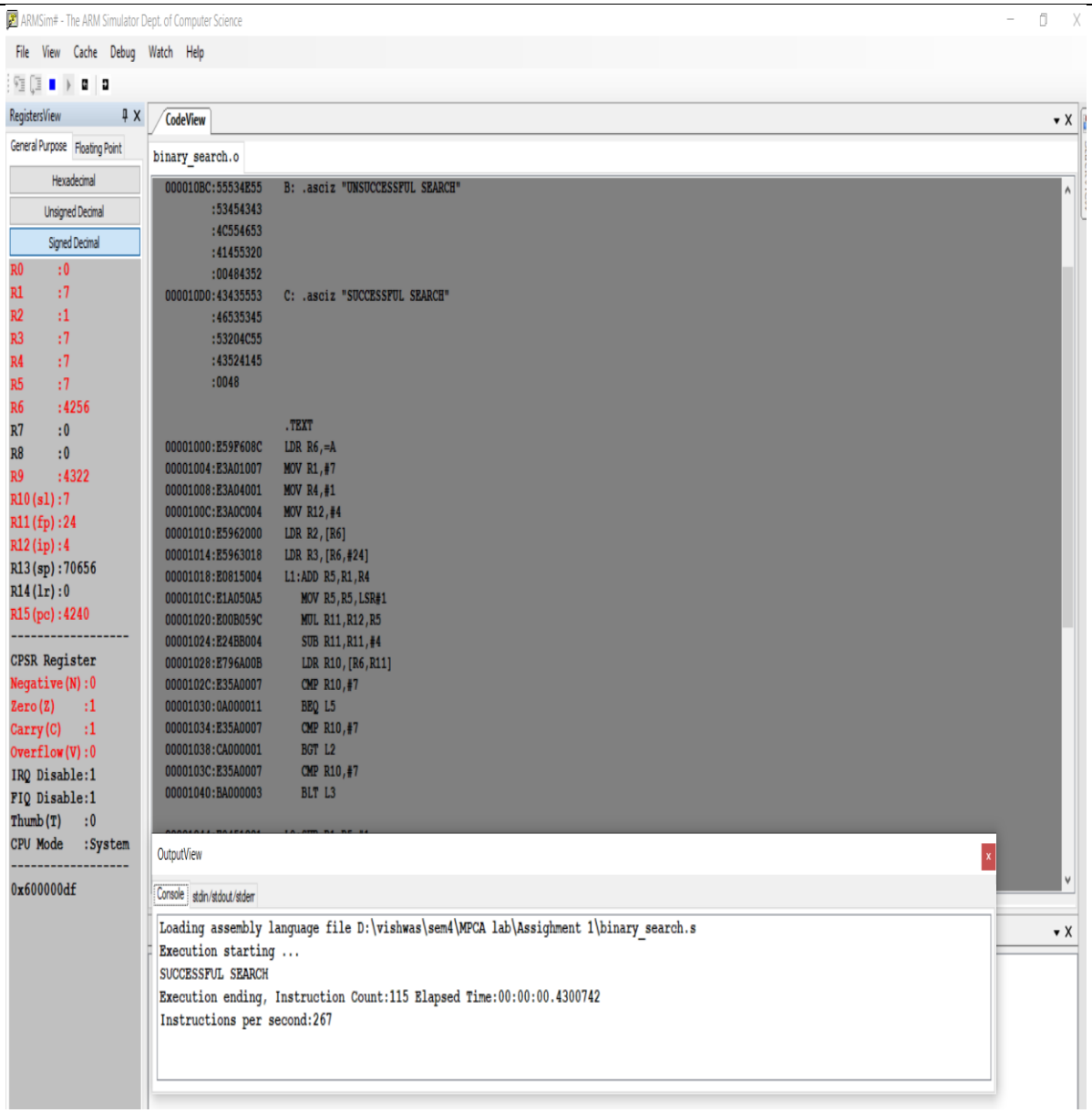
Sl #	Question

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.
	
	PROGRAM:
	SUCCESSFUL SEARCH:
	//SEARCHING FOR NUMBER 7 IN THE GIVEN SORTED ARRAY
	.DATA
	A: .WORD 1,2,3,4,5,6,7
	B: .asciz "UNSUCCESSFUL SEARCH"
	C: .asciz "SUCCESSFUL SEARCH"
	
	.TEXT
	LDR R6,=A
	MOV R1,#7
	MOV R4,#1
	MOV R12,#4
	LDR R2,[R6]
	LDR R3,[R6,#24]
	L1:ADD R5,R1,R4
	MOV R5,R5,LSR#1
	MUL R11,R12,R5
	SUB R11,R11,#4
	LDR R10,[R6,R11]
	CMP R10,#7
	BEQ L5
	CMP R10,#7
	BGT L2
	CMP R10,#7
	BLT L3

```
L2:SUB R1,R5,#1
  CMP R4,R1
  BLE L1
  B L4

L3:ADD R4,R5,#1
  CMP R4,R1
  BLE L1
  B L4

L4:LDR R7,=B
strprints:  LDRB  R0, [R7], #1
        CMP R0, #0
        SWINE 0x00
        BNE   strprints
        SWI  0x11
L5:LDR R9,=C
strprint:  LDRB  R0, [R9], #1
        CMP R0, #0
        SWINE 0x00
        BNE   strprint
        SWI  0x11
```



UNSUCCESSFUL SEARCH:

//SEARCHING FOR NUMBER 9 IN THE GIVEN SORTED ARRAY

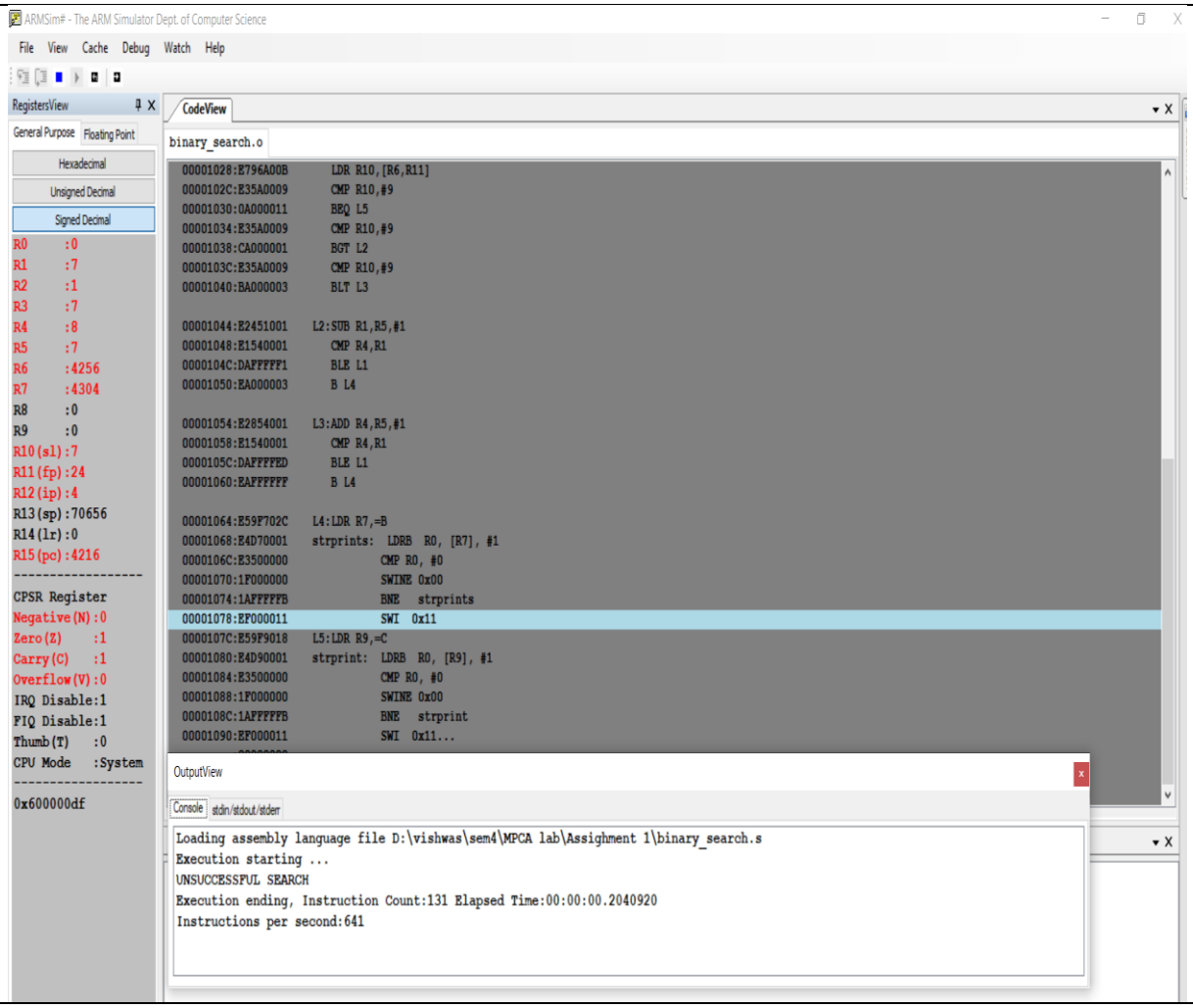
```
.DATA
A: .WORD 1,2,3,4,5,6,7
B: .asciz "UNSUCCESSFUL SEARCH"
C: .asciz "SUCCESSFUL SEARCH"
.TEXT
LDR R6,=A
MOV R1,#7
MOV R4,#1
MOV R12,#4
LDR R2,[R6]
LDR R3,[R6,#24]
L1:ADD R5,R1,R4
  MOV R5,R5,LSR#1
  MUL R11,R12,R5
```

```
  SUB R11,R11,#4
  LDR R10,[R6,R11]
  CMP R10,#9
  BEQ L5
  CMP R10,#9
  BGT L2
  CMP R10,#9
  BLT L3

L2:SUB R1,R5,#1
  CMP R4,R1
  BLE L1
  B L4

L3:ADD R4,R5,#1
  CMP R4,R1
  BLE L1
  B L4

L4:LDR R7,=B
strprints:  LDRB  R0, [R7], #1
        CMP R0, #0
        SWINE 0x00
        BNE   strprints
        SWI  0x11
L5:LDR R9,=C
strprint:  LDRB  R0, [R9], #1
        CMP R0, #0
        SWINE 0x00
        BNE   strprint
        SWI  0x11
```



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"

PROGRAM:

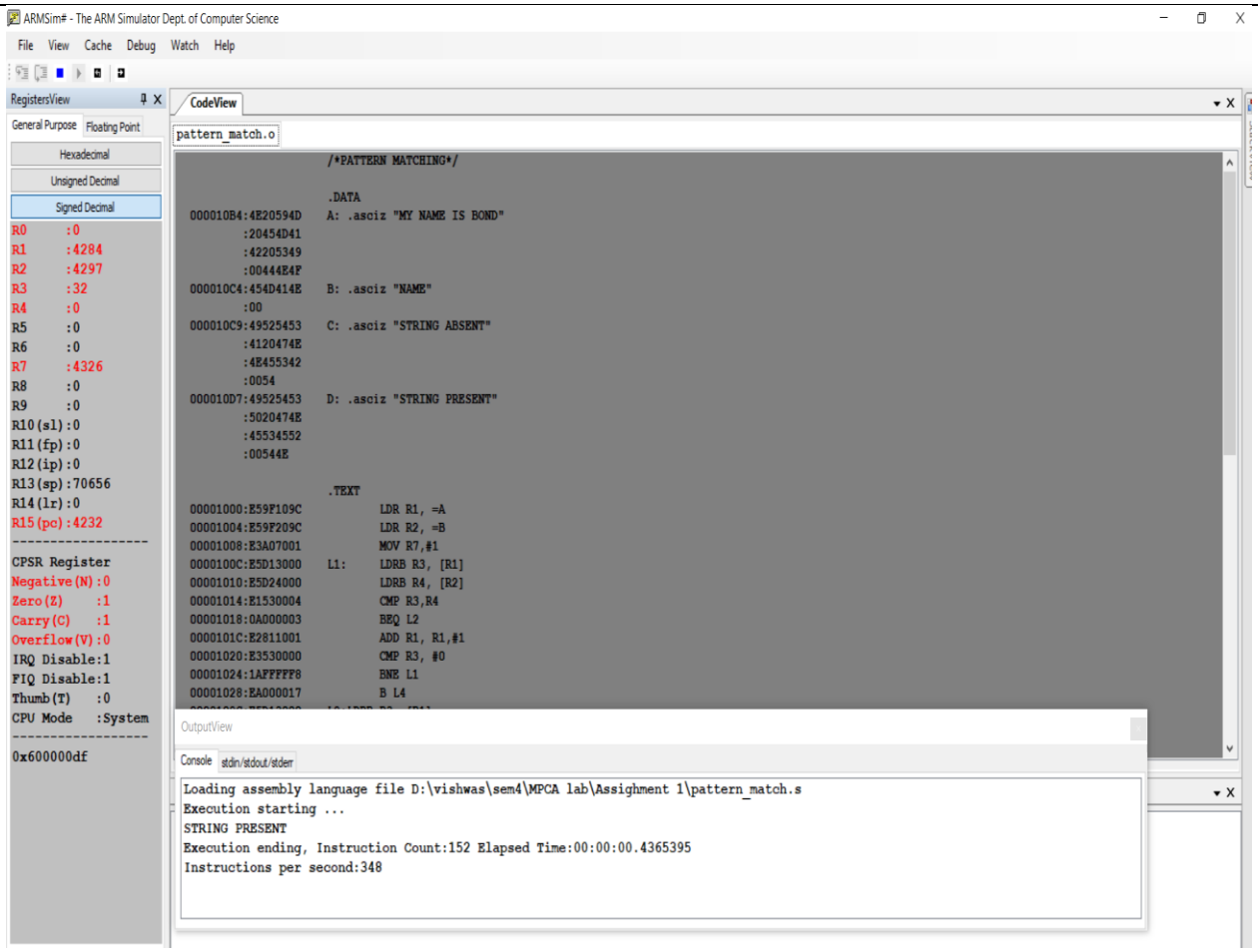
```
/*PATTERN MATCHING*/
//SUCCESSFUL SEARCH
.DATA
A: .asciz "MY NAME IS BOND"
B: .asciz "NAME"
C: .asciz "STRING ABSENT"
D: .asciz "STRING PRESENT"

.TEXT
        LDR R1, =A
        LDR R2, =B
    MOV R7,#1
L1:     LDRB R3, [R1]
        LDRB R4, [R2]
    CMP R3,R4
    BEQ L2
```

```
      ADD R1, R1,#1
         CMP R3, #0
         BNE L1
      B L4
L2:LDRB R3, [R1]
  LDRB R4, [R2]
  CMP R3,R4
  ADDEQ R7,R7,#1
  CMP R7,#6
  BEQ L3
  CMP R3,#0
  BEQ L4
  CMP R3,R4
  ADD R1, R1,#1
  ADD R2, R2,#1
  BEQ L2
  CMP R4,#0
  BEQ L3
  LDR R2, =B
  ADD R1, R1,#1
  ADD R2, R2,#1
  B L1

L3:LDR R7,=D
strprints:  LDRB  R0, [R7],#1
        CMP R0, #0
        SWINE 0x00
        BNE strprints
        SWI  0x11

L4:LDR R9,=C
strprint:  LDRB  R0, [R9], #1
        CMP R0, #0
        SWINE 0x00
        BNE   strprint
        SWI  0x11
```



Example2: Main string : My name is Bond.
Character : 'James'.

Expected Output : "String Absent"

PROGRAM:

/*PATTERN MATCHING*/

//FOR UNSUCCESSFUL SEARCH

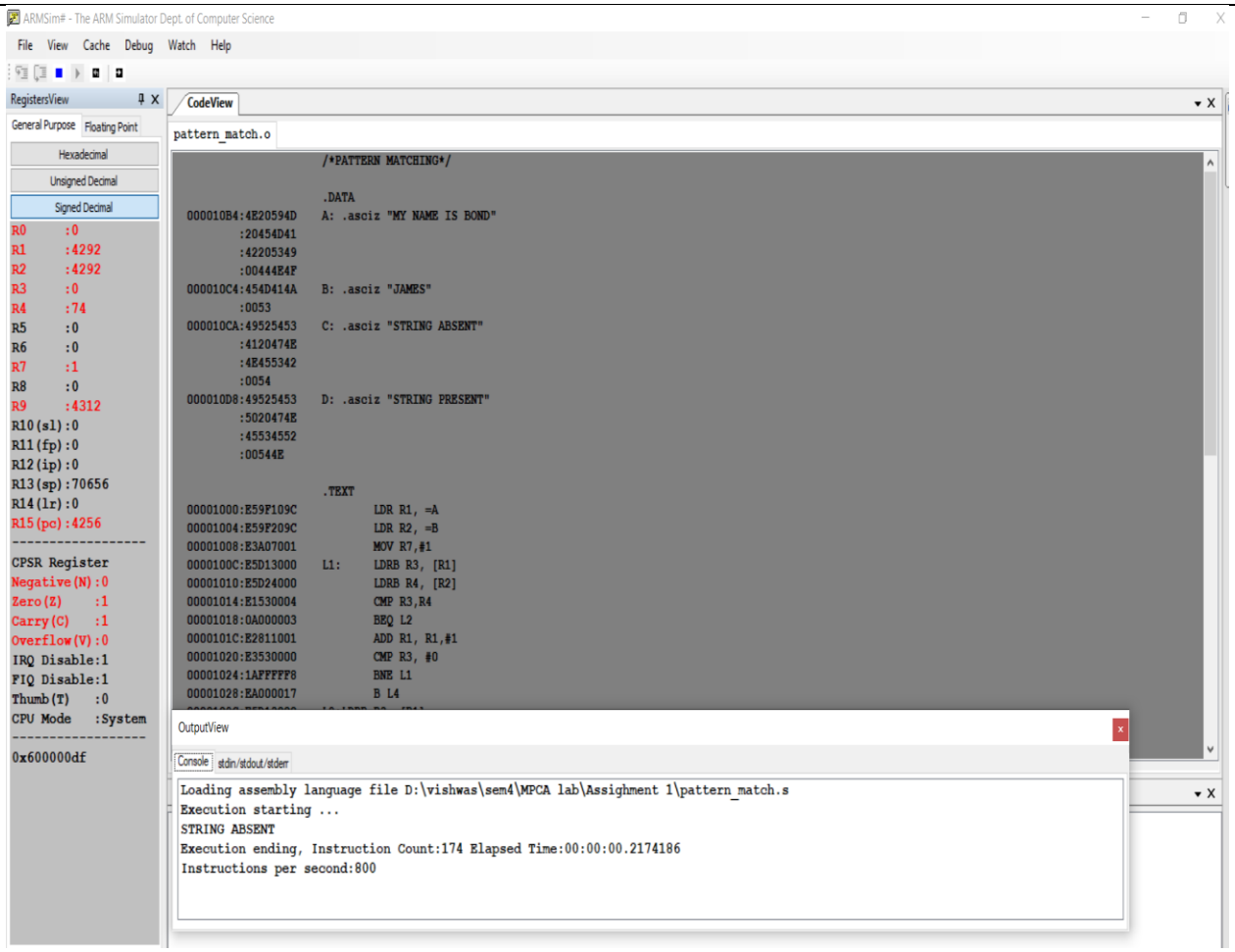
```
.DATA
A: .asciz "MY NAME IS BOND"
B: .asciz "JAMES"
C: .asciz "STRING ABSENT"
D: .asciz "STRING PRESENT"

.TEXT
        LDR R1, =A
        LDR R2, =B
    MOV R7,#1
L1:     LDRB R3, [R1]
        LDRB R4, [R2]
    CMP R3,R4
    BEQ L2
    ADD R1, R1,#1
        CMP R3, #0
        BNE L1
```

```
      B L4
L2:LDRB R3, [R1]
  LDRB R4, [R2]
  CMP R3,R4
  ADDEQ R7,R7,#1
  CMP R7,#6
  BEQ L3
  CMP R3,#0
  BEQ L4
  CMP R3,R4
  ADD R1, R1,#1
  ADD R2, R2,#1
  BEQ L2
  CMP R4,#0
  BEQ L3
  LDR R2, =B
  ADD R1, R1,#1
  ADD R2, R2,#1
  B L1

L3:LDR R7,=D
strprints:  LDRB  R0, [R7],#1
        CMP R0, #0
        SWINE 0x00
        BNE strprints
        SWI  0x11

L4:LDR R9,=C
strprint:  LDRB  R0, [R9], #1
        CMP R0, #0
        SWINE 0x00
        BNE   strprint
        SWI  0x11
```


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 :0
R1 :4292
R2 :4292
R3 :0
R4 :74
R5 :0
R6 :0
R7 :1
R8 :0
R9 :4312
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):70656
R14(lr):0
R15(pc):4256
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
CodeView
pattern_match.o
/*PATTERN MATCHING*/
.DATA
000010B4:4E20594D A: .asciz "MY NAME IS BOND"
:20454D41
:42205349
:00444E4F
000010C4:454D414A B: .asciz "JAMES"
:0053
000010CA:49525453 C: .asciz "STRING ABSENT"
:4120474E
:4E455342
:0054
000010D8:49525453 D: .asciz "STRING PRESENT"
:5020474E
:45534552
:00544E
.TEXT
00001000:E59F109C LDR R1, =A
00001004:E59F209C LDR R2, =B
00001008:E3A07001 MOV R7,#1
0000100C:E5D13000 L1: LDRB R3, [R1]
00001010:E5D24000 LDRB R4, [R2]
00001014:E1530004 CMP R3,R4
00001018:0A000003 BEQ L2
0000101C:E2811001 ADD R1, R1,#1
00001020:E3530000 CMP R3, #0
00001024:1AFFFFF8 BNE L1
00001028:EA000017 B L4
OutputView
Console stdin/stdout/stderr
Loading assembly language file D:\vishwas\sem4\MPCA lab\Assighment 1\pattern_match.s
Execution starting ...
STRING ABSENT
Execution ending, Instruction Count:174 Elapsed Time:00:00:00.2174186
Instructions per second:800

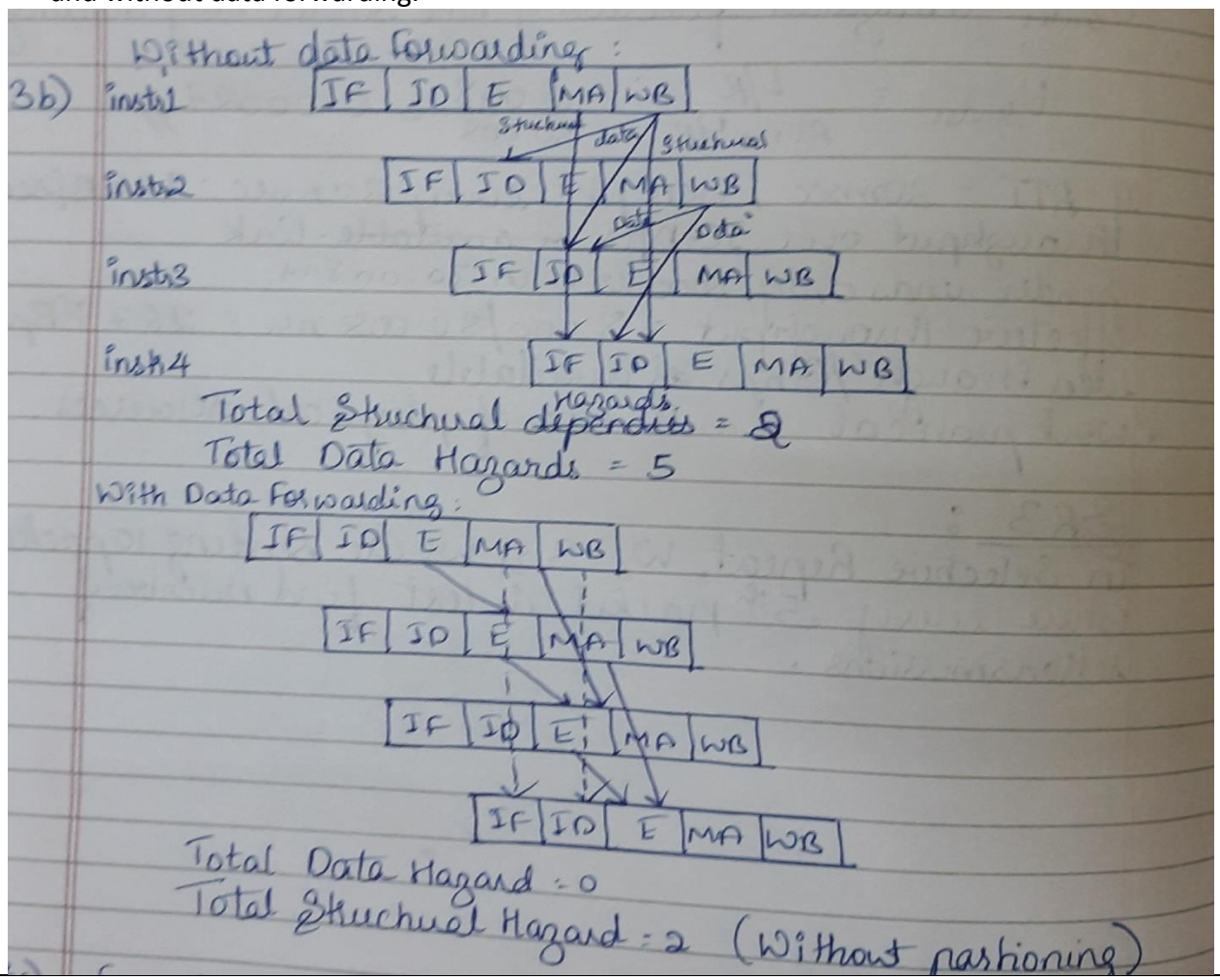
3 Consider the following sequence of instructions in MIPS architecture.

LDR R1, [R2,#40]
ADD R2, R3, R3
ADD R1, R1, R2
STR R1, [R2,#20]

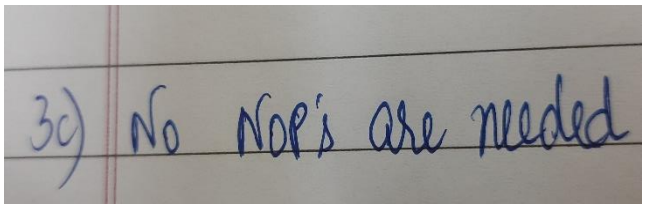
a. Find all dependencies in this instruction sequence.

3a) The data dependencies are:
a) i1 & i2 → WAR
b) i2 & i3 → RAW
c) i2 & i4 → WAW
d) i3 & i4 → WAR (R2)
e) i1 & i3 → RAW
f) i3 & i4 → RAW (R1)

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



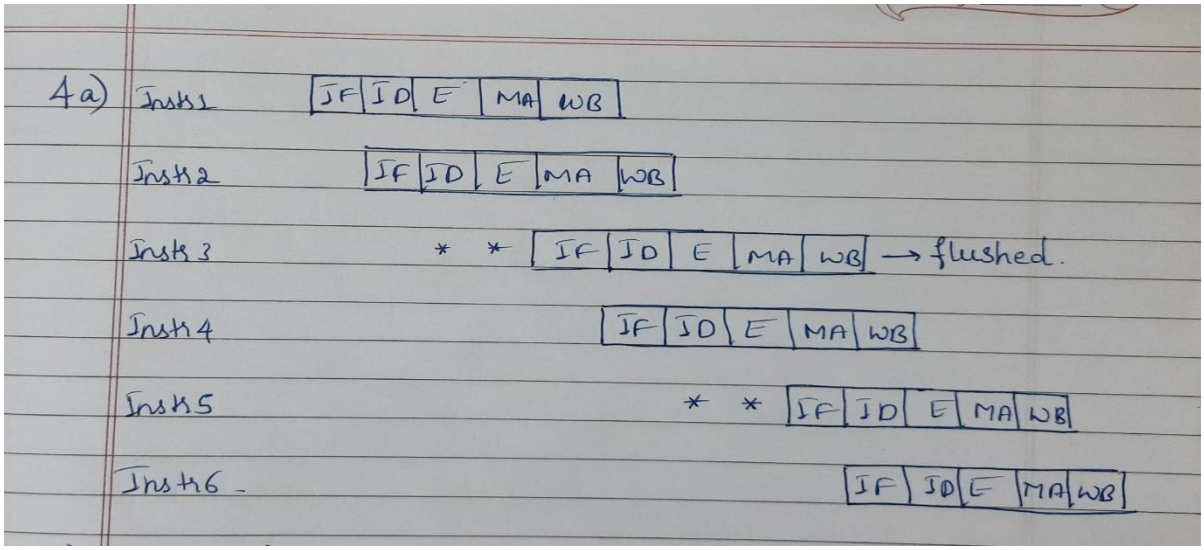
c. Find whether NOPs are required to be introduced inspite of data forwarding in this instruction sequence.



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.