PES University, Bangalore



(EstablishedunderKarnatakaActNo.16of2013)

**B.Tech., 4thSemester, March 2022**

**UE20CS252: Microprocessor and Computer Architecture**

**Assignment -1**

**Last Date of Submission : 20th March 2022.**

**Sriram Radhakrishna PES1UG20CS435 Section : ‘H’**

|  |  |
| --- | --- |
| **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 :**  .data  a: .word 10, 11, 15, 17, 20, 2528, 30, 35, 40  b: .word 30  s: .asciiz "successful search"  us: .asciiz "unsuccessful search"    .text  ldr r1, =a  ldr r6, =b  ldr r7, [r6]  mov r3, #10  mov r4, #0  mov r10, #4    loop:  add r5, r4, r3  mul r11, r10, r5  mov r5, r5, lsr #1  sub r11, r11, #4  ldr r8, [r1, r11]  cmp r7, r8  beq successful  bpl high  b low    high:  mov r4, r5, lsr #1  sub r9, r3, r4  cmp r9, #1  beq rcheck  b loop    low:  mov r3, r5, lsr #1  sub r9, r3, r4  cmp r9, #1  beq rcheck  b loop    rcheck:  mul r11, r10, r3  ldr r8, [r1, r11]  cmp r8, r7  beq successful  b unsuccessful    lcheck:  mul r11, r10, r4  ldr r8, [r1, r11]  cmp r8, r7  beq successful  b unsuccessful    successful:    ldr r0, =s  swi 0x02  swi 0x11    unsuccessful:  ldr r0, =us  swi 0x02  swi 0x11    .end  **Screenshot :** |
| **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”**  **Program :**  .data  a: .asciz "abcd efgh"  b: .asciz "ef"  s: .asciz "substring found"  us: .asciz "substring not found"    .text  ldr r1, =a  ldr r2, =b  mov r5, #14  mov r6, #3  sub r7, r5, r6  mov r8, #1    outer:    ldrb r3, [r1]  ldrb r4, [r2]  cmp r3, r4  mov r9, #0  beq inner  add r1, r1, #1  cmp r8, r7  beq exit  add r8, r8, #1  b outer    inner:  cmp r4, #0  beq successful  add r9, r9, #1  add r1, r1, #1  add r2, r2, #1  ldrb r3, [r1]  ldrb r4, [r2]  cmp r3, r4  beq inner  sub r1, r1, r9  add r1, r1, #1  sub r2, r2, r4  b outer    successful:  ldr r0, =s  swi 0x02  swi 0x11    exit:  ldr r0, =us  swi 0x02  swi 0x11    **Screenshot :** |
| **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]   1. Find all dependencies in this instruction sequence.   **Answer :**   * WAR in LDR R1, [R2,#40] & ADD R2, R3, R3 * WAW in ADD R2, R3, R3 & STR R1, [R2,#20] * WAW in ADD R1, R1, R2 & STR R1, [R2,#20] * RAW in ADD R2, R3, R3 & ADD R1, R1, R2 * RAW in LDR R1, [R2,#40] & ADD R1, R1, R2 * RAW in ADD R1, R1, R2 & STR R1, [R2,#20]   b. Find all hazards in this instruction sequence for a five stage pipeline with  and without data forwarding.  **Answer :**   * Data hazard * Structural hazard : Clash in memory writeback in LDR R1, [R2,#40] & STR R1, [R2,#20]   c. Find whether NOPs are required to be introduced inspite of data  forwarding in this instruction sequence.  **Answer :** NOPs aren’t required for this instruction set. |
| **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.  **Answer :**  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.  **Answer :** |