**Imam Mohammad Ibn Saud Islamic University**

**College of Computer and Information Sciences**

**Computer Science Department**

**Bachelor of Science in Computer Science**

|  |  |
| --- | --- |
| **Course Title:** | *Computer Architecture* |
| **Course Code:** | CS228 |
| **Course Instructor:** | Dr. Mostafa Ibrahim |
| **Assignment:** | High level to MIPS Assembly Translation |
| **Semester:** | 3rd Semester, 2024 |
| **Due Date:** | Day (2024-04-27( |
| **Marks:** | 5 |

|  |  |
| --- | --- |
| **Student Name (in English):** | **Abdulaziz Fahad Askar** |
| **Student ID:** | 443015550 |
| **Section No.:** | 172 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Official Use Only** | | | |
| **Question** | **CLOs** | **Student Marks** | **Question Marks** |
| 1 |  |  | 2 |
| 2 |  |  | 3 |
| **Total** | |  | **5** |

**Question 1: Write the MIPS instructions for the following statement: [2 marks]**

**if ( B[2]>=0)**

**X = A[B[2]] \* 2;**

Assume that:

* A is an array stored in memory and its starting address is in $a0.
* B is an array stored in memory and its starting address is in $a1.
* X is a variable associated with $s0.

|  |
| --- |
| # Setting up the environment  .data  A: .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10  # Array A  B: .word 0, -1, 5, -3, 4                 # Array B  .text  # Main function for SPIM Execution  .globl main  main:      la $a0, A    # Load base address of A into $a0      la $a1, B    # Load base address of B into $a1      # Load B[2] into $t0      addi $t1, $zero, 2          # $t1 = 2 (index)      sll $t1, $t1, 2             # $t1 = $t1 \* 4 (byte offset)      add $t1, $a1, $t1           # $t1 = base address of B + byte offset      lw $t0, 0($t1)              # $t0 = B[2]        # Check if B[2] is greater than or equal to 0      bgez $t0, compute           # if B[2] >= 0, jump to compute      # If B[2] is less than 0, we do nothing and exit      j exit      compute:      # Load A[B[2]] into $t2      sll $t0, $t0, 2             # $t0 = B[2] \* 4 (byte offset)      add $t0, $a0, $t0           # $t0 = base address of A + byte offset      lw $t2, 0($t0)              # $t2 = A[B[2]]      # Print A[B[2]]      move $a0, $t2  # Move the value of $t2 to $a0      li $v0, 1      # Load the system call code for print\_int into $v0      syscall        # Make the system call      # Print New Line      li $a0, 10     # Load ASCII value of newline into $a0      li $v0, 11     # Load the system call code for print\_char into $v0      syscall        # Make the system call      # Compute A[B[2]] \* 2 and store in X ($s0)      sll $s0, $t2, 1             # $s0 = A[B[2]] \* 2  exit:      # Print X ($s0)      move $a0, $s0  # Move the value of $s0 to $a0      li $v0, 1      # Load the system call code for print\_int into $v0      syscall        # Make the system call      # Exit point of the program      li $v0, 10     # Load the system call code for exit into $v0      syscall        # Make the system call |

**Question 2: Translate the following high-level code segment into MIPS assembly [3 marks]**

**int t0 = 3;**

**if (t0 < 7)**

**t1 = 7 + t0;**

**else**

**t1 = t0 + t0;**

Assume that:

The variable names in the high-level code are the same registers in MIPS code.

|  |
| --- |
| #Main function for SPIM Execution  main:  addi $t0, $zero, 3  # int t0 = 3;  addi $t1, $zero, 7  # int t1 = 7;  slt $t2, $t0, $t1   # if (t0 < 7)  beq $t2, $zero, Else  add $t1, $t1, $t0   # t1 = 7 + t0;  j Exit  Else:  add $t1, $t0, $t0   # t1 = t0 + t0;  Exit:  # Print $t0  move $a0, $t0       # Move the value of $t0 to $a0  li $v0, 1           # Load immediate value 1 to $v0 for print integer service  syscall             # Perform the system call  # Print newline  li $v0, 11          # Load immediate value 11 to $v0 for print character service  li $a0, 10          # Load ASCII value of newline character to $a0  syscall             # Perform the system call  # Print $t1  move $a0, $t1       # Move the value of $t1 to $a0  li $v0, 1           # Load immediate value 1 to $v0 for print integer service  syscall             # Perform the system call  jr $ra              # Return from main function |