CDA 4205 Computer Architecture

Assignment 2: MIPS Instructions and Assembly Language

1. (4 pts) Bits have no inherent meaning. Given the 32-bit pattern:

**1010 1101 0001 0000 0000 0000 0000 0010**

What does it represent, assuming it is …

1. A 2's complement signed integer?
2. A MIPS instruction?

**Solution**

1. **-1,391,460,350**
2. **Op = 1010112 = 0x2b = sw - store word (I-Type format)**

**rs = 010002 = $8 = $t0**

**rt = 100002 = $16 = $s0**

**immediate16 = 0000 0000 0000 00102 = 2**

**MIPS instruction = sw $s0, 2($t0)**

1. (4 pts) Determine the absolute value of a signed integer. Show the implementation of the following pseudo-instruction using three real instructions:

**Solution**

**addu $t1, $t2, $zero**

**bgez $t2, next**

**subu $t1, $zero, $t2**

**next:**

1. (16 pts) For each pseudo-instruction in the following table, produce a minimal sequence of actual MIPS instructions to accomplish the same thing. You may use the $at for some of the sequences. In the following table, imm32 refers to a 32-bit constant.
2. **move $t1, $t2**
3. **clear $t5**
4. **li $t5, imm32**
5. **addi $t5, $t3, imm32**
6. **beq $t5, imm32, Label**
7. **ble $t5, $t3, Label**
8. **bgt $t5, $t3, Label**
9. **bge $t5, $t3, Label**

**Solution**

|  |  |
| --- | --- |
| **Pseudo-Instruction** | **Solution** |
| **move $t1, $t2** | **addu $t1, $t2, $zero** |
| **clear $t5** | **addu $t5, $zero, $zero**  **or**  **xor $t5, $t5, $t5** |
| **li $t5, imm32** | **lui $t5, upper16**  **ori $t5, $t5, lower16** |
| **addi $t5, $t3, imm32** | **lui $at, upper16**  **ori $at, $at, lower16**  **add $t5, $t3, $at** |
| **beq $t5, imm32, Label** | **lui $at, upper16**  **ori $at, $at, lower16**  **beq $t5, $at, Label** |
| **ble $t5, $t3, Label** | **slt $at, $t3, $t5**  **beq $at, $zero, Label** |
| **bgt $t5, $t3, Label** | **slt $at, $t3, $t5**  **bne $at, $zero, Label** |
| **bge $t5, $t3, Label** | **slt $at, $t5, $t3**  **beq $at, $zero, Label** |

1. (8 pts) Translate the following statements into MIPS assembly language. Assume that a, b, c, and d are allocated in $s0, $s1, $s2, and $s3. All values are signed 32-bit integers.
2. **if ((a > b) || (b > c)) {d = 1;}**

**Solution:**

**bgt $s0, $s1, L1**

**ble $s1, $s2, next**

**L1: ori $s3, $zero, 1**

**next:**

1. **if ((a <= b) && (b > c)) {d = 1;}**

**Solution:**

**bgt $s0, $s1, next**

**ble $s1, $s2, next**

**ori $s3, $zero, 1**

**next:**

1. (8 pts) Consider the following fragment of C code:

**for (i=0; i<=100; i=i+1) { a[i] = b[i] + c; }**

Assume that a and b are arrays of words and the base address of a is in $a0 and the base address of b is in $a1. Register $t0 is associated with variable i and register $s0 with c. Write the code in MIPS.

**Solution:**

**addu $t0, $zero, $zero # i = 0**

**addu $t1, $a0, $zero # $t1 = address a[i]**

**addu $t2, $a1, $zero # $t2 = address b[i]**

**addiu $t3, $zero, 101 # $t3 = 101 (max i)**

**loop: lw $t4, 0($t2) # $t4 = b[i]**

**addu $t5, $t4, $s0 # $t5 = b[i] + c**

**sw $t5, 0($t1) # a[i] = b[i] + c**

**addiu $t0, $t0, 1 # i++**

**addiu $t1, $t1, 4 # address of next a[i]**

**addiu $t2, $t2, 4 # address of next b[i]**

**bne $t0, $t3, loop # exit if (i == 101)**

1. (8 pts) Add comments to the following MIPS code and describe in one sentence what it computes. Assume that $a0 is used for the input and initially contains n, a positive integer. Assume that $v0 is used for the output.

**begin: addi $t0, $zero, 0 # $t0 = sum = 0**

**addi $t1, $zero, 1 # $t1 = i = 1**

**loop: slt $t2, $a0, $t1 # if (n < i)**

**bne $t2, $zero, finish # exit loop if ( i > n )**

**add $t0, $t0, $t1 # sum = sum + i**

**addi $t1, $t1, 2 # i = i + 2**

**j loop # repeat loop**

**finish: add $v0, $t0, $zero # result = sum**

**Solution:**

**Result $v0 is the sum of the odd positive integers 1 + 3 + 5 + … which are less than or equal to n.**

1. (12 pts) The following code fragment processes an array and produces two important values in registers $v0 and $v1. Assume that the array consists of 5000 words indexed 0 through 4999, and its base address is stored in $a0 and its size (5000) in $a1. Describe what this code does. Specifically, what will be returned in $v0 and $v1?

**add $a1, $a1, $a1 # $a1 = 5000 \* 2**

**add $a1, $a1, $a1 # $a1 = 5000 \* 4**

**add $v0, $zero, $zero # $v0 = 0**

**add $t0, $zero, $zero # $t0 = 0**

**outer: add $t4, $a0, $t0 # $t4 = address A[i]**

**lw $t4, 0($t4) # $t4 = A[i]**

**add $t5, $zero, $zero # $t5 = count = 0**

**add $t1, $zero, $zero # $t1 = 0**

**inner: add $t3, $a0, $t1 # $t3 = address A[j]**

**lw $t3, 0($t3) # $t3 = A[j]**

**bne $t3, $t4, skip # if (A[i]!=A[j]) skip**

**addi $t5, $t5, 1 # count++**

**skip: addi $t1, $t1, 4 # j = j+4**

**bne $t1, $a1, inner # inner loop = 5000**

**slt $t2, $t5, $v0 # if (count < $v0)**

**bne $t2, $zero, next # then goto next**

**add $v0, $t5, $zero # $v0 = count**

**add $v1, $t4, $zero # $v1 = A[i]**

**next: addi $t0, $t0, 4 # i = i+4**

**bne $t0, $a1, outer # outer loop = 5000**

**Solution:**

**This code compares every element in the array against all elements for identical matches. It counts the frequency of occurrence of each value in the array. The count of the most frequently used value is returned in $v0 and the value itself is returned in $v1.**

* **Submission Requirements**
* Your solutions must be in a single file with a file name yourname-hw1.
* If scanned from hand-written copies, then the writing must be legible, or loss of credits may occur.
* Only submissions via the link on Canvas where this description is downloaded are graded. Submissions to any other locations on Canvas will be ignored.
* Late submissions are accepted for a maximum of 3 late days with 20% assignment credit off as late penalization. Assignments submitted after 3 late days will not be accepted.