## **Computer Sciences Department**

## **University of Wisconsin-Madison**

## **CS/ECE 552 – Introduction to Computer Architecture**

## **In-Class Exercise (01/30)**

**Answers to all questions should be uploaded on Canvas.**

1. From the textbook: [4 points]

(a) [3 points] (Check Yourself 2.10)

I. What is the range of addresses for conditional branches in MIPS (K = 1024)? [1 point]

1. Addresses between 0 and 64K-1

2. Addresses between 0 and 256K-1

3. Addresses up to about 32K before the branch to about 32K after

4. Addresses up to about 128K before the branch to about 128K after

**1**

II. [1 point] What is the range of addresses for jump and JAL (jump and link) in MIPS (K = 1024)?

1. Addresses between 0 and 64M – 1
2. Addresses between 0 and 256M – 1
3. Addresses up to about 32M before the branch to about 32M after
4. Addresses up to about 128M before the branch to about 128M after
5. Anywhere within a block of 64M addresses where the PC supplies the upper 6 bits
6. Anywhere within a block of 256M addresses where the PC supplies the upper 4 bits

6

III. What is the MIPS assembly language instruction corresponding to the machine instruction with the value 0x0000 0000? [1 point]

1. j
2. R-format
3. addi
4. sll
5. mfc0
6. Undefined opcode: there is no legal instruction that corresponds to 0

**4**

(b) [1 point] (Challenge problem) For the machine instruction from (a) III, if there is a valid opcode, give the full MIPS instruction (e.g., add $s0, $s1, $s2). Are there any problems with this instruction? If there is no valid opcode, what would the processor do when it encounters this instruction?

**0000001000010001100020000020000**

2. [2 points] Consider the following MIPS assembly code:

Loop: sll $t1, $s3, 2

add $t1, $t1, $s6

lw $t0, 0($t1)

bne $t0, $s5, Exit

addi $s3, $s3, 1

j Loop

Exit:

1. [1 point] Write down the machine code of these assembly instructions:

|  |  |
| --- | --- |
| **Assembly Program Instruction** | **Machine code (in hexadecimal)** |
| add $t1, $t1, $s6 | 0x01364820 |
| addi $s3, $s3, 1 | 0x12730001 |
| lw $t0, 0($t1) | 0x8D280000 |
| bne $t0, $s5, Exit | 0x16A80003 |

1. (b) [1 point] The above assembly code is compiled from the following C statement:

**while (A[i] == k) i++;**

Which registers contain:

* Integer variable i? $s3
* Integer variable k? $s5
* Base address of integer array A (&A[0])? $s6

3. [2 points] Rewrite the following MIPS assembly code such that it produces the same output (*$s2*) but executes less instructions:

addi $t3, $s3, 4

add $t3, $s5, $t3

lbu $t2, 0($t3)

add $s2, $s4, $t2

add $t3, Ts5, $s3

lbu $t2, 4($t3)

add $s2, $s4, $t2

4. [2 points] You are assigned a new role as a verification engineer at your company that uses MIPS for their assembly language. Your first task is to verify that this MIPS assembly code:

sub $s0, $zero, $s0

addi $s0, $s0, -4

add $s0, $s0, $s1

Is being correctly converted to the machine language code. Currently, your assembler emits the following for the above assembly code:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Op** | **Rs** | **Rt** | **Rd** | **Address/shamt** | **Funct** |
| 000000 | 00000 | 10000 | 10000 | 00000 | 100010 |
| 100011 | 10000 | 10000 | 0000 1111 1111 1100 | | |
| 000000 | 10000 | 10001 | 10000 | 00000 | 100000 |

Given your prior experience with MIPS from CS/ECE 552, you spot at least one mistake in the above translation!

1. [1 point] What is the corresponding MIPS assembly code for the buggy machine language code?

**00000000000100001000000000010110**

**10001110000100001111111111111100**

**00000010000100011000000000000000**

1. [1 point] What is the correct machine language code for the original MIPS assembly code in this problem?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Op** | **Rs** | **Rt** | **Rd** | **Address/shamt** | **Funct** |
| 000000 | 00000 | 10000 | 10000 | 00000 | 010110 |
| 100011 | 10000 | 10000 | 1111111111111100 | | |
| 000000 | 10000 | 10001 | 10000 | 00000 | 000000 |