Written Assignment 1

CS 281 Section B - Professor Katsinis

Joseph Hines (jth95)

April 3, 2019

# Part A

## Q2.1

Given that f, g, and h are in stores $s0, $s1, and $s2 respectively.

addi $t0, $zero, 5 # store the value 5 in t0  
sub $t1, $s2, $t0 # store h - 5 in t1  
add $s0, $s1, $t1 # store g + (h - 5) in s0 (f)

## Q2.3

Assuming that each element in the array is 4 bytes in size

sub $t0, $s3, $s4 # store i - j in t0  
sll $t1, $t0, 2 # calculate offset 4 \* (i - j) and store it in t1  
add $t1, $t1, $s6 # address of A[i - j], which is (4 \* (i - j)) + A  
lw $t2, 0($t1) # store value A[i - j] in t2  
sw $t2, 32($s7) # store A[i -j] in B[8]

## Q2.11

Referencing *Figure 2.14* and *Figure 2.19*

Some operations have the same opcode, and are differentiated by their **funct** block. Depending on the operation type I vs. R, the structure of the 32 bits is different. Each register is associated with a decimal and therefore binary value. I first look up the value of each token in one of the aforementioned tables and write it into the decimal table. I then convert each to binary and make sure they each use the appropriate amount of bits, depending on the type.

### addi $t0, $s6, 4

*Decimal*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OP | RS | RT | I | RD |
| 8 | 22 | 8 | 4 | N/A |

*Binary*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OP | RS | RT | I | RD |
| 001000 | 10110 | 01000 | 0000 0000 0000 0100 | N/A |

### add $t1, $s6, $0

*Decimal*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OP | RS | RT | I | RD |
| 0 | 22 | 0 | N/A | 9 |

*Binary*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OP | RS | RT | I | RD |
| 000000 | 10110 | 00000 | N/A | 01001 |

### sw $t1, 0($t0)

*Decimal*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OP | RS | RT | I | RD |
| 43 | 8 | 9 | N/A | N/A |

*Binary*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OP | RS | RT | I | RD |
| 101011 | 01000 | 01001 | N/A | N/A |

### lw $t0, 0($t0)

*Decimal*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OP | RS | RT | I | RD |
| 35 | 8 | 8 | N/A | N/A |

*Binary*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OP | RS | RT | I | RD |
| 100011 | 01000 | 01000 | N/A | N/A |

### add $s0, $t1, $t0

*Decimal*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OP | RS | RT | I | RD |
| 0 | 9 | 8 | N/A | 16 |

*Binary*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OP | RS | RT | I | RD |
| 000000 | 01001 | 01000 | N/A | 10000 |

## Q2.15

Referencing *Figure 2.4* and the tables underneath the heading *MIPS Fields* in section 2.5.

As with above, I tokenized the operation and converted from decimal to binary. From binary, I used the referenced table to do the hexadecimal conversion (4 bits of binary per hex character).

sw $t1, 32($t2)

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Decimal | Binary | Hexadecimal |
| I | 43 10 9 32 | 1010 1101 0100 1001 0000 0000 0010 0000 | ad49 0020 |

## Q2.39

Referencing the first example in 2.10 and *Figure 2.17*

Our desired constant is :

0010 0000 0000 0001 0100 1001 0010 0100

First we must convert the “top” or first 16 bits to decimal.

|  |  |
| --- | --- |
| Binary | Decimal |
| 0010 0000 0000 0001 | 2^13 + 2^0 = 8192 + 1 = **8193** |

Now we can must convert the “bottom” or last 16 bits as well.

|  |  |
| --- | --- |
| Binary | Decimal |
| 0100 1001 0010 0100 | 2^14 + 2^11 + 2^8 + 2^5 + 2^2 = 16384 + 2048 + 256 + 32 + 4 = **18724** |

Now that we have the values, we can write the MIPS code.

lui $t0, 8193 # set the upper part of $t0 to 8193  
 # t0 = 0010 0000 0000 0001 0000 0000 0000 0000  
ori $t0, $t0, 18724 # set the lower part to 18734  
 # t0 = 0010 0000 0000 0001 0100 1001 0010 0100