EN605.204.82.SU17 Computer Organization

Caleb Bowers

Assignment 5a

2.20

To find the shortest sequence of MIPS instructions that extract bits 16 down to 11 from register `$t0` and uses the value of this field to replace bits 31 down to 26 in register `$t1` we need to first isolate bits 16 down to 11 in register `$t0`:

```
srl $t2, $t0, 11 #shift $t0 right 11 bringing byte 11 to byte 0
sll $t2, $t2, 26 #shift $t2 left 26, placing orig. bits in 31-26
```

We then need to open the 31-26 bit spot in register `$t1`:

```
srl $t1, $t1, 6 # Zero out 31-26 on $t1
```

Then we add in register `$t2`

```
add $t2, $t2, $t1 #combine regs, placing orig. 16-11 into 31-26
```

So, altogether our operations are:

```
srl $t2, $t0, 11 #shift $t0 right 11 bringing byte 11 to byte 0
sll $t2, $t2, 26 #shift $t2 left 26, placing orig. bits in 31-26
srl $t1, $t1, 6 # Zero out 31-26 on $t1
add $t2, $t2, $t1 #combine regs, placing orig. 16-11 into 31-26
```

2.25

The following instruction is not included in the MIPS instruction set:

```
rpt $t2, loop # if (R[rs]>0) R[rs]=R[rs]-1, PC=PC+4+BranchAddr
```

2.25.1

If we were to implement this instruction into MIPS it would be in an I-Type instruction format where the immediate field is populated with the address to which the command is branching.

This command is very similar to the BNE command, where it tests the values of two registers and branches if they are not equal. The rpt command tests register `$t2` and `$0` and branches if not equal, so it would have the same format as BNE: I-Type.

2.25.2

What is the shortest sequence of MIPS instructions that perform this operation?

```
loop: beq $t2, $0, Exit # exit loops if $t2==$zero
      sub $t2, $t2, 1
      j loop
```

2.26

Consider the following MIPS loop:

```
LOOP: slt $t2, $0, $t1
      beq $t2, $0, DONE
      sub $t1, $t1, 1
      addi $s2, $s2, 2
      j loop
DONE
```

2.26.1

Assuming `$t1` is initialized to the value of 10, the resulting value in `$s2` is 20. After one iteration `$s2` has the value 2 since it was initialized to 0. The register `$t1` has the value of 9. So, 10 iterations occur before the statement

```
slt $t2, $0, $t1
```

assigns `$t2` a value of 0 and the loop exits. Therefore, 2 * 10 = 20.

2.26.2

The equivalent c code for the loop above is:

```
while (i > 0)
{
     i = i - 1; // Decrement i by one each iteration.
     B = B + 2; // Increment B by two on each iteration.
}
```

2.26.3

If the loop initially sets register `$t1` equal to `N` then the loop will iterate `5*N+2` times. If register `$t1` is equal to zero or lower, then two instructions are executed. This is the base case. If `$t1` is greater than 0, then five instructions are executed. No matter what the base case will be performed and whatever value is assigned to `$t1` that number of iterations will occur, so the resulting number of instructions executed is `5*N+2`.

For the following pseudoinstructions what is the actual MIPS code:

```
abs $s4, $s1 #pseudoinstruction, absolute value
```

Actual code:

```
     bgez $s1, ABS # If $s1 is greater than or eq to 0, goto ABS
     nor  $s4, $s1, $0 # Invert neg bits since for pos 2's comp
     addi $s4, $s4, 1 # add one to get full 2's comp
ABS: add $s4, $s1, $0 # Store $s1 in $s4 since positive
```

Pseudoinstruction:

```
rol $t7, $t3 8  #rotate left
```

Actual code:

```
sll $t0, $t3, 8  #shift 8 bits left to get new lead bit
srl $t1, $t3, 24 #shift 24 bits right to get new trailing bits
or $t7, $t0, $t1 #combine registers to get newly rotated reg.
```

Pseudoinstruction:

```
ld $t5, 0($t8)
```

Actual code:

```
lw $t5, 0($t8)     # Load beginning of $t8 into %t5
lui $t1, $0        # Load the upper imm of $t1 with 0
addu $t1, $t1, $t8 # add $t8 to $t1
lw $t6, 0($t1)     # Load %t6 with beginning of $t1
```