CDA 3101 Assignment 3 Solution

Due in class on Friday, Sept. 24

Note all question numbers are from 4th edition of the textbook.

1. (2 points) Problem 2.6.4b from the textbook. Assume A and B are integer arrays.
   \[ s6 = s6 - 20; \]
   \[ s6 = s6 + g; \]
   so the offset from the base address now is \( g-20+8 \), which is \( g/4-3 \) in the index.
   \[ so f = A[g/4 – 3]; \]

2. (4 points) Problem 2.17.1 from the textbook. For each instruction, provide how it can
   be implemented using other MIPS basic instructions
   \[
   \begin{align*}
   \text{abs} & \quad \text{slt} \quad \text{bne} \\
   & \quad \text{add} \quad \text{NEGATION} \\
   & \quad \text{sub} \quad \text{NEGATION:}
   \end{align*}
   \]
   \[ \text{slt} \quad \text{sgt} \]
   All of these instructions are either supported by an existing instruction, or
   sequence of existing instructions.

3. (4 points) Problem 2.17.4 from the textbook.
   a. 20
   b. 200

4. (6 points) Problem 2.18.2 from the textbook.
   a.
   \[
   \begin{align*}
   \text{addi} & \quad \text{add} \quad \text{add} \\
   \text{LOOP:} & \quad \text{TEST:} \\
   & \quad \text{slti} \quad \text{slti} \quad \text{bne} \\
   & \quad \text{addi} \quad \text{addi} \\
   & \quad \text{addi} \quad \text{bne} \\
   & \quad \text{addi} \quad \text{addi} \\
   & \quad \text{addi} \quad \text{addi}
   \end{align*}
   \]
b. 
```
addi $s0, $s0, 10
```
```
LOOP:  
    slti $t2, $s0, 10
    beq $t2, $0, DONE
    add $t3, $s1, $s0
    sll $t2, $s0, 2
    add $t2, $s2, $t2
    sw $t3, ($t2)
    addi $s0, $s0, 1
    j LOOP
```
```
DONE:
```

5. (4 points) Problem 2.18.4 from the textbook.
   a. 501
   b. 301

6. (6 points) Problem 2.18.5 from the textbook.
   a. 
   ```
   for(i=100; i>0; i--){
       result += MemArray[s0];
       s0 += 1;
   }
   ```
   b. 
   ```
   for(i=0; i<100; i+=2){
       result += MemArray[s0 + i];
       result += MemArray[s0 + i + 1];
   }
   ```

7. (4 points) Problem 2.18.6 from the textbook.
   a. 
   ```
   addi $t1, $s0, 400
   LOOP:  
       lw $s1, 0($s0)
       add $s2, $s2, $s1
       add $s0, $s0, 4
       bne $s0, $t1, LOOP
   ```
   b. Already reduced to minimum instructions

8. (7 points) For each of the pseudo instructions below, produce a minimal sequence of actual MIPS instructions to accomplish the same thing. In the following table, big refers to a specific number that requires 32 bits to represent and small refers to a number that can fit in 16 bits.
   a. beq $t1, small, L # if($t1 == small) go to L
       ```
       add $at, $zero, small
       beq $t1, $at, L
       ```
b. beq $t1, big, L # if($t1 == big) go to L
   lui $at, upper(big)
   ori $at, $at, lower(big)
   beq $t2, $at, L

c. addi $t0, $t2, big # $t0=$t2+big
   lui $at, upper(big)
   ori $at, $at, lower(big)
   add $t0, $t2, $at

d. lw $t5, big($t2) # $t5 = Mem[$t2+big]
   lui $at, upper(big)
   ori $at, $at, lower(big)
   add $at, $at, $t2
   lw $t5, 0($at)

This question investigates the format of instructions.

All these instructions take 32-bit registers as input, so the first step is to put these immediate small and big into a 32-bit register in a proper way. Notice that addi and lw only have 16-bit immediate field, so big will not be accepted as an immediate for these two instructions.

We have split big into two parts, each with 16 bit and then combine them together into a 32-bit register.

9. Consider the MIPS assembly program given below. Refer to Appendix A, section 10 for assembler directives like .data, .text etc used in this program.

.data
plaintext: .asciiz "This is fun"

.text
.globl main
main:  la $t1, plaintext
      # loads address of
      # the string labeled
      # by plaintext in $t1

loop:  lb $t2, 0($t1)
      beq $t2, $zero, endfor
      addiu $t2, $t2, 3
      sb $t2, 0($t1)
      addiu $t1, $t1, 1
      j loop
endfor: li $v0, 4 #loads 4 in $v0
         la $a0, plaintext
         syscall
         jr $ra

a. (3 points) Write the equivalent C code.
char plaintext[]="This is fun";
int main() {
    int counter=0;
    while (plaintext[counter] != 0){
        plaintext[counter++] += 3;
b. (4 points) Hand simulate the first 3 iterations of the loop (starting at the loop label) as we did in class, tracking each change to a register as it happens. Write all the values in hexadecimal. The first iteration has been written down for you in a table on next page (and is colored gray), along with the instructions for all 3 loop iterations. You should complete this table by showing any change to either a register or the first 3 characters of plaintext. You only have to write a value in a row/column when it changes.

c. (2 points) For the last column of the table, you should reference the ASCII table on the green insert at the front of your book (or any other ASCII table you have handy) to convert from the hex numbers to the ASCII text. Make sure to show both as done in the first line of the table on the next page!

Remember that all the numbers are in hex, for example, t1 are all in hex.

d. (2 points) Run the program in SPIM and watch each byte of memory change as you step through each loop iteration. What are the contents of plaintext (in hex) as shown in SPIM data window after the program completes execution?

- `0x766c6b57 0x23766c23 0x00717869`
- `0x576b6c76 0x236c7623 0x6978`

e. (2 points) What will the program change the following string to.

Qefp*fp*fjmibjbqfkd*`^bp^o*`fmebo

This is implementation caesar cipher
<table>
<thead>
<tr>
<th>$t1$</th>
<th>$t2$</th>
<th>First 3 bytes of plaintext</th>
</tr>
</thead>
<tbody>
<tr>
<td>10010000</td>
<td>-</td>
<td>54 68 69 T h i</td>
</tr>
<tr>
<td>lb $t2$, 0($t1$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>beq $t2$, $zero$, endfor</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>Addiu $t2$, $t2$, 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sb $t2$, 0($t1$)</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>Addiu $t1$, $t1$, 1</td>
<td>10010001</td>
<td></td>
</tr>
<tr>
<td>j loop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lb $t2$, 0($t1$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>beq $t2$, $zero$, endfor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Addiu $t2$, $t2$, 3</td>
<td>6b</td>
<td></td>
</tr>
<tr>
<td>sb $t2$, 0($t1$)</td>
<td>57 6b 69 W h i</td>
<td></td>
</tr>
<tr>
<td>Addiu $t1$, $t1$, 1</td>
<td>10010002</td>
<td></td>
</tr>
<tr>
<td>j loop</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lb $t2$, 0($t1$)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>beq $t2$, $zero$, endfor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Addiu $t2$, $t2$, 3</td>
<td>6c</td>
<td></td>
</tr>
<tr>
<td>sb $t2$, 0($t1$)</td>
<td>57 6b 6c W k l</td>
<td></td>
</tr>
<tr>
<td>Addiu $t1$, $t1$, 1</td>
<td>10010003</td>
<td></td>
</tr>
<tr>
<td>j loop</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>