CSC 256 - Machine Structures
Project 6

Assigned : April 12th, 2017
Due : April 22nd, 2017 @ midnight
Total Points: 100 Points

Description For project six, your objective is to convert the given C++ code into MIPS assembly. Please do not modify the C++ code itself. You are only allowed to make modifications to the assembly file. Start writing your code below the sumOfDoubleEvenPlace: label.

When doing a C++ to MIPS conversion with functions, it can be done in the following steps:

1 Assign variables to registers. When inspecting code, any constant values in expressions may need to be assigned to temporary registers.

2 Initialize variables to registers. (actually put the values into the registers.)

3 Then move onto the rest of the code.

4 For functions, remember for non-leaf functions(functions that call other functions, you must saved the values of the $s? registers you are using. You will save these values on the stack. Pushing values is done via storeword(sw) and popping values is done via loadword(lw).

5 Remember that $t, $v, $a, and $ra registers are not preserved across function calls.

6 YOU ONLY NEED TO Implement the function sumOfDoubleEvenPlace. DO NOT MODIFY MAIN OR GETDIGIT. Doing this may result it incorrect values. If you find or beleive there is a mistake in the base code please do not hesitate to email me or ask in class.

Expected Output:

```
Expected Value: 23 Value: 23
Expected Value: 21 Value: 21
```

Submission
When you have completed the assignment please upload your .s file to ilearn. PLEASE DO NOT UPLOAD ANY OTHER TYPE OF FILE.

Base MIPS Code

```
.data
  expVal23:     .asciiz  "Expected Value : 23  Your Value : "
  expVal21:     .asciiz  "Expected Value : 21  Your Value : "
  endl:         .asciiz  "\n"

.text

##
# int getDigit(int number);
# number --> $t0
# result of modulo --> $t1
# result of div --> $t2
# 10 --> $t3
# sum  --> $v0
getDigit:
    add $t0, $a0, $0 #make a copy of arg0
    li $v0 , 0
    li $t3, 10
    bge $t0, $t3, else
    add $v0, $t0, $0
    j func_return
  else:
    rem $t1, $t0, $t3
    div $t2, $t0, $t3
    add $v0, $t1, $t2
  func_return:
    jr $ra


##
# int sumOfDoubleEvenPlace(int number);
# List Used Registers Here:
##
sumOfDoubleEvenPlace:

##
# test1   --> s0
# test2   --> s1
# result1 --> s2
# result2 --> s3
##
main:
  li $s0, 89744563  # int test1 = 89744563;
  li $s1, 98756421  # int test2 = 98756421;
  li $s2, 0         # int result1 = 0;
  li $s3, 0         # int result2 = 0;


  # code for first function call

```

```
51  add $a0, $0, $s0
52  jal sumOfDoubleEvenPlace
53  add $s2, $0, $v0
54
55  la   $a0, expVal23
56  addi $v0, $0, 4
57  syscall
58
59  move $a0, $s2
60  addi $v0, $0, 1
61  syscall
62
63  la   $a0, endl
64  addi $v0, $0, 4
65  syscall
66
67   # code for first function call
68
69  add $a0, $0, $s1
70  jal sumOfDoubleEvenPlace
71  add $s3, $0, $v0
72
73  la   $a0, expVal21
74  addi $v0, $0, 4
75  syscall
76
77  move $a0, $s3
78  addi $v0, $0, 1
79  syscall
80
81  la   $a0, endl
82  addi $v0, $0, 4
83  syscall
84
85  li $v0, 10
86  syscall
```

C++ Equivalent

```cpp
#include <bits/stdc++.h>
using namespace std;

int sumOfDoubleEvenPlace(int number);
int getDigit(int number);

int main(void)
{
    int test1 = 89744563;
    int test2 = 98756421;
    int result1 = 0;
    int result2 = 0;

    result1 = sumOfDoubleEvenPlace(test1);
    cout << "Expected Value: 23  Value: " << result1 << endl;

    result2 = sumOfDoubleEvenPlace(test2);
    cout << "Expected Value: 21  Value: " << result2 << endl;

}
/*
* Function returns the sum of the even placed
* digits(after being doubled) starting from the left.
* Note that the algorithm starts counting from 1 not 0. Therefore,
* given the number 1234, 4 is the first digit from the left.
* So the even placed digits are 3 and 1 and the odd place digits are
* 4 and 2 from the left.
*/
int sumOfDoubleEvenPlace(int number) {
    int sum = 0;
    int digit;

    //Remove first odd digit
    number = number / 10;

    while (number > 0) {
        //Grab even placed digit
        digit = (number % 10);
        //Double the digit and pass it to getDigit,
        //Add result to sum
        sum += getDigit(digit*2);
        //Remove current even digit and the next odd digit.
        number = number/100;
    }
    return sum;
}

/* getDigit returns the sum of the digits in
 * a 1 or 2 digit number.
 * if number is < 10, a single digit number we
```

```
51  * we return the number.
52  * else we return the sum of the digits in the 2 digit
53  * number.
54  * For example:
55  * 1 would return 1
56  * 11 would return 2
57  * 18 would return 9
58  */
59 int getDigit(int number) {
60     int sum = 0;
61     if (number < 10) {
62         sum = number;
63     } else {
64         sum = number%10 + number/10;
65     }
66     return sum;
67 }
```