Hacettepe University
Department of Computer Engineering
BBM234 Computer Organization
Spring 2022
MIPS Project

Deadline : Sunday, 27/03/2022 (at 23:59)

Purpose: In this project, you will learn how to write and simulate MIPS code. You will also experience how to handle multiple function calls. You will understand how stack is used.

MIPS Simulator: You must use MARS MIPS simulator. You can download the simulator from the following link:
http://courses.missouristate.edu/kenvollmar/mars/

There is also good tutorial that helps installation process and explains how to use MARS: https://www.youtube.com/watch?v=jzfzvmnFbdw

Important note about automatic grading: Your submissions will be graded automatically using command line execution. Therefore, you must make sure you understand how to use MARS from a terminal so that you can test if your code is producing the expected outputs. Furthermore, you must follow the instructions to the letter regarding which registers or memory locations should hold the results of your programs once they finish executing. **Failing to read the input numbers as user inputs or failing to have the results placed in the specified locations will result in grade of 0 points**.

Using MARS from a command line:
http://courses.missouristate.edu/kenvollmar/mars/Help/Help_4_1/MarsHelpCommand.html

Instructions you are allowed to use: You can only use the instructions given below. We will visually inspect your code. If you use instructions other than the ones below, you will lose 5 points for each of them.

Instruction set: **add, addi, sub, beq, bne, slt, j, jal, jr, sll, lw, sw.**

QUESTIONS:

1) Arrays using for loops

Assume you have an array with 10 elements (e.g., A[10]) and an integer value *K*. Write a MIPS code that returns the total number of elements that are less than *K*. **The integer value *K* must be read as the user input**. You can assume that the array **A** will always be the same, so you can initialize (define) it in your program's `.data` part as shown in the example below. **The result must be returned in register *$v1***.

Example: Assume A={2, 3, 1, 4, 8, 20, 30, 7, 9, 15}.

- **For *K*=5**, your program should return **4** since we have four numbers (2, 3, 1, and 4) that are less than 5.
- **For *K*=50**, you program should return **10** since all ten numbers are less than 50.

How to define arrays, load array address and how to read user input:

You can define your array, load the address of A[0] and read the user input as shown below. The part that you should write your code is also shown below.

```
.data
A: .word 2, 3, 1, 4, 8, 20, 30, 7, 9, 15 # Initialize the array A in memory

.text
.globl main

main:
        # Load the address of A[0] to $t1
        la $t1, A

        # Getting user integer input K into register v0
        li $v0, 5
        syscall

        # Moving the integer input to another register: $t0 <- K
        move $t0, $v0

        # Your code goes here…
        # Don't forget that the final result must be in $v1 before exit

        # Exit from the simulator function
        li $v0, 10
        syscall
```

How your program will be executed from the terminal (assume the input file `<inputfile>` will have one integer K stored in the first line):

```
      cat <inputfile> | java -jar Mars4_5.jar v1 dec array.asm
```

The expected terminal output for K = 5:

```
      MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar


      $v1      4
```

2) Function calls and stacks:

Write a MIPS code for the following C code fragment. In your code, **you are not allowed to use multiplication instructions (mult or mul)**. Note that you must use stack to keep *$ra* register values when you have multiple function calls. You MUST use *jal* instruction for function calls. **Integers *a* and *b* must be read as the user inputs**. **The result must be returned in register *$v1***.

```
int main() {
        int a, b;
        scanf("%d", &a);
        scanf("%d", &b);
        int result = 0;

        if(a == b)
                result = 2*(a + b);
        else
                result = compare(a, b);
        return result;
}

int compare(int a, int b){
        if(a<b)
                return punish(a, b);
        else
                return award(a, b);
}

int punish(int a, int b){
        return (a-3*b);
}

int award(int a, int b){
        return (3*a+b);
}
```

Starter code below:

```
.text
.globl main

main:
        # Getting user integer input a into register v0
        li $v0, 5
        syscall
        # Moving the integer input to another register: $t0 <- a
        move $t0, $v0
        # Getting the second user integer input a into register v0
        li $v0, 5
        syscall
        # Moving the integer input to another register: $t1 <- b
        move $t1, $v0

        # Your code goes here…
        # Don't forget that the final result must be in $v1 before exit

        # Exit from the simulator function
        li $v0, 10
        syscall
```

How your program will be executed from the terminal (assume the input file `<inputfile>` will have two integers corresponding to the input numbers ***a*** and ***b*** respectively, stored in the first two lines, each on a separate line):

```
cat <inputfile> | java -jar Mars4_5.jar v1 dec function.asm
```

A sample expected terminal output for a = 3, b = 13:

```
MARS 4.5  Copyright 2003-2014 Pete Sanderson and Kenneth Vollmar


$v1     -36
```

What to Turn In:

Please turn the following items through **submission system**:

1. Assembly code files in .asm file format (array.asm, function.asm) in a single zip file. **You must add your name and student id as a comment at the beginning of each file**. You should make a single zip file since the submit system accepts only zip files.

   Accepted zip file naming format:

   - **b***studentid***.zip**
     - array.asm
     - function.asm

Plagiarism Control Notice:

Students must implement their solutions individually. All submissions will be submitted to a plagiarism check. Any submissions that show a high level of similarity will be reported as plagiarism attempts to the ethics committee.