Computer Organization, Spring 2017

Lab 1: MIPS Programming

Due Date : 2017/3/9

1. Goal：

In Lab 1, students will learn how to write MIPS code, and know the difference between assembly and high-level languages. In order to test the correctness of program, students should use a MIPS simulator – SPIM to simulate the programs.

2. Attached Files：

- "factorial.c"：C code for "factorial", modified from the example given in textbook.
- "factorial.s"：MIPS code for "factorial", modified from the example given in textbook.
- "bubble_sort.c": C code for "bubble sort", modified from the example given in textbook.
- "pascal.c"：C code for "pascal triangle".
- "gcd.c"：C code for "greatest common divisor".
- "Tutorial of MIPS Simulator-SPIM.ppt"：The instruction of MIPS simulator SPIM

3. Format of MIPS Executable Files:

A. The basic structure of a MIPS program:

.text **.globl main** **main:** **…**	 **#starting point of the main program**
.data **name: .data_type data** **…**	 **#name, type, and value of a datum**
.text **label:**	 **#starting point of a procedure**

B. Data types:

1. .word: 4-byte integer
   Example 1: int1: .word 5 #declare and set an integer variable
   Example 2: array1: .word 1, 3, 9, 7 #declare and set an integer array with 4 elements
2. .half: 2-byte integer
3. .float: single-precision floating-point number
4. .double: double-precision floating-point number
5. .ascii: string
   Example: String1: .ascii "print string \n" #(\n) newline, (\t) tab, ( ) space, …
6. .asciiz: string end with NULL

C. Invoking of system calls:

- Steps of invoking a system call:
  i. Load system call service code into register $v0.
  ii. Load arguments into registers $a0~$a3, if necessary.
  iii. Invoke system call "*syscall*".
  iv. Return value in register $v0, if necessary.

- System call services:

Service	Code in $v0	Arguments	Results
print_int	1	$a0 = integer to be printed	
print_float	2	$f12 = float to be printed	
print_double	3	$f12 = double to be printed	
print_string	4	$a0 = address of string in memory	
read_int	5		integer returned in $v0
read_float	6		float returned in $v0
read_double	7		double returned in $v0
read_string	8	$a0 = memory address of string input buffer $a1 = read length of string buffer	
sbrk	9	$a0 = amount	address in $v0

exit	10	

- print_int : print an integer on the console interface (similar to *printf* in C)
- read_int: read an integer from the console interface (similar to *scanf* in C)
- exit: finish the execution of the program

D. Example of using system call:

```
# print a string on the console interface
li $v0, 4            # set service code (print_string service) into $v0
la $a0, string       # load the address of the string to be printed into $a0
syscall              # print the string
```

E. Supplementary instructions:

Division may be used in this lab. It is an example about division and remainder bellowed.

```
div $t1, t2      #t1 / t2
mflo $t3         #copy quotient to $t3
mfhi $t4         #copy remainder to $t4
```

F. Detailed information:

Connect to http://logos.cs.uic.edu/366/notes/mips%20quick%20tutorial.htm for detailed information.

4. Lab Description:

This lab can be divided into four parts. In Part A, a simple example “factorial.s” written in MIPS language (also given the corresponding C code “factorial.c”) is given for the practice of the MIPS simulator, SPIM.

In part B, part C, and part D “bubble_sort.c”, “pascal.c”, and “gcd.c” written in C are given, respectively. You have to write the corresponding MIPS codes according to these programs.

A. Factorial： (0%) (Example of Lab1)

The attached files factorial.c and factorial.s are modified from the example given in textbook for computing n!. This part is an example of hw1, which make you familiar to SPIM.

B. Bubble Sort： (30%)

1. According to “bubble_sort.c”, please write the corresponding MIPS program, named “bubble_sort.s”.
2. There are two procedures, swap and sort, in “bubble_sort.c”. Refer to the textbook for the detailed description of these two procedures.
3. Please declare and set the array before sorting in the .data of the program directly.
4. Output: Print the message shown below on the console interface of the simulator.

```
Console
The array before sort :
5 3 6 7 31 23 43 12 45 1
The array after  sort :
1 3 5 6 7 12 23 31 43 45
```

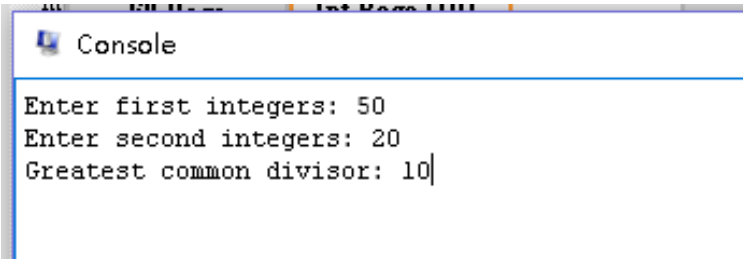
C. Pascal triangle： (40%)

1. According to “pascal.c”, please write the corresponding MIPS program, named “pascal.s”. When testing, the levels of the pascal triangle will be limited to 1 to 30.
2. Input : Please input the number of levels of the pascal triangle on the console interface.
3. Output : Print the message shown below on the console interface of the simulator.

```
Console
Pascal Triangle
Please enter the number of levels(1~30): 10
 1
 1 1
 1 2 1
 1 3 3 1
 1 4 6 4 1
 1 5 10 10 5 1
 1 6 15 20 15 6 1
 1 7 21 35 35 21 7 1
 1 8 28 56 70 56 28 8 1
 1 9 36 84 126 126 84 36 9 1
```

D. Greatest common divisor： (30%)

1. According to “gcd.c”, write the corresponding MIPS program, named “gcd.s”.
2. Input : Please input two integers on the console interface.
3. Output : Print the message shown below on the console interface of the simulator.



5. Deadline:

A. One person per group for this lab. Please upload your files onto E3 (eCampus) platform.

B. The files you should hand in include:
   1. bubble_sort.s
   2. pascal.s
   3. gcd.s

   Please compress these files into one zip file, and name your zip file as “Lab1_ID.zip” (only zip file can be accepted).

C. Deadline：2017/3/9 23:59.

D. Any assignment work by fraud will get a zero point.