國立清華大學資訊工程學系

CS 4100 -- 計算機結構

103 學年度下學期 Homework #2

- There are two parts in this homework.

1. (Load, Store, Add, Sub)
   Please load the data 0($gp) as A, 4($gp) as B, and do the following calculations.
   C = A + B, store C to 8($gp)
   D = A – B, store D to 12($gp)
   P.s. just fill up the ####### block in the asm file

The “Run I/O” will show like this:

```
This it the first part of output. (Load,Store,Add,Sub)

A+B   A-B
60
30
```

2. (Branch Loop, System call)
   Please convert C code to assembly code. Write a new assembly file for this part.
   Must have: one input (positive int)

```
#++++++++          C       code          ++++++++++
# if ($t0<=0 || $t0>10)
#     printf("$t0 is %d\n", $t0);
# else
#     for (i=0; i<$t0; i++)
#           printf("***%d***\n", i );
#++++++++++++++++++++++++++++++++++++++++++++
```

The “Run I/O” may show like this: (input : 15)

```
Please enter a int:15
$t0 is 15
```

The “Run I/O” may show like this: (input : 5)

```
Please enter a int:5
***0***
***1***
***2***
***3***
***4***
```

- Submission (two assembly programs)

Please name your assembly program with your student ID, for example:
"hw2_p1_100000001.asm" & "hw2_p2_100000001.asm".
Use the iLMS (http://lms.nthu.edu.tw/) to submit your program.

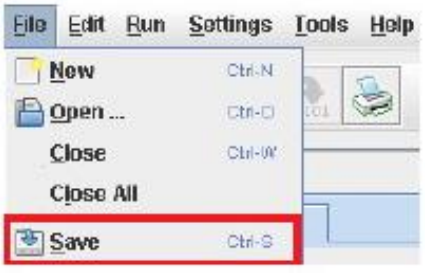
- Grading Criteria

Correctness: 80%
Comment in program: 10%
Output format: 10%

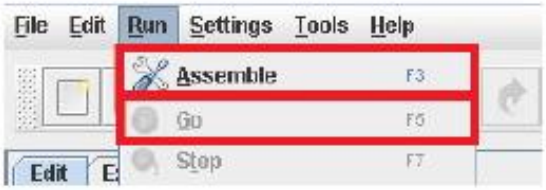
- MARS (MIPS Assembler and Runtime Simulator)

1. MARS can assemble and simulate the execution of MIPS assembly language programs.
   Please go to the website to download MARS:
   http://courses.missouristate.edu/KenVollmar/MARS/
2. MARS is developed with Java language, and it requires JRE (Java Runtime Environment) installed on your computer. Please go to the website to download JRE: http://www.oracle.com/technetwork/java/javase/downloads/index.html
3. Usage of MARS:

Open a new file



Save your file



Assemble
Run

- Appendix

System Calls

SPIM provides a small set of operating-system-like services through the system call (`syscall`) instruction. To request a service, a program loads the system call code (see Figure A.9.1) into register $v0 and arguments into registers $a0–$a3 (or $f12 for floating-point values). System calls that return values put their results in register $v0 (or $f0 for floating-point results). For example, the following code prints "the answer = 5":

```
        .data
str:
        .asciiz  "the answer= "
        .text

        li       $v0, 4     # system call code for print_str
        la       $a0, str   # address of string to print
        syscall             # print the string

        li       $v0, 1     # system call code for print_int
        li       $a0, 5     # integer to print
        syscall             # print it
```

The `print_int` system call is passed an integer and prints it on the console. `print_float` prints a single floating-point number; `print_double` prints a double precision number; and `print_string` is passed a pointer to a null-terminated string, which it writes to the console.

The system calls `read_int`, `read_float`, and `read_double` read an entire line of input up to and including the newline. Characters following the number are ignored. `read_string` has the same semantics as the UNIX library routine `fgets`. It reads up to $n - 1$ characters into a buffer and terminates the string with a null byte. If fewer than $n - 1$ characters are on the current line, `read_string` reads up to and including the newline and again null-terminates the string.

Service	System call code	Arguments	Result
print_int	1	$a0 = integer	
print_float	2	$f12 = float	
print_double	3	$f12 = double	
print_string	4	$a0 = string	
read_int	5		integer (in $v0)
read_float	6		float (in $f0)
read_double	7		double (in $f0)
read_string	8	$a0 = buffer, $a1 = length	
sbrk	9	$a0 = amount	address (in $v0)
exit	10		
print_char	11	$a0 = char	
read_char	12		char (in $a0)
open	13	$a0 = filename (string), $a1 = flags, $a2 = mode	file descriptor (in $a0)
read	14	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars read (in $a0)
write	15	$a0 = file descriptor, $a1 = buffer, $a2 = length	num chars written (in $a0)
close	16	$a0 = file descriptor	
exit2	17	$a0 = result	

FIGURE B.9.1 System services.