

Computer Organization Project

Lab1 – MIPS Programming

1. 目標 Goal：

在單元一中，同學們將練習撰寫 MIPS 程式，以便了解組合語言與高階語言間的差異。為了驗證程式的正確性，也將練習使用 MIPS 的 simulator – MARS(or SPIM)來模擬程式的執行。

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 –MARS (or SPIM) to simulate the programs.

2. 附檔 Attached Files：

- “MIPS Simulator – MARS.pdf”：MIPS 模擬器 MARS 之使用說明。
The instruction of MIPS simulator MARS.
- “factorial.c”：factorial 程式之 C code，修改自教科書範例。
C code for “factorial”, modified from the example given in textbook.
- “factorial.s”：factorial 程式之 MIPS code，修改自教科書範例。
MIPS code for “factorial”, modified from the example given in textbook.
- “bubble_sort.c”：Bubble sort 程式之 C code，修改自教科書範例。
C code for “bubble sort”, modified from the example given in textbook.
- “fibonacci_f.c”：以 loop 形式所撰寫的 Fibonacci 程式之 C code。
C code for “Fibonacci number” with loops.
- “fibonacci_r.c”：以 recursive call 形式所撰寫的 Fibonacci 程式之 C code。
C code for “Fibonacci number” with recursive calls.

3. MARS 的安裝與使用 Installation and Using of MARS：

A. Download and installation:

1. Download the page:
<https://courses.missouristate.edu/KenVollmar/mars/download.htm>
2. Download the jar file.
(You might need to install Java JDK and add it to PATH for running the simulator. Otherwise, you can choose to use SPIM.)

B. Interface of MARS:

1. Edit/Execute section: for showing MIPS code and corresponding text and data segments.
 - A. “Text Segment”: for showing the MIPS instructions loaded into memory to be executed (your code).
 - B. “Data Segment”: for showing data in memory, stack, etc.
2. Register section “Regs”: for showing allocation of registers .
3. Run I/O: the interface (input/output) between program users and computer

C. Steps for running a MIPS code on MARS:

1. File -> Open
2. Run-> Assemble (F3)
3. Run-> Go (F5)

4. Check Registers section, Data section, and Run I/O, if necessary.

D. Debugging:

MARS supports two methods for debugging:

1. Run -> Step (F7):
You can execute your program step by step for debugging, but it may take a lot of time.
2. Set breakpoint(s):
In “Text Segment”, mark the “Bkpt” of a specific memory address to set a breakpoint.

E. References:

Please refer to the attached file “MIPS Simulator – MARS.pdf” and <http://courses.missouristate.edu/kenvollmar/mars/index.htm> for detailed information.

4. 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	Result
print integer	1	\$a0 = integer to print	
print float	2	\$f12 = float to print	
print double	3	\$f12 = double to print	
print string	4	\$a0 = address of null-terminated string to print	
read integer	5		\$v0 contains integer read
read float	6		\$f0 contains float read
read double	7		\$f0 contains double read
read string	8	\$a0 = address of input buffer \$a1 = maximum number of characters to read	See note below table
sbrk (allocate heap memory)	9	\$a0 = number of bytes to allocate	\$v0 contains address of allocated memory
exit (terminate execution)	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

- **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
```

D. Detailed information:

Connect to <https://www.mips.com/products/architectures/mips32-2/> for detailed information.

5. Lab 說明 Lab Description :

本實驗單元共分成四個部分；A 部分是練習用，將提供一個階乘運算程式之 C code (factorial.c) 與 MIPS program (factorial.s) 供同學參考，作為練習 MARS 之用；B 與 C 部分，將提供 C code (bubble_sort.c 與 fibonacci_f.c)，同學們必須照著附檔的 C code 形式轉成 MIPS 的 code；D 部分則為加分題，也會提供 C code (fibonacci_r.c)，作為轉換成 MIPS code 之參考。

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, MARS.

In part B and part C, “bubble_sort.c” and “fibonacci_f.c” written in C are given, respectively. You have to write the corresponding MIPS codes according to these programs. Part D is the bonus and the C code “fibonacci_r.c” is also given for your reference of writing the corresponding MIPS code.

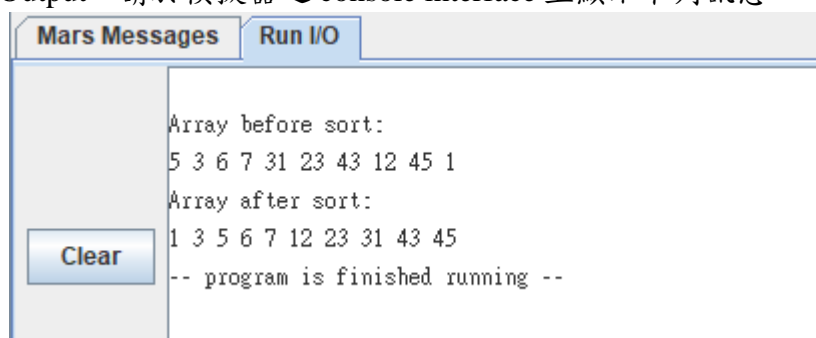
A. Factorial : (0%)

此程式為階乘運算($n!$)，修改自教科書中的範例。附檔中的 factorial.c 與 factorial.s 供參考，作為練習 MIPS 模擬器之用。

The attached files factorial.c and factorial.s are modified from the example given in textbook for computing $n!$. In this part, please execute factorial.s on MIPS simulator SPIM for practice.

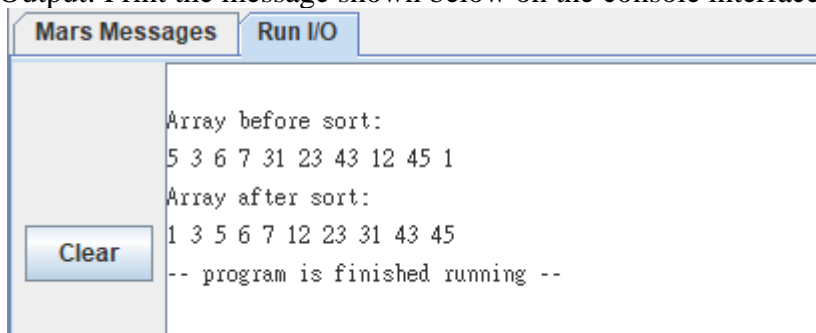
B. Bubble Sort : (40%)

1. 請根據“bubble_sort.c”，撰寫出相對應的 MIPS 程式，命名為“bubble_sort.s”。
2. “bubble_sort.c”中有兩個 procedures，swap 與 sort；詳細說明請參見教科書。
3. 待排序之數列資料請於程式的 .data 中直接設定。
4. Output：請於模擬器之 console interface 上顯示下列訊息



```
Mars Messages Run I/O
Array before sort:
5 3 6 7 31 23 43 12 45 1
Array after sort:
1 3 5 6 7 12 23 31 43 45
-- program is finished running --
```

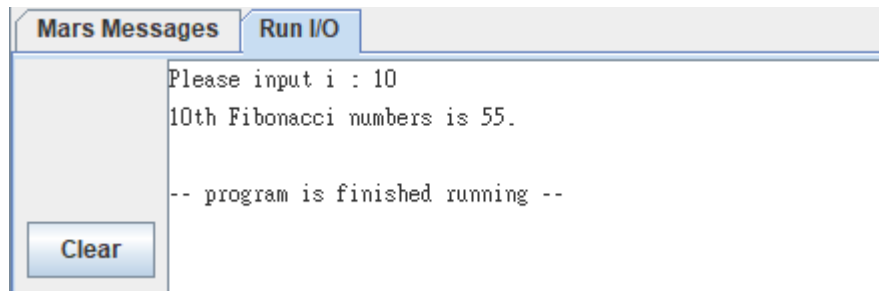
1. According to “bubble_sort.c”, please write the corresponding MIPS program, named “bubble_sort.s”.
2. There are two procedures, swap and sort, in “bubble_sort.c”. Refer to the textbook for the detailed description of these two procedures.
3. Please declare and set the array before sorting in the .data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



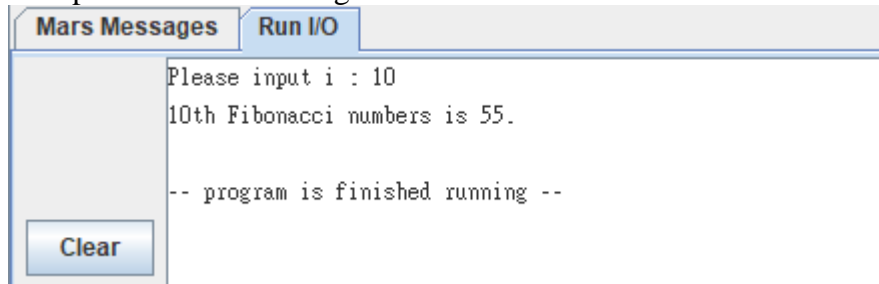
```
Mars Messages Run I/O
Array before sort:
5 3 6 7 31 23 43 12 45 1
Array after sort:
1 3 5 6 7 12 23 31 43 45
-- program is finished running --
```

C. Fibonacci number , non-procedure-call version : (60%)

1. 請根據“fibonacci_f.c”，撰寫出相對應的 MIPS 程式。請命名為“Fibonacci_f.s”。
2. Input：透過模擬器之 console interface 輸入一個整數。
3. Output：請於模擬器之 console interface 上顯示下列訊息

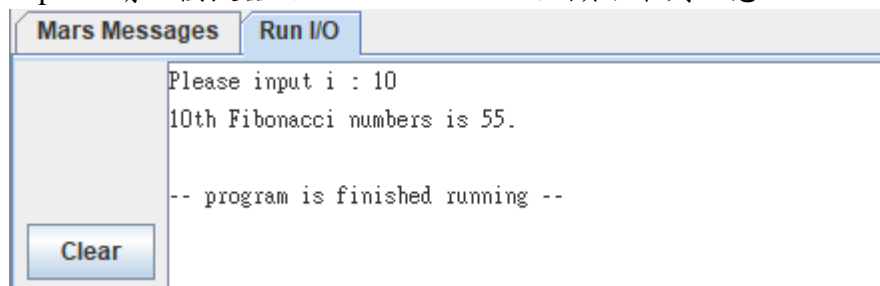


1. According to “fibonacci_f.c”, please write the corresponding MIPS program, named “fibonacci_f.s”.
2. Input : Please input one integer on the console interface.
3. Output : Print the message shown below on the console interface of the simulator.

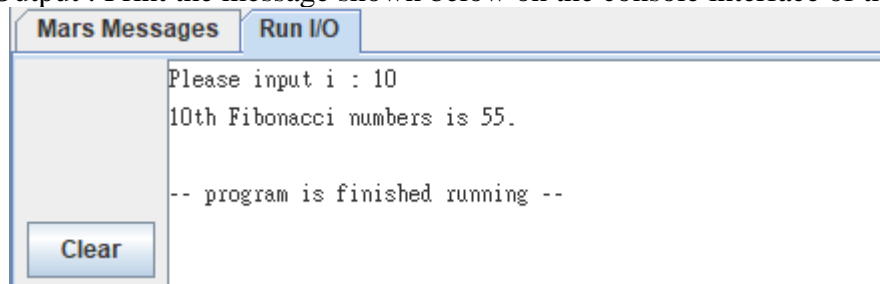


D. Fibonacci number , **recursive-call** version : (Bonus 20%)

1. 請根據”fibonacci_r.c” ，撰寫出相對應的 MIPS 程式。請命名為 “fibonacci_r.s”。
2. Input：透過模擬器之 console interface 輸入一個整數。
3. Output：請於模擬器之 console interface 上顯示下列訊息



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



6. 評分方式 Grading:

各部分(B, C, and D)評分方式為：

- 程式正確性：70%
- 程式註解：30%

助教會以多筆測資測試你的程式，並檢查你的程式是否符合作業要求。程式執行流程、函式呼叫必須與提供之.c 檔相同且輸出正確，則可獲得程式正確性部分之分數；**註解部分，則依註解詳細度斟酌給分。輸出若不正確，則 0 分(註解亦不給分)。**

Grading policies for each part (B, C, and D) are listed as follows:

- Correctness of the program: 70%
- Comments of the program: 30%

We will test many test cases for grading and check if the requirement is met. While the control flow and function call of your program are the same as the provided .c file and the output is correct, you will get the full grade for code correctness, and **the comments of the program will also be graded; otherwise, you will get zero score even if you have written comments for the program.**

7. Deadline:

- A. 本實驗單元為一人一組，請將作業檔案上傳至 e3 平台。

This lab unit is one student per group. Please upload your homework onto e-Campus platform.

- B. 繳交日期為 **2022/7/17 (日) 23:59**，**準時繳交者可獲得 10% 的額外加分**；最後繳交期限為 **2022/8/15 (一) 23:59**，**之後不接受逾期繳交。**

The due date is **2022/7/17 (Sunday) 23:59** and you will **get extra 10% points for on-time award**. Deadline is **2022/8/15 (Monday) 23:59, no late hand-in is allowed.**

- C. 須繳交的檔案為：

You should hand in the files listed as follows:

`bubble_sort.s`

`fibonacci_f.s`

`fibonacci_r.s` (bonus)

請將上述作業 **MIPS 檔(.s)**全部壓縮成一個 **zip 檔** (禁止上傳 rar 檔或是其他檔案格式)，並以「**Lab1_學號_姓名**」的方式命名，如：「Lab1_110551600_王大明」。

Please compress these .s files into one **zip** file (rar file or other format is not accepted), and name the zip file as “**Lab1_StudentID_Name**”, for example, “Lab1_110551600_KentChang”.

- D. **禁止抄襲，違者(抄襲者與被抄襲者)以 0 分計算。**

Any assignment work by fraud will get a zero point.