Computer Organization (CCIT4026)

HKU SPACE Community College

Student ID:		Name:		Date:	

(Lab 05 – CL01/CL07/CL08)

Instructions:

1. Fill in your personal information on the provided answer sheet (WORD) file.
2. The lab. submission consists of two components: a) A lab. report in PDF format which includes the material stated in the lab. sheet and b) The original assembly source code file for output verification.
3. Submission missing **EITHER** the lab. report (.pdf) **OR** the final source code (.asm) would **NOT** be graded.
4. Screen captures your final assembly source codes (with colors) and the input/output messages onto a WORD file and then convert to PDF format. The file would be considered as lab. report.
5. **Source code listing shown in the lab. report file CANNOT be superseded by the source code ".asm" file. Mark deduction would be imposed if no source code listing is shown in the report file.**
6. Upload **BOTH** the lab. report PDF file **AND** the final source code to SOUL.
7. Make sure the source codes are readable and neatly formatted. Unclear, incorrectly-sized or codes of poor readability would NOT be graded.
8. Each instruction should be associated with suitable comments to improve readability.
9. All problems are expected to be answered by using the ***instructions*** mentioned in the course materials ONLY. If you have to bring in instructions out of this scope, **prior consent** from the lecturer should be sought and a proper citation has to be provided otherwise the answers would be considered as INCORRECT.
10. Make sure to run (and single-step) your codes on MARS simulator to confirm its functionality before submission.
11. **Photo capture of the results** shown on the monitor is **NOT** acceptable. Use screen capturing software such as "Snipping Tool" on Window, then copy and paste the images onto the WORD file.
12. **No late submission will be accepted.**
13. Please read the Student Handbook for issues related to academic honesty. The College places very high importance on honesty in academic work submitted by students, and adopts a policy of zero tolerance on cheating and plagiarism. Any related offense may be passed to the Guidance Committee. Possible disciplinary action, including termination of study, may follow.
14. By making a submission, you also confirm to have read and understood the above-listed instructions and HKU SPACE Community College's academic honesty policy, and agree to comply with them. Failure to comply with the above instructions may lead to disqualification of your assignment results and possible disciplinary actions.

Introduction

In this lab, we are going to study the procedure call using MIPS programming. The program code that you will write in this lab must conform to the procedure calling convention discussed in the lecture. Each main program or function *must conform to this convention strictly, and use the stack to save and restore the necessary register values*.

15. *Program specifications*

Given two vectors C and D defined as below,

$$C = \{c_0\ c_1\}$$

and

$$D = \{d_0\ d_1\}$$

Write a MIPS assembly program to calculate the value *M* defined as follows,

$$M = \sum_{i=0}^{1} d_i + int(\frac{(c_i + d_i)^2}{(d_i + 1)})$$

The calculation of the value *M* should be implemented by two functions which are called "comp_sum" and "square_div". The tasks of all functions are defined as below:

- In the "main" program, it should pass all four vector components of both *C* and *D* to the function "comp_*sum*" as arguments.
- The function, "*comp_sum*" should accept four input parameters to perform the sum of each component between them.
- Then pass EACH sum and one vector component as arguments to another function "*square_div*".
- In the function "*square_div*", which takes the input parameters, squares and then divides the result by another parameter according to the equation. The integer part of the result should be returned back to the function "*comp_sum*".
- In the function "*comp_sum*", it should call the function "*square_div*" twice for each pair of vector components, sum each returned results together with d_i and return back to the "*main*" function.
- The main function should lastly print the result *M* on the console.

Below is the Pseudo-code of this program:

For "*main*",

Initialization

- Reserve 4 words in data segment to store the two vectors *C* and *D*.
- Reserve sufficient space for stack.

Step 1

Step 1.1 Setup the stack pointer $sp to the top of stack.

Step 1.2 Read all four vector components onto the four argument registers.

Step 1.3 Take all procedure calling convention and save caller-saving registers if necessary.

Step 1.4 Call the function "comp_sum".

Step 1.5 Restore any caller-saving registers if necessary.

Step 1.6 Print the result prompt message and the return value from the function "comp_sum" according to the sample output format.

Step 1.7 Exit program.

For function "*comp_sum*",

Step 1

Step 1.1 Take all procedure calling convention and save callee-saving registers if necessary.

Step 1.2 Find the sum of the first components of C and D and

Step 1.3 Copy the sum and the first component of D to the argument registers for the function "square_div".

Step 1.4 Take all procedure calling convention and save caller-saving registers if necessary.

Step 1.5 Call the function "square_div".

Step 1.6 Restore any caller-saving registers if necessary.

Step 1.7 Sum the first component of D with the return value and store to a register.

Step 1.8 Find the sum of the second components of C and D and

Step 1.9 Copy the sum and the second component of D to the argument registers for the function "square_div".

Step 1.10 Take all procedure calling convention and save caller-saving registers if necessary.

Step 1.11 Call the function "square_div".

Step 1.12 Restore any caller-saving registers if necessary.

Step 1.13 Add the two return values together

Step 1.14 Add the second component of D and store to return register.

Step 1.15 Restore callee-saving registers if necessary.

Step 1.16 Return to caller.

For function "*square_div*",

Step 1

Step 1.1 Save callee-saving registers if necessary.

Step 1.2 Find the square of the first parameter.

Step 1.3 Increment the second parameter by 1

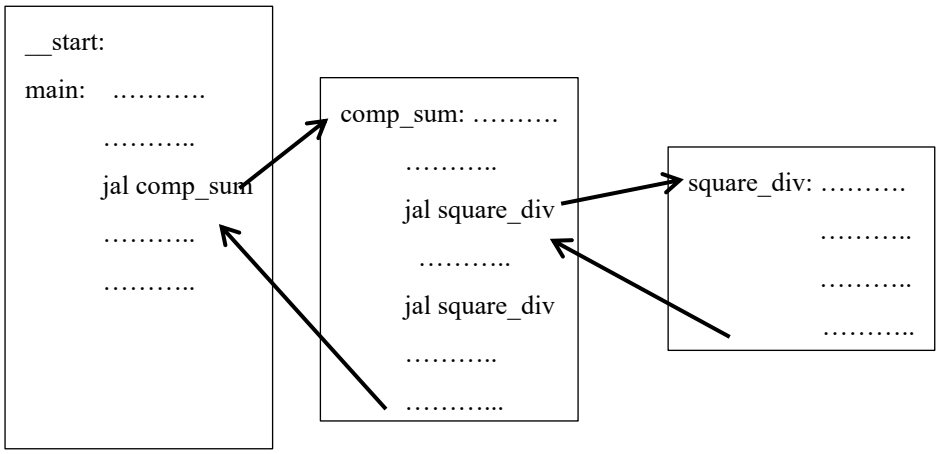
Step 1.4 Divide the squared result with the incremented result.

Step 1.5 Copy the division result to return register.

Step 1.6 Restore callee-saving registers if necessary.

Step 1.7 Return to caller.

The program execution flow is shown below,



The vector elements of A and B should be pre-set in data segment and defined based on your SID. For example, assume your student ID (SID) is 1234**5678** then vector A and B are defined as follows:

C = {	5	6	}
D = {	7	8	}

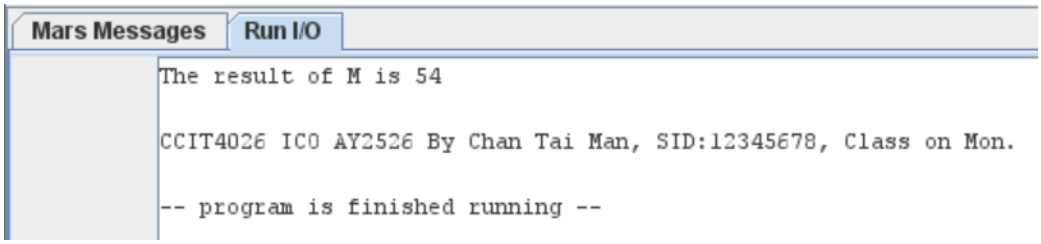
Thus the value *M* should be calculated as follows,

$$M = 7 + int\left(\frac{(5+7)^2}{7+1}\right) + 8 + int\left(\frac{(6+8)^2}{8+1}\right) = 54$$

NOTES:

- A suitable size of stack should be defined in the .data segment.
- Properly initialize the $sp to the Top of Stack (TOS).
- The assignment of vector components to argument registers could be ARBITRARILY defined for convenient.
- Parameters of any function can NOT be passed through stack.
- In the function "*comp_sum*", the following registers CAN ONLY be used with restrictions:
  - $a0-$a3: Read AND Write.
  - $v0-$v1: Read AND Write.
  - $sp and $ra: Read AND Write (for stack operation ONLY).
- In the function "*square_div*", the following registers CAN ONLY be used with restrictions:
  - $a0-$a1: Read ONLY.
  - $s0-$s1: Read AND Write.
  - $v0-$v1, Read AND Write.
  - $sp and $ra: Read AND Write (for stack operation ONLY).
- Only 32-bit results should be considered in all arithmetic operations.
- All procedure calling, caller-saving registers and callee-saving registers convention MUST be followed.

Sample Output:



Submission instructions:

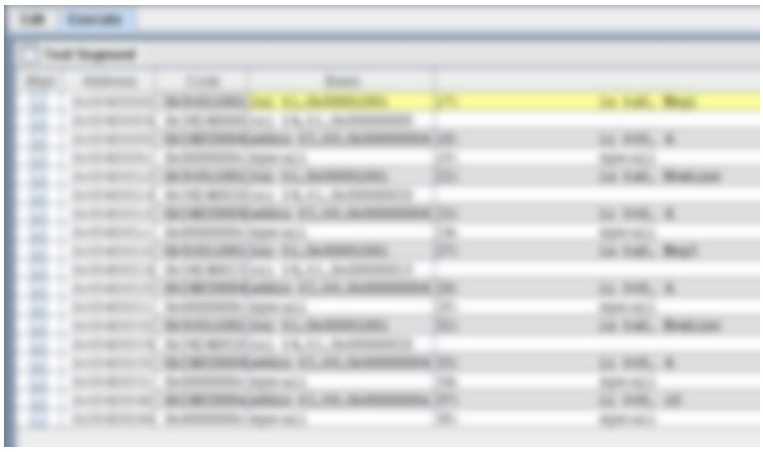
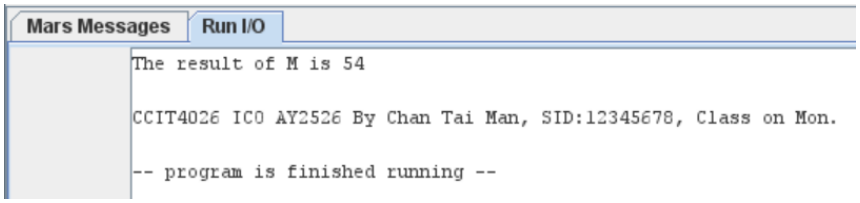
i. Write the MIPS assembly program to complete the lab. tasks above.
ii. **Strictly follow** the steps shown in the Pseudo-code.
iii. Copy each of the step descriptions shown in the **Pseudo code as comments** in your program.
iv. Complete each step with your MIPS instructions.
v. Copy and paste the screen capture of the source code from the MARS editor onto the provided answer sheet template in WORD file.
vi. Copy and paste the screen capture of the assembled window that shows your assembled machine codes.
vii. Copy and paste the screen capture of the "Run I/O" window that shows your program execution output (Program termination message MUST be shown).
viii. Fill in your name, class, student ID and a photo of your student ID card at the beginning of the WORD file. A sample submission format is shown below.
ix. Convert the provided answer sheet template WORD file to PDF format as your lab. report.
x. Rename your revised ".asm" and lab. report ".pdf" file to the file names according to the following format,
    i. *"ICO_Lab5_CL<classNO>_<yourname>_<studentID>.asm"and*
    ii. *"ICO_Lab5_CL<classNO>_<yourname>_<studentID>.pdf"*
    iii. e.g., For CL01,
        1. *"ICO_Lab5_CL01_CHANTaiMan_20002000.asm"and*
        2. *"ICO_Lab5_CL01_CHANTaiMan_20002000.pdf"*

xi. Upload BOTH the lab. report PDF file and your ".asm" to SOUL before the deadline.
xii. No mark would be given if no source code file ".asm" is submitted OR missing your student ID card photo.
xiii. Late submission will NOT be graded.

Submission Sample:

		(Insert a photo of your student ID card below)
Student Name:		(No mark would be given if no student ID is present here)
Student no.:		
Class Number:		

Manual calculations	Fill in your SID below: C = { 5 6 } **D** = { 7 8 } $M = 7 + int\left(\frac{(5+7)^2}{7+1}\right) + 8 + int(\frac{(6+8)^2}{8+1}) = 54$
Source Code (Captured from MARS)	Screen capture the output directly from MARS Editor.
Assembled Results (Captured from MARS)	Screen capture the output directly from "Execute" window:

	
Run I/O results (Captured from MARS)	Screen capture the output directly from “Run I/O” window: 

— THE END —