CS 2318 (Assembly Language, §260, §263 & §264, Spring 2018) by Lee S. Koh

Assignment 2

(separate post/due dates for each (sub-)part)

■ Part 1 (30 points, 7.5 points per program)

For this part (which has *4 sub-parts*: **Part 1A**, **Part 1B**, **Part 1C** and **Part 1D**) of the assignment, you are to use **MARS** to work on and test small MIPS assembly language programs - 4 ***separate programs***, one ***each*** for **Part 1A**, **Part 1B**, **Part 1C** and **Part 1D**.

⇒ IMPORTANT:
You must use the version of MARS *modified* for use in this class posted under **Other Resources** (MARS).

⇒ CAUTION:
If you use a different version of MARS and your code doesn't assemble when tested using the modified version indicated above, the highest score you can get is 30% of the maximum attainable score.

⇒ IMPORTANT:
You do NOT need to know any *branching/jumping* instructions (for doing *conditional and unconditional goto's*) to do the four Part 1 programs.

If you happen to know about such instructions, you **MUST NOT** use them in any of the Part 1A, Part 1B, Part 1C and Part 1D programs.

- Examples of instructions (to be covered later on so you are not expected to know them now) you must NOT use:
  - `j ...` (for **j**ump)
  - `jal ...` (for **j**ump **a**nd **l**ink)
  - `b ...` (for **branch**)
  - `beq ...` (for **b**ranch on **eq**ual)
  - `blt ...` (for **b**ranch on **l**ess **t**han)
  - `bne ...` (for **b**ranch on **n**ot **e**qual)

Since each program is quite trivial in this assignment, you don't have to use/follow the documentation guide (templates) given in the lecture note. Instead, include the following items (as comments) at the ***top*** of each program:

⇒ Your **`name, CS 2318-26?, Assignment 2 Part 1 Program`** **`?`**.

- **`?`** should be 0, 3 or 4, and **`?`** should be A or B or C or D.

⇒ The description of the task.

- Simply adapt the given description and include it *as comments*.
  Make sure that each line is not too long (limit each line to no more than 80 characters, counting white spaces, seems to be good rule) as to cause line wrap when printing.

CAUTION:
Too many past students regretted having points taken off for failing to do this.

For *each sub-part* (*i.e.*, Part 1A, Part 1B, Part 1C or Part 1D), you are to print and turn in *hardcopy* of

the following items (be sure to print ***each item starting on a new sheet***):

⇒ Your program (MARS' **`Edit`** window).

⇒ The test output (MARS' **`Run I/O`** window).

Be sure to collate and staple you printouts together with a cover page - a ***separate cover page for each part***.

Be sure to also *email me your programs* (don't have to email me the test output).

⇒ Be sure to read Programming Assignments (What/How to Submit) on how to properly submit softcopies of your programs (especially in regard to the inclusion of ***subject lines in the specified format***).

Be sure to refer to http://cs.txstate.edu/~lk04/2318/OthRes/Using_Microsoft_Excel_to_Print_Program_Nicely.pdf to print your MARS program nicely.

A (posted 02/21/2018, DUE 2/28/2018 for Sec.260, 3/1/2018 for Sec.263 and Sec.264)

Prompt the user to enter each of the following, *read* the user's input, and *display* a *labeled output* about the user's input.

⇒ An integer.

⇒ A string of up to 50 characters long.

- String entered MUST be stored in *separate storage space* (*i.e.*, not overwriting the storage space used by prompt and label strings) allocated *just enough (no more, no less) to accommodate up to the maximum number* of characters indicated.

⇒ A character.

- Note that it involves a character, NOT a *one-charater string*. You will get no credits if you do it using a *one-character string* even though the same output is obtained.

Be sure to do the "prompting-reading-displaying" for each type separately, *i.e.* first prompt, read and display the integer, then prompt, read and display the string, and finally prompt, read and display the character.
(You will incur penalty if you prompt for all three together, then read all three together and finally display all three together.)

Be sure to introduce appropriate spaces and newlines to make what appears on the **Run I/O** window (of MARS) reasonably readable.
(You will incur penalty, for instance, if different items appear running together.)

B (posted 02/21/2018, DUE 2/28/2018 for Sec.260, 3/1/2018 for Sec.263 and Sec.264)

Study "this supplied program" and fill in the missing code as indicated (via comments).

IMPORTANT: As indicated in the program's comments:

⇒ You ARE to write no more than certain number of lines of code as indicated.
(One instruction per line, and there will be penalty if your code consumes more lines.)

⇒ Your code MUST use only instructions that are allowed (*i.e.*, only *bit manipulating* instructions).

⇒ You MUST NOT in any way change the lines of code already written.

⇒ You MUST print your actual test output (MARS' **`Run I/O`** window), NOT the sample test runs

included at the bottom of the supplied file.

C (**posted 02/21/2018, DUE 2/28/2018** for **Sec.260**, **3/1/2018** for **Sec.263** and **Sec.264**)

The following tasks, in order:

⇒ Allocate a global array (*i.e.*, space in the *data segment*) enough for storing 3 integers and initialize the array with 1001, 2002 and 3003 *at the same time* (*i.e.*, DON'T write code to put in these values instead).

⇒ Display a labeled output about the array's initial contents (from **1st to 3rd** element).

⇒ Re-order the values in the array so that the contents of the array *in memory* (from 1st to 3rd element) becomes 3003, 2002 and 1001, using the following operations in the order listed (to not defeat the goals of this exercise, you must NOT change the specified operations and order, even if doing so will accomplish the same effect more efficiently):

- Swap the contents *in memory* of the 1st and 2nd elements of the array.
- Swap the contents *in memory* of the 1st and 3rd elements of the array.
- Swap the contents *in memory* of the 2nd and 3rd elements of the array.

IMPORTANT:

When performing the *second* and *third* swap operations above, you can re-use the array's base address in register (loaded when performing the *first* swap operation) but you MUST *re-load the values of the associated array elements fresh from memory* (*i.e.*, assuming *no knowledge* that certain values might have already existed in some registers due to prior swap operations).

⇒ Display a labeled output about the array's contents (from **3rd to 1st** element) after the 3 swapping operations above.

- NOTE: It is from **3rd to 1st** element and not from **1st to 3rd** element.

IMPORTANT:

Similar to when performing the *second* and *third* swap operations above, you can re-use the array's base address in register (loaded when performing the prior operations) but you MUST *re-load the values of the array elements fresh from memory* (*i.e.*, assuming *no knowledge* that certain values might have already existed in some registers due to prior operations).

D (**posted 02/21/2018, DUE 2/28/2018** for **Sec.260**, **3/1/2018** for **Sec.263** and **Sec.264**)

Prompt the user to enter the *integer* scores for Exam 1, Exam 2 and Final Exam, read the scores, compute the weighted average score (using the following formula), and display a labeled output about the weighted average score.

$$avgScore = \frac{410}{2048} e1Score + \frac{256}{853} e2Score + \frac{finScore}{2}$$

IMPORTANT: For the purpose of this exercise, be sure to observe the following:

⇒ You MUST perform (in the appropriate order, of course) ALL the additions, multiplications and divisions shown in the given formula. (You should NOT resort to simplifying the formula in some way, perhaps to make the computation more efficient.)

⇒ You MUST use *bit-shifting* to effect multiplications and divisions involving powers of 2.

- Note that 2, 256 and 2048 correspond to some powers of 2 (but not 410 and 853).
- You are NOT to replace 410 and 853 (that are not powers of 2) with their "sum-of-powers-of-2" equivalents.

⇒ Assume it is the intent to simply discard the fractional portion when a division is performed. When evaluating the *first* and *second* terms on the right hand side (*i.e.*, the Exam 1 and Exam 2 contributions, respectively), however, you MUST perform (in each case) the division *after* (NOT *before*) the multiplication (otherwise, accuracy may be unnecessarily lost).

⇒ For any multiplication and division operation that cannot be effected with simple (one-time) bit-shifting (*i.e.*, without resorting to doing "sum-of-powers-of-2" replacments), you MUST use another "*true*" *instruction* (NOT a *pseudoinstruction*) instead.

- **Part 2 (20 points)**
  (to be filled in)

. . .

- **Part 3 (50 points)**
  (to be filled in)

. . .

NOTES:

- Be sure to check back often for updates and/or further information.
- Please let me know (best through email) if you spot anything that looks incorrect.

Use your browser's **Back** button to go back to previous page. (click here to return to my homepage)
