B: SE 2XA3 (2018/19, Term I) Major Lab 4 -- lab section L01

Back To Lab Menu | Back To Main Menu | Submissions | Log Out

sample solutions: proj4_1.asm and proj4_2.asm and bonus.txt

If you are in the correct lab room and during the SCHEDULED time slot for your lab section for Major Lab 4 and working on one of the lab's workstations, go into Marks+Comments to see if your attendance was registered. If you signed on the course website too early (the most common problem), your attendance would not be recorded; in such case, please log out and sign on again. If your attendance is still not recorded, please, ask your TA to record your attendance and email Prof. Franek right away. If you are not using one of the lab's workstations, please, have the TA record your attendance.

The Major Labs are open book labs, you can bring and use any notes etc. You can access any website, Google, Wikipedia etc. The only thing that is not allowed is cooperation with other people, inside or outside the lab. You must submit your own work. The TA can only help with administrative and technical aspects and not with the solutions of the problems of the Major Lab.

In this lab, there are two tasks and two deliverables: NASM programs `proj4_1.asm` and `proj4_2.asm` , and a text file `bonus.txt` for the bonus if you decide to do the bonus question -- note that the bonus is not mandatory and the bonus mark of 2% will be applied to the overall grade. Each file can be submitted either via the course website, or using `2xa3submit`. For `2xa3submit` submission please use `2xa3submit AAA proj4 BBB` where `AAA` is your student number and `BBB` the name of the file you want to submit. You can submit every file as many times as you wish, the latest submission will be used for marking. The submission is opened from 8:30-11:20 or 14:30-17:20 depending on the lab section; after the closing time no submission is possible. If submission is not possible for whatever reason (typically right after the submission closes), email the file or files as an attachment immediately (needed for the time verification) to Prof. Franek (franek@mcmaster.ca) with an explanation of the problem; you must use your official McMaster email account for that. Include the course code, your lab section, full name, and your student number. Note that if there is no problem with submission (typically the student using a wrong name for the file), you might be assessed a penalty for email submission, depending on the reason you used email.

Task 1. NASM program named `proj4_1.asm` (Do proj4_1.asm before proj4_2.asm)

The name of your file must be `proj4_1.asm` and below is a description of what it should do when executed.

Before you start working on the program `proj4_1.asm`:

- Decide what will be your working directory.
- Download a text file asm_io.asm and save it on your workstation, then transfer it to ***moore*** to your working directory and convert it to a unix text file (using `dos2unix`).
- Download a text file asm_io.inc and save it on your workstation, then transfer it to ***moore*** to your working directory and convert it to a unix text file.
- Download a text file cdecl.h and save it on your workstation, then transfer it to ***moore*** to your working directory and convert it to a unix text file.
- Download a C++ program driver.c and save it on your workstation, then transfer it to ***moore*** to your working directory and convert it to a unix text file.
- Download a bash script makefile and save it on your workstation, then transfer it to ***moore*** to your working directory and convert it to a unix text file.

What should `proj4_1.asm` do:

1. In the program in the `.data` section define a byte array named `string` and initialize it with `Hello World` and

terminate it with 0.
2. The program traverses `string` counting the number of characters in the string and also counting the number of the lower case letters.
3. When the count of characters reaches 20 and the string had not been entirely traversed, an error message is displayed, and the program terminates.
4. If the traversal was successfully completed, the string is displayed.
5. Then a message about the length of the string is displayed followed by display of the string's length.
6. Then the message about the number of lower case letters is displayed, followed by display of the number of lower case letters.
7. The program terminates.

A sample run

```
Hello World
the length of the string is 11
the number of lower case letters is 8
```

A sample run if the value of `string` in the code is changed to `HelloHelloHelloHello WorldWorldWorldWorldWorld` and the whole program is re-compiled.

```
too many characters
```

Hint: to display a string use `print_string` *and to display a number use* `print_int` *. To check if a letter is a lower case letter use two comparisons (we are assuming that* `ebx` *is pointing to the letter in the string*) :

```
cmp byte [ebx],'a'
jb L1
cmp byte [ebx],'z'
ja L1
....        ;;code here for what to do when byte[ebx] is a lower case letter
jmp L2
L1:
....        ;;code here for what to do when byte[ebx] is not a lower case letter
L2:
```

Task 2. NASM program named `proj4_2.asm`

The name of your program must be `proj4_2.asm` and below is a description of what it should do when used.

What should `proj4_2.asm` do:

1. It is a modification of `proj4_1.asm`, see above.
2. After `string` is displayed, and its length, and the number of lower case letters as in `proj4_1.asm`, the string is crossed displayed, i.e. find the first blank (space) in the string and display all the letters after the space, then the space, then all the letters before the space.
3. Then it displays the string in its original form.

Hint: For the crossed display: to display all the letters after the first blank is easy, just use `print_string` *and pass it as the address of the string the* `address of the blank+1`*. We cannot use a similar trick for displaying the first part before the blank, as it is not terminated by the* `NULL` *character as required by* `print_string` *. So we temporarily do it, i.e. we put the* `NULL` *character instead of the blank in the string. Then we can use* `print_string` *and pass it the original address of the string. Then we have to "heal" the string so it can be again displayed in its entirety, i.e. we put blank back where we temporarily put the* `NULL` *.*

A sample run

```
Hello World
the length of the string is 11
the number of lower case letters is 8
World Hello
Hello World
```

A sample run when the value of string is changed to `A B C D` and the whole program is recompiled.

```
A B C D
the length of the string is 7
the number of lower case letters is 0
```

```
B C D A
A B C D
```

Bonus task.

Download a text file bonus.asm and save it on your workstation, then transfer it to ***moore*** to your working directory and convert it to a unix text file. Compile the program by `make bonus` . Now you can execute the program, for instance `bonus 123`

The output is `the length of the string is 3`

The program loops through the string given as the first command line argument, in this case `123`. A variable `N` is used as a counter of the number of characters traversed. Then the length message and the length are displayed.

Now execute the program with a longer string `bonus 12345`

The output is `the length of the string is 5`

As in the first run, everything looks fine.

Now execute the program with a yet longer string `bonus 123456`

The output is `the length of the string is 55`

The output indicates something is wrong. But what? Describe the nature of the error in the program and how to fix it in a text file `bonus.txt` that you submit in the usual way.