C CS 2XA3/SE 2XA3 (2015/16, Term I) Proj 3 -- lab section L03

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

sample solutions str1.asm str2.asm str3.asm makefile

In this project, there are three deliverables, i.e. three files are to be created and submitted either via the Submission button at the top of this page, or using the `2xa3submit` command (see Lab 1). All three files are NASM assembler programs called `str1.asm`, `str2.asm`, and `str3.asm`. The description of what these programs are supposed to do is given below.

Before you start working on these programs:

1. Download asm_io.inc and save it on your workstation, then transfer it to ***moore*** and convert it to a unix text file (using dos2unix). *This file is necessary for the assembly and compilation to work.*
2. Download asm_io.asm and save it on your workstation, then transfer it to ***moore*** and convert it to a unix text file (using dos2unix). *This file is necessary for the assembly and compilation to work.*
3. Download cdecl.h and save it on your workstation, then transfer it to ***moore*** and convert it to a unix text file (using dos2unix). *This file is necessary for the assembly and compilation to work.*
4. Download driver.c and save it on your workstation, then transfer it to ***moore*** and convert it to a unix text file (using dos2unix). *This file is necessary for the assembly and compilation to work.*
5. Download makefile and save it on your workstation, then transfer it to ***moore*** and convert it to a unix text file (using dos2unix).
6. Download argv.asm and save it on your workstation, then transfer it to ***moore*** and convert it to a unix text file (using dos2unix).
7. Crate the executable argv by make argv. Executing argv hello world, you should see
```
3
argv
hello
world
```
8. The program argv.asm illustrates how to "obtain" and "work with" the number of command line arguments (in the example above it is 3: argv hello world) -- argc in colloquial terminology, how to "obtain" and "work with" the 1st command line argument (in the example above argv), how to "obtain" and "work with" the 2nd command line argument (in the example above hello), and how to "obtain" and "work with" the 3rd command line argument (in the example above world).
9. The three programs you are to write are based on this program.

What should **`str1.asm`** do

1. The program `str1.asm` expects 1 command line argument.
2. In a loop it traverses the 2nd line argument (remember, it is a string terminated by null), and displays it one letter at a time, with a space between two consecutive letters.
3. Moreover, every digit less than 9 (i.e. 0, 1, ..., 8) is displayed as the next digit.
   E.g. `str1 Hello2` will display `h e l l o 3`, or `str1 Hello9` will display `H e l l o 9` or `str1 1234` will display `1 2 3 4`.
4. The I/O routines used must be from asm_io.asm (i.e. print_char, print_nl etc.)
5. ***Some useful tips***:
   Append the downloaded makefile, what is done for argv, *should be exactly done for* str1.

How to determine digits and siplayed them incremented:

```
CMP AL, '0'                ; compare the low byte of EAX with '0'
 JB NOT_DIGIT              ; if it is a smaller value, jump to NOT_DIGIT
 CMP AL, '9'               ; compare the low byte of EAX with '9'
 JAE NOT_DIGIT             ; if it is a greater or equal value, jump to NOT_DIGIT
 ; so it is digit
 SUB AL, '0'               ; subtract '0' and add '1'
 ADD AL, '1'               ;
 NOT_DIGIT:
 call print_char           ; display the letter
```

What should **`str2.asm`** do

1. The program `str2.asm` expects 1 command line argument.
2. It displayes the 2nd line argument (remember, it is a string terminated by null).
3. In a loop it traverses the 2nd line argument and counts th enumber of letters.
4. Then iy displays the number of letters and terminates. E.g. `str2 Hello2` will display
```
Hello2
6
```
   or `str2 buy` will display
```
buy
3
```
5. The I/O routines used must be from asm_io.asm (i.e. print_char, print_nl etc.)
6. ***Some useful tips****:*
   Append the downloaded makefile, what is done for argv*, should be exactly done for* str2*.*

What should **`str3.asm`** do

1. The program `str3.asm` expects 2 command line argument.
2. In a loop it traverses the 2nd line argument (remember, it is a string terminated by null), and displays it one letter at a time while counting the number of letters.
3. In a loop it traverses the 3rd line argument (remember, it is a string terminated by null), and displays it one letter at a time while counting the number of letters.
4. If the number of letters in both strings together is greater than 6, the error message `concatenation too long` is displayed and the program terminates; e.g. `str3 Hello world` will display
```
Helloworld
concatenation too long
```
5. If the number of letters in both strings together is at most 6, then there is no additional display, e.g. `str3 Hel wor` world will display
```
Helwor
```
6. The I/O routines used must be from asm_io.asm (i.e. print_char, print_nl etc.)
7. ***Some useful tips****:*
   Append the downloaded makefile, what is done for argv*, should be exactly done for* str3*.*