Computer Architecture
Assignment 2

Submission Guidelines

To reduce likelihood of misunderstandings, please follow these guidelines:

1. **Read the whole document carefully.**
2. Work individually.
3. Submission date is 9/12/2024, 23:59.
4. Submission is through the "submit" system.
5. Make sure that your solution compiles and runs without any errors and warnings on BIU servers.
6. In the first line of every file you submit, write in a comment your id and full name. For example: "/* 123456789 Israela Israeli */".
7. Do not use any AI agents (such as ChatGPT or Github Copilot) when writing your homework. Getting caught using tools as such will count as cheating just like any other method.
8. Use only AT&T syntax.
9. For every question you have, please use the course site forum.
10. **Good luck!**

General Background

The goal of this assignment is to practice writing more advanced assembly. This assignment will be about strings.

Assignment Details

In this assignment, we'll write a simple string library. To get working on this assignment, unzip the **"pstrings.zip"** file attached to this assignment, and open the directory in your text editor.

First, let's understand the structure of the program:

- "pstrings.h" - header file where the main structure of the program and the library functions are declared.
- "main.c" - implementation of the main function.
- "Makefile" - project builder for this part.
- "pstring.s", "func_select.s" - the two files you are in charge of implementing :).

The flow of the program is as follows: the main function receives from the user two Pstrings (by length and characters), and builds them. Then, it prompts the user with a menu - showcasing the available library functions, and which number should be inputted for provoking the matching function. After scanning in the requested number from the user, it calls the *run_func* passing references to both Pstrings and the inputted number.

The run_func function, which you'll implement in func_select.s, calls one of the pstrings library functions, according to the choice number. These library functions will also be implemented by you, in the pstring.s file.

In total, you need to submit the files "func_select.s" and "pstrings.s" only.

run_func in more detail

As stated above, the run_func function receives a choice integer and two Pstring references. You can see the signature in main.c. The function should behave the following way, according to the choice integer:

If choice is 31: The function should call pstrlen and calculate the length of each string, then print the lengths using the following format:

"first pstring length: %d, second pstring length: %d\n"

If choice is 33: The function should use swapCase to turn every capital case to its lower case, and vice versa. Then, print the two Pstrings in the following format:

"length: %d, string: %s\n"

If choice is 34: The function should receive from the user two integers as start and end indices. Then, call pstrijcpy with i being the start and j being the end index. After copying is completed, both Pstrings should be printed using the format:

"length: %d, string: %s\n"

If choice is 37: The function should call pstrcat, which, if possible, concatenates both strings by appending the second to the end of the first. After copying is completed, both Pstrings should be printed using the format:

"length: %d, string: %s\n"

For every other choice: the function should print the following error message:

"invalid option!\n”

pstring.s functions

char pstrlen(Pstring* pstr);

Given a pointer to a Pstring, the function should return the length.

Pstring* swapCase(Pstring* pstr);

Given a pointer to a Pstring, the function turns every capital case to its lower case and vice versa.

Pstring* pstrijcpy(Pstring* dst, Pstring* src, unsigned char i, unsigned char j);

Given pointers to two Pstrings, and two indices, the function copies src[i:j] into dst[i:j] (**including** j, where index is counted from 0) and returns the pointer to dst. If either i or j are invalid given src and dst sizes, no changes should be made to dst, and the following message should be printed: "invalid input!\n".

Pstring* pstrcat(Pstring* dst, Pstring* src);

Given pointers to two Pstrings, the function initially checks if (dst->len)+(src->len) ≤ 254. If so, (src->str) is appended to the end of (dst->str), and (dst->len) is updated correspondingly. Otherwise, dst remains the same, and the following message should be printed: “cannot concatenate strings!\n”. In either case, pointer to dst has to be returned.

Notes:

1. when implementing these functions, do **not** spoil the fun and use string.h functions…
2. Only libc functions allowed to use are scanf and printf. If you need more space for your program, don’t dynamically allocate using malloc, but use the stack.
3. It’s okay to assume that for each Pstring received, length inputted will match the actual length of input string, and that input length will always be less than 254.
4. Pay attention to conventions! Don’t forget the calling conventions we learn about in class, and don’t forget writing a prolog and an epilog for each function.
5. For this part of the assignment, do not use the data segment to store variable values, **use the stack!** You may still use the *rodata* segment in order to define format strings and so as literals.

Example outputs

```
Enter Pstring length: 11
Enter Pstring: Shalom Olam
Enter Pstring length: 12
Enter Pstring: Hello World!
Choose a function:
        31. pstrlen
        33. swapCase
        34. pstrijcpy
        37. pstrcat
31
first pstring length: 11, second pstring length: 12
```

```
Enter Pstring length: 7
Enter Pstring: aBcDeFg
Enter Pstring length: 4
Enter Pstring: !W3H
Choose a function:
        31. pstrlen
        33. swapCase
        34. pstrijcpy
        37. pstrcat
33
length: 7, string: AbCdEfG
length: 4, string: !w3h
```

```
Enter Pstring length: 11
Enter Pstring: Hello World
Enter Pstring length: 5
Enter Pstring: AAAAA
Choose a function:
        31. pstrlen
        33. swapCase
        34. pstrijcpy
        37. pstrcat
34
1 4
length: 11, string: HAAAA World
length: 5, string: AAAAA
```

```
Enter Pstring length: 11
Enter Pstring: Hello World
Enter Pstring length: 5
Enter Pstring: BBBBB
Choose a function:
        31. pstrlen
        33. swapCase
        34. pstrijcpy
        37. pstrcat
34
1 7
invalid input!
length: 11, string: Hello World
length: 5, string: BBBBB
```

```
Enter Pstring length: 4
Enter Pstring: ABCD
Enter Pstring length: 3
Enter Pstring: efg
Choose a function:
        31. pstrlen
        33. swapCase
        34. pstrijcpy
        37. pstrcat
37
length: 7, string: ABCDefg
length: 3, string: efg
```

```
Enter Pstring length: 250
Enter Pstring: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAA
Enter Pstring length: 5
Enter Pstring: Hello
Choose a function:
        31. pstrlen
        33. swapCase
        34. pstrijcpy
        37. pstrcat
37
cannot concatenate strings!
length: 250, string: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAA
length: 5, string: Hello
```

```
Enter Pstring length: 1
Enter Pstring: a
Enter Pstring length: 2
Enter Pstring: bb
Choose a function:
        31. pstrlen
        33. swapCase
        34. pstrijcpy
        37. pstrcat
32
invalid option!
```

```
Enter Pstring length: 1
Enter Pstring: a
Enter Pstring length: 1
Enter Pstring: a
Choose a function:
        31. pstrlen
        33. swapCase
        34. pstrijcpy
        37. pstrcat
42
invalid option!
```