Computer Organization
Assignment 1

Submission Guidelines

To reduce likelihood of misunderstandings, please follow these guidelines:

1. Work individually.
2. Submission date is 8/2/2024, 23:59.
3. Submission is through the "submit" system.
4. Make sure that your solution compiles and runs without any errors and warnings on BIU servers.
5. In the first line of every file you submit, write in a comment your id and full name. For example: "/* 123456789 Israela Israeli */".
6. 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.
7. Use either AT&T or Intel syntax, but be consistent.
8. No grace days!

General Background

The goal of this exercise is to practice writing both basic assembly and more advanced assembly. This exercise is composed of 2 unrelated parts. The first part will be writing a simple game, and the second one will be about strings. Each part will have you write different files, and use different files for its environment. Overall, remember you need to submit the files: **"main.s"**, **"pstrings.s"** and **"func_select.s"**.

Part I - Basic Introduction

In the first part of the homework assignment we'll write a basic program using assembly, to demonstrate 'everyday' functionality. E.g: calling library functions, basic loops, conditions and defining global variables.

We'll implement a simple number guessing game :).

A number guessing game is a simple game that acts like the following: first, the program prompts the user to enter a configuration value - the seed for the rand() library function. Then, it generates a random number between 0 and N. After that, the program constantly prompts the user with the age-old question "What is your guess?". Each time, the user then inputs whatever number he thinks of. If the guess is incorrect, the program prints "Incorrect.". The application comes to a stop when either M iterations of the game are passed, or the user guesses the number correctly. Whichever comes first. If the user guesses the correct number after less than M tries, the program prints "Congratz! You won!". Otherwise, the program prints "Game over, you lost :(. The Correct answer was X." where X was the random number.

Your task for this part is to implement a number guessing with N = 10 and M = 5. Write your code under a file named "main.s" and make sure it compiles using the command "gcc main.s -no-pie".

Example outputs

```
Enter configuration seed: 1
What is your guess? 8
Incorrect.
What is your guess? 0
Incorrect.
What is your guess? 1
Incorrect.
What is your guess? 5
Incorrect.
What is your guess? 4
Incorrect.
Game over, you lost :(. The correct answer was 3
```

```
Enter configuration seed: 1
What is your guess? 2
Incorrect.
What is your guess? 6
Incorrect.
What is your guess? 3
Congratz! You won!
```

```
Enter configuration seed: 2
What is your guess? 4
Incorrect.
What is your guess? 3
Incorrect.
What is your guess? 2
Incorrect.
What is your guess? 1
Incorrect.
What is your guess? 0
Congratz! You won!
```

Part II - A more advanced task

For the second part of the homework 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 of choice.

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

- "pstrings.h" - header file where the main structure of the program and the library function's signatures 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.

You'll need to submit the files "func_select.s" and "pstrings.s" for this part.

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 the 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 letter to small letter and vice versa. Then, print the two Pstrings in the following format:

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

If choice is 33: 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 "

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 letter to a little one and vice versa.

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

Given pointers to two Pstrings, and two indices, the function copies src[i:j] into dst[i:j] 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 message should be printed: "invalid input!\n".

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
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
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
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
34
1 7
invalid input!
length: 11, string: Hello World
length: 5, string: BBBBB
```

```
Enter Pstring length: 1
Enter Pstring: a
Enter Pstring length: 2
Enter Pstring: bb
Choose a function:
        31. pstrlen
        33. swapCase
        34. pstrijcpy
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
42
invalid option!
```