CS 330 Homework

Freql: Low-level Assembly

1 Overview

Your responsibility in this homework is to gain some familiarity with the language of languages: assembly code. You will do this in the context of writing code to produce a frequency plot—a histogram—of some arbitrary numeric data.

Assembly code is very close to machine code, into which many high-level languages compile. You will gain experience with the underlying typelessness of data, the memory-oriented processing of the von Neumann architecture, and the stack management used to support subroutines. You will also look at high-level languages with enlightened gratitude.

2 Requirements

In order to complete this homework, please satisfy the requirements below.

1. Create all source and output files in a directory named `freql`.
2. Write all your code in a single assembly language file `freql.s`.
3. In your `.data` section, include two data fields of this form:

   ```
   count:
     .long 4
   numbers:
     .long 100, 299, 4, 156
   ```

   Follow this form and order exactly, and reference these names rather than literals elsewhere in your code. Testing scripts used by the grader will swap out these lines with alternate definitions to test the generality of your solution. The `numbers` field contains `count` numbers in $[0, 300)$.

4. For output, first compute the least and great values in `numbers`. Print them to stdout in the following form:

   ```
   Least: 4
   Greatest: 299
   ```

   Second, print a histogram with 30 horizontal bars. The first bar represents how frequently numbers 0–9 appear in `numbers`. If 17 such numbers occur, then the bar is comprised of 17 asterisks. The second bar represents numbers 10–19. The last bar represents numbers 290–299. Precede each bar with a space-padded, 3-digit number showing the bar's lower bound. For each bar with 1 or more asterisks, succeed the asterisks with a parenthesized count. As a comprehensive example, consider a list of numbers where 0 is the least number and 297 is the greatest. Correctly formatted output has the following form:

```
Least: 0
Greatest: 297
  0 ****** (6)
 10 ***** (5)
 20 ***** (5)
 30 **** (4)
 40
 50 ** (2)
 60 ****** (6)
 70 *** (3)
 80 ******** (8)
 90 *** (3)
100 * (1)

...

280
290 *** (3)
```

5. Create and use at least three functions in your solution:

   (a) `findmin`, which finds and returns (but does not print) the minimum value in `numbers`.

   (b) `findmax`, which finds and returns (but does not print) the maximum value in `numbers`.

   (c) `printnstars`, which takes a number as a parameter and prints that many asterisks. It prints nothing else.

   These functions must be invoked with `call` and returned from with `ret`.

6. Compile your code directly to an executable with `gcc -m32 -o freql -g freql.s`. Your code must compile and run without warnings on `thing-[012]`. Since we are using `gcc`—and not the assembler `as` and linker `ld`—use the label `main`, not `_start`. We deviate from the text I've pointed you to. The `-m32` forces a 32-bit executable. The university machines have a 64-bit operating system installed, which requires slightly different assembly than the 32-bit examples in the text and lectures.

7. Zip up your directory to `freql.zip` with the following:

```
(cd .. && zip -r freql.zip freql)
```

This command assumes your current working directory is `freql`. The *.zip file is placed in the parent directory.

8. You may wish to put the definitions of variables `w330` and `my330` in your `.bashrc` or `.zshrc` file so you don't need to type them in more than once.

```
w330=/network_shares/w_drive/c\ s/CJohnson/cs330
my330=$w330/submissions/$USER
```

9. To test your code, run the following:

```
(cd .. && $w330/freql/test_freql)
```

This script only tests a few things like proper names and basic functionality. This script does not test all requirements. You need to do your own testing too. Failure to do so will likely result in a rejected submission. After you pass these tests, the script will prompt you to submit and notify us of your submission.