Arquitetura de Computadores 2019/20

TPC 2

Deadline: 23:59, April 28, 2020

This homework consists of two individual programming exercises. You can discuss general doubts with colleagues but the solution and the code writing should be strictly individual. All solutions will be automatically compared and plagiarism cases will be punished in accordance with the regulations.

Your solutions are to be submitted to DI's Mooshak (`http://mooshak.di.fct.unl.pt/~mooshak/`). You are limited to 10 submissions for each exercise (more will be ignored!). The OS is a Linux and your program is compiled with the following command to ensure 32bits architecture is used (in case of errors or warnings your program will fail!):

```
as --32 -o prog.o prog.s
ld -m elf_i386 -o prog prog.o
```

Note that your program's output must be exactly the same as in the examples. The points obtained by mooshak will be used as guide to your grade (100 points = 20 mark).

Problem 1 (50%)

Write a program in assembly that inverts the order of the words in a sentence in memory (`msg`) and places the transformation into another vector in memory (`msgInvert`). The word separator in the original sentence is only the blank space, likewise for the result sentence. As simplification, it is assumed that the original sentence only contains letters. For instance, consider the following representations for the data section in your program:

Example 1:

```
.data       # data section (variables)
  msg: .ascii "March was an awkward month"
  MSGLEN = (. - msg)
  .comm msgInvert,MSGLEN
```

At the end of your program, the final result in the memory region `msgInvert` has to be:

```
"month awkward an was March"
```

Example 2:

```
.data       # data section (variables)
  msg: .ascii "Mary is schoolsick"
  MSGLEN = (. - msg)
  .comm msgInvert,MSGLEN
```

At the end of your program, the memory region `msgInvert` contains†:

```
"schoolsick is Mary"
```

Start your solution using the provided template (ex1.s).

† Of course, your code does not validate or correct words… ☺

Problem 2 (50%)

Write a program in assembly that sums a selection of numbers present in a vector of integers named "`numbers`". The indices of the numbers to be selected from the vector "`numbers`" are defined in a second vector named "`select`". The indices represent the positions in the vector "`numbers`" like is commonly stated in the C language. For instance, the index '0' represents the first element in the vector, i.e. `numbers[0]`, the index index '1' represents the second element in the vector, i.e. `numbers[1]`, and so on.
The vector "`select`" is not ordered and may contain repetitions, meaning that the indices may appear in any position and more than once in this vector. It is assumed that the vector only contains valid indices. For instance, consider the memory representation in the following two examples:

Example 1:

```
.data       # data section (variables)
  numbers: .int 1, 2, 3, 4
  LEN = (. - numbers)/4
  select: .int 3, 0   # indices of the numbers to be selected
  LEN2 = (. - select)/4
  sum: .int 0    # should contain the sum of the selected numbers
```

The numbers to be added are the ones in positions 3 and 0, i.e. the values 4 and 1, respectively. At the end of the execution of your program, the `sum` variable should contain the value 5.

Example 2:

```
.data       # data section (variables)
  numbers: .int 35, 2, 5, 17, 10
  LEN = (. - numbers)/4
  select:  .int 2, 0, 4, 2  # indices of the numbers to be added
  LEN2 = (. - select)/4
  sum: .int 0    # should contain the sum of the valid selected numbers
```

The numbers to be added from the vector "`numbers`" in this case are, the ones with indexes 2 (two), 0 (zero), 4 (four), and 2 (two), i.e. the values 5, 35, 10, and 5, respectively. At the end of the execution of your program, the `sum` variable should contain the value 55.

Start your solution using the provided template (ex2.s).