Assignment 1 – String Search

Computer Architecture Fall 2015

Part 1 - Programming

Your task is programming a small piece of software only using assembly. You must not link to available libraries or use a higher programming language in order to generate your assembly code. You get access to a couple of implemented functions which will help you to finish the project.

The software has to fulfill the following requirements:

- Pass the input as command line arguments
- Open and read the file into memory
- Search for a given String in the file
- Return the position of the substring on stdout

Specifics

The Program

You are free to implement a string search algorithm of your choice. You will have to read two input files. One file consists of the query strings (the strings you have to find the text) separated by the newline sign, the second file consists of text in which you should search.

You might provide your code in multiple files. All those files should end with ".asm". The program itself must be able to be compiled with:

```
> as *.asm -o searcher.o
> ld searcher.o -o searcher
```

Ensure that this can be done on the machines in the terminal room. Your program is then called like

```
> ./searcher /path/to/file/queries.dat /path/to/file/text.dat
```

The output has to appear be on stdout with the query and in a next line the position of the occurrence in the text. Return -1 when the query is not part of the text. Let us assume we are looking for "banana" and "balloon", the output would look like the following:

```
> ./searcher /path/to/file/queries.dat /path/to/file/text.dat
banana
-1
balloon
789
>
```

Don't output anything else, neither any greeting message nor debug information.

Multiple Occurrences

What happens if the search string occurs several times in text? Here, you have two possibilities:

1. **Minimal**: Output the first occurrence and proceed to the next query
2. **Extensive**: Output all occurrences of the search string to stdout. Following the example from above, the output should like like:
   ```
   > ./searcher /path/to/file/queries.dat /path/to/file/text.dat
   banana
   145
   287
   …
   balloon
   789
   …
   >
   ```

To pass the assignment, the simple variant is enough. To take part in the competition (see at the end of the document), you have to implement the extensive variant.

Tips & Tricks

Svend will provide you with relevant code snippets in due course. Furthermore, you will need a lot of the code of the lab classes for your assignment.

Part 2 – Evaluation & Report

Together with the source code, you have to assemble a short report of what you have done (max. 4 pages total). Describe the algorithm you have decided to implement, and its complexity (in terms of big O notation) for preparation and searchtime, dependent on text length n and query length m.

The second part of your report presents a short evaluation of the actual runtime of your program. Make sure, you are testing a meaningful combination of the following parameter:

- Varying text length
- Varying query length
- Varying numbers of queries

Specify the exact protocol you are using and how you created your testdata. Make sure to present your findings appropriately (e.g., in tables, diagrams, heatmaps). Measure the runtime of your program using the command `time`, or use any other suitable method. Specify the CPU and setting you have used for your measurement.

Grading, Submission & Deadlines

Grading

This project will be evaluated with pass/fail marks. **You can work in teams up to 3 persons**. In order to pass, you must provide the following items:

Zip-File containing:

- A README file containing your team's names
- All source code files in order to assemble your program in the subfolder asm/
- Your report as pdf in the subfolder report/
- Do not include any parent folders or any other content

Your source code:

- Must be able to be compiled as described above (no further steps or parameters are allowed) and run on the terminal machines as described
- You have implemented at least the minimal requirement version
- **Has to be commented** and you have to explain your steps and decisions in the code
- We highly discourage the use of the Intel syntax, as we won't be able to provide any help to you if you are not using the AT&T syntax.

The report:

- The report (4 pages max) must present the results of your evaluation in a structured, scientific way, following the protocol given above
- Also comment and discuss the results instead of plainly presenting them

Submission

Please upload the zip-File with the contents exactly as described above to blackboard using the to the according SDU assignment tool. We will not accept incomplete zip-Files or files provided by different means than the blackboard system.

Deadline

Upload your zip not later than

Sunday, 8th of November, 23:59:59 Danish Time

Is there something else?

Yes, indeed. As searching is a quite long standing problem and the algorithms are not exactly what we would call cutting edge, we still want to give you a motivation to maybe not use the most straight forward algorithm. We hereby announce two challenges:

- Write the fastest program fulfilling all requirements
- Write the smallest program, i.e., which program results in the smallest binary, fulfilling all requirements

We will test the speed with an increasing length of the queries, text and number of queries. That means, we will test sufficiently enough that the time spent for preprocessing the text will provide you an advantage.

If you can achieve both goals, you will win both prices (Yes, there will be prices). You can also hand in two versions, one optimized for speed, one for size. Put the program you want to be graded on into the folder asm/, your additional program in the folder challenge/. For the challenge it is highly recommended to NOT use the snippets provided, as they are neither efficient nor compact but easy-to-understand. Of course, you can also use MMX or SSE instruction, if you like. The only requirement is, that the program still is compile-able on the computers in the terminal room and do not use any libraries.