Technische Universität München
Lehrstuhl für Rechnertechnik und Rechnerorganisation

Carsten Trinitis

Course IN2075: Microprocessors

Winter Course 14/15

Exercise 1

Overview

In this exercise, we will investigate the C-routine ‘toupper’, which changes all lower-case characters of a string into upper-case characters. The goal is to optimise the routine at source code level, first by using different compiler optimisations, and finally by hand-crafted assembler code.

Which team implements the fastest ‘toupper’-routine?

Preparation

For now, open a Terminal on your favourite Linux machine or log into a Linux machine remotely!
If you are using a Microsoft operating system, you can use the programs ‘putty‘ and ‘winscp’ to log in and copy files to and from the Linux machine.

Transfer the file ex1.tar.gz to the Linux machine and log in. Then type:

```
tar xzvf ex1.tar.gz
```

Then type:

```
cd uProc1415
make -s PARAMS="-d -l 8000"
```

for a first test (As you can see, ‘toupper’ is not working yet).

Exercise 1.1

Implement the function **`toupper_simple`** using a simple loop which checks for each character of text whether it is in the range ‘a-z’ and subtracts the value 0x20 from it if necessary.
Test your implementation.

Furthermore, implement the function **`gettime`**. It should deliver the current system time in a double. You should use the system call **`gettimeofday`** for implementing this function.

You can now measure the time the function takes to perform the ‘toupper’ operation, try:

```
make PARAMS="-d -l 200000 400000 10000"
```

In case the times are too short, adjust your parameters accordingly.

Exercise 1.2

Implement the function **`toupper_optimised`** using assembly code or intrinsics and try to make it as fast as possible.
Compare the runtime results to those of the different optimisation stages of **`toupper_simple`**.

Create some slides which document your progress, the measured results and possible reasons for the differences in runtime. Hint: You can get information on the system you are working on by typing **`cat /proc/cpuinfo`**.
You can vary the length of the character arrays to be processed by adapting the command line argument:

```
make PARAMS="-d -l <size_min> <size_max> <size_step>"
```

Use **`gnuplot`** to visualise the results. For instance, type

```
gnuplot
```

and then

```
plot 'results1' using 2:8 title 'simple' with linespoints, '' using 2:10 title 'optimised' with linespoints
```

within **`gnuplot`** to create a graph of your measurements.

To create a postscript file of your graph:

```
set terminal postscript
set output "graph.eps"
plot 'results1' using 2:8 title 'simple' with linespoints, '' using 2:10 title 'optimised' with linespoints
```

If you want to manually change the colours etc. of the graph, try

```
set terminal xfig
set output "graph.fig"
plot 'results1' using 2:8 title 'simple' with linespoints, '' using 2:10 title 'optimised' with linespoints
```

exit gnuplot, and type:

```
xfig graph.fig
```