Universidade de Aveiro

Mestrado Integrado em Engenharia de Computadores e Telemática

Arquitectura de Computadores Avançada

Lesson 7: Using MMX and SSE

Academic year: 2015/2016 Nuno Lau / José Luís Azevedo

The use of multimedia instruction extensions of current processors allows increasing the performance of multimedia applications and of some common operations in general computing. The **`gcc`** compiler allows the insertion of assembly instructions embedded in C code through the **`__asm__`** directive (for more details see [1]), and this feature will be used in this lab assignment for various experiments using instructions from MMX and SSE multimedia extensions.

The **`simd.tgz`** archive (available in the moodle course site) contains the files: **`sumarray.cpp`**, **`TheMMXinstructionSet.pdf`** and **`Intel-IA32-Manual.pdf`**.

For the next exercises, compile the programs with and without optimization options and take note of the execution times (e.g. **`gcc -O1 sumarray.cpp -o sumarray`**).

1. In this first exercise we are going to test the relative performance of two different functions to add arrays, and to verify the functionality of 4 different MMX arithmetic instructions.
   1.1. Analyze the **`sumarray.cpp`** program and identify the implemented functionality in the sum functions (**`sumarrayX86()`** and **`sumarrayMMX()`** functions).
   1.2. Compile and run the **`sumarray.cpp`** program. Draw conclusions about the relative performance of the sum functions.
   1.3. Replace the **`paddd`** instruction by: 1) **`paddw`**; 2) **`paddb`**; 3) **`paddusw`**; 4) **`paddusb`**. For each of the above instructions, run the program, observe the produced results and explain them.
2. Implement a function to add the contents of two integer arrays by using the **`movdqa`** and **`paddd`** instructions and SSE registers. Name that function **`sumarraySSE()`**. Test the implemented function and compare its performance with the performance of the other two available functions.
3. Create a copy of the **`sumarray.cpp`** file and change it so that the three available functions operate on **`char`** type arrays.
4. Change the program obtained in exercise 2 so that the **`sumarray`** functions can operate with arrays of any size.
5. Consider now a program to sum all elements of an array.
   5.1. Implement 3 versions (C, MMX and SSE) of a function to add all elements of an array. The function prototype should be: **`int sumelems(int *a, int size)`**.
   5.2. Optimize the functions implemented in the previous exercise by using loop unrolling and by carefully scheduling MMX and SSE instructions.
   5.3. Change the SSE version of the function implemented in exercise 2.1, in order to use the horizontal add instruction available in SSE3.

Bibliography

[1] http://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HOWTO.html (last visited on 10/10/2014)

[2] Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture, Intel Corporation, September 2009