TD SIMD

November 16, 2020

Contents

1 Introduction **1**

2 Travail a effectuer **2**
2.1 Code 2
2.2 Options de compilation 2

3 Rendu **3**

4 References: **3**

Directives:

- Ce TD sera a rendre pour au plus tard Dimanche 00:00 (minuit) sur mon courriel UVSQ: *mohammed-salah.ibnamar@uvsq.fr* avec comme objet: `[AP TD_simd - NOM PRENOM]`

Notes:

- Vous pouvez effectuer votre exploration sur une VM car l'execution du code n'est pas importante.
- Evitez de copier du code, je le verrai tres rapidement :)

1 Introduction

L'objectif de ce TD est d'explorer le jeu d'instructions SIMD present sur les processeurs x86 (Intel et AMD) dont vous disposez.

2 Travail a effectuer

2.1 Code

Vous devez implementer plusieurs versions du code **dotprod** ci-dessous en deroulant autant de fois que vous desirez la boucle interne et comparer les codes assembleurs produits par le compilateur pour les differentes versions avec differentes options de compilation.

```
double dotprod(double *restrict a, double *restrict b, unsigned long long n)
{
    double d = 0.0;

    for (unsigned long long i = 0; i < n; i++)
       d += a[i] * b[i];

    return d;
}
```

2.2 Options de compilation

Vous pouvez utiliser un ou plusieurs compilateurs, par exemple:

- GNU C Compiler **gcc**
- Intel C Compiler **icc**
- LLVM **clang**
- AMD Optimizing C Compiler **aocc**.

Il faudra compiler le programme en suivant les etapes suivantes (pour **gcc** et **clang**):

- Version de base: -O1
- Version legerement optimisee: -O2
- Version optimisee: -O3
- Version fortement optimisee: -Ofast
- Version kamikaze: -march=native -mtune=native -Ofast -funroll-loops -finline-functions -flto -ftree-vectorize

3 Rendu

Vous devez fournir une archive contenant les codes sources et les codes assembleurs analyses en plus d'un **README** qui fournit des explications pour chacune des versions de la fonction et si les instructions SIMD x86 (jeux d'instructions SSE et AVX) ont ete utilisees de facon scalaire (i.e. add[sd] ou add[ss]) ou vectorielle (i.e. add[pd] ou add[ps]).

- [ss | sd]: **ss** pour **scalar simple** precision et **sd** pour **scalar double** precision
- [ps | pd]: **ps** pour **packed simple** precision et **pd** pour **packed double** precision

Il est preferable de creer un **makefile** avec une regle par version.

Il faudra aussi fournir un fichier contenant la version du compilateur utilise:

```
#Pour GCC
$ gcc --version > gcc_ver.txt

#Pour clang
$ clang --version > clang_ver.txt
```

Et un fichier contenant les informations sur votre processeur:

```
$ cat /proc/cpuinfo > cpuinfo.txt
```

4 References:

- RTFM: man gcc
- `https://gcc.godbolt.org/`
- `https://software.intel.com/sites/landingpage/IntrinsicsGuide/`
- `https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html`