Titel des Vortrags

Name des Autors

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

2015-01-01

Gliederung (Agenda)

1 The problem at hand

2 What is vectorization?

3 Vectorizing code

4 Conclusion

5 Literatur

The Program:
Simulation/Game/Analytics which processes huge amounts of data.
It is already written in an data oriented style.

The Problem:
The execution time is way to high.

What can we do?

Steps of making code faster:

- reduce cash misses
- manual optimizations
- parallelization
- reduce overhead
- buying better hardware
- buying more hardware

Steps of making code faster:

- reduce cash misses
- manual optimizations
- parallelization
- => vectorization <=
- reduce overhead
- buying better hardware
- buying more hardware

What does vectorization mean?

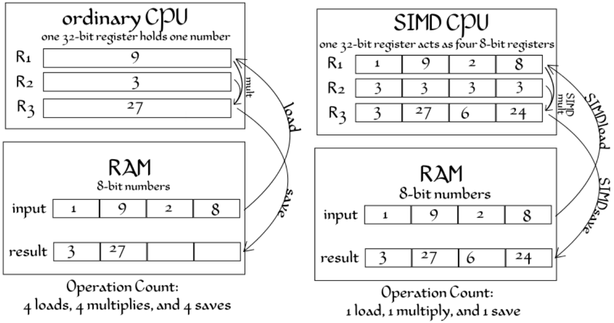
Vectorization

What is Vectorization?
The use of a cpu‘s:

- vector units

What are those units?

- special computation units
- every modern cpu implements them
- calculate multiple results from multiple inputs in one cycle

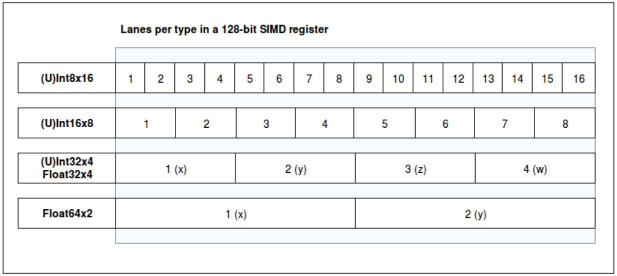


Vectorization

What is Vectorization?
The use of a cpu‘s:

- vector units
- full vector registers

Vector Registers


Lanes per type in a 128-bit SIMD register

(U)Int8x16	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(U)Int16x8	1		2		3		4		5		6		7		8	
(U)Int32x4 Float32x4	1 (x)				2 (y)				3 (z)				4 (w)			
Float64x2	1 (x)								2 (y)							

Vectorization

What is Vectorization?

The use of a cpu‘s:

- vector units
- full vector registers
- extended set of cpu instructions

Extended vector instructions

MOVAPS(x,y)z
moves a memory line starting with x and with y size
to vector register z
MOV(x,y)
moves single value from memory x into scalar register y

Instruction naming

Example:

movaps
mov = move
u = unaligned
p = packaged
s = single precision

Extended cpu vector instructions

avx/sse/avx512
different architectures provide different instruction sets (each new version introduced more)
so older architectures do not have all instructions
mention double precision instruction (when added)

What makes my code eligible for vectorization?

- calculations over arrays
- code must be in the innermost loop
- no if statements
- no uninlined function calls

How can I use vectorization?

The compiler does that for us if we tell him to.
Example for gcc:

- gcc standard optimizations does not vectorize
- -O3 enables auto vectorization
- -O3 does it by using the -ftree-vectorize flag
- -fopt-info-vec enables vectorization report

```
void test(float * vec1, float * vec2, float * res) {
    for (unsigned long i = 0; i < vector_size; i++) {
        res[i] += vec2[i] * vec1[i];
        res[i] /= vec2[i];
        res[i] -= vec1[i];
    }
}
```

- TODO add restrict explanation(compiler checks for overlaping arrays -> more assem code) restrict can be dangerous since it trusts the programmer to be right. use with caution. While explaining show assembler code

```
void test(float *__restrict vec1, float *__restrict vec2, f
    for (unsigned long i = 0; i < vector_size; i++) {
        res[i] += vec2[i] * vec1[i];
        res[i] /= vec2[i];
        res[i] -= vec1[i];
    }
}
```

TODO add explanation for alignment (compiler does not know automatically how long a type is and checks for it resulting in more assembler code)

```
typedef float_32 attribute((aligned(32)))
void test(float_32 *__restrict vec1,
          float_32 *__restrict vec2,
          float_32 *__restrict res)
{
    for (unsigned long i = 0; i < vector_size; i++) {
        res[i] += vec2[i] * vec1[i];
        res[i] /= vec2[i];
        res[i] -= vec1[i];
    }
}
```

Zusammenfassung

- Zusammenfassung 1
  - Unterpunkt 1
  - Unterpunkt 2
- Zusammenfassung 2
  - Unterpunkt 1
  - Unterpunkt 2
- Quelle: [**?**]

Literatur