Subject: Software optimization resources
Posted by Novo on Thu, 06 Jan 2011 03:03:39 GMT
View Forum Message <> Reply to Message

http://agner.org/optimize/

This web site contains five manuals, which describe everything you need to know about optimizing code for x86 and x86-64 family microprocessors, including optimization advices for C++ and assembly language, details about the microarchitecture and instruction timings of Intel, AMD and VIA processors, and details about different compilers and calling conventions.

If you liked Bit Twiddling Hacks you might like these manuals

Subject: Re: Software optimization resources
Posted by dolik.rce on Thu, 06 Jan 2011 12:33:32 GMT
View Forum Message <> Reply to Message

Thanks Novo, it looks quite interesting.

Especially the assembler library caught my attention. It replaces some common c functions (mem{cpy,move,set},str{cat,copy,len,cmp}) with assembler versions using advanced instruction sets. Out of curiosity I launched a benchmark code from the NTL/STL comparison page. But to my great disappointment, it turns out there was no recognizable gain in speed If there is someone interested enough to try to figure out why, I would be very interested

But anyway, the manuals look good

Best regards,
Honza

Subject: Re: Software optimization resources
Posted by Didier on Thu, 06 Jan 2011 21:52:41 GMT
View Forum Message <> Reply to Message

Hi Novo,

I haven't tried the samples you are talking about but for special assembler functions to work (SSE, or whatever, ...) the memory has to be aligned on 4 bytes or 64 bytes or 128 bytes or more maybe.
The alignement depends on the assembler instructions used.

If the memory is not aligned correctly either you get bad results or just poor execution timings.

Maybe this is what happens in you're case

Subject: Re: Software optimization resources
Posted by dolik.rce on Fri, 07 Jan 2011 10:01:40 GMT
View Forum Message <> Reply to Message

Didier wrote on Thu, 06 January 2011 22:52Hi Novo,

I haven't tried the samples you are talking about but for special assembler functions to work (SSE, or whatever, ...) the memory has to be aligned on 4 bytes or 64 bytes or 128 bytes or more maybe.
The alignement depends on the assembler instructions used.

If the memory is not aligned correctly either you get bad results or just poor execution timings.

Maybe this is what happens in you're case
Hi Didier,
I believe you react on my post, even though I'm not Novo

The asmlib functions take the alignment into consideration. Their internals first take care of the unaligned part using clasic instructions and then process the rest using SSE or whatever available.

After some more thinking I believe that the real reason why there was no noticeable change was badly chosen benchmark. There was probably majority of the time spent in other functions than memory and string handling. I will try again with better constructed test code.

Honza

Subject: Re: Software optimization resources
Posted by Didier on Fri, 07 Jan 2011 16:44:15 GMT
View Forum Message <> Reply to Message

Oups !!!!

Sorry for the mistake Honza.

Subject: Re: Software optimization resources
Posted by Novo on Mon, 17 Jan 2011 04:53:20 GMT
View Forum Message <> Reply to Message

I found another interesting resource related to vectorization. It was referenced in a book about the Cell processor.

http://www.freevec.org/

This is a C library, which has vectorized code for most popular functions. And it can handle unaligned data (in a petty simple way). The hardest part was to figure out how to download it.

http://www.ohloh.net/p/libfreevec (there is a "Download Page" link on top right)
