

Re: SSE2

Source: <http://coding.derkeiler.com/Archive/Assembler/comp.lang.asm.x86/2004-02/0059.html>

From: Matt Taylor (para_at_tampabay.rr.com)

Date: 01/27/04

Date: Tue, 27 Jan 2004 01:57:42 +0000 (UTC)

"Phil Carmody" <thefatphil_demunged@yahoo.co.uk> wrote in message news:8765ez0z43.fsf@nonospaz.fatphil.org...

<snip>

- > 2/3 of the *pmuludq*'s are only required for low32 results, and only
- > 1/3 need full 64 bit results. However I'm unable to find an
- > instruction that maximises 32*32->32 throughput. It seems that
- > two 32*32->64s via *pmuludq* is as good as it gets, but at least
- > providing two of them.

Yes, that is unfortunate. Both *pmullw* and *pmulhuw*/*pmulhw* are 8 cycles latency and 1 throughput, so *pmuludq* gives you the same throughput for the most part. It *might* be possible to use *pmullw*/*pmulhuw* with some swizzling to do what you want, but I think the swizzles will be more costly than any gains from parallelism. I was looking at MMX; perhaps it is worthwhile with SSE 2...

- > What do the ; n/n/n comments mean? issue cycle/latency/refractory period?

I'm not too consistent with them. With P4 I do issue/latency/throughput, and on Athlon it's decode/issue/latency. (AFAIK throughput is always 1.)

- > I don't like halving the throughput by looking only at 64-bit MMX.
- > However, if SSE2, MMX and FP can all happily co-exist, then I might
- > try to have MMX, FPU, and int units taking on some of the mults if
- > it appears that the SSE2 unit is making the other units idle.

SSE 2 is a possibility although it has higher latency. I usually favor MMX because the underlying implementation favors it. Perhaps with SSE 2 it may be worthwhile to pack/unpack to do 4 simultaneous 32x32 multiplies using *pmullw*/*pmulhuw*. The best way to find out really is to time both sequences. You are probably right; *emms* is supposed to be 12 cycles worst-case, but it is still best avoided.

- >> Intel doesn't document the latencies of most of these instructions, but
- I
- >> think it's 12 cycles total. That's less than a single *imul* (14-18). With
- >> full unrolling, you can get a large number of *pmuludqs* running in

parallel

- > > *with better throughput than imul. I looked at an SSE 2 version, but SSE 2*
- > > *latency isn't as good as MMX latency, and I don't see any easy way to get*
- > > *better throughput.*
- >
- > *With 4 of my blocks running in parallel, the latency is effectively*
- > *quartered, Do you suspect that SSE2 latency is twice MMX latency?*
- > *If so then you might be right (although some combination would*
- > *probably be best).*

In some cases it's much worse, but in others it's the same. I didn't see figures listed for pmuludq on MMX registers, but pmullw/pmulhuw were the same for both. I was assuming that pmuludq would be identical as well. Other ops like unpacks are much faster for MMX (2/1 vs. 4-6/2).

- > *When I've got code that actually works, I'll post it here for forensics.*
- > *(May be a while, I don't have a machine I can test on!)*
- >
- > *Does anyone have any good ideas about*
- > *if(a<0) a+=b;*
- > *for 64 bit values, such that a is already in [63-0] of XMMn, and b isn't?*
- > *(I might just let the 32-bit int unit do this stage rather than idling.)*

Ah, if only there were a pcmpltq. If you already have a in an SSE register, I would keep it there. The latency of adc combined with moving data between register files is much worse than a pcmpltd with appropriate logic to handle 64-bits.

-Matt