

Re: Share Your Experience with 3DNow, SSE, SSE2 etc.

## Re: Share Your Experience with 3DNow, SSE, SSE2 etc.

---

*Source:* <http://sci.tech-archive.net/Archive/sci.image.processing/2008-08/msg00043.html>

---

- *From:* Martin Brown <|||newspam|||@nezumi.demon.co.uk>
  - *Date:* Tue, 05 Aug 2008 12:29:27 +0100
- 

aruzinsky wrote:

I forgot prefetchN = 8.

On Aug 4, 11:42 am, aruzinsky <aruzin...@xxxxxxxxxxxxxxxxxxxxxxxx> wrote:

On Aug 4, 12:15 am, Hendrik van der Heijden <h...@xxxxxx> wrote:

aruzinsky schrieb:

Thank you for your input. I would feel more confident if you tested my code, though.

Exact behaviour will depend on which CPU architecture it runs on.

Then post a pointer to your code, sources preferred.  
Hendrik

Simple enough to cut and paste here, but this is an untested excerpt that assumes the arrays are a multiple of 16 floats aligned on 16 byte boundaries.

Only really worth looking at the innermost loop. It hits [edx + 64\*prefetchN] with prefetch hints every time around. I confess some surprise that it makes so much difference.

[snip]

```
align 16;
$L2:
movaps xmm0, [edx]
movaps xmm1, [edx+16]
```

Re: Share Your Experience with 3DNow, SSE, SSE2 etc.

```
movaps xmm2, [edx+32]
movaps xmm3, [edx+48]

PREFETCHNTA [edx+ecx]

movntpd [eax], xmm0
movntpd [eax+16], xmm1
movntpd [eax+32], xmm2
movntpd [eax+48], xmm3

add edx, 64
add eax, 64

dec esi
jnz $L2
```

At first glance there is a possible [AGI] partial stall on the first instruction in the loop depending on how good the branch prediction/speculative execution is. Try moving add edx, 64 up a bit.

Did you establish experimentally that a loop unrolled by 4 was optimum?

Tempting to suggest the usual peephole optimisations, and also to try out the shortest loop to see how well or badly that performs. eg.

\$L2:

```
movaps xmm0, [esi]
movntpd [ebx+esi], xmm0
add esi, 16
dec ecx ; or loop $L2
jnz $L2
```

Registers used all different ebx contains old(eax-edx) etc.

And then try loop unrolling.

But rather than optimise by trial and error, blindfolded why not enable the performance monitoring counters and do it properly by monitoring cache misses and stalls.

The Ring0 driver ia32.sys is at the University of Texas site (playing up at the moment) but I think Google cache still has a copy – including a new class library around it.

Try at your own peril but it can be very useful for tuning critical code.

<http://216.239.59.104/search?q=cache:H9fYu1GGy0EJ:iss.ices.utexas.edu/ia32.php+ia32.sys+performance+monitoring>

Regards,

Martin Brown

\*\* Posted from <http://www.teranews.com> \*\*

.