Subject: Re: [BRIDGE] Unaligned access on IA64 when comparing ethernet addresses
Posted by Eric Dumazet on Thu, 19 Apr 2007 20:29:28 GMT
View Forum Message <> Reply to Message

> From: Eric Dumazet <dada1@cosmosbay.com>
> Date: Thu, 19 Apr 2007 16:14:23 +0200
>
>> On Wed, 18 Apr 2007 13:04:22 -0700 (PDT)
>> David Miller <davem@davemloft.net> wrote:
>>
>>> Although I don't think gcc does anything fancy since we don't
>>> use memcmp(). It's a tradeoff, we'd like to use unsigned long
>>> comparisons when both objects are aligned correctly but we also
>>> don't want it to use any more than one potentially mispredicted
>>> branch.
>> Again, memcmp() *cannot* be optimized, because its semantic is to compare bytes.
>>
>> memcpy() can take into account alignement if known at compile time, not memcmp()
>>
>> http://lists.openwall.net/netdev/2007/03/13/31
>
> I was prehaps thinking about strlen() where I know several
> implementations work a word at a time even though it is
> a byte-based operation:
>
> --------------------
> #define LO_MAGIC 0x01010101
> #define HI_MAGIC 0x80808080
> ...
> sethi %hi(HI_MAGIC), %o4
> ...
> or %o4, %lo(HI_MAGIC), %o3
> ...
> sethi %hi(LO_MAGIC), %o4
> ...
> or %o4, %lo(LO_MAGIC), %o2
> ...
> 8:
> ld [%o0], %o5
> 2:
> sub %o5, %o2, %o4
> andcc %o4, %o3, %g0
> be,pt %icc, 8b
> add %o0, 4, %o0
> --------------------
>

> I figured some similar trick could be done with strcmp() and
> memcmp().
>
>

Hum, I was refering to IA64 (or the more spreaded x86 arches), that is litle endian AFAIK.

On big endian machines, a compiler can indeed perform some word tricks for memcmp() if it knows at compile time both pointers are word aligned.

PowerPc example (xlc compiler)

```
int func(const unsigned int *a, const unsigned int *b)
{
return memcmp(a, b, 6);
}

.func:                          # 0x00000000 (H.10.NO_SYMBOL)
        l       r5,0(r3)
        l       r0,0(r4)
        cmp     0,r5,r0
        bc      BO_IF_NOT,CR0_EQ,__L2c
        lhz     r3,4(r3)
        lhz     r0,4(r4)
        sf      r0,r0,r3
        sfze    r3,r0
        a       r0,r3,r0
        aze     r3,r0
        bcr     BO_ALWAYS,CR0_LT
__L2c:                          # 0x0000002c (H.10.NO_SYMBOL+0x2c)
        sf      r0,r0,r5
        sfze    r3,r0
        a       r0,r3,r0
        aze     r3,r0
        bcr     BO_ALWAYS,CR0_LT
```

But to compare 6 bytes, known to be aligned to even addresses, current code is just fine and portable. We *could* use arch/endian specific tricks to save one or two cycles, but who really wants that ?