

## Re: NdisInterLockedIncrement/Decrement macros

---

*Source:*

<http://www.tech-archive.net/Archive/Development/microsoft.public.development.device.drivers/2007-01/msg00428.html>

---

- *From:* [soviet\\_bloke@xxxxxxxxxxxx](mailto:soviet_bloke@xxxxxxxxxxxx)
  - *Date:* 14 Jan 2007 03:05:10 -0800
- 

Stephan,

How about "LOCK MOV ..."?

You are allowed to use LOCK prefix only with the instructions that I mentioned above, and MOV is not among them. Therefore, "LOCK MOV ..." is undefined opcode. You will get an exception if you try something like that – even if your code passes through compiler, you still have no chance to get it actually executed. Please check Intel Developer's Manual, Volume 2 for more info....

Anton Bassov

Stephan Wolf [MVP] wrote:

soviet\_bl...@xxxxxxxxxxxx wrote:  
[..]

The problem that, in order to use interlocked operation, you need bus locking, which applies to the following instructions:

ADD, ADC, AND, BTC, BTR, BTS, CMPXCHG, DEC, INC, NEG, NOT,  
OR, SBB,  
SUB, XOR, XADD, and XCHG.

However, reads and writes are MOV instruction, so that there is no way to lock a bus on read and write operations....

[..]

How about "LOCK MOV ..."?

Anyway, if the variable in question uses the "usual" alignment (i.e. 16-bit aligned if 16 bits wide, 32-bit aligned if 32 bits wide, don't care for 8 bits), any read operation is for sure always atomic. Simply

## Re: NdisInterLockedIncrement/Decrement macros

because the hardware will always read 16 or 32 bits as one entity then (even 64 bits on 64 bit machines).

Also, any other synchronization is useless for read operations by nature (if all bits are read at once). There is nothing to synchronize with here.

If some other task/CPU/DMA/whatever writes to the same location, well, you have no control as to when this happens. This is true even if you *would* add some locking mechanism.

```
foo()
{
AcquireSomeLock();
ReadData();
ReleaseSomeLock();
}
```

The use of the locking mechanism is just useless here. So this is equivalent to:

```
foo()
{
ReadData();
}
```

If (and only if) the read variable is *not* aligned, e.g. if it crosses any 32-bit alignment boundary, then you need to add some locking mechanism (like the LOCK instruction on x86).

Stephan