

Re: The Linear Address Space

Source: <http://coding.derkeiler.com/Archive/Assembler/comp.lang.asm.x86/2004-04/0356.html>

From: Matt Taylor (para_at_tampabay.rr.com)

Date: 04/27/04

Date: Tue, 27 Apr 2004 15:26:40 +0000 (UTC)

"Cameron Gibbs" <odly@hotmail.com> wrote in message
news:408e0de6@news.comindico.com.au...

> *Hey people.*

>

<snip>

> *The problem is that in a computer there is Hard Drive storage, RAM storage
> and a heap of other devices and the like that all require byte addresses.*

> *Say for example you had a 180GByte HDD in your system, even with the 36
bit*

> *address extension mechanisms which bring the Physical Address Space up to
> 64GBytes, there are still just not enough Logical Addresses to reference
the*

> *whole of the addresses on a system.*

>

> *Also with Hard Drive partitions and with every program having to be put in
a*

> *particular spot on the Hard Drive, it is a very rare thing that one would
> write a program that is to be put at address zero.*

<snip>

Physical address space has nothing to do with the hard drive. The IDE controller (or SCSI if you prefer) is part of the physical address space. Some of its registers are accessible as memory. When for some reason data must be read off the hard drive, the OS sends a command to the IDE or SCSI controller to read the data into physical memory (RAM).

Just to make things more confusing, there is also an I/O address space. It is essentially the same thing, but it was used exclusively for devices in the 8086 days. Legacy PC hardware still uses it. I avoided the distinction between Physical address space and I/O address space above; I believe the IDE controller hardware still uses the I/O address space for commands, but I'm not too sure.

> *Also, I think that paging has something to do with RAM management and the
> "checking in" and "checking out" of data to and from RAM but frankly I can
> neither confirm nor deny this and hence I just cannot figure out what is
> actually going on with paging.*

> *Am I on the right track with paging? – Please correct!*

Partly. For the CPU, paging refers only to the mapping between linear addresses and physical addresses. OSes use the paging mechanism to implement virtual memory, using the hard drive to expand the RAM limit in the system.

> *Might the mapping to real addresses have something to do with the Page Directory Base Register (PDBR) which is Control Register 3 (CR3)?*

It most certainly does. The cr3 register points to a table (page directory). That table is indexed with the first 10 bits of the linear address. The entry indexed then points to another table (page table). That table is indexed with the next 10 bits of the linear address. The entry indexed this time points to a 4K frame of physical memory, and the lower 12 bits are then used as an offset into that frame.

The more complicated PSE/PSE36 schemes allow the translation process to stop at the page directory, and the remaining 22 bits are used as the index into a 4M frame of physical memory. PAE allows this too, but there are 21 remaining bits instead of 22, so the frame is only 2M in size.

–Matt