Intel Virtual Memory

Note: there is a figure that goes along with this web page; if you have popups disabled, it didn't appear. Here's a link to it.

In order to give an example of a practical, widely-used, and very nice multilevel paged virtual memory scheme, let's look at how Intel does it.

We'll discuss the following items:

Address Breakdown

A 32 bit virtual address is divided into a ten bit page table number, a ten bit page number, and a twelve bit offset into a page:

Page Table Number	Page Number	Offset
31-22	21-12	11-0

Translation Algorithm

The "short form" of the translation algorithm is:

1. Use the page table number to index a directory, getting the address (in physical memory) of a page table. Notice that a process may have 1024 page tables.

2. Use the page number to index the page table, getting the address in physical memory of the page containing the data. Each page table also has 1024 entries.

3. Use the offset to index the page, getting the actual data.

In the "long form" of the algorithm, the numbers following correspond to the numbers in the figure in the popup:
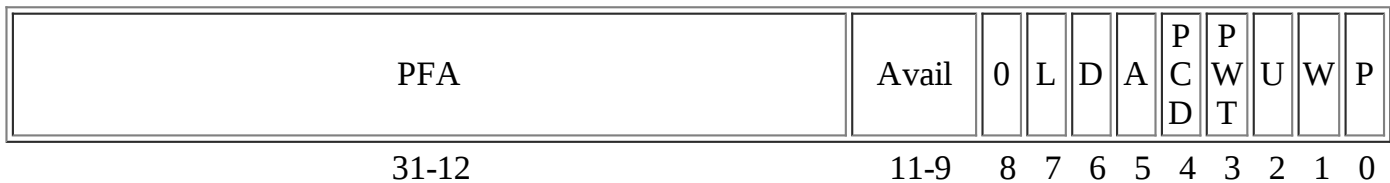
1. The incoming virtual address is divided into a page table number, a page number, and an offset.
2. The process desriptor base register (PDBR) in the CPU tells where the directory starts.
3. The page table number is multiplied by four to use as an offset into the directory, and the directory entry is looked up.
4. The directory entry contains the address of the page table, and validity and protection information. If this information says that either the page table isn't present in memory or the protections aren't OK,

the translation stops and an exception is raised.

5. The page number is multiplied by four to use as an offset into the page table, and the page table entry is looked up.
6. The page table entry contains the address of the page, and validity and protection information. If this information says that either the page isn't present in memory or the protections aren't OK, the translation stops and an exception is raised.
7. The offset is used as an index into the page.
8. The data is at the address finally arrived at.

Directory and Page Table Entry Format

Directory entries and page table entries share a common format:

PFA	Avail	0	L	D	A	PCD	PWT	U	W	P
31-12	11-9	8	7	6	5	4	3	2	1	0

Bits	Name	Interpretation
31-12	PFA	page frame address
11-9	Avail	available to OS
8	0	must be 0
7	L	PTE -- Must be 0. Dir Entry -- 4MB page
6	D	dirty (PTE only -- documented as undefined in directory entry)
5	A	accessed
4	PCD	page cache disable (can't cache data on this page)
3	PWT	page write transparent (tell external cache to use write-through strategy for this page)
2	U	user accessible
1	W	writeable
0	P	present

Note: "present" is actually checked first. If it's not present, the entire remainder of the PTE (or directory entry) is available.

Intel VM Translation Examples

Assume the following partial contents of memory for both examples. PDBR contains 001b3000.

Address	Contents
0x0001a038	0x000b4045
0x000b4b9c	0x236b12c1
0x000b91a0	0x1b9d8fc5
0x001b31cc	0x003a9067
0x001b3458	0x0001a067
0x003a9054	0x000b9067

Example 1: VM address 1cc151a0

1. Dividing the VM address up into two ten bit fields and a twelve bit field gives the following:

   Page Table Number: 0x073

   Page Number: 0x015

   Offset: 0x1a0

2. According to the PDBR, the directory starts at 0x001b3000.
3. We get the offset into the directory by multiplying the page table number by four (equivalently, left-shifting two places), giving 0x001b3000 + 0x1cc = 0x001b31cc.
4. This address contains 0x003a9067. Decoding gives us:

Bit(s)	Contains	Means
31-12	0x003a9	Page Table starts at 0x003a9000
11-7	0x00	Nothing relevant
6	1	Nothing in directory
5	1	Page has been accessed (lately)
4-3	0	Nothing relevant
2	1	User accessible
1	1	Writeable
0	1	Present

5. Adding the address of the page table (0x003a9000) to the offset into the page table (0x0015 * 4 = 0x0x054) gives 0x003a9054 as the address of the page table entry.
6. This address contains 0x000b9067. Decoding gives us:

Bit(s)	Contains	Means
31-12	0x000b9	Page starts at 0x000b9000
11-7	0x00	Nothing relevant
6	1	Page is dirty
5	1	Page has been accessed (lately)
4-3	0	Nothing relevant
2	1	User accessible
1	1	Writeable
0	1	Present

7. Adding the address of the start of the page (0x000b9000) to the offset (0x1a0) tells us the data is at 0x000b91a0 in physical memory.
8. The data at that address is 0x1b9d8fc5

Example 2: VM address 0x4580eb9c. This one is your's to figure out.

Last modified: Wed Apr 13 13:11:28 MDT 2005