Setting Up Paging With PAE

From OSDev Wiki

This page is a work in progress and may thus be incomplete. Its content may be changed in the near future.

Difficulty level

Beginner

This is a guide to setting up paging with PAE enabled. You should read Setting Up Paging first.

Contents

Differences between PAE-Paging and Legacy-Paging

- PAE allows you to access more physical memory, which is usually 64GiB (in fact, this is implementation specific).
- A new data structure is added, the so called 'Page-Directory-Pointer-Table'
- An entry is now 8-byte-wide (Legacy: 4-byte), so the number of entries is halved to 512 (Legacy: 1024)
- If the CPU supports it you can use the NoExecute-bit

Setting Up The Data Structures

As mentioned above the 'Page-Directory-Pointer-Table' is added, which contains 4 Page-Directory-Entries

```
uint64_t page_dir_ptr_tab[4] __attribute__((aligned(0x20))); //
    // ... turning out that you can put more of them into one pa
```

Keep in mind that the size of the CR3 register remains at 4byte, meaning that a PDPT must be located below 4GiB in physical memory.

Now we need our Page-Directory/-Table

```
// 512 entries
uint64_t page_dir[512] __attribute__((aligned(0x1000)));  // mus
```

```
uint64_t page_tab[512] __attribute__((aligned(0x1000)));
```

Making it run

Ok, now we have our structures. Now we have to make it run.

```
page_dir_ptr_tab[0] = (uint64_t)&page_dir | 1; // set the page d
page_dir[0] = (uint64_t)&p_tab | 3; //set the page table into th
```

Ok, let's map the first 2MiB.

```
unsigned int i, address = 0;
for(i = 0; i < 512; i++)
{
    page_tab[i] = address | 3; // map address and mark it presen
    address = address + 0x1000;
}
```

Ok, pages are mapped. Now we have to set the PAE-bit and load the PDPT into CR3

```
asm volatile ("movl %cr4, %eax; bts $5, %eax; movl %eax, %cr4");
asm volatile ("movl %%eax, %%cr3" :: "a" (&page_dir_ptr_tab)); /
```

The last thing we need to do is activating paging. Simply done:

```
asm volatile ("movl %cr0, %eax; orl $0x80000000, %eax; movl %eax
```

PAE-Paging should now be enabled.

Mapping the PD to itself

In Legacy-Paging this is quite easy. Just map the PD to the last entry of itself.

In PAE-Paging we have 4 entries and the PDPT, so how does it work? Depending on where you want to set it you just map all 4 directories into one of those! Example (PD's at end of virtual memory)

```
uint64_t * page_dir = (uint64_t*)page_dir_ptr_tab[3]; // get the
page_dir[511] = (uint64_t)page_dir; // map pd to itself
page_dir[510] = page_dir_ptr_tab[2]; // map pd3 to it
```

```
page_dir[509] = page_dir_ptr_tab[1]; // map pd2 to it
page_dir[508] = page_dir_ptr_tab[0]; // map pd1 to it
page_dir[507] = (uint64_t)&page_dir_ptr_tab; /* map the PDPT to
```

Now you can access all structures in virtual memory. Mapping the PDPT into the directory wastes quite much virtual memory as only 32 bytes are used, but if you allocate most/all PDPT's into one page then you can access ALL of them, which can be quite useful.

You can also statically allocate the PDPT at boot time, put the 4 page directory addresses in your process struct, and then just write the same PDPT address to CR3 on a context switch after you've patched the PDPT.

Categories: Level 1 Tutorials | In Progress | X86 CPU | Tutorials

---