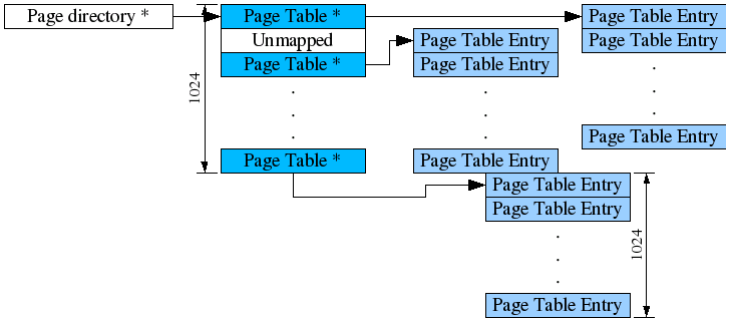
Report 4
System and device programming
Prof. Pietro Laface

Nicola Sabino
mat:253839
Date: 04/04/2019

Laboratory 4.3



Possibly you've been tapping on your calculator and have worked out that to generate a table mapping each 4KB page to one 32-bit descriptor over a 4GB address space requires 4MB of memory. Perhaps, perhaps not - but it's true. 4MB may seem like a large overhead, and to be fair, it is. If you have 4GB of physical RAM, it's not much. However, if you are working on a machine that has 16MB of RAM, you've just lost a quarter of your available memory! What we want is something progressive, that will take up an amount of space proportionate to the amount of RAM you have.

Well, we don't have that. But intel did come up with something similar - they use a 2-tier system. The CPU gets told about a page directory, which is a 4KB large table, each entry of which points to a page table. The page table is, again, 4KB large and each entry is a page table entry, described above.

This way, The entire 4GB address space can be covered with the advantage that if a page table has no entries, it can be freed and it's present flag unset in the page directory.

```
#include "monitor.h"
#include "descriptor_tables.h"
#include "timer.h"
#include "paging.h"

int main(struct multiboot *mboot_ptr)
{
    // Initialise all the ISRs and segmentation
    init_descriptor_tables();
    // Initialise the screen (by clearing it)
    monitor_clear();

    initialise_paging();
    monitor_write("Hello, paging world!  ");

        u32int *ptr = (u32int*) 0x00000000;
        u32int page = 0;
        while(page<3){
                *ptr = page;
                monitor_write("  ptr ");
                monitor_write_hex(ptr);
                monitor_write(" (page ");
                monitor_write_dec(page);
```

```
                monitor_write(") contains ");
                monitor_write_dec(*ptr);
                monitor_write(" ");
                //u32int point_something = *ptr;
                ptr += 0x00007D00;
                page += 1;
        }

        u32int *ptr2 = (u32int*) 0x00000000;
        u32int tmp = *ptr2;
        *ptr2 = *(ptr2 + 0x00007D00);
        monitor_write("page 0 now contains ");
        monitor_write_dec(*ptr2);
        monitor_write(". ");
        ptr2 += 0x00007D00;
        *ptr2 = tmp;

        monitor_write("page 1 now contains ");
        monitor_write_dec(*ptr2);
        monitor_write(". ");

    return 0;
}
```