

If this program is run on a non-VM platform,²⁴ then it will merrily execute without any apparent error. It will simply write to the 1800 words which follow the end of the array `q`. This may or may not be harmful, depending on what those words had been used for.

But on a VM platform, in our case UNIX, an error will indeed be reported, with a “Segmentation fault” message. However, as we look into how this comes about, the timing of the error may surprise you. The error is not likely to occur when `i = 200`; it is likely to be much later than that.

To illustrate this, I ran this program under `gdb` so that I could take a look at the address of `q[199]`.²⁵ After running this program, I found that the seg fault occurred not at `i = 200`, but actually at `i = 728`. Let’s see why.

From queries to `gdb` I found that the array `q` ended at `0x80497bf`, i.e. the last byte of `q[199]` was at that address. On Intel machines, the page size is 4096 bytes, so a virtual address breaks down into a 20-bit page number and a 12-bit offset, just as in Section 5.5.1 above. In our case here, `q` ends in virtual page number `0x8049 = 32841`, offset `0x7bf = 1983`. So, after `q[199]`, there are still `4096-1984 = 2112` bytes left in the page. That amount of space holds `2112/4 = 528` `int` variables, i.e. elements “200” through “727” of `q`. Those elements of `q` don’t exist, of course, but as discussed in an earlier unit the compiler will not complain. Neither will the hardware, as we will be writing to a page for which we do have write permission. But when `i` becomes 728, that will take us to a new page, one for which we don’t have write (or any other) permission; the hardware will detect this and trigger the seg fault.

5.6 Improving Performance

Virtual memory comes at a big cost, in the form of overhead incurred by accessing the page tables. For this reason, the hardware will also typically include a **translation lookaside buffer** (TLB). This is a special cache to keep a copy of part of the page table in the CPU, to reduce the number of times one must access memory, where the page table resides.

5.7 Intel Page Tables

On Intel machines, each process actually has many tables, not just one. And some of them may even be nonresident currently; in other words, even the page tables are paged!

There is a “page table table,” which serves as directory of page tables for the current process. A special register, CR3, points to the page table table.

A virtual address is broken down into three fields, rather than the two in our descriptions above:

- Bits 31-22: `i`, the index into the page table table
- Bits 21-12: `j`, index into some page table

²⁴Recall that “VM platform” requires both that our CPU has VM capability, and that our OS uses this capability.

²⁵Or I could have added a `printf()` statement to get such information. Note by the way that either running under `gdb` or adding `printf()` statement will change the load locations of the program, and thus affect the results.