If we don’t set up the interrupt descriptor table before, then pic will give us in form of an interrupt but without table interrupt will just create a general protection fault and then probably your computer will restart.

Before we tell pic to give us information, we first have to set up the interrupt descriptor table (IDT).

Entry at IDT needs

* what interrupt it is for
  + uint8\_t 🡪 0 for timer interrupt
* address in the RAM where to jump to, code for the event handler for this interrupt is somewhere in RAM
  + void\* handler
  + You have different code for different interrupts.
  + These codes should be written in assembly because you should not jump into C++ function directly because compiler might decide to put stuff which changes the registers and this might screw up the code that you have been executing before.
* some flags
* segment to tell to processor to switch to certain segment before executing the handler
  + for example if the processor currently executing code in user space and handler is in the kernel space so it needs to switch segment to kernel space
* access rights
  + number from 0 to 3
  + 3 is user space
  + 0 is kernel space

Entries are called gate descriptors.

A picture containing text

Description automatically generated

keyboard

We will write

* interruptstubs.s
  + mostly taken from <http://www.lowlevel.eu/wiki/Tyndur>
* interrupts.h
* interrupts.cpp

A screenshot of a computer

Description automatically generated with medium confidence

This is the name of the handle interrupt static method. This is what it is called to the outside. We didn’t use extern “C”.

If we get interrupt and we don’t have handler for it, we will get global protection fault and OS will crash.

We have 2 different pics:

* Master pic
* Slave pic

We need to communicate with both of them.

If you press a key on the keyboard, you will get interrupt 1.

Problem is that CPU uses interrupt 1 internally for exceptions also.

So what we are doing now is basically telling the master pic, if you get any interrupt, just add 0x20 to it and slave pic is told to add to 0x28 to it. Because both of the pics have 8 interrupts that they can use. 20 to 27, 28 to 30 (as hexadecimal). ------> check “We need to tell pic to give us interrupts” part at interrupts.cpp

We only get 1 interrupt right now, timer interrupt should give us more interrupts.

Reason is because we haven’t told the pic “thank you for the interrupt, I have finished working with this, so handling of this interrupt is finished”. We need to tell this.