Operating System

Practice 4

Домашнее задание №2

Написать программу **ticker**, которая:

1. Будет загружена multiboot загрузчиком
2. Создаст и загрузит глобальную таблицу дескрипторов
3. Настроит контроллер прерываний Intel 8252a
4. Запрограммирует интервальный таймер
5. Зарегистрирует обработчик прерываний таймера, который будет раз в секунду выводить сообщение ‘*tick*’ на экран.

Результат: Архив **<фамилия>_<группа>_os_hw2.zip**, внутри **Makefile** ---*make*--> **ticker**

Тема письма: **OS_HW2**

Дедлайн: 19.10.15 (3 недели)

Global Descriptor Table

Global Descriptor Table

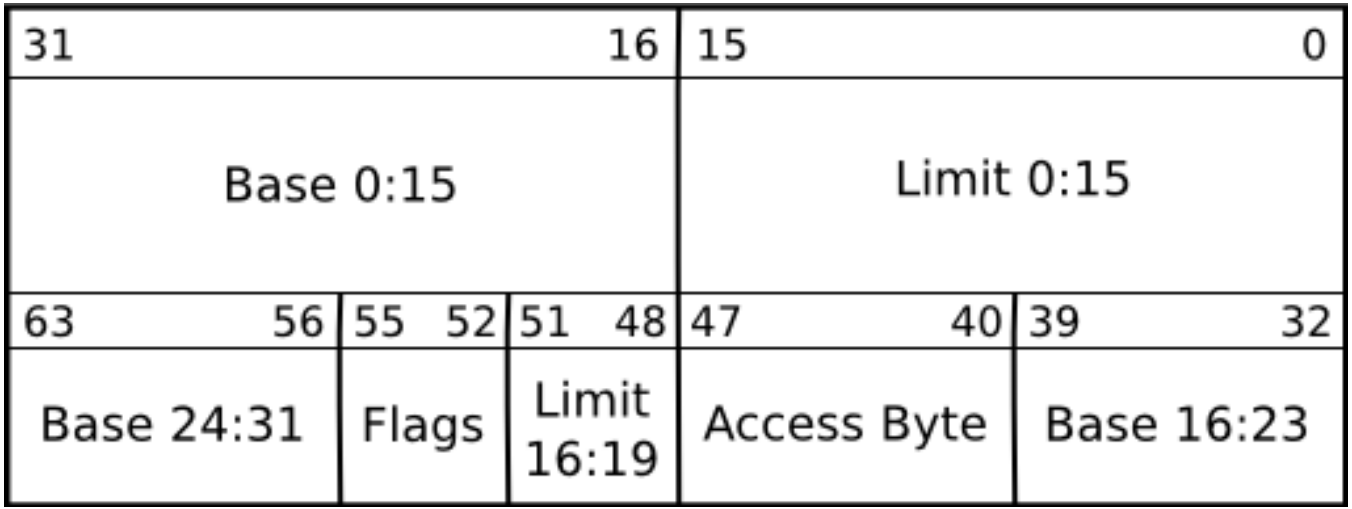
Location of GDT (address and size): **GDTR** register

Load GDT: **LGDT** <pointer to GDT>

```
struct GDTPointer {
   uint16_t size;
   uint32_t base;
}
```

http://wiki.osdev.org/Interrupt_Descriptor_Table

Global Descriptor Table



Interrupt Descriptor Table

Interrupt Descriptor Table

Location of IDT (address and size): **IDTR** register

Load IDT: **LIDT** <pointer to IDT>

```
struct IDTPointer {
   uint16_t size;
   uint32_t base;
}
```

http://wiki.osdev.org/Interrupt_Descriptor_Table

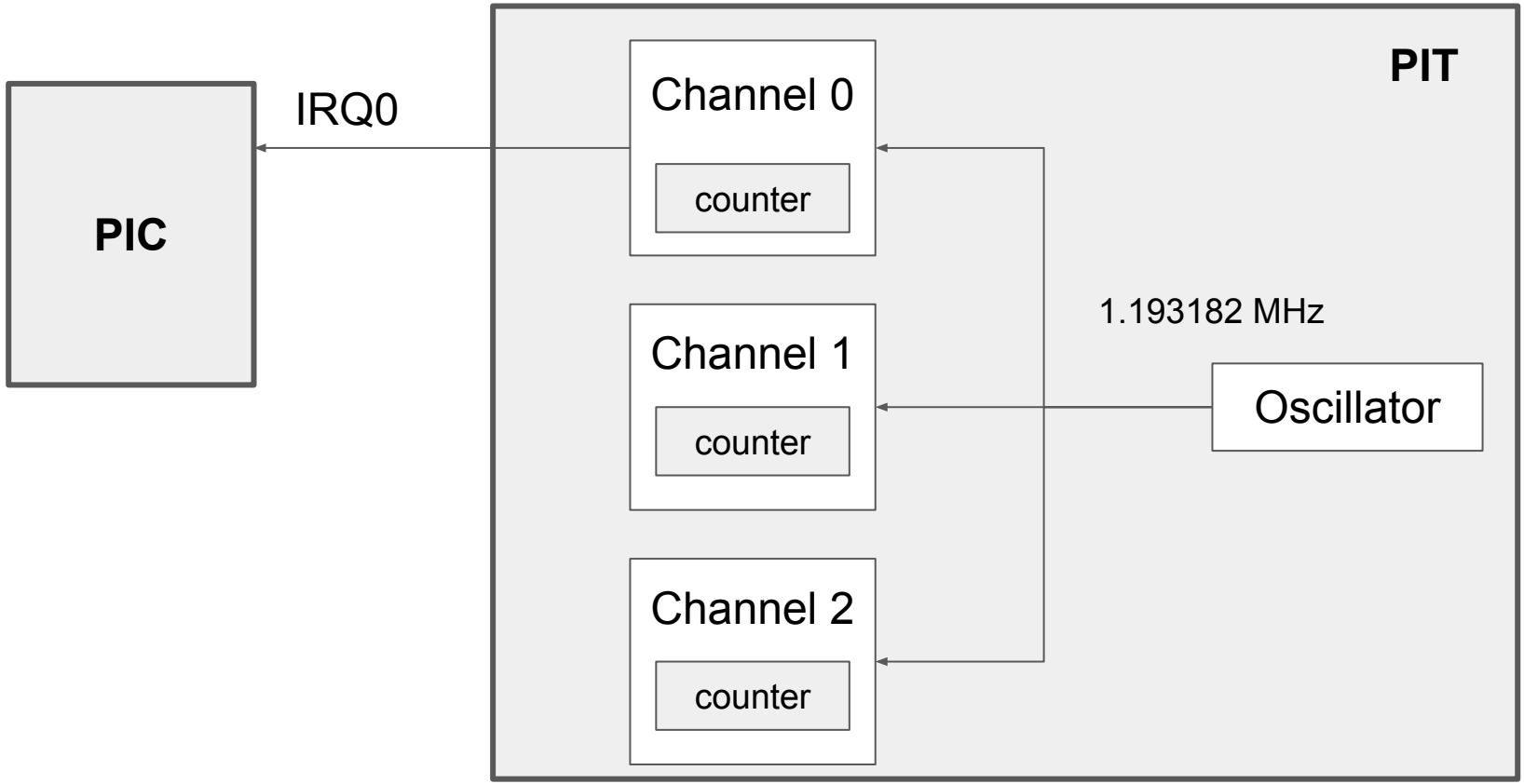
Interrupt Descriptor Table

The table contains 8-byte Gate entries:

```
struct IDTDescriptor {
   uint16_t offset_1; // offset bits 0..15
   uint16_t selector; // a code segment selector in GDT or LDT
   uint8_t zero;      // unused, set to 0
   uint8_t type_attr; // type and attributes, see below
   uint16_t offset_2; // offset bits 16..31
};
```

Programmable Interval Timer

Programmable Interrupt Timer: Intel 8253/8254



Programmable Interrupt Timer

I/O port	Usage
0x40	Channel 0 data port (read/write)
0x41	Channel 1 data port (read/write)
0x42	Channel 2 data port (read/write)
0x43	Mode/Command register (write)

Programmable Interrupt Timer

Bits	Usage	Bits	Usage
6 and 7	Select channel :	1 to 3	Operating mode :
	0 0 = Channel 0		0 0 0 = Mode 0
	0 1 = Channel 1		0 0 1 = Mode 1
	1 0 = Channel 2		**0 1 0** = Mode 2 (rate generator)
	1 1 = Read-back cmd		0 1 1 = Mode 3
4 and 5	Access mode :		1 0 0 = Mode 4
	0 0 = Latch count value cmd		1 0 1 = Mode 5
	0 1 = Access mode: low byte		1 1 0 = Mode 2 (rate generator)
	1 0 = Access mode: high byte		1 1 1 = Mode 3
	1 1 = Access mode: low/high	0	BCD/Binary mode: **0** = 16-bit binary

Programmable Interval Timer

http://wiki.osdev.org/Programmable_Interval_Timer

http://www.scs.stanford.edu/10wi-cs140/pintos/specs/8254.pdf

Programmable Interrupt Controller

Programmable Interrupt Controller

Chip	Purpose	I/O port
Master PIC	Command	0x0020
Master PIC	Data	0x0021
Slave PIC	Command	0x00A0
Slave PIC	Data	0x00A1

Programmable Interrupt Controller

http://www6.in.tum.de/pub/Main/TeachingWs2009Echtzeitsysteme/intel-8259.pdf

http://wiki.osdev.org/8259_PIC

Debugging with QEMU

Debugging with QEMU

- Open QEMU's monitor: **Ctrl-Alt-2**

or

- Run qemu with "**-monitor stdio**"

Some useful commands:

info registers : dump register state

help : list all commands

Debugging with QEMU

Run qemu with **-s -S** command line options: **qemu -s -S -kernel <mykernel>**

-s - run gdb server

-S - “freeze” CPU until ‘continue’ command

Run GDB and connect to port **1234**

gdb <mykernel>

(gdb) **target remote :1234**

Note: use GCC option **-g** to include debug information