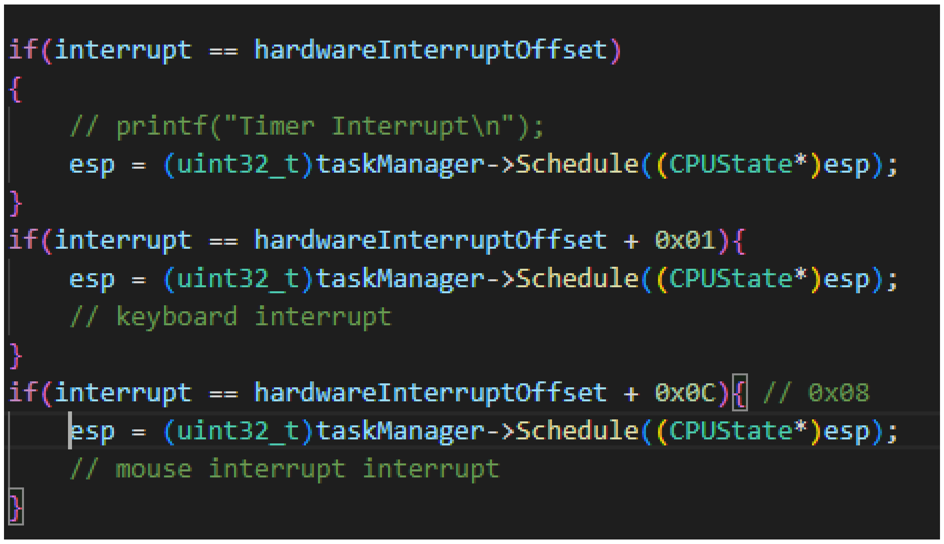
2023

Operating System Hw Report

ELIFNUR KABALCI
1801042617

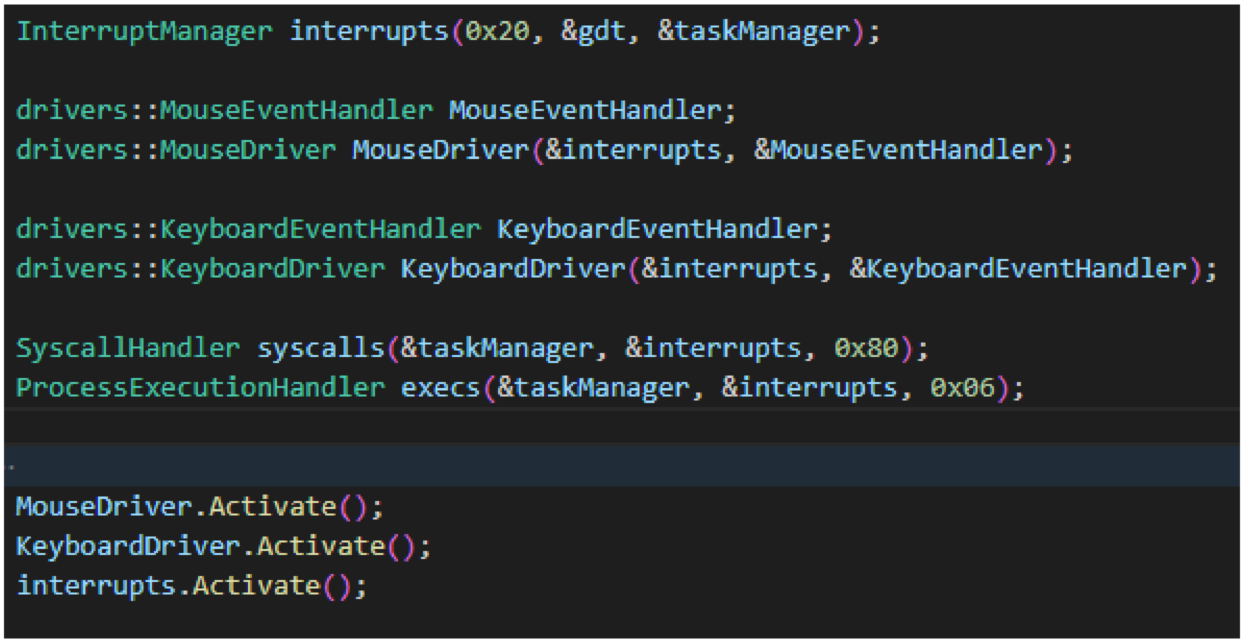
1) In this homework, I used the hw2 file shared with me as a source assignment. Because most of the required hardware interrupt handler definitions have already been made in this file and it was the one that best suited my understanding.

2) According to my research, the mouse and keyboard files in the driver contain everything that needs to be done. There was an interrupt handler belonging to a microkernel in the interrupt file. In addition, I also wrote keyboard and mouse interrupt handlers. The interrupt handler number for the keyboard is 1, and the interrupt handler number for the mouse is 12. Their hexadecimal equivalents are 0x01 and 0x0C, respectively. Each is called with CPU state and task manager scheduling method.



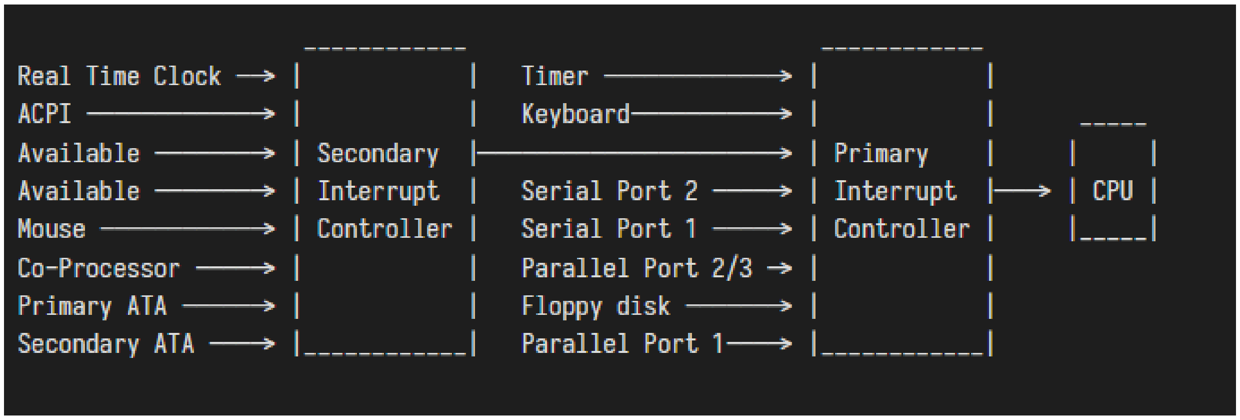
```
if(interrupt == hardwareInterruptOffset)
{
    // printf("Timer Interrupt\n");
    esp = (uint32_t)taskManager->Schedule((CPUState*)esp);
}
if(interrupt == hardwareInterruptOffset + 0x01){
    esp = (uint32_t)taskManager->Schedule((CPUState*)esp);
    // keyboard interrupt
}
if(interrupt == hardwareInterruptOffset + 0x0C){ // 0x08
    esp = (uint32_t)taskManager->Schedule((CPUState*)esp);
    // mouse interrupt interrupt
}
```

3) In order for the interrupt handler file of the created operating system to be able to intercept interrupts, the kernel needs to make a definition and call. Therefore, at the interrupt manager calling point, I made event handler definitions. I defined mouse and keyboard drivers with the defined event handlers and activated these drivers. The syscall and process execution handlers that were left over from before remained unchanged.



```cpp
InterruptManager interrupts(0x20, &gdt, &taskManager);

drivers::MouseEventHandler MouseEventHandler;
drivers::MouseDriver MouseDriver(&interrupts, &MouseEventHandler);

drivers::KeyboardEventHandler KeyboardEventHandler;
drivers::KeyboardDriver KeyboardDriver(&interrupts, &KeyboardEventHandler);

SyscallHandler syscalls(&taskManager, &interrupts, 0x80);
ProcessExecutionHandler execs(&taskManager, &interrupts, 0x06);


MouseDriver.Activate();
KeyboardDriver.Activate();
interrupts.Activate();
```

4) This code captures the user's keyboard and mouse movements and creates an interrupt accordingly. The keys pressed by the user in the keyboard movements are defined with a switch case, which returns a result based on the key pressed, and handles are made for each character in each case. Therefore, there is no need to make such a definition in interrupt.cpp. When handling the mouse, the status should be kept according to the movement of the mouse, and the data received from the buffer should be kept according to the status value. While each key on the keyboard has a corresponding value, each interrupt on the mouse creates a constant value. Therefore, data needs to be stored. Pressing and releasing each button on the mouse creates a handler. Each movement creates a handler. I found that all of these needed to be done through my research, but when I examined the code, I saw that they were already done in the mouse.cpp and keyboard.cpp files. Therefore, I did not write anything additional in the interrupt file.



```
                     ____________                           ____________
Real Time Clock --> |            |   Timer -------------> |            |
ACPI -------------> |            |   Keyboard-----------> |            |      _____
Available --------> | Secondary  |----------------------> | Primary    |     |     |
Available --------> | Interrupt  |   Serial Port 2 -----> | Interrupt  |---> | CPU |
Mouse ------------> | Controller |   Serial Port 1 -----> | Controller |     |_____|
Co-Processor -----> |            |   Parallel Port 2/3 -> |            |
Primary ATA ------> |            |   Floppy disk -------> |            |
Secondary ATA ----> |____________|   Parallel Port 1----> |____________|
```

Note: This table taken from this site: https://os.phil-opp.com/hardware-interrupts/