Travis Cripe
11519554
Lab 3 Prelab

LAB3 pre-work
DUE: 9-15-2020
Submit a (text-edited) file to TA

1. In order to use an I/O device by interrupts, 4 conditions must be met.

---

(1). The device must be allowed to generate interrupts.
(2). The SIC and VIC interrupt controllers must be programmed to route the
device interrupt through to CPU's IRQ line.
(3). When a device interrupt occurs, the CPU must be in a state to accept
interrupts.
(4). An interrupt handler (function) must be installed to handle the device
interrupts.

For the KBD driver program C3.2, answer the above questions by identifying the
lines of code that do (1) to (4):

(1). Where and How is KBD allowed to generate interrupts?
Offset Register ------ ----------

0x00 Control 0x04 Status 0x08 Data
0x0C ClkDiv 0x10 IntStatus

Bits Assignment ---------- bit 5=0(AT) 4=IntEn 2=Enable bit 4=RXF 3=RXBUSY
input scan code
(a value between 0-15)
bit 0=RX interrupt

And using the driver provided

(2). Which code lines route KBD interrupts to CPU?

```
// VIC status BITs: timer0=4, uart0=13, uart1=14, SIC=31:
KBD at 3 if (vicstatus & (1<<4)) // bit 4 => timer
interrupt
timer_handler(0);

if (vicstatus & (1<<31)){ // SIC interrupts=bit_31=>KBD at bit 3 if (sicstatus & (1<<3)){ // SIC bit 3 =>
KBD interrupt

kbd_handler();

}
```

(3). CPU runs main() in SVC mode. It enters IRQ mode to handle IRQ interrupts.

Which code line allows CPU to accept interrupts?

```
sub lr, lr, #4
stmfd sp!, {r0-r12, lr} // stack ALL registers
bl IRQ_handler // call IRQ_hanler() in C ldmfd sp!, {r0-r3,
r12, pc}^ // return
```

(4). Identify IRQ interrupts handler entry point

```
if (vicstatus & (1<<31)){ if (sicstatus & (1<<3))

kbd_handler(); }

}
```

Which function determines it's a KBD interrupt? HOW?

IRQ_handler() it scans the bits of the status register to determine the interrupt source

Where is the KBD interrupt handler?

kbd_hander()? I don’t get this question

2. In an ARM system supporting IRQ interrupts, e.g. KBD interrupts,
   the following components are needed/provided:

(1). Vector table at memory address 0
```
     0x18: LDR PC, irq_handler_addr
     irq_handler_addr: .word irq_handler
```

(2). irq_handler:
```
       sub lr, lr, #4
       stmfd sp!, {r0-r12, lr}
       bl  IRQ_handler
       ldmfd sp!, {r0-r12, pc}^
```

(3). IRQ_handler{
```
       if (VIC.statusBit31 && SIC.statusBit3)
           kbd_handler();
     }

   int hasData = 0;
   char c;
```

(4). kbd_handler()
```
     {
       get scancode;
       c = ASCII char mapped by scancode;
       hasData = 1;
     }
```

(5). char kgetc()
```
     {
     while(hasData==0);
     hasData = 0;
        return c;
     }
```

(6). main()
```
     {
        unlock();    // allow CPU to accept IRQ interrupts
```

```
        kgetc();     // CPU executes this
     }
```

Assume: the CPU executes kgetc() in main().

1. Draw a diagram to show the control flow of CPU when a KBD key is pressed

KCW's BAD Answer Example:

```
-------------------------------------------------------------
------------
                                key
In (5) at while(hasData==0); =====> (1) Reason: GOD says so
   (1) ============================> (4) Reason: CPU has a
mind of its own

--------------- YOU finish the diagram with valid reasons
-----------
```

It starts at (1) to enable the VIC and then goes to (5) to get data which then goes to (3) to make sure that it is a proper request which goes to (2) to reset the handler and get the data for it which then goes to (4) to read the bytes of the input