Day 7

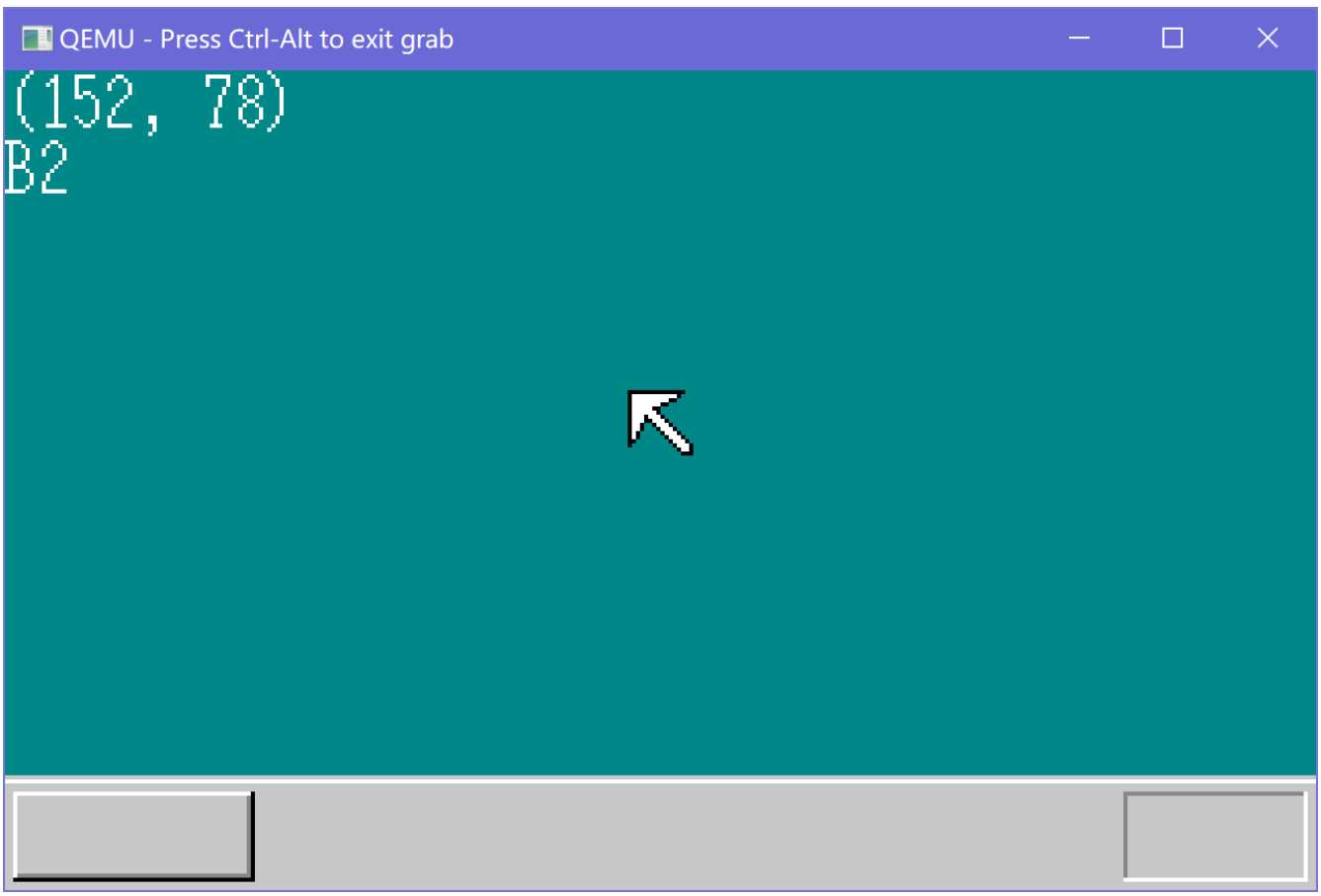
Phase 1

SubPhase 1

获取按键编码。为了获取按键编码，我们需要使用`io_in8`函数从针脚读入键盘提供的数据。读入地址是0x0060

我们要先通知PIC程序已知悉IRQ1中断（相当于重启监视），然后再从0x0060设备获取数据。代码如下

```
void inthandler21(int *esp) {
    struct BOOTINFO *binfo = (struct BOOTINFO *) ADR_BOOTINFO;
    unsigned char data, s[4];
    io_out8(PIC0_OCW2, 0x61);   /* IRQ-01受付完了をPICに通知 */
    data = io_in8(PORT_KEYDAT);

    sprintf(s, "%02X", data);
    boxfill8(binfo->vram, binfo->scrnx, COL8_008484, 0, 16, 15, 31);
    putfonts8_asc(binfo->vram, binfo->scrnx, 0, 16, COL8_FFFFFF, s);

    return;
}
```



按下`m`键的结果

SubPhase 2

使用队列来处理键盘事件

对于已经学习过数据结构的我们，在只是用数组和整形变量的情况下手撸出来一个循环队列是一个再简单不过的事情了。

```
struct FIFO8 {
    int head, tail;
    char data[QSIZE];
};

void fifo8_init(struct queue *q) {
    q->data[q->head = q->tail = 0] = 0;
}

void fifo8_status(struct queue *q) {
    return (q->tail + QSIZE - q->head) % QSIZE;
}

int fifo8_put(struct queue *q, char dta) {
    if (qsize(q) >= QSIZE - 1) return -1;
    q->data[q->tail] = dta;
    if (++q->tail >= QSIZE) q->tail = 0;
    return 0;
}

int fifo8_get(struct queue *q, char *dta) {
    if (qsize(q) <= 0) return -1;
    *dta = q->data[q->head];
    if (++q->head >= QSIZE) q->head = 0;
    return 0;
}
```

我们稍微修改一下`inthandler21`和主函数的循环就可以在保证不丢数据的情况下加快中断的响应了。

```
// partial content of "int.c"
void inthandler2c(int *esp) {
    struct BOOTINFO *binfo = (struct BOOTINFO *) ADR_BOOTINFO;
    boxfill8(binfo->vram, binfo->scrnx, COL8_000000, 0, 0, 32 * 8 - 1, 15);
    putfonts8_asc(binfo->vram, binfo->scrnx, 0, 0, COL8_FFFFFF, "INT 2C (IRQ-12) : PS/2
mouse");
    for (;;) {
        io_hlt();
    }
}

// partial content of function HariMain in "bootpack.c"
for (;;) {
        io_cli();
```

```
        if (fifo8_status(&keyfifo) == 0) {
            io_stihlt();
        } else {
            i = fifo8_get(&keyfifo);
            io_sti();
            sprintf(s, "%02X", i);
            boxfill8(binfo->vram, binfo->scrnx, COL8_008484, 0, 16, 15, 31);
            putfonts8_asc(binfo->vram, binfo->scrnx, 0, 16, COL8_FFFFFF, s);
        }
    }
```

注意我们在对队列读取数据之前用`io_cli`暂时关闭了中断，这是因为我们不希望在我们读取队列的时候因为外部中断的打断再次改变队列的内容。这样会造成混乱。我们在读取完队列的内容之后再恢复允许中断。

测试，发现运行正常。但截图上无法体现，故不再重复贴图。

Phase 2

SubPhase 1

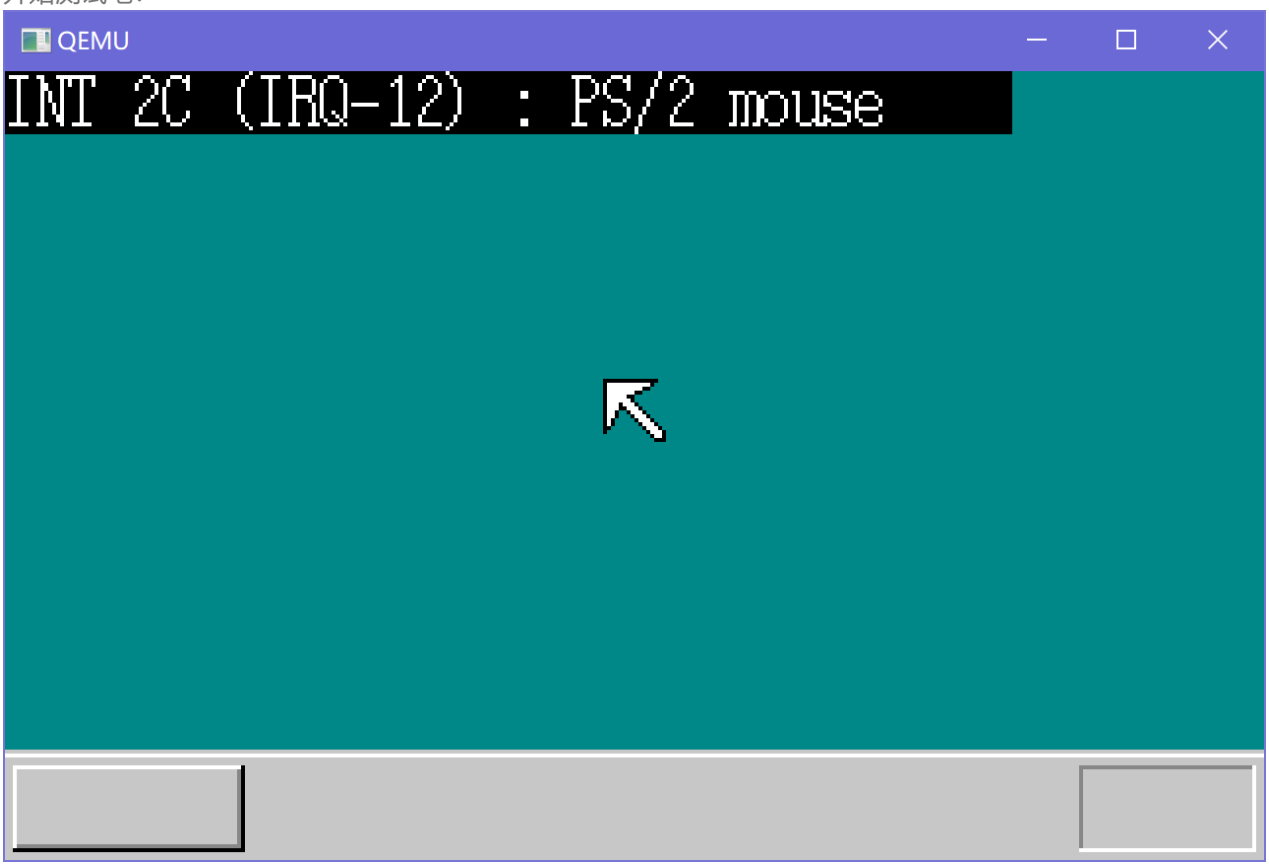
由于历史原因，当鼠标只是接入到机器上之后并不会马上工作，计算机必须先激活鼠标控制电路，然后激活鼠标。完成这些工作之后鼠标才会开始工作，鼠标控制电路才会送入中断和数据。

```
#define PORT_KEYDAT             0x0060
#define PORT_KEYSTA             0x0064
#define PORT_KEYCMD             0x0064
#define KEYSTA_SEND_NOTREADY    0x02
#define KEYCMD_WRITE_MODE       0x60
#define KBC_MODE                0x47

void wait_KBC_sendready(void) {
    for (;;) {
        if ((io_in8(PORT_KEYSTA) & KEYSTA_SEND_NOTREADY) == 0) {
            break;
        }
    }
    return;
}

void init_keyboard(void) {
    wait_KBC_sendready();
    io_out8(PORT_KEYCMD, KEYCMD_WRITE_MODE);
    wait_KBC_sendready();
    io_out8(PORT_KEYDAT, KBC_MODE);
    return;
}

#define KEYCMD_SENDTO_MOUSE     0xd4
#define MOUSECMD_ENABLE         0xf4

void enable_mouse(void) {
    wait_KBC_sendready();
```

```
    io_out8(PORT_KEYCMD, KEYCMD_SENDTO_MOUSE);
    wait_KBC_sendready();
    io_out8(PORT_KEYDAT, MOUSECMD_ENABLE);
    return;
}
```

`wait_KBC_sendready`这个函数是干什么的呢？他是让键盘控制电路做好准备动作，等待控制指令的到来。其中的道理是，键盘控制电路速度相比CPU比较慢，如果我们在它没有准备好的时候就给他发送指令，就无法保证指令能够执行，从而出现错误，得不到我们想要的结果。

> 书上小错误好多啊
>
> 另一方面，一直等着机会露脸的鼠标先生，收到激活指令以后，马上就给CPU发送答复信息：“OK，从现在开始就要不停地发送鼠标信息了，拜托了。”这个答复信息就是0xfa。
>
> 这里应该是`0xf4`吧?

> 开始测试吧!
>
> 
>

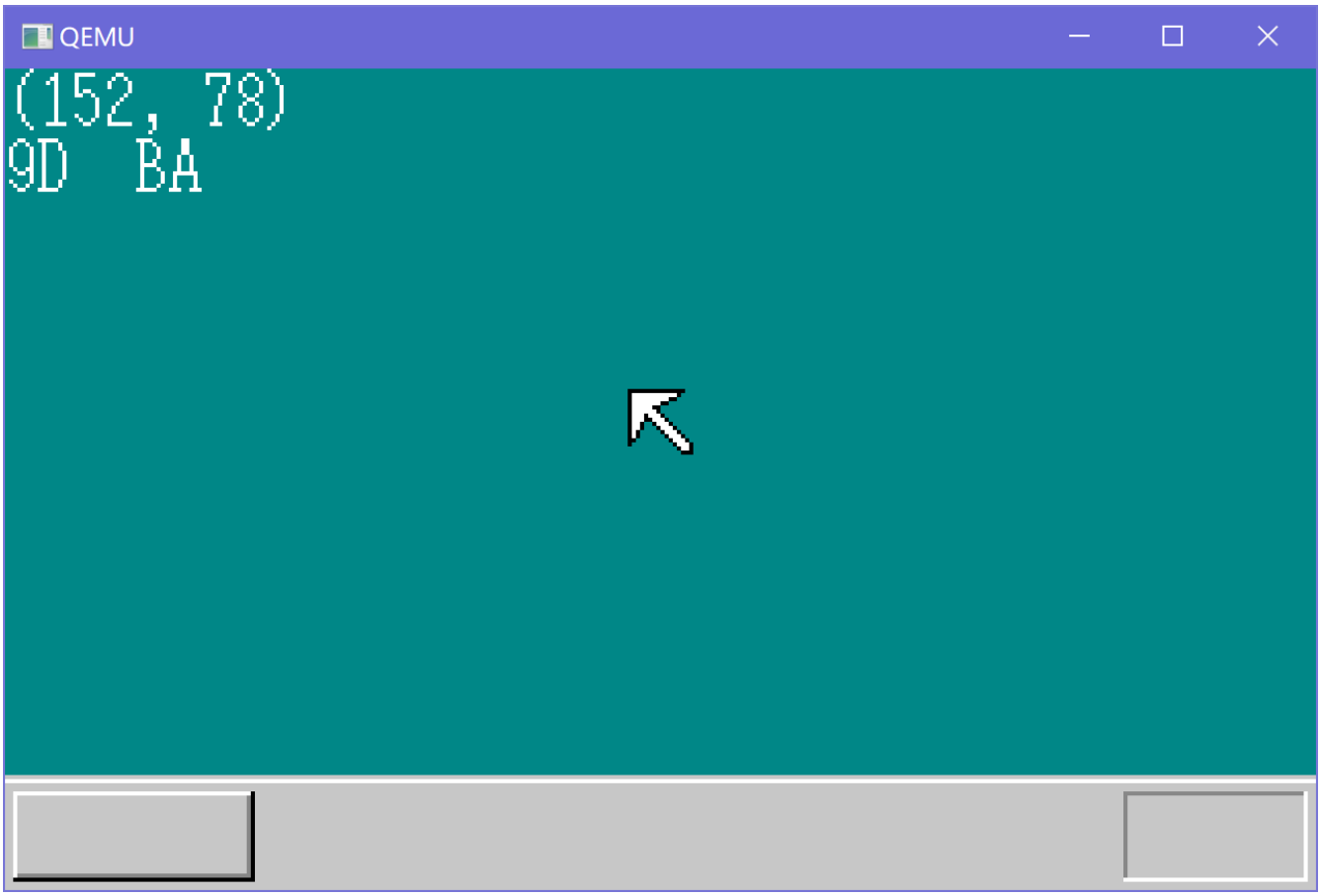
SubPhase 2

从鼠标接收数据的方法和从键盘接收数据的方法大同小异。不同之处是要先通知从PIC中断已经处理，然后再通知主PIC。原因是两个PIC之间的协调不能自动完成，需要CPU来完成。

```
// partial content of "int.c"
void inthandler2c(int *esp) {
    unsigned char data;
    io_out8(PIC1_OCW2, 0x64);
    io_out8(PIC0_OCW2, 0x62);
```

```
    data = io_in8(PORT_KEYDAT);
    fifo8_put(&mousefifo, data);
    return;
}
// partial content of function HariMain in "bootpack.c"
for (;;) {
    io_cli();
    if (fifo8_status(&keyfifo) + fifo8_status(&mousefifo) == 0) {
        io_stihlt();
    } else {
        if (fifo8_status(&keyfifo) != 0) {
            i = fifo8_get(&keyfifo);
            io_sti();
            sprintf(s, "%02X", i);
            boxfill8(binfo->vram, binfo->scrnx, COL8_008484,  0, 16, 15, 31);
            putfonts8_asc(binfo->vram, binfo->scrnx, 0, 16, COL8_FFFFFF, s);
        } else if (fifo8_status(&mousefifo) != 0) {
            i = fifo8_get(&mousefifo);
            io_sti();
            sprintf(s, "%02X", i);
            boxfill8(binfo->vram, binfo->scrnx, COL8_008484, 32, 16, 47, 31);
            putfonts8_asc(binfo->vram, binfo->scrnx, 32, 16, COL8_FFFFFF, s);
        }
    }
}
```

测试运行一下吧！



工作正常。今天就先这样吧。