L.EIC
#2.2

Laboratório de Computadores

Resumos

1. Timer
2. Interrupções
3. Keyboard
- Mouse
5. Video Card

1. Timer

Ler status / configuração

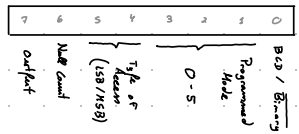
1. Escrever o **Read-Back Command** no **control register** (0x83)

Read-Back Command

Distinguir RBC de um "control word"	1
	1
Count	0
Status	0
Counter 2	1
Counter 1	1
Counter 0	1
(Reservado)	0

2. Ler do registo do timer (0x40 + timer)

↳

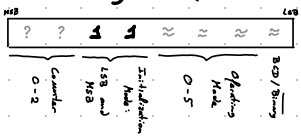


Read-Back Status

Modificar Frequências

1. Mudar o modo do timer para **MSB-after-LSB**
   a) Buscar a configuração do timer
   b) Escrever **control word** (com base na configuração) no **control register** (0x83)

↳



2. Escrever o **valor inicial** no registo do timer (0x40 + timer)

(*) valor inicial = $\frac{\text{frequência}}{1193182}$

(Freq. interna do timer)

2. Interrupções

1. Subscrever

```
HOOK_ID = bit_no;
sys_irqsetpolicy (<IRQ>, <Policy>, &HOOK_ID);
```

- bit_no | nº selecionado pelo programador de 0 a 31.
- Hook_ID | nº dado pelo Minix. único por driver.
- Linhas IRQ:
  - Timer | 0
  - Keyboard | 1
  - Mouse | 12
  - Video |
- Policies:
  - IRQ_REENABLE
    - 0x001
  - IRQ_EXCLUSIVE
    - 0x002

2. Ciclo

```
int ipc_status;
message msg;

while (<condição de saída>) {
   if (driver_receive(ANY, &msg, &ipc_status)
      continue;
   if (is_ipc_notify(ipc_status)) {
      switch (_ENDPOINT_P(msg.m_source)) {
         case HARDWARE:
            if (msg.m_notify.interrupts & BIT(bit_no)) {
               < Código a executar quando
                 receber uma notificação >
            }
            break;
         default:
            break;
      }
   } else { }
}
```

3. Cancelar subscrição

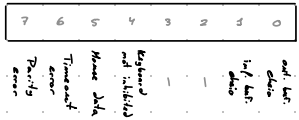
```
sys_irqsetpolicy (&HOOK_ID);
```

LC

3. Keyboard

Ler status (KBC)

Ler o conteúdo do **status register** (0x64)



Ler output buffer (KBC)

1. Verificar bits 0, 6 e 7 do **status buffer**
2. Ler do **output buffer** (0x60)

No caso do teclado:

- Envia 1 ou 2 bytes a representar a tecla
- Makecode | MSB = 0 (premir)
- Breakcode | MSB = 1 (largar)

Comandos KBC

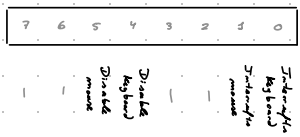
- **Comandos** enviados para o registo 0x64
- **Argumentos** enviados para o registo 0x60
- **Respostas** recebidas no **output buffer** (0x60)

0x20 | Ler Command Byte
0x60 | Escrever Command Byte
...

Command Byte

Ex. Ativar interrupções do teclado:

1. Escrever 0x20 no registo 0x64
2. Ler Command Byte do registo 0x60
3. Ativar bit 0 do Command Byte
4. Escrever 0x60 no registo 0x64
5. Escrever o Command Byte no registo 0x60



Avisos:

- O KBC é lento: ao ler/escrever em registos do KBC, faça várias tentativas (≃ 10) e acrescente um atraso de ≃ 20 ms em cada tentativa.
- Ao escrever nos registos do KBC também é preciso ter o cuidado de ler os bits 1, 6 e 7 para verificar se é ou não possível escrever.

LC

5. Vídeo

Inicializar memória

1. Popular uma struct "vbe_mode_info_t" com a informação sobre o modo de VBE usado:

```
vbe_mode_info_t* vi;
vbe_get_mode_info(vi);
```

2. Buscar o endereço base e o endereço limite do buffer:

```
unsigned vram_size = (vi->XResolution * vi->YResolution * vi->BitsPerPixel) / 8;
struct minix_mem_range mr;
mr.mr_base = (phys_bytes) vi->PhysBasePtr;
mr.mr_limit = mr.mr_base + vram_size;
```

3. Pedir permissão ao Minix

```
if(sys_privctl(SELF, SYS_PRIV_ADD_MEM, &mr)) ...
```

4. Mapeamento

```
uint8_t* video_mem = vm_map_phys(SELF, (void*) mr.mr_base, vram_size);
```

Inicializar modo gráfico

1. Preparar a instrução VBE

```
struct reg86* r;
memset(r, 0, sizeof(struct reg86));
r->intno = 0x10;
r->ah = 0x4F;
r->al = 0x02;
r->bx = video_mode | BIT(14);
```

2. Executar instrução

```
sys_int86(r);
```

3. Confirma valor de retorno

```
if(r->al == 0x4F && r->ah == 0x00) ✓ ...
```

Posição de um pixel = base_video_address[(XResolution * y + x) * bytes_per_pixel];

Pintar um pixel

```
memcpy(memory_position, (uint32_t) &color, bytes_per_pixel);
```

Cor no modo direto = (R << vi.r_pos) | (G << vi.g_pos) | (B << vi.b_pos);

⚠ Nos modos 0x110 e 0x11A, é necessário normalizar as cores (colocar bits extra a zero)

⚠ No modo indexado, as cores são representadas por índices em vez de RGB.

LC