Arquitectura de Sistemas

Práctica 3: Controlador de teclado

Gustavo Romero López

Updated: 17 de marzo de 2025

Departamento de Ingeniería de Computadores, Automática y Robótica

Objetivos

Objetivos:

- Recordar el funcionamiento de las interrupciones.
- Describir el funcionamiento del teclado.
- Escribir un controlador de teclado básico...
  1. Mínimo: que imprima cualquier cosa al pulsar una tecla.
  2. Otro que imprima los códigos de búsqueda de las teclas pulsadas.
  3. Uno último capaz de traducir códigos de búsqueda a ASCII.

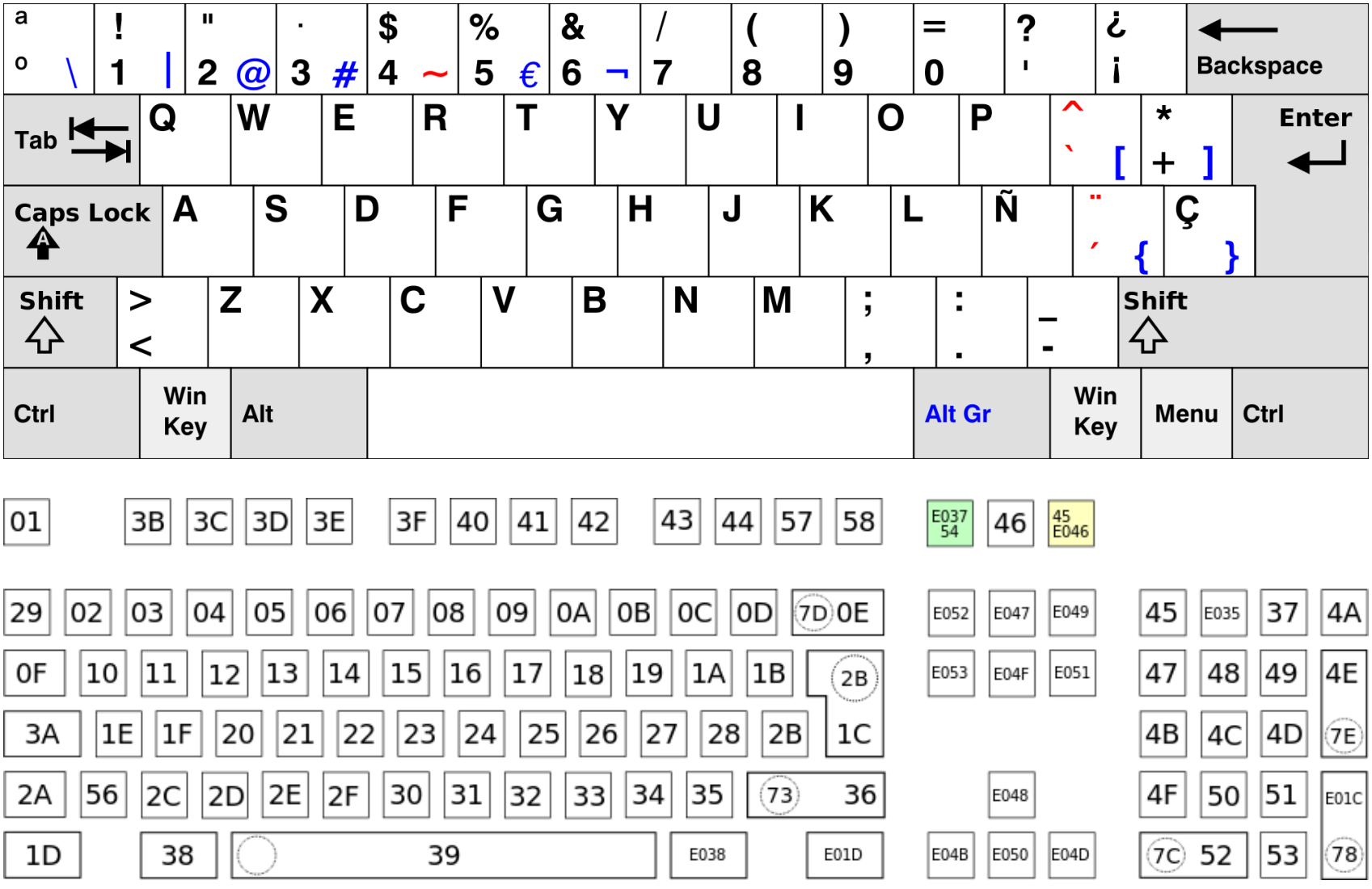
Fuentes:

- Hardware:
  `http://www.seasip.info/VintagePC/ibm_1391406.html`
- Software: `http://wiki.osdev.org/Babystep5`

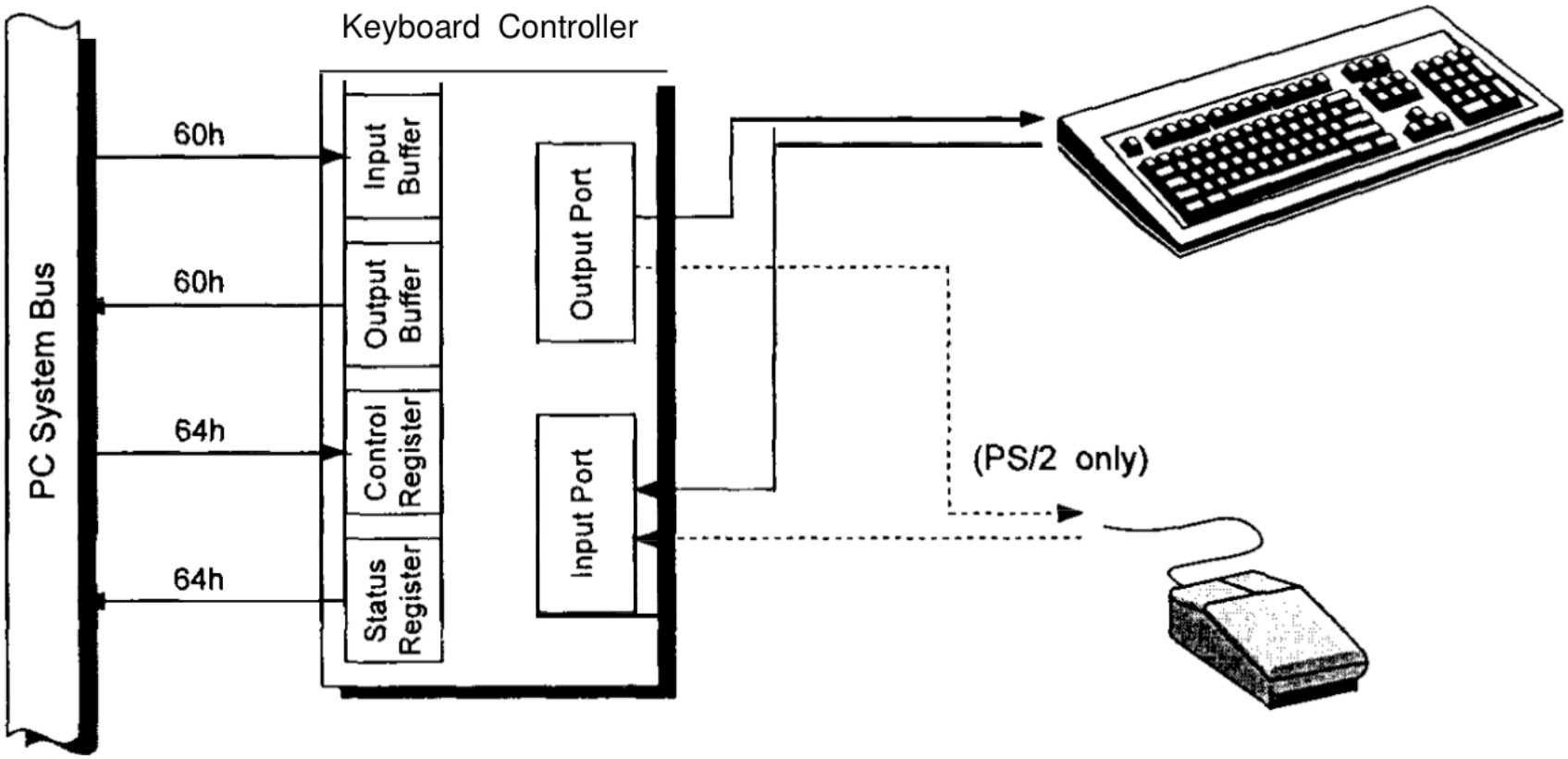
Recursos x86:

- Arquitectura
- Lenguaje ensamblador

El teclado del PC



Puertos utilizados por el teclado



Funcionamiento del teclado

- El teclado de los PCs no está hecho para generar directamente ASCII sino dos **códigos de búsquedas**: uno se emite al pulsar y otro al soltar cualquier tecla.
- Si el código de pulsación es n, al soltar se emite n+128 ó n+0x80.
- El controlador del teclado debe traducir el código de cada pulsación a su correspondiente valor en ASCII.
- Las teclas de control 

, 

, 

, 

,... deben ser tenidas en cuenta porque modifican el carácter final obtenido.

Ejemplo de funcionamiento del teclado

¿Qué pasa al intentar obtener la letra 'A' mayúscula?

1. Pulse 

, con lo que se emite el código **0x2a**.
2. Pulse 

con lo que se emite el código **0x1e**.
3. Suelte 

y se emite **0x9e** = 0x1e + 0x80.
4. Suelte 

y se emite el código **0xaa** = 0x2a + 0x80.
5. El controlador calcula el código ASCII de la 'A', **0x41**.

Cambio de manejador de interrupción

- Debemos cambiar la dirección de salto almacenada en el vector de interrupción.
- El vector de interrupción es una tabla almacenada al principio de la memoria: desplazamiento + segmento.
- La interrupción de teclado es la **9**.

Código para establecer el controlador del teclado

```
cli                          # deshabilitar interrupciones
mov $9, %bx                  # interrupción hardware del teclado
shl $2, %bx                  # dirección del vector en 9 * 4 = 36
mov $controlador, (%bx)      # cambia el desplazamiento
mov %cs, 2(%bx)              # cambia el segmento
sti                          # habilitar interrupciones
```

Fin de interrupción

- Cada vez que ejecuta el manejador de una interrupción hemos de emitir la orden de fin de interrupción, EOI, para que el controlador de interrupciones 8259 sepa que ya ha sido atendida.
- Escriba el valor **0x20** en el puerto **0x20**.
- Hay que hacerlo lo antes posible para no perder nuevas peticiones de interrupción.

orden de fin de interrupción (EOI)

```
mov $0x20, %al          # código EOI
out %al, $0x20          # enviar EOI
```

makefile I

```
.ONESHELL:

ASM = $(wildcard *.s)
ATT = $(BIN:.bin=.att)
BIN = $(OBJ:.o=.bin)
OBJ = $(ASM:.s=.o)

all: qemu | $(ATT)

clean: kill
    -rm -fv $(ATT) $(BIN) $(OBJ) core.* *~

curses: $(BIN)
    gnome-terminal --geometry 80x25 -- qemu-system-i386 -display curses
    ↪  -drive file=$<,format=raw -k es -serial mon:stdio &

debug: $(BIN) | kill
    qemu-system-i386 -drive file=$<,format=raw -k es -s -S &> /dev/null &
```

makefile II

```
    cd ..
    gdb -ix gdb_init_real_mode.txt \
        -ex 'set tdesc filename target.xml' \
        -ex 'target remote localhost:1234' \
        -ex 'hbr *0x7c00' -ex 'hbr *0x7c19'
    cd $$OLDPWD

kill:
    -killall -q qemu-system-i386 || true

qemu: $(BIN)
    qemu-system-i386 -drive file=$<,format=raw &> /dev/null &

%.att: %.bin
    objdump -D -b binary -mi8086 -Maddr16,data16 -z $< > $@

%.bin: %.o
    ld -T../linker.ld $< -o $@
```

El más sencillo: basico.s I

```
.code16                 # código de 16 bits

.text                   # sección de código
    .globl _start       # punto de entrada

#############################################################

_start:
    mov $0xb800, %ax    # 0xb800 --> ax |
    mov %ax, %es        # ax --> es     | video --> es:di = 0xb8000
    xor %di, %di        # 0 --> di      |

    cli                         # deshabilitar interrupciones
    mov $9, %bx                 # interrupción hardware del teclado
    shl $2, %bx                 # dirección del vector en 9 * 4 = 36
    mov $controlador, (%bx)     # cambia el desplazamiento
```

El más sencillo: basico.s II

```
    mov %cs, 2(%bx)          # cambia el segmento
    sti                      # habilitar interrupciones

stop:
    hlt                      # ahorra energía sin
    jmp stop                 # detener el procesador

################################################################

controlador:
    in $0x60, %al          # leer código de tecla pulsada

    mov $0x0f, %ah         # color: blanco sobre negro
    stosw                  # imprimir caracter: %ax --> %es:(%di++)

    mov $0x20, %al         # código EOI
```

El más sencillo: basico.s III

```
    out %al, $0x20          # enviar EOI

    iret                    # volver de la interrupción

#########################################################################

    .org 510                # posición de memoria 510
    .word 0xAA55            # marca del sector de arranque
```

2ª versión: impresión de códigos numéricos

1. Cree un nuevo directorio con una copia de basico.s y makefile.
2. Modifique basico.s de forma que se imprima el **código numérico** que se obtiene al pulsar cada tecla. Recuerde que al liberarla también se emite otro código diferente.

3ª versión: impresión de caracteres ASCII

1. Cree un nuevo directorio con una copia de basico.s y makefile.
2. Modifique basico.s de forma que obtenga el código de cada pulsación, lo traduzca al **carácter ASCII** equivalente e imprima dicho carácter.