Introducción a la programación para ingeniería de computadores

El teclado del PC

Gustavo Romero López

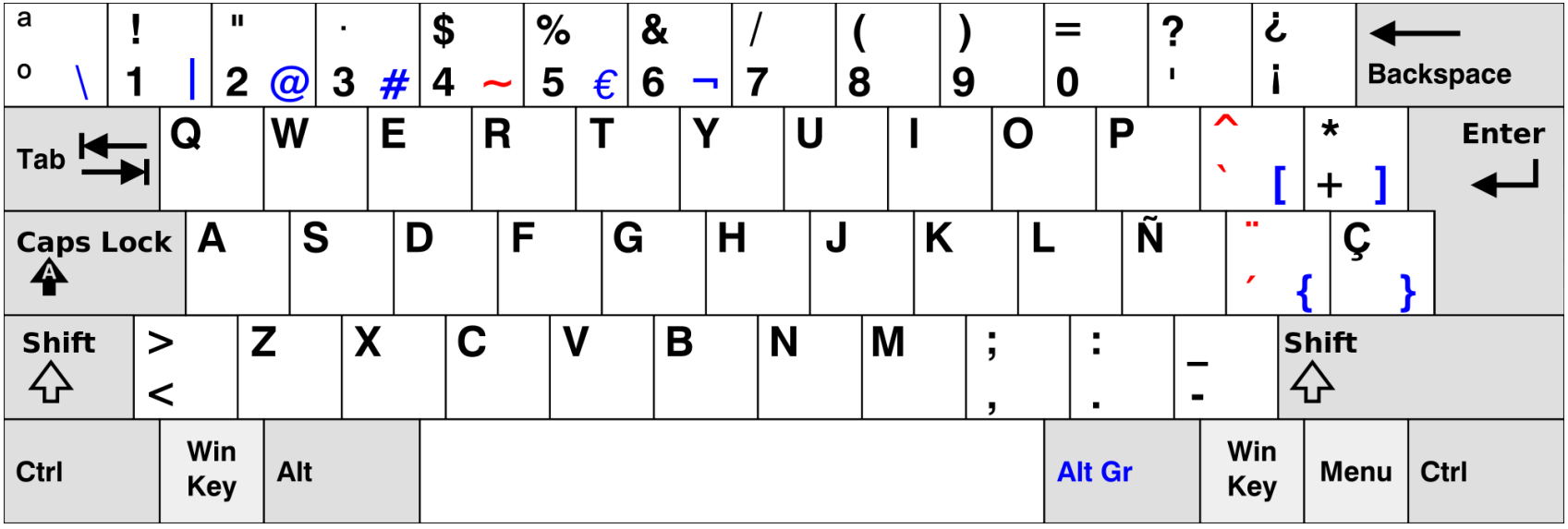
Updated: 24 de septiembre de 2024

Arquitectura y Tecnología de Computadores

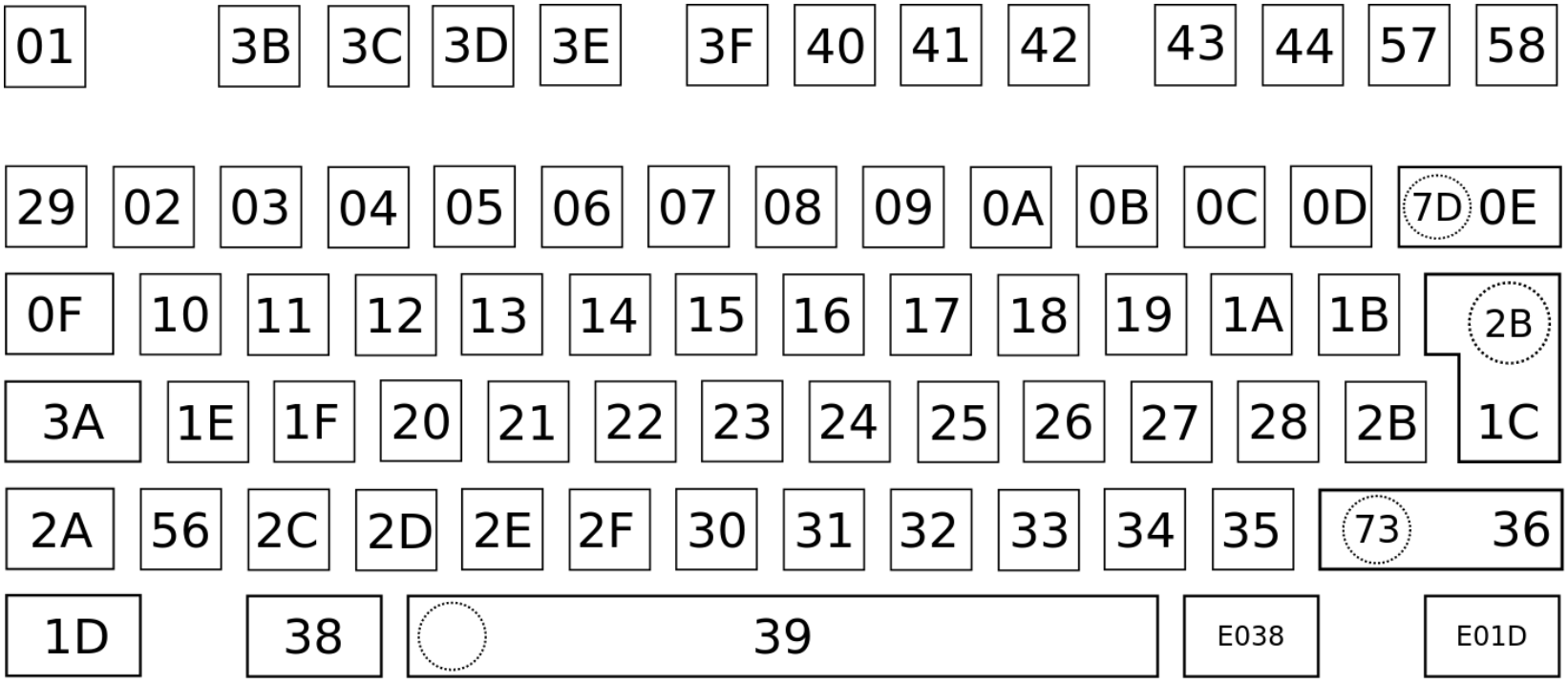
Objetivos

- Objetivos:
  - Recordar el funcionamiento de las interrupciones.
  - Describir el funcionamiento del teclado.
  - Crear una controlador de teclado mediante manejo de interrupciones en varias fases:
    1. Mínimo: que imprima cualquier cosa al pulsar una tecla.
    2. Otro que imprima los códigos de las teclas pulsadas.
    3. Uno último capaz de escribir caracteres ASCII.
- Fuentes:
  - Hardware:
    `http://www.seasip.info/VintagePC/ibm_1391406.html`
  - Software: `http://wiki.osdev.org/Babystep5`
- Recursos x86:
  - Arquitectura
  - Lenguaje ensamblador

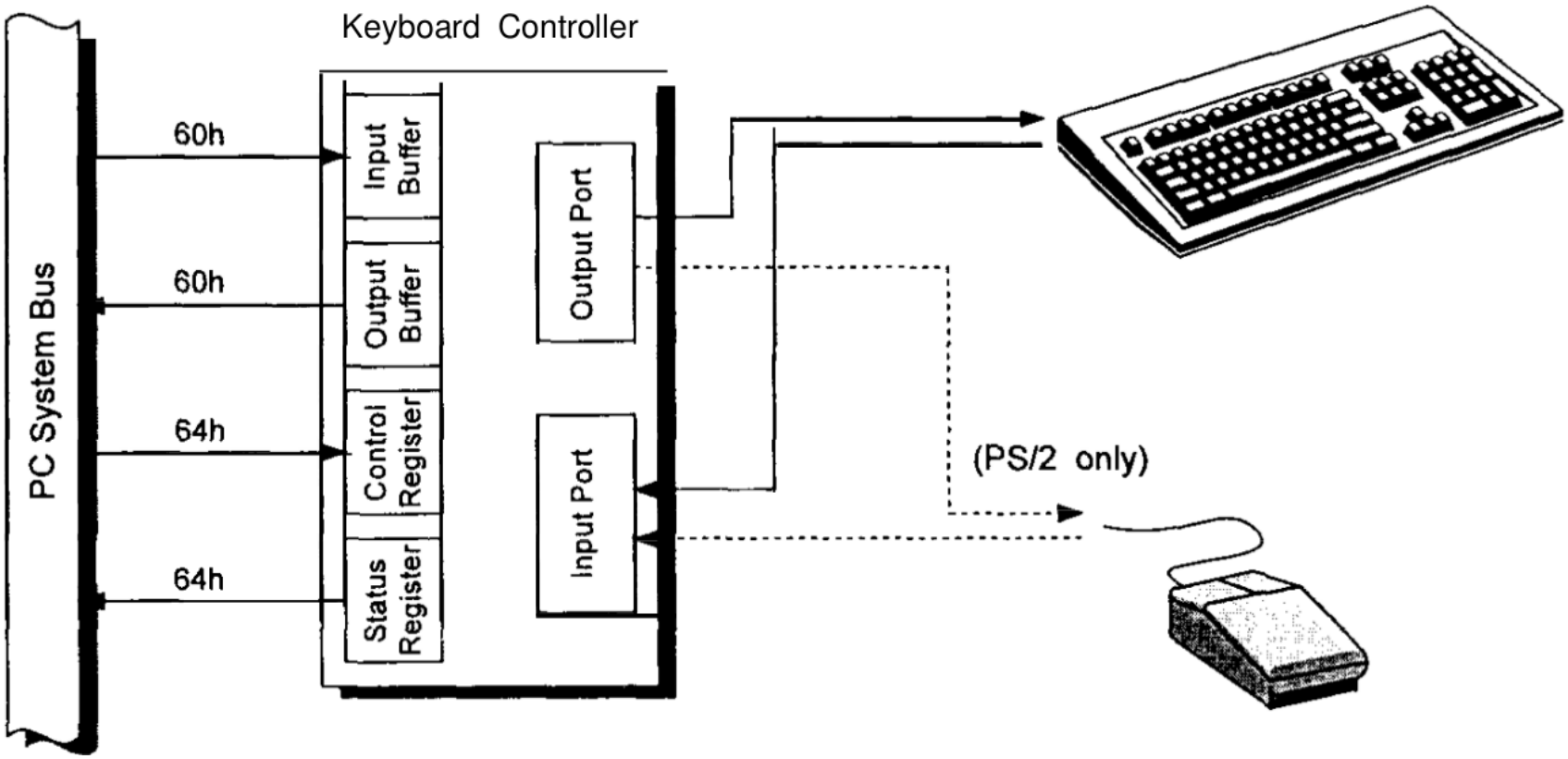
El teclado del PC



Códigos de búsqueda



Puertos utilizados por el teclado



Funcionamiento del teclado

- El teclado de los PCs no está hecho para generar directamente ASCII sino un código de búsqueda, en realidad dos: uno que se emite al pulsar una tecla y otro al soltarla. Si el código de la pulsación de una tecla es n, al soltarla se emite el código 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 (MAYÚS/CTRL/ALT) deben ser tenidas en cuenta porque modifican el carácter final obtenido.
- Ejemplo de como obtener la letra 'A' mayúscula:
  1. Pulsar la tecla [Mayús] emite el código **0x2a**.
  2. Pulsar la tecla [A] emite el código **0x1e**.
  3. Soltar la tecla [A] emite **0x9e** = 0x1e + 0x80.
  4. Soltar la tecla [Mayús] emite el código **0xae** = 0x2a + 0x80.
  5. Calcular el código ASCII de la letra 'A', **0x41**, y mostrarlo.

Cambio de manejador de interrupción

- Para cambiar el manejador de interrupción hemos de cambiar la dirección de salto almacenada en el vector de interrupción.
- Esta tabla se almacena al principio de la memoria. Primero el desplazamiento y luego el segmento del manejador.
- La interrupción de teclado es la **0x09**.
- Para evitar ser interrumpidos hemos de deshabilitar las interrupciones durante todo el proceso.

```
cli                          # deshabilitar interrupciones
mov $0x09, %bx               # interrupción hardware del tecla
shl $2, %bx                  # bx = bx * 4, dirección del vect
movw $controlador, (%bx) # cambiar el desplazamiento la in
movw %cs, 2(%bx)             # cambiar el segmento de la int.
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. Para ello es necesario escribir el valor **0x20** en el puerto **0x20**.

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

makefile I

```
.ONESHELL:

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

all: $(ATT) qemu

clean:
    -killall -q gdb qemu-system-i386 || echo "nothing to kill"
    -rm -rfv $(ATT) $(BIN) $(OBJ) core.* *~
```

makefile II

```
curses: $(BIN)
    qemu-system-i386 -display curses -drive file=$<,format=raw

debug: qemu
    pushd ..
    gdb -q \
        -ix gdb_init_real_mode.txt \
        -ex 'set tdesc filename target.xml' \
        -ex 'target remote localhost:1234' \
        -ex 'hbr *0x7c00' -ex 'hbr *0x7c20'
    popd

qemu: $(BIN)
```

makefile III

```
    qemu-system-i386 -drive file=$<,format=raw -s &> /dev/nul

%.bin: %.o
    $(LD) -T../linker.ld $< -o $@

%.att: %.bin
    objdump -D -b binary -mi386 -Maddr16,data16 $< > $@

.PHONY: all clean curses debug qemu
```

El más sencillo: basico.s I

```
.code16                 # código de 16 bits

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

_start:
    xor %ax, %ax        # ax = 0                                  |  8k c
    mov %ax, %ss        # ss = 0                                  | pila
    mov $0x9c00, %sp    # sp = 0x09c00 = 0x7c00 + 0x2000          |  ss:s

    mov $0xb800, %ax    # 0xb800 --> ax |
    mov %ax, %es        # ax --> es     | video --> es:di = 0xb8
    xor %di, %di        # 0 --> di      |
```

El más sencillo: basico.s II

```
    cli                         # deshabilitar interrupciones
    mov $0x09, %bx              # interrupción hardware del tecla
    shl $2, %bx                 # bx = bx * 4, dirección del vect
    movw $controlador, (%bx) # cambiar el desplazamiento la ir
    movw %cs, 2(%bx)            # cambiar el segmento de la int.
    sti                         # habilitar interrupciones

stop:
    hlt                         # ¿hace falta?
    jmp stop                    # bucle vacío

##################################################################
```

El más sencillo: basico.s III

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

    iret             # volver de la interrupción

##################################################
```

El más sencillo: basico.s IV

```
    .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.