

# Arquitectura de Sistemas

## Práctica 3: El teclado del PC

---

Gustavo Romero López

Updated: 7 de marzo de 2019

Arquitectura y Tecnología de Computadores

# Objetivos

## Objetivos:

- ⊙ Recordar el funcionamiento de las interrupciones.
- ⊙ Describir el funcionamiento del teclado.
- ⊙ Escribir un controlador de teclado...
  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 códigos ASCII.

## Fuentes:

- ⊙ Hardware:  
[http://www.seasip.info/VintagePC/ibm\\_1391406.html](http://www.seasip.info/VintagePC/ibm_1391406.html)
- ⊙ Software: <http://wiki.osdev.org/Babystep5>

## Recursos x86:

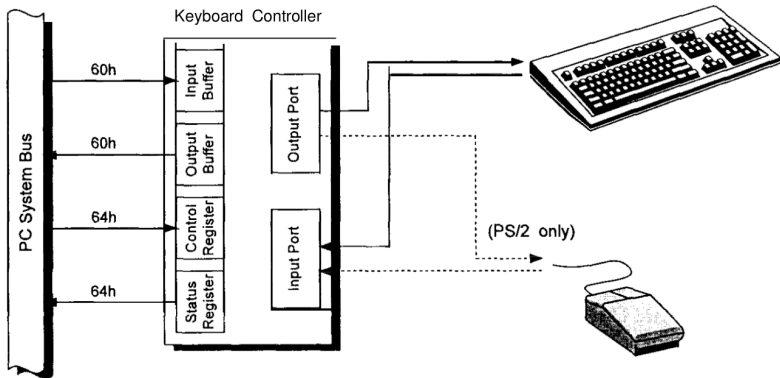
- ⊙ Arquitectura
- ⊙ Lenguaje ensamblador

# El teclado del PC

a	!	"	.	\$	%	&	/	(	)	=	?	¿	← Backspace	
o	\	1	2 @	3 #	4 ~	5 €	6 ~	7	8	9	0	'	i	
Tab	↔	Q	W	E	R	T	Y	U	I	O	P	^	*	Enter
												[	+	↩
Caps Lock	⬆	A	S	D	F	G	H	J	K	L	Ñ	"	ç	
												{	}	
Shift	⬆	>	Z	X	C	V	B	N	M	;	:	-	Shift	
		<								,	.	-	⬆	
Ctrl		Win Key	Alt								Alt Gr	Win Key	Menu	Ctrl

01	3B	3C	3D	3E	3F	40	41	42	43	44	57	58	E037 54	46	45 E046						
29	02	03	04	05	06	07	08	09	0A	0B	0C	0D	7D	0E	E052	E047	E049	45	E035	37	4A
0F	10	11	12	13	14	15	16	17	18	19	1A	1B	2B		E053	E04F	E051	47	48	49	4E
3A	1E	1F	20	21	22	23	24	25	26	27	28	2B	1C					4B	4C	4D	7E
2A	56	2C	2D	2E	2F	30	31	32	33	34	35	73	36		E048			4F	50	51	E01C
1D	38					39				E038		E01D			E04B	E050	E04D	7C	52	53	78

# 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 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 **0xae** = **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 **0x09**.

## cambio del manejador de interrupción

```
cli                # deshabilitar interrupciones
mov $0x09, %bx     # interrupción hardware del teclado
shl $2, %bx        # bx = bx * 4, dirección del vector int.
movw $controlador, (%bx) # cambiar el desplazamiento la int. teclado
movw %cs, 2(%bx)   # cambiar el segmento de la int. teclado
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 “olvidar” peticiones de interrupción.

## orden de fin de interrupción (EOI)

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



# makefile

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

default: $(ATT) qemu

clean:
    -killall -q qemu-system-i386 || true
    -rm -fv $(ATT) $(BIN) $(OBJ) core.* *~

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

%.bin: %.o
    $(LD) --oformat binary -Ttext 0x7c00 $< -o $@

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

.PHONY: clean default 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
    mov %ax, %ss        # ss = 0
    mov $0x9c00, %sp    # sp = 0x09c00 = 0x7c00 + 0x2000 | pila en ss:sp

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

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

stop:
    hlt                 # ¿hace falta?... sí!!!
    jmp stop            # bucle casi vacío
```

# El más sencillo: basico.s II

```
#####

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

    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.

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.