Escribir una rutina en ensamblador que se quede bloqueada en un bucle hasta que el bit petición de interrupción del ATC se ponga a uno

```
Esperar_IRQF PROC FAR
push ax
Inicio: MOV AL, 0Ch
OUT 70h, AL
IN AL, 71h, se guarda en el el registro C
TEST AL, 80h; and para actualizar banderas
jz inicio ; si da 0 el and significa en el bit IRQF valía 0
pop ax ; recuperamos valor de ax
ret
Esperar_IRQF ENDP
```

2) Escribir una rutina en ensamblador que lea en el registro DX la dirección base del puerto paralelo LPT2

```
Leer_LPT2 PROC FAR
push es
mov dx, 0
mov es, dx
mov dx, es:[040Ah]
pop es
ret
```

3) Usando la BIOS que se adjunta, escribir en ensamblador el código de la función de C cbrm KeyStroke(), que retorna 1 si se ha pulsado alguna tecla, 0 en caso contrario

```
_KeyStroke PROC FAR
MOV AX, 0
MOV AH, 01h
INT 16h
MOV AX, 0
jz retorno_1 , si es 0 (no se pulsó tecla)
MOV AX, 1
RET
retorno_1
MOV AX, 0
RET
_KeyStroke ENDP
```

KEYBOARD - CHECK FOR KEYSTROKE
AH = 01h
return
ZF set if no keystroke available
AH = BIOS SCAN CODE
AL = ASCII character
BIOS - Int 16h - K

4) Escribir en ensamblador el código que sea necesario para ejecutar un EOI en los manejadores de las siguientes interrupciones (una EOI + OCW2

76h (Disco Duro)
76h es del PIC-1 por lanto debe mandar a esclavo y maestro

```
mov al, 20h
out 20h, al
out A0h, al
```

02h (NMI)
no es enmascarable no necesario mandar

OCh (COM1)
OCh es del PIC-0 por lanto no es necesario mandar EOI al esclavo

```
mov al, 20h
out 20h, al
```

5) Escribir subrutina en ensamblador que programe la oscilación del RTC a 512 interrupciones por segundo

$RS = 1 + \log_2 \frac{32768}{512} = 7 = 0111\,b$

```
Rutina_int PROC FAR
push ax
mov al, 70h
out 70h, al   ; UIP  RS
mov al, 00100111b
out 71h, al   ; DV
pop ax
ret
Rutina_int ENDP
```

6) Usando la BIOS adjuntada, escribir en ensamblador el código de la función C void Imprimir_letra(char ascii)

```
-Imprimir_letra PROC NEAR
    Push bp
    mov bp, sp
    Push ax
    Push bx
    push cx
    mov al, 4[bp]
    mov bl, 10101100b
    xor bh, bh
    mov cx, 1
    mov ah, 09h
    int 10h
    pop...
    ret
Imprimir_letra ENDP
```

7) Implementar en ensamblador de 8086 la subrutina llamada Imprimir_ASCIIZ que sea de imprimir por el puerto LPT 1 una cadena ASSCIIZ cuya dirección reside en BX. La impresión debe seguir el protocolo BUSY

```
Imprimir_ASCIIZ PROC FAR
pop ax, es, dx

mov ax,0
mov es, 0
mov dx, es:[0408h] ; en dx se guarda dir base de LPT1
cont: mov al, [bx]
cmp al, 0 ; comprueba si terminó de imprimir la cadena
je final
inc dx ; obtenemos dirección de reg de estado
busy: in al, dx ; leemos registro de estado
test al, 10000000b; comprobamos busy
jz busy
dec dx, dirección de registro de datos
mov al, [bx]; leemos el caracter != 0
out dx, al colocamos caracter en reg de datos
add dx, 2 ; dirección de registro de control
in al, dx ; leemos registro de control
or al, 00000001b ; ponemos strobe a 1
out dx , al
dec dx ; registro de estado
busy-2 in al, dx
test al, 1000 0000b; comprobamos busy
jz busy-2 ; si está ocupada
inc dx registro de control
in al, dx
AND AL, 11111110b bajamos el strobe
out dx , al
inc DX , siguiente caracter
SUB dx, 2 , volvemos a dir base
jmp cont

final
    push dx, es, ax
    ret
Imprimir_ASCIIZ ENDP
```

8) Escribir en ensamblador de 80x86 el código necesario para, el puerto paralelo LPT2 genere interrupciones

```
mov ax, 0
mov es, ax
mov bx, es: [0408h]
ADD bx, 2   bx contiene dirección de registro de control
in  al, bx  al contiene el valor del registro
OR  AL, 00010000b activar IRQEN
OUT bx, AL activamos registro
```

9) Escribir en 80x86 una RSI del RTC que llame a la subrutina - Actualizar cada vez que se reciba una interrupción de actualización de la hora/fecha, se supone que RTC tiene habilitadas sus interrupciones

```
RSI_RTC PROC FAR
    sti ; activa interrupciones
pushax  mov al, 0Ch
    out 70h, al
    in al, 71h ; leemos registro C
    test al, 00010000h
    jz final ; si es 0 el and es pq UF == 0
    call _Actualizar si no es 0 es pq se debe actualizar
final
    mov al, 20h ; mandar EOI
    out 20h, al
    out 0A0h, al
    pop ax
    iret
RSI_RTC ENDP
```

12) Escribir en ensamblador codigo para escribir un EOI en los manejadores de las siguientes interrupciones

1Ch (Tic temporizador)	70h (RTC)	16h (E/S Teclado)
No es enmascarable	mov al, 20h	EOI innecesario
EOI innecesario	out 20h, al	
	out 0A0h, al	

13) Escribir en ensamblador una subrutina que ponga a 0 el bit IRQEN del LPT1 dejando intacto el resto

```
IRQENO_LPT1 PROC FAR
push ax, es, dx
mov ax, 0
mov es, 0
mov dx, es:[0408h] ; guarda en dx dirección base de LPT1
ADD dx, 2 , dirección de registro de control
in al, dx leemos valor del registro
AND AL, 11101111b ponemos a 0 IRQEN sin cambiar el resto
out dx, al
pop dx, es, ax
RET
IRQENO_LPT1 ENDP
```

14) Escribir en ensamblador 80x86 una subrutina llamada que programe las interrupciones periódicas del RTC a una frecuencia de 2 MHz

$RS = 1 + \log_2 \frac{4193404}{2} = 21.0681 \approx 22$

```
push...
mov AL, 0Ah
out 70h, AL
MOV AL, 00000010b
out 71h, AL
pop...
ret
```

19) Escribir en ensamblador de 80x86 una rutina llamada que configure la frecuencia de las interrupciones periódicas del RTC a 4Hz y habilite dichas interrupciones

$RS1 = 1 + \log_2 \frac{32768}{4} = 14 = 0000\ 1110$

```
Rutina PROC FAR
push ...
mov AL, 0Ah
out 70h, AL
mov AL, 0010 1110b
out 71h, AL
mov AL, 0Bh
out 70h, AL
IN AL, 70h
OR AL, 01000000b
OUT 70h, AL  ← hay que volver a configurar para reg C
pop...
ret
Rutina ENDP
```

Escribir en ensamblador de 8086 una rutina de servicio de la interrupción periódica del RTC que cada vez que se ejecute modifique la frecuencia actual del contador 2 del temporizador. Aumentará en 100 Hz si la frecuencia es menor que 1000 Hz.
El valor 1193182 equivale a 001234DEh, se supone que el contador 2 del timer ha sido inicializado en modo 3, cod binaria y RW=11b

Valor inicial = 1193182 / frec deseada
frec = 1193182 / Valor inicial

```
RTC_rsi PROC FAR
    sti ; activa interrupciones
    pushs                    ; Bucle para cambiar valor inicial
    MOV BX, OFFFFh
buscar: MOV DX, BX           ; en DX se guarda contador anterior
    MOV AL, 10000000b
    OUT 43h, AL      en AL, 42h   modo RW=11 por lo tanto
    en AL, 42h       MOV BH, AL   es 2 bytes el número
    mov BL, AL
    CMP BX, DX
    JBE buscar       ; mientras esté decrementando
    ; en el momento en el que el nuevo valor es mayor
      que el anterior significa que está en su máximo
    MOV AX, 34DEh
    MOV DX, 0012h
    DIV BX ; frecuencia está en AX
    CMP AX, 1000
    JGE fin
                ; si es menor que 1000 Hz
    ADD AX, 100      se suman 100 Hz
    MOV BX, AX
    MOV AL, 10110000b ; solicitamos escritura
    OUT 43h, AL
    MOV BL, AL
    IN 42h, AL
    MOV BH, AL
    IN 42h, AL
    ; la interrupción del RTC es de PIC-1 por lo que se manda EOI
      a maestro y esclavo
    MOV AL, 20h
    OUT 20h, AL
    OUT 0A0h, AL
    pops
    IRET
RTC-rsi ENDP
```

23] Los pines 2 a 9 del LPT1 controlan una hilera de 8 LEDs, de modo que cuando uno está activo a +5V, su LED está encendido. Escriba en ensamblador la función cercana VUmeter que reciba en AH el número

```
VUmeter PROC NEAR
pushs...
mov CL, AH ; guarda el nº de leds
XOR AX, AX ; ponemos AX a 0
MOV ES, AX
MOV DX, ES:[0408]
MOV AL, 0
TONLED: CMP CL, 0
JLE fin
SHL AL, 1
INC AL
JMP TONLED
OUT DX, AL ; ponemos AL en registro de datos
pops...
Ret
VUmeter ENDP
```

```
mov cl, ah
mov ax, 0FF00h  1 byte todo 1 y otro todo 0
rol ax, cl      al rotar a la vez los 1 irán
                apareciendo por la derecha todos
1 consecutivos como indique cl
```

27) Escribir en ensamblador una rutina de servicio de la interrupción del RTC que cada vez que lea un byte del puerto 1234h y lo almacene en un buffer circular de 512 bytes.

```
RSC_RTC PROC FAR                    ; lee registro C del RTC (reset de flags)
  push ... STI
  jmp inicio
  buffer DB 512 dup (?)
  index DW 0
inicio:
  in AL, 1234h
  MOV DI, CS:index
  MOV CS:[buffer+DI], AL
  CMP DI, 512 ; va de 0 a 512 y es circular
  JNE fin
  MOV DI, 0
fin
  mov al, 20h   mandar el EOI
  out 20h, al
  out 0A0h, al
IRET
RSC_RTC ENDP
```

28) Escribir en ensamblador una rutina lejana que llame a la función TICK (int hour, int minute, int second) cada vez que haya transcurrido 1 segundo en el RTC, emplear la int 1Ah con bucle ocupado

```
RTC PROC FAR                        ; Time - Get Real
push...                             ; AH = 02h
MOV AH 02h  MOV AL, 0               ; Return
Iterate  INT 1Ah                    ; CF clear if successful
  jc Iterate ; si hay error         ; CH = hour
  CMP DH, AL                        ; CL = minute
  JE Iterate (si aún no ha pasado 1 sec)   ; DH = second
  (si pasó un segundo)              ; DL = Daylight saving
  MOV BX, 0                         ; CH set if invalid
  MOV BL, DH                        ; INT 1Ah
  PUSH BX
  MOV BL, CL
  PUSH BX
  MOV BL, CH
  PUSH BX
  call - TICK
  ADD SP 6 ; volvemos a poner la pila en su sitio
  JMP Iterate
RTC ENDP
```

29] Escribir en ensamblador una rutina lejana que recibe en AH un número binario de 4 bits y lo convierta a código Gray de 4 bits mediante la función unsigned int BinaryToGray(unsigned); compilada en modo largo y escriba el resultado en los 4 bits de + peso del reg de datos del LPT2

```
WriteGray_LPT2 PROC FAR
 push...
 mov al, ah
 mov ah, 0
 push ax
 call _BinaryToGray
 add SP,2
 mov ah, al    ;ah codigo Gray
 mov dx,0
 mov es, dx
 mov dx, es:[040Ah]
 in  al, [dx] registro de datos
 mov cl, 4
 SHL ah, cl  poneros los 4 bits en los de más peso
 and ah, 11110000b por si acaso
 and al, 00001111b
 or  al, ah
 out dx, al
 pop...
 ret
WriteGray_LPT2 ENDP
```