SIMULACRO DE PARCIAL

1ER PARCIAL ARQUI 2024

anabel paez la valle

1er parte: pila, subrutinas y pasaje de parámetros

EJERCICIO 1:

El siguiente programa envía un parámetro por la pila a SUBRUT. ¿Qué valor queda almacenado en DL?

```
ORG 1000h
    CAD DB "Hola"
ORG 2000H
    MOV DX, OFFSET CAD
    PUSH DX
    CALL SUBRUT
    ...
```

```
        ORG 3000H
SUBRUT:  MOV BX, SP
        ADD BX, 3
        MOV DL, [BX]
            ...
```

Respuesta:

DL=_____

```
ORG 1000h
    CAD DB "Hola"
ORG 2000H
    MOV DX, OFFSET CAD
    PUSH DX
    CALL SUBRUT
    ...
```

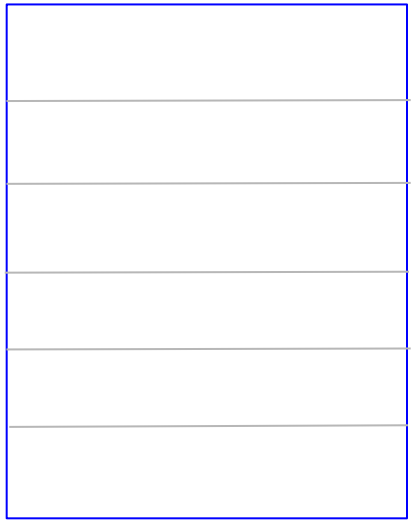
```
        ORG 3000H
SUBRUT:  MOV BX, SP
        ADD BX, 3
        MOV DL, [BX]
             ...
```

Respuesta:

DL=_____

AX =
BX=
CX=
DX = 10 00

PILA



8000H ← SP

```
ORG 1000h
    CAD DB "Hola"
ORG 2000H
    MOV DX, OFFSET CAD
    PUSH DX
    CALL SUBRUT
    ...
```

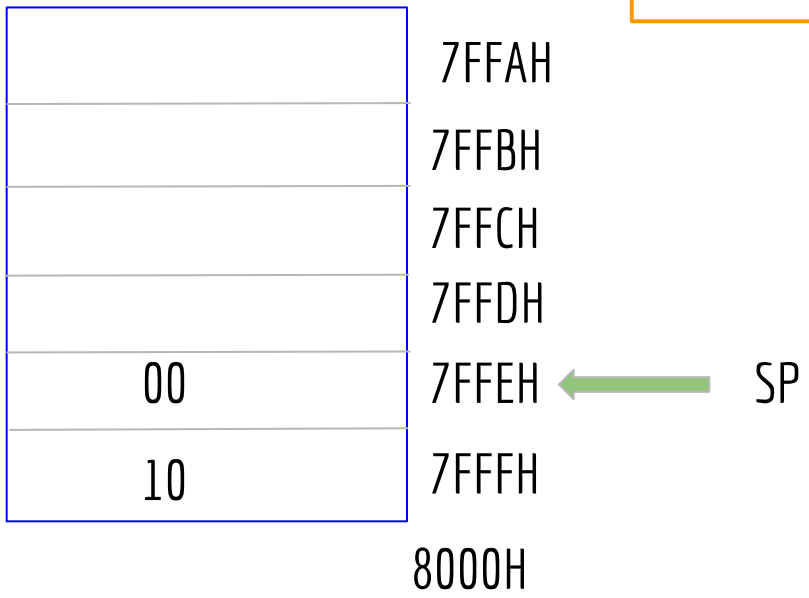
```
        ORG 3000H
SUBRUT:  MOV BX, SP
        ADD BX, 3
        MOV DL, [BX]
            ...
```

Respuesta:

DL=_____

AX =
BX=
CX=
DX = 10 00

PILA



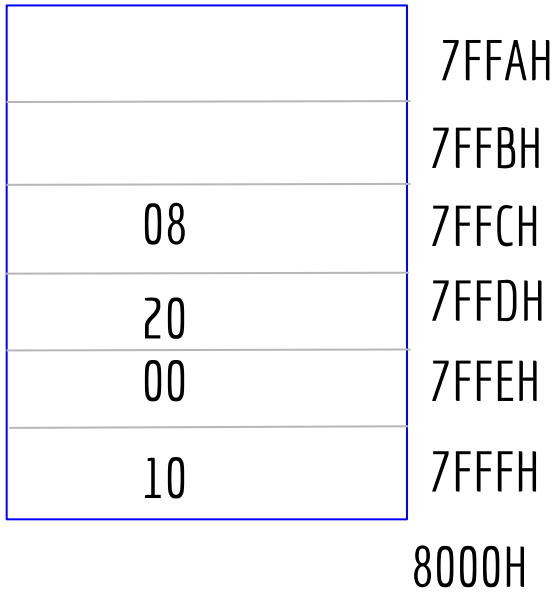
```
ORG 1000h
    CAD DB "Hola"
ORG 2000H
    MOV DX, OFFSET CAD
    PUSH DX
    CALL SUBRUT
     ...

        ORG 3000H
SUBRUT:  MOV BX, SP
        ADD BX, 3
        MOV DL, [BX]
             ...
```

Respuesta:

DL=_____

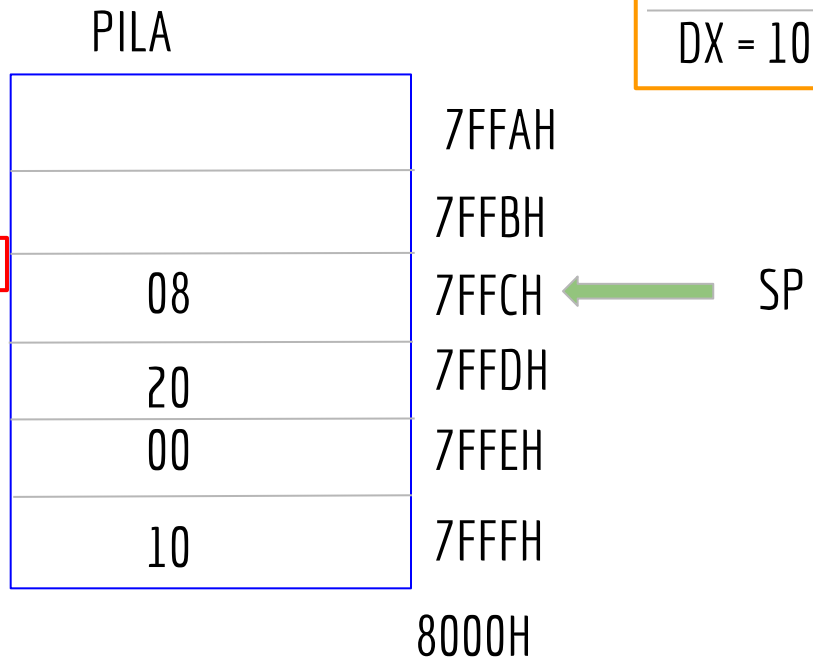
AX =
BX=
CX=
DX = 10 00

PILA



```
ORG 1000h
    CAD DB "Hola"
ORG 2000H
    MOV DX, OFFSET CAD
    PUSH DX
    CALL SUBRUT
    ...

        ORG 3000H
SUBRUT:  MOV BX, SP
        ADD BX, 3
        MOV DL, [BX]
            ...
```

AX =
BX= 7F FC
CX=
DX = 10 00



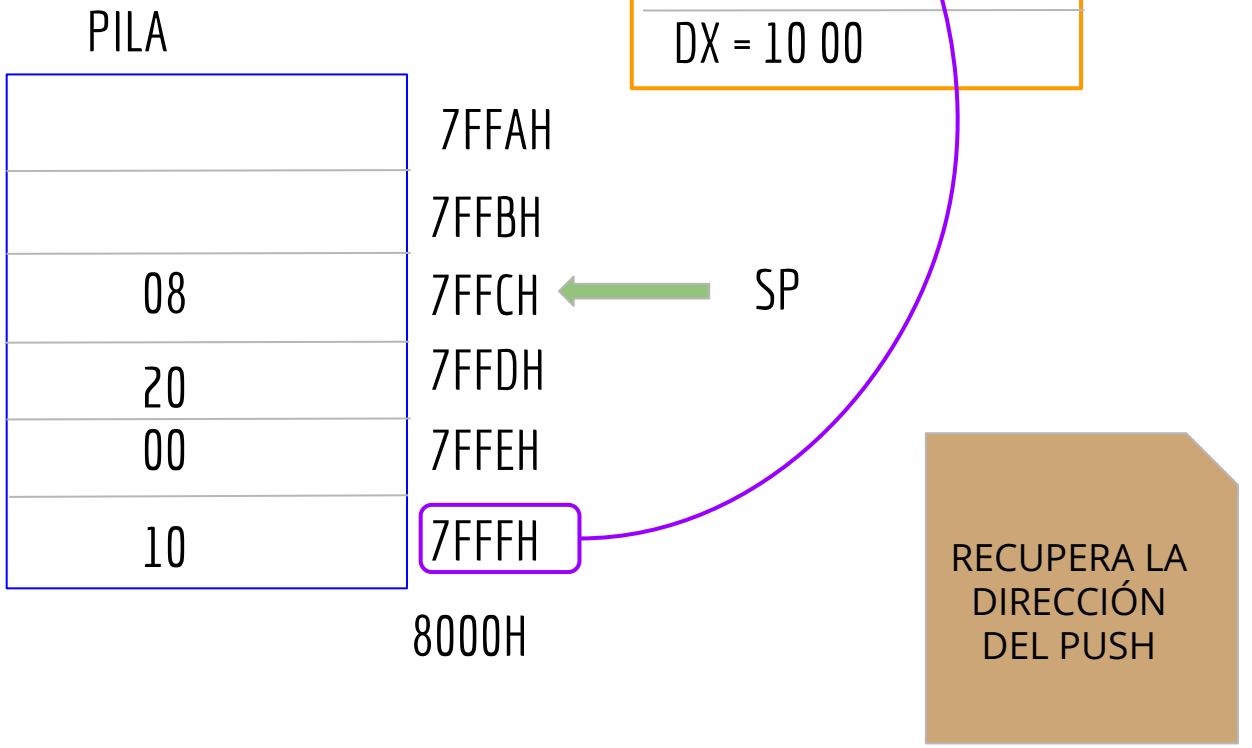
Respuesta:

DL=_____

```
ORG 1000h
    CAD DB "Hola"
ORG 2000H
    MOV DX, OFFSET CAD
    PUSH DX
    CALL SUBRUT
    ...

        ORG 3000H
SUBRUT:  MOV BX, SP
        ADD BX, 3
        MOV DL, [BX]
            ...
```

Respuesta:

DL=_____



```
ORG 1000h
    CAD DB "Hola"
ORG 2000H
    MOV DX, OFFSET CAD
    PUSH DX
    CALL SUBRUT
    ...
```

```
        ORG 3000H
SUBRUT:  MOV BX, SP
        ADD BX, 3
        MOV DL, [BX]
            ...
```

PILA

	7FFAH
	7FFBH
08	7FFCH ← SP
20	7FFDH
00	7FFEH
10	7FFFH

8000H

AX =

BX= 7F FF

CX=

DX = 10 10

ALTA BAJA

Respuesta:

DL= 10

EJERCICIO 2:

En el siguiente fragmento de programa se hace un llamado a una subrutina, a la cual se le pasan tres parámetros a través de los registros AX, BX y CX. Indicar qué parámetro se pasa en cada registro y si es pasado por valor o por referencia. No completar con los valores sino con: nombreVariable, formaPasaje.

```
org 1000h
NRO     DB    2
COD1    DB    “A”
COD2    DB    “J”
DIG     DB    7
```

```
org 2000h
mov CX, offset NRO
mov BX, CX
ADD BX, 3
MOV AL, [BX]
mov BX, offset COD2
call SUBRUTINA
```

AX: _____

BX: _____

CX: _____

PASAJE DE PARÁMETROS:

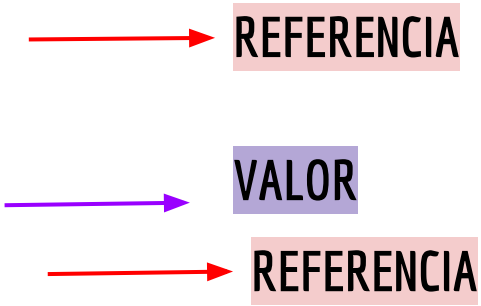
A TRAVÉS DE: PILA/REGISTRO
POR: VALOR/REFERENCIA

```
org 1000h
NRO     DB    2
COD1    DB    “A”
COD2    DB    “J”
DIG     DB    7
```

```
org 2000h
mov CX, offset NRO
mov BX, CX
ADD BX, 3
MOV AL, [BX]
mov BX, offset COD2
call SUBRUTINA
```

```
org 1000h
NRO     DB    2
COD1    DB    “A”
COD2    DB    “J”
DIG     DB    7
```

```
org 2000h
mov CX, offset NRO
mov BX, CX
ADD BX, 3
MOV AL, [BX]
mov BX, offset COD2
call SUBRUTINA
```



- **Por valor**: Si trabajas con el contenido de la dirección (`[bx]`), estás tratando con el valor directamente.
- **Por referencia**: Si estás utilizando `offset` o directamente el registro `bx` para trabajar con direcciones, se considera un acceso por referencia porque estás manipulando o pasando la dirección de la memoria.

```
org 1000h
NRO     DB    2
COD1    DB    “A”
COD2    DB    “J”
DIG     DB    7
```

```
org 2000h
mov CX, offset NRO
mov BX, CX
ADD BX, 3
MOV AL, [BX]
mov BX, offset COD2
call SUBRUTINA
```

REFERENCIA

VALOR

REFERENCIA

- **Por valor**: Si trabajas con el contenido de la dirección (`[bx]`), estás tratando con el valor directamente.
- **Por referencia**: Si estás utilizando `offset` o directamente el registro `bx` para trabajar con direcciones, se considera un acceso por referencia porque estás manipulando o pasando la dirección de la memoria.

AX: dig, VALOR

BX: cod2, REFERENCIA

CX: nro, REFERENCIA

2da parte: completar sentencias de código

EJERCICIO 3:

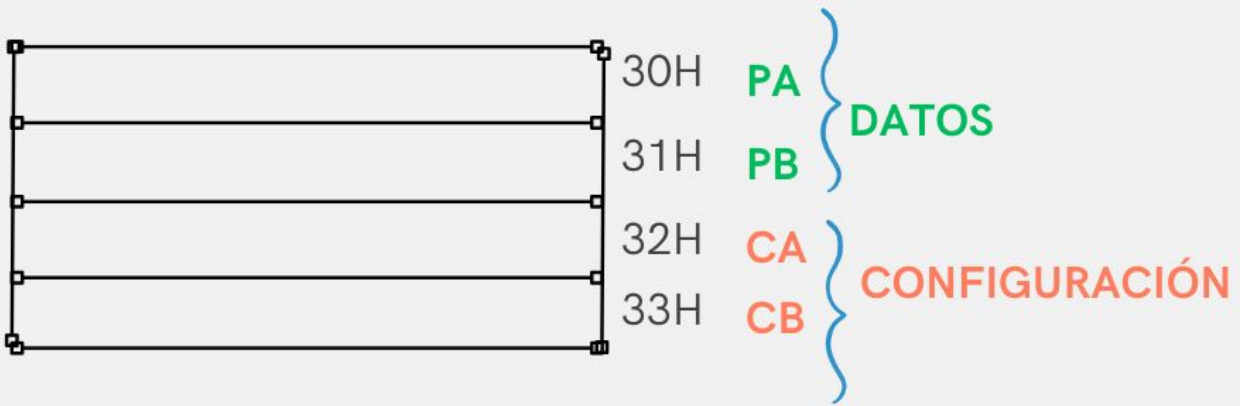
El siguiente programa para VonSim lee carácteres por teclado y los envía a la impresora a través del PIO a medida que se van leyendo. El programa termina cuando se lee el carácter "." Completar las instrucciones faltantes.

```
PA EQU 30h
PB EQU 31h
CA EQU 32h
CB EQU 33h

org 1000h
car db ?
```

```
org 2000h
mov al, 0
out CB, al

_____________
out CA, al
in al, PA
and al,11111101b
out PA, al
mov bx, offset car
loop: int 6

      _____________
      jz fin
      call imp
      jmp loop
fin: INT 0
end
```

```
imp: in al, PA

      _____________
      jnz imp
      mov al, car
      out PB, al
      in al, PA

      _____________
      out PA, al
      in al, PA
      and al,11111101b
      out PA, al
      ret
```

PIO (puerto paralelo de e/s)



- IN: leer -> entrada
- OUT: escribir -> salida
- solo se pueden usar con el registro AL

PA <----------> ESTADO

PB ----------> DATOS

Veamos cuáles bits del registro **estado** son de **entrada** y cuáles de **salida**...

- **Bit 0 (busy)** - 1 si está ocupada la impresora, 0 si está libre
- **Bit 1 (strobe)** - seteando el bit en 1 le avisamos a la impresora que dejamos un caracter en **DATO** para que lo imprima

MASCARAS

OR

FORZAR STROBE EN 1 PARA AVISAR A LA IMPRESORA QUE MANDE UN DATO

AND

FORZAR STROBE EN 0

```
PA EQU 30h
PB EQU 31h
CA EQU 32h
CB EQU 33h

org 1000h
car db ?
```

CONFIGURACIÓN DEL PIO

FUERZO STROBE A 0

```
org 2000h
mov al, 0
out CB, al

_____________
out CA, al
in al, PA
and al,11111101b
out PA, al
mov bx, offset car
loop: int 6

      _____________
      jz fin
      call imp
      jmp loop
fin: INT 0
end
```

MIENTRAS NO SEA PUNTO, SIGO IMPRIMIENDO

```
imp: in al, PA

      _____________
      jnz imp
      mov al, car
      out PB, al
      in al, PA

      _____________
      out PA, al
      in al, PA
      and al,11111101b
      out PA, al
      ret
```

POOL, MIENTRAS NO ESTE DISPONIBLE SIGO PREGUNTANDO

FUERZO STROBE A 1

FUERZO STROBE A 0

El siguiente programa para VonSim lee carácteres por teclado y los envía a la impresora a través del PIO a medida que se van leyendo. El programa termina cuando se lee el carácter "." Completar las instrucciones faltantes.

```
PA EQU 30h
PB EQU 31h
CA EQU 32h
CB EQU 33h

org 1000h
car db ?
```

CONFIGURACIÓN DEL PIO

FUERZO STROBE A 0

```
org 2000h
mov al, 0
out CB, al
mov al, 1111 1101b
out CA, al
in al, PA
and al,11111101b
out PA, al
mov bx, offset car
loop: int 6
cmp byte ptr [bx], 00101110b
      jz fin
      call imp
      jmp loop
fin: INT 0
end
```

MIENTRAS NO SEA PUNTO, SIGO IMPRIMIENDO

```
imp: in al, PA
     and al, 1
     jnz imp
     mov al, car
     out PB, al
     in al, PA
     or al, 00000010b
     out PA, al
     in al, PA
     and al,11111101b
     out PA, al
     ret
```

POOL, MIENTRAS NO ESTE DISPONIBLE SIGO PREGUNTANDO

FUERZO STROBE A 1

FUERZO STROBE A 0

EJERCICIO 4:

Se cuenta con un programa para VonSim que usa el TIMER (mediante la posición del vector 7) y el F10 (mediante la posición del vector 8). En un momento dado de la ejecución se está atendiendo la interrupción de TIMER y hay un pedido de interrupción de F10 (no hay más pedidos que ese). Complete los datos en dicho momento.

```
org _____________        IMR:  _____________        ISR: _____________
DW   RUT_TIMER           INT0: _____________        IRR: _____________
```

Se cuenta con un programa para VonSim que usa el TIMER (mediante la posición del vector 7) y el F10 (mediante la posición del vector 8). En un momento dado de la ejecución se está atendiendo la interrupción de TIMER y hay un pedido de interrupción de F10 (no hay más pedidos que ese). Complete los datos en dicho momento.

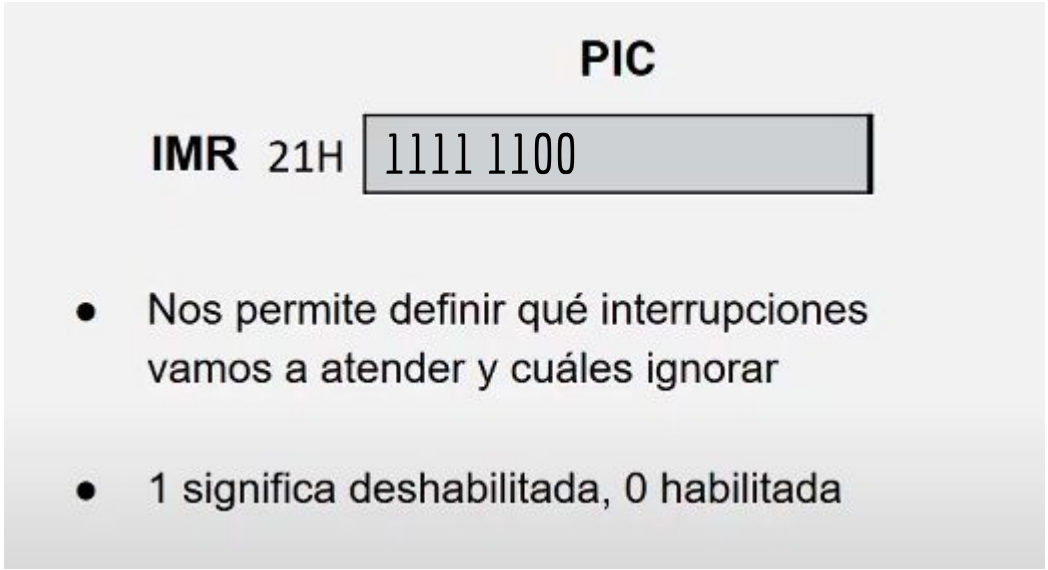
```
org _____________        IMR:  _____________        ISR: _____________
DW   RUT_TIMER           INT0: _____________        IRR: _____________
```

PIC

EOI	20H	_____	Le avisa al **PIC** que la interrupción ya fue atendida
IMR	21H	_____	Para habilitar o deshabilitar alguna interrupción
IRR	22H	_____	Indica cuáles dispositivos externos solicitan interrumpir
ISR	23H	_____	Indica cuál dispositivo externo está siendo atendido
INT 0	24H	_____	Contiene *ID* asignado al F10
INT 1	25H	_____	Contiene *ID* asignado al Timer
INT 2	26H	_____	Contiene *ID* asignado al Handshake
INT 3	27H	_____	Contiene *ID* asignado al CDMA

Se cuenta con un programa para VonSim que usa el TIMER (mediante la posición del vector 7) y el F10 (mediante la posición del vector 8). En un momento dado de la ejecución se está atendiendo la interrupción de TIMER y hay un pedido de interrupción de F10 (no hay más pedidos que ese). Complete los datos en dicho momento.

```
org _____________        IMR:  _____________        ISR: _____________
DW   RUT_TIMER           INT0: _____________        IRR: _____________
```



Se cuenta con un programa para VonSim que usa el TIMER (mediante la posición del vector 7) y el F10 (mediante la posición del vector 8). En un momento dado de la ejecución se está atendiendo la interrupción de TIMER y hay un pedido de interrupción de F10 (no hay más pedidos que ese). Complete los datos en dicho momento.

```
org _____________        IMR:  1111 1100_____        ISR: _____________
DW   RUT_TIMER           INT0: _____________        IRR: _____________
```

24h	INT0	**ID de Línea INT0.** Almacena el ID de la interrupción asociada al dispositivo *F10* para buscar en el vector de interrupciones la dirección de comienzo de la subrutina que lo atiende.

INT0

8

Se cuenta con un programa para VonSim que usa el TIMER (mediante la posición del vector 7) y el F10 (mediante la posición del vector 8). En un momento dado de la ejecución se está atendiendo la interrupción de TIMER y hay un pedido de interrupción de F10 (no hay más pedidos que ese). Complete los datos en dicho momento.

```
org _____________        IMR:  1111 1100_____     ISR: 0000 0010_____
DW   RUT_TIMER           INT0: 8_____________     IRR: _____________
```

23h	ISR	**Registro de Interrupción en Servicio.** El bit que representa la línea de interrupción que se está atendiendo estará en 1 interrupción (bit 0 se asocia a la entrada INT0 ... bit 7 se asocia a la entrada INT7), el resto en 0. Si no hay interrupciones atendiéndose todos los bits valen 0.

ISR

0000 0010

Se cuenta con un programa para VonSim que usa el TIMER (mediante la posición del vector 7) y el F10 (mediante la posición del vector 8). En un momento dado de la ejecución se está atendiendo la interrupción de TIMER y hay un pedido de interrupción de F10 (no hay más pedidos que ese). Complete los datos en dicho momento.

```
org _____________        IMR:  1111 1100        ISR:  0000 0010
DW   RUT_TIMER           INT0:  8               IRR:  _____________
```

22h	IRR	**Registro de Petición de Interrupciones.** Almacena las interrupciones pendientes. Cada bit de este registro representa una línea de interrupción (bit 0 se asocia a la entrada INT0 ... bit 7 se asocia a la entrada INT7). Cuando la interrupción se satisface, el bit de dicha línea se pone a cero.

IRR

0000 0001

Se cuenta con un programa para VonSim que usa el TIMER (mediante la posición del vector 7) y el F10 (mediante la posición del vector 8). En un momento dado de la ejecución se está atendiendo la interrupción de TIMER y hay un pedido de interrupción de F10 (no hay más pedidos que ese). Complete los datos en dicho momento.

```
org  28
DW   RUT_TIMER
```

IMR: 1111 1100

INT0: 8

ISR: 0000 0010

IRR: 0000 0001

ORG (7 * 4)
DW RUT_TIMER

ORG 28
DW RUT_TIMER

EJERCICIO 5:

Un programa que imprime "INGENIERIA E INFORMATICA" en la impresora a través del HAND-SHAKE. La comunicación se establece por consulta de estado (polling).

```
DATO EQU 40h
ESTADO EQU 41h

     ORG 1000h ; Memoria de datos

msj DB "INGENIERIA E INFORMATICA"
fin DB ?

     ORG 3000h
poll: in al,ESTADO
     ____________
     jnz poll
     ret

     ORG 3200h
imprimir_caracter_hand: ____________
     call poll
     pop ax
     out DATO, al
     ret
```

```
     ORG 2000H
     in al, ESTADO
     ____________
     out ESTADO, al
     mov bx, offset msj
     mov cl, offset fin - offset msj
lazo: mov al, [bx]
     call imprimir_caracter_hand
     ____________
     dec cl
     jnz lazo
     int 0
     end
```

Un programa que imprime “INGENIERIA E INFORMATICA” en la impresora a través del HAND-SHAKE. La comunicación se establece por consulta de estado (polling).

```
DATO EQU 40h
ESTADO EQU 41h

    ORG 1000h ; Memoria de datos

msj DB "INGENIERIA E INFORMATICA"
fin DB ?

    ORG 3000h
poll: in al,ESTADO
    and al,1
    jnz poll
    ret

    ORG 3200h
imprimir_caracter_hand: push ax
    call poll
    pop ax
    out DATO, al
    ret
```

```
    ORG 2000H
    in al, ESTADO
    and al, 7Fh
    out ESTADO, al
    mov bx, offset msj
    mov cl, offset fin - offset msj
lazo: mov al, [bx]
    call imprimir_caracter_hand
    inc bx
    dec cl
    jnz lazo
    int 0
    end
```

3er parte: programar

EJERCICIO 6:

Escribir un programa para VonSim que deberá utilizar las luces y llaves de la siguiente forma:

a. Cada vez que las llaves cambien de valor, se actualizan las luces a su estado opuesto. De modo que si las llaves están en el estado “00011010” las luces tendrán el estado “11100101”.

b. Cada vez que encuentre la primera llave (la del bit menos significativo) prendida, mostrar en pantalla el mensaje “Arquitectura de Computadoras”

c. En el caso particular en que todas las llaves estén apagadas, mostrar en pantalla el mensaje “Fin de programa” y finalizar el mismo.

Las funciones ”a”, “b” y “c” deben implementarse utilizando subrutinas.

EJERCICIO 7:

Escribir un programa para VonSim que envíe la cadena de caracteres “Universidad Nacional de La Plata” a un dispositivo nuevo, conectado a los 8 bits del puerto PA. Este dispositivo recibe la cadena de a un carácter a la vez. Para que el dispositivo reconozca que se va a enviar un dato, **antes** de enviar un carácter debe enviar el valor 0. El programa debe finalizar cuando se han enviado todos los caracteres de la cadena, o cuando se presiona la tecla F10, cancelando el envío cancelando el envío de los caracteres que restan.

Ejemplo para enviar la cadena “ASDF”: Envío de 0 → Envío de la “A” → Envío de 0 →Envío de la “S” → Envío de 0 → Envío de la “D” → Envío de 0 → Envío de la “F”

Nota: para comunicarse con el dispositivo no es necesario realizar una consulta de estado ni hacerlo mediante interrupciones, el protocolo solo requiere que envíe un 0 y un carácter de forma alternada.

EJERCICIO 8:

Escribir un programa para VonSim que muestre el mensaje “Escriba 5 caracteres:”, después lea un texto de 5 caracteres y lo almacene en memoria. Luego, el texto leído debe ser enviado a una subrutina llamada "SUB_IMP" por referencia a través de la pila. La subrutina debe mostrar el texto en pantalla de a 1 carácter por segundo (utilizando el TIMER).