1. Si antes de invocar a la subrutina **RUT\_SIMPLE** el registro **SP** *[Stack Pointer, puntero de pila]* contenía el valor **8000H**, **AX = 12FFH**, **BX = 0011H**, **CX = ABCDH**. Complete el contenido de la memoria después de ejecutar las siguientes instrucciones:

**RUT\_SIMPLE:** PUSH AX [1] *;AX vale 12 FF (hexadecimal) donde 12H es la parte alta y FFH es la parte baja*

*La subrutina apila primero la parte alta de AX y después la parte baja moviendo*

*El SP dos celdas hacia arriba (orden decreciente en memoria);*

PUSH BX [2]  *;BX tiene como valor 00H en su parte alta y 11H en su parte baja, también se*

*apilan y el SP se mueve dos celdas más hacia arriba;*

XOR BH, AL [3] ;*Se hace un XOR lógico con la parte alta de BX y la parte baja de AX.*

***El resultado se va a guardar en BH****, es decir, se sobrescribirá su valor*;

POP CX [4] ;*Se desapilan 16 bits de la pila, es decir, se van a vaciar dos celdas a partir de*

*donde quedó apuntando SP la última vez. Estos nuevos valores se cargan en CX*

*sobrescribiendo o “pisando” lo que contenía antes****. CX ahora vale 0011H***;

PUSH BX [5]  *;Se guarda el valor de BX en la pila (recordar que su parte alta fue modificada*

*por una operación XOR);*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[3]

0000 0000 esto es 00H en binario (el valor de BH)

XOR  
 1111 1111 esto es FFH en binario (el valor de AL)

**1111 1111** 🡪 Resultado del XOR, es el valor **FFH** (en el XOR cuando ambos bits son diferentes el resultado lógico es 1; si los valores de los bits son iguales da como resultado 0). **Por ende ahora BH valdrá FFH.**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**SP después de ejecutar subrutina** 🡪

|  |  |
| --- | --- |
| **DIRECCION** | **CONTENIDO** |
| 7FF9H |  |
| 7FFAH | ~~11H~~ *;BL;*  **11H ;BL;** |
| 7FFBH | ~~00H~~ *;BH;*  **FFH ;BH;** |
| 7FFCH | FFH *;AL;* |
| 7FFDH | 12H *;AH;* |
| 7FFEH | IP\_L |
| 7FFFH | IP\_H |
| **8000H** |  |

**[0]\*** 🡪

**SP antes de ejecutar subrutina** 🡪

[0]\*: **IMPORTANTE**

El CALL en el programa principal apila la próxima dirección ejecutable que contiene el Contador de Programa (o también IP: Instruction Pointer), así sabe a qué dirección de memoria debe retomar el flujo de ejecución una vez que se termine de ejecutar la subrutina. Al igual que los registros primero se apila la parte alta de la dirección y después la parte baja (puesto que las direcciones son datos de 16bits, entonces ocupan dos celdas de memoria consecutivas también).

Por las dudas aclaro que la dirección 8000H nunca pero **NUNCA** se va a ocupar, puesto que (además de ser una convención de los informáticos que decidieron que esta dirección de memoria nunca se ocuparía), el sentido en el que apila la pila es siempre: **SUBIR, APILAR, SUBIR, APILAR** (lo mismo para desapilar pero en el orden inverso). Es decir, la pila jamás va a apilar primero el contenido en la dirección que está parada y luego va a subir.

2) ¿Cuál será el valor del **registro SP** luego de ejecutadas las instrucciones anteriores?

SP = **7FFAH** *;Porque la última instrucción fue un PUSH que dejó al SP ahí;*

3) ¿Cuál será el valor de los siguientes registros luego de ejecutadas las instrucciones anteriores?

AX = **12FFH** BX = **FF11H** CX = **0011H**

4) ¿Cuál debe ser la última instrucción de la subrutina?

\_**RET\_** *;Esa instrucción es necesaria en todas las subrutinas y sirve para devolverle el flujo de ejecución al*

*programa principal;*

5) Dadas las siguientes constantes

**PA = 30H**

**PB = 31H**

**CA = 32H**

**CB = 33H**

Escribir las 7 instrucciones necesarias para inicializar la **PIO** para poder imprimir a través de la impresora.

*Las constantes que acaba de dar el ejercicio son los 4 registros que pertenecen a la PIO en memoria y esos valores indican en qué dirección de memoria están ubicados. Los primeros dos registros* ***PA y PB*** *son los puertos de la PIO que generalmente se usan como* ***entrada*** *(al hacer un IN se obtienen sus datos del periférico PA y se copian en el registro AL de la CPU), y como* ***salida*** *respectivamente (al hacer un OUT sobre PB podemos pasarle los datos que habíamos escrito en AL sobre el mismo. Es decir es una transferencia CPU – periférico o registro – periférico).*

*Los siguientes dos registros se llaman* ***registros de control.*** *Según lo que se escriba en los bits de estos vamos a poder configurar a PA y PB como entrada o salida (los* ***bits en 1*** *hacen que configuremos los mismos bits de los puertos como entrada y los* ***bits en 0*** *los vuelven salida).*

*[****ENTRADA*** *significa que podemos obtener o mejor dicho leer datos desde un periférico y los podemos copiar en registros de la CPU (el cual es AL), usando la instrucción IN.*

*Mientras que la* ***SALIDA*** *hace que podamos escribir los datos que queramos poner en un periférico moviéndolos en el registro AL y luego pasándolos a dicho periférico; esto puede hacerse con la instrucción OUT]*

\_MOV\_\_AL,\_0FDH\_ *;Escribo* **1111 1101** *en AL;*

\_OUT\_\_CA,\_\_AL\_\_ *;Escribo el contenido de AL en el registro de control que configura el puerto PA.*

*Convierte el bit1 de PA en salida para que podamos escribir el valor que queramos en dicho bit*

*el cual es el* **STROBE** *(da pulsaciones de reloj) durante la transferencia con la impresora).*

\_IN\_\_AL,\_\_PA\_ *;Copio los bits que tiene el puerto PA en el registro AL;*

\_AND\_AL,\_0FDH\_ *;Los AND me aseguran tener ceros. Yo quiero que* el bit STROBE valga cero *antes de comenzar la*

*transferencia con la impresora, esto lo logro poniendo un cero en el bit1 donde va a estar*

*ubicado el STROBE que obtuvimos copiándolo en AL;*

\_OUT\_PA,\_AL\_ *;Escribo el resultado de la operación lógica en PA;*

\_MOVE\_AL,\_0\_

\_OUT\_CB,\_AL\_ *;Configuro a* PB como puerto de salida *haciendo que todos los bits de CB estén en cero. Haciendo*

*Me aseguro que durante la transferencia yo puedo escribir datos sobre PB, el cual los envía a*

*la impresora;*

[Cuando se quiere imprimir usando la PIO el bit0 (el menos significativo, el que se encuentra en el extremo derecho) del puerto PA cumple la función de **BUSY**. Este bit no se puede escribir (no podemos darle el valor que queramos), sino que sólo se puede leer haciéndole un IN para saber si la impresora está ocupada o no (Busy = 1, la impresora está ocupada. Busy = 0, la impresora está desocupada).

El bit1 como ya dijimos es el bit **STROBE** que da las pulsaciones de reloj. Este bit sí que podemos escribirlo pero primero tenemos que configurarlo desde CA como hicimos haciendo que valga 0). Cuando STROBE vale 1 está activado y da una pulsación de reloj y cuando vale 0 está desactivado.]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| X | X | X | X | X | X | strobe | busy |

Bit 7 bit 6 bit 5 bit 4 bit 3 bit 2 bit 1 bit 0

6) Explicar por qué los registros IMR e ISR pueden tener simultáneamente los valores: **IMR = F7H** e **ISR = 08H**:

\_Los valores de esos registros son válidos puesto que F7H en IMR indica que está habilitada INT3 (puede interrumpir al procesador), mientras que 08H en ISR indica que precisamente INT3 estaría siendo atendida (es decir que se está ejecutando su rutina de atención).

F7H = 1111 0111 (los bits en 1 en el registro IMR enmascaran interrupciones y en 0 las habilitan).

08H = 0000 1000 (los bits en 1 en ISR indican que se está interrumpiendo una interrupción).

7) Si **PUERTO** es una constante que vale **41H** y el registro **AL** contiene el valor **80H**, ¿Qué efecto produce la ejecución de la instrucción **OUT PUERTO, AL** ?

\_Se escribe el contenido de AL en PUERTO. Es decir, PUERTO corresponde al registro de Estado del HAND-SHAKE, por tanto al escribirle el valor 80H se está diciendo que la impresora se comunicará con el procesador mediante interrupciones.

El HAND-SHAKE posee dos registros en memoria mapeados a partir de la dirección 40H. El primer registro es el de DATOS donde se escriben los datos que se le quieren enviar a la impresora y el segundo es el de ESTADO o CONTROL. Según lo que se escriba en el último registro se va a establecer cómo va a ser la comunicación entre la impresora y el procesador:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| INT | X | X | X | X | X | STROBE | BUSY |

Bit 7 bit 6 bit 5 bit 4 bit 3 bit 2 bit 1 bit 0

Los bits del 2 al 6 no tienen uso asi que no importa qué valor tengan escritos. Los bits STROBE y BUSY sólo pueden ser manipulados por la impresora, es decir nosotros sólo vamos a poder leerlos para saber qué valor tienen.

En cambio sí podemos escribir el bit 7 el cual corresponde a la línea de interrupciones. Escribiendo un bit en 1 ahí mismo estamos diciendo que la impresora va a trabajar mediante interrupciones cuando la línea BUSY no esté activa (es decir cuando esta valga cero). Si INT vale 0 entonces la impresora no va a trabajar mediante interrupciones sino que va a trabajar con el procesador mediante consulta de estado (el procesador va a entrar en un loop revisando todo el tiempo el bit BUSY para ver si la impresora desocupada así le puede enviar el dato a imprimir).

8) Indicar qué dato se debe cargar en el registro de Control del CDMA (controlador de acceso directo a memoria) para realizar una transferencia de memoria a periférico por demanda:

(CTRL) = \_04H\_\_(0000 0100 B)\_

Según lo que escribamos en el registro de CONTROL del CDMA podemos configurar:

Tipo de transferencia 🡪 0 transferencia periférico – memoria o al revés

🡪 1 transferencia memoria – memoria

Sentido de transferencia 🡪 0 periférico – memoria

🡪 1 memoria – periférico (estos valores solo tienen sentido cuando TT vale 0)

Modo de transferencia 🡪 0 por demanda (por robo de ciclo)

🡪 1 por bloque (modo ráfaga)

Los registros STOP y Terminal Count no se escriben, sólo se leen para saber su estado. STOP indica si la transferencia total de todos los datos terminó o no. Cuando STOP vale 0 significa que la transferencia está en curso y cuando vale 1 la transferencia está finalizada.

Terminal Count cumple una función similar pero indica si la palabra que se estaba transfiriendo terminó de transferirse o no. Tiene los mismos valores que STOP.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC | X | X | X | MT | ST | TT | STOP |

Bit 7 bit 6 bit 5 bit 4 bit 3 bit 2 bit 1 bit 0