INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

Organización y Programación de Computadoras

Ejercicios CB

Grupo de Trabajo

Las Chicas

Integrantes

Andrea Marín Alarcón – 158999

Luis Felipe Landa Lizarralde – 158228

Tábata Ailé González Alvarado – 155999

20 de noviembre de 2019

Ejercicios CB 14nov19 >> 20nov19

En el reporte, por cada ejercicio, muestre como resultado el despliegue de la ventana Console (cmd.exe) y explique su contenido. Entre su despliegue y lo que usted responda, deberá visualizarse la correcta justificación. Incluya en su respuesta todos los valores que se necesiten.

En cada ejercicio también despliegue la parte importante del programa en lenguaje ensamblador.

--------------------------------------------------------------------------------------- --

Los datos y resultados deben ser precedidos por un texto adecuado.

Lea siempre, todo el ejercicio completo antes de empezar a resolverlo.

1. A partir del programa contenido en el archivo “MultiShf.asm”.

Supongamos que un entero superlargo de n\*32 bits se desplazará entre 1 y 4 bits a la derecha. n valdrá entre 2 y 8.

Haga entrar los n dwords hexadecimal que componen el entero superlargo y almacénelos en un arreglo.

Al igual que en caso de “MultiShf.asm” imprima en la representación inicial de cada dword y al final los dword una vez desplazados a la derecha.

Al final imprimirá ADIOS.

Por ejemplo:

DATOS

N: 4

Dword 1o: 12345678

Dword 2o: ABCDEF12

Dword 3o: 87654321

Dword 4o: 21FEDCBA

Bits a desplazar: 2

RESULTADOS

Inicial

; Los cuatro dwords en binario

Una vez desplazados

; Los cuatro dwords en binario

ADIOS

*Hint*: instrucciones shift y rotate, etc. ReadHex, WriteBin, WriteString, etc.

Usted deberá programar, además del procedimiento principal y del procedimiento que despliega en binario, los siguientes procedimientos:

Uno para leer los n valores dword, imprimiendo el texto de pedido. Otro más para hacer los desplazamientos a la derecha entre los n valores dword.

1. Lleve a cabo la multiplicación binaria, sin signo, M \* N, multiplicando la M por potencias de 2 adecuadas tomadas de N.

Una vez leidos M y N, sólo verifique que N<=M y de no ser así, intercámbielos.

Por ejemplo:

M: 5

N: 10

Potencia de 2, peso 0

Potencia de 2, peso 2

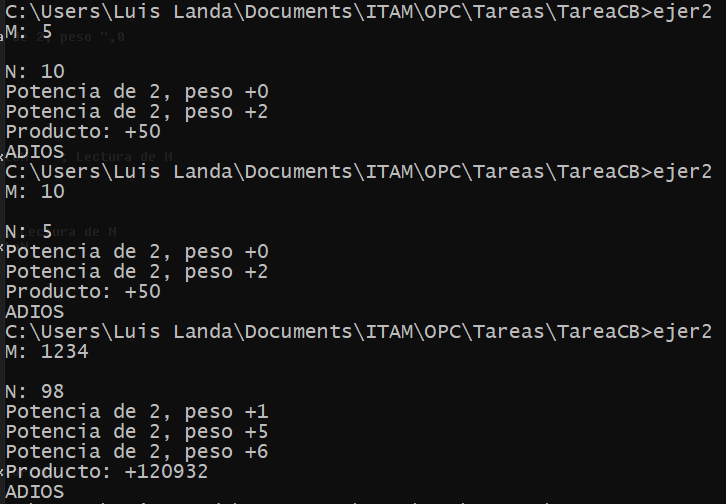
Producto: 50

ADIOS

El resultado debe caber en 32 bits.

Hint: instrucciones shift y rotate.

Además del procedimiento principal, usted deberá programar el procedimiento que lleve a cabo la multiplicación binaria a partir de M y N.



En la captura se muestra la ejecución del programa. En la primera ejecución se usan los valores de M = 5 y N = 10 que es el caso que nos ejemplifica el problema. En la segunda ejecución invertimos los valores, entonces M = 10 y N = 5. Esto lo hicimos para verificar que si N <= M no se cumple los valores se inviertan y el programa nos entregue el resultado correcto. Por último, probamos que M = 1234 y N = 98 para verificar que el programa funcionara con números de tamaño mayor.

1. Elabore un programa que lea un valor entero a 32 bits, positivo e imprima su representación hexadecimal en 8 caracteres representando los 8 dígitos hexadecimales. La tabla ASCII viene en el archivo de ejercicios de la clase pasada.

Este ejercicio sería similar en funcionamiento al procedimiento de Irvine32, WriteHex.

Esta conversión deberá hacerse lo más automática y general posible, que involucre ciclos.

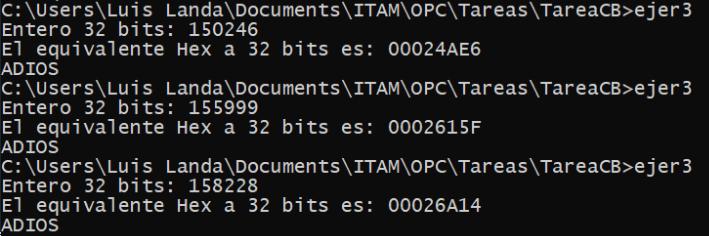
Por ejemplo:

Entero32bits: 150246

El equivalente Hex a 32bits es 00024AE6

ADIOS

Además del procedimiento principal elabore un procedimiento que se encargue de la conversión, dejando el string que representa al valor hexadecimal y desde el programa principal imprima el string “hexadecimal” con WriteString.



En la imagen se muestra como al ingresar enteros de 32 bits nos regresa el equivalente en hexadecimal. Primero probamos el ejemplo descrito con la definición del problema para verificar que el programa regrese el resultado correctamente. Por lo tanto, ingresamos 150246 que en hexadecimal es 00024AE6. Después, se muestran las pruebas que hicimos con dos enteros diferentes. El equivalente en hexadecimal lo corroboramos con un convertidor de decimal a hexadecimal.

1. Construya un programa en ensamblador “\_\_\_\_\_.asm” donde tengamos dos procedimientos, uno que convierta un DWORD a una cadena ASCII, y el otro que convierta un SDWORD a una cadena ASCII; ambos en representación decimal.

Para esto desarrolle dos procedimientos: uno *dwToStr* y el otro *sdwToStr*, respectivamente. Los argumentos deberán ser pasados por el stack. Se pasaran dos argumentos, el valor DWORD/SDWORD a convertir y la dirección inicial del buffer donde quedara el string, en representación decimal. Con la dirección del buffer usted podrá dejar directamente lo convertido desde el procedimiento correspondiente.

Rango DWORD: 0 a 4,294,967,295

Tome en cuenta que el buffer que contendrá el número convertido, debe considerar una cantidad de bytes máximo que incluya al final el carácter nulo.

p.e. DWORD 0Ah o DWORD 10; producirá el siguiente string: “10”,0

Rango SDWORD: -2,147,483,648 a 2,147,483,647

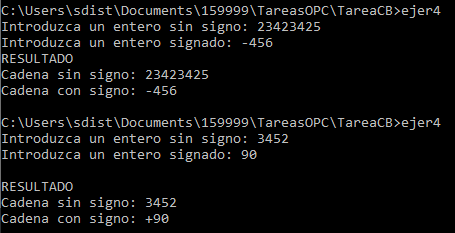
Tome en cuenta que el buffer que contendrá el número convertido, debe considerar, una cantidad de bytes máximo que incluya al principio un carácter signo (+ / -) y al final el carácter nulo.

p.e. SDWORD 0Ah o SDWORD 10; producirá el siguiente string: “+10”,0

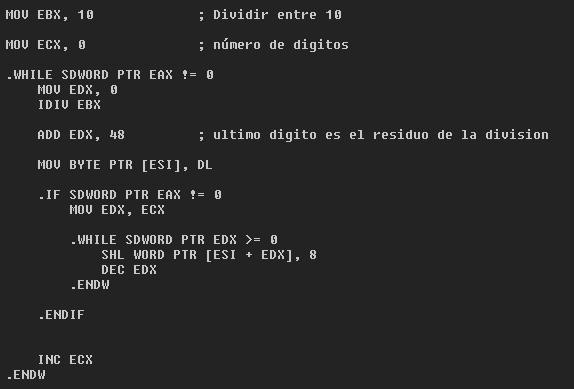
Después desde el programa principal imprima con WriteString las dos cadenas, la convertida de DWORD y la de SDWORD.

*Hint*: puede usar las instrucciones shift, rotate, multiplicación y división. Use las instrucciones que le permitan optimizar su algoritmo.

ADIOS



En la imagen se puede observar cómo el programa toma como entrada un número con signo y un número no signado y lo convierte a su representación en ASCII. En el caso de los números con signo, también despliega el signo del número. A continuación se muestra el código relevante para resolver el problema:



Para poder obtener el último dígito de cualquier número en base 10, basta con dividir el número entre 10 y el residuo es el dígito deseado. Para poder convertir el número en una cadena necesitamos obtener cada dígito del número, por lo que se divide entre 10 sucesivamente hasta que el resultado de la división sea 0, pues eso significa que ya tenemos todos los dígitos.

La representación de los dígitos del 0-9 en ASCII son los números del 48-57, es decir, el “0” es 48, el “1” es 49 y así hasta el “9” que es 57. Por lo tanto para llevar un dígito a su representación en ASCII basta sumarle 48 a EDX (que es donde se encuentra el dígito) y el resultado se guarda en el arreglo donde va a estar el *string.*

Como los dígitos se leen de derecha a izquierda tenemos que ir recorriendo lo que está guardado en memoria. Se tiene que recorrer 1 byte cada dígito que hemos guardado para que el nuevo dígito quede en el orden correcto. Es decir, si el nuevo dígito es “9” y hasta el momento en memoria tenemos “14”, necesitamos recorrer los dígitos para ahora tener 0,”14” y poder guardar el “9” en la primera posición (“914”.).

**OBSERVACIONES:**

* La respuesta a esta tarea deberá subirla a Comunidad, a la sección de TRABAJOS Y EXAMENES, a más tardar este martes 20 de noviembre, antes de las 23:30 hs.
* La primera página de este reporte deberá contener la portada que hemos manejado.
* El nombre del archivo zip donde usted pondrá el archivo respuesta, deberá llamarse con el nombre del grupo de trabajo seguido del nombre de este ejercicio, p.e. **“SparkEjerciciosBI.zip”,** donde “Spark” sería el nombre de su grupo de trabajo.
* También deberá incluir los programas fuente (archivos .asm).