Guía Práctica 4 - *Assembler*

Requerimientos: Esta práctica está basada en la utilización de lenguaje Assembly, en particular NASM. Para obtenerlo puede descargarlo de su página oficial para diferentes arquitecturas: http://www.nasm.us.
Utilizaremos un entorno de desarrollo SASM, el mismo puede obtenerse de su página **https://dman95.github.io/SASM/english.html**

También utilizaremos compilación y ejecución mediante línea de comandos.

1. Escribir las instrucciones que representa cada inciso en el lenguaje NASM e indicar el modo de direccionamiento de cada una de ellas. En este caso utilizaremos NASM desde la línea de comandos. Para ello deberá descargar los archivos driver.c, asm_io.asm y asm_io.inc de la página de su autor o del aula virtual.

    a. Incrementar el valor del registro EAX.
    b. Cargar el registro EBX con el decimal 18 utilizando una etiqueta label binaria.
    c. Sumar al registro EAX el valor 200 (expresado en decimal).
    d. Cargar el registro AX (parte baja del registro EAX) con el contenido de la celda de memoria cuya dirección está almacenada en el registro EBX.
    e. Multiplicar el valor 52 almacenado en una etiqueta en formato hexadecimal, con el valor cuya dirección de memoria está almacenada en el registro EBX.
    f. Sumar al registro EAX el contenido de la celda de memoria cuya dirección es calculada incrementando el registro EBX en 4 unidades.

2. Indicar con qué valores queda representado el registro extendido EAX de la arquitectura 80386, según las operaciones aplicadas:

(a)
```
1 L1 db 0
2 mov eax, L1
```

(b)
```
1 L6 db 18h
2 mov eax, [L6]
3 add eax, [L6]
```

(c)
```
1 L1 dd 1Ah
2 mov AL, [L1]
```

(d)
```
1 mov ax, 5h
2 shl ax, 1
3 shr ax, 1
```

(e)
```
1 mov ax, 0110b
2 rol ax, 1
3 ror ax, 2
```

3. ¿Qué diferencia existe entre el segmento de programa .data y el .bss? Dé un ejemplo que muestre la utilidad de cada uno.

4. Revise la sección 2.2 del Libro “Lenguaje Ensamblador para PC” de Paul A. Carter. Determine cómo simularía los esquemas de las estructuras de control en assembler. Escriba de ejemplos en NASM

5. Escriba un programa en assembler que utilice tres arreglos definidos en el segmento de datos, pero cada uno con un tamaño diferente de elementos (byte, word, dword). Realice un programa en assembler que muestre los elementos de los tres arreglos. **Nota: revise el capítulo 5 del libro de Paul A. Carter para conocer en detalle el tratamiento de arreglos en NASM**.

6. Defina cómo se puede determinar si un número es par utilizando operadores a nivel de bits. Luego, construya un programa assembler que dado un arreglo de 10 enteros de 16 bits (words), definido en el segmento .data, sume los números pares y muestre el resultado por pantalla.

7. Construya un programa assembler que calcule de manera iterativa el factorial de un número dado.

8. Construya un programa assembler que dado un número entero positivo N , calcule la serie de Fibonacci hasta N y la muestre por pantalla (resuelva de manera iterativa). Ejemplo: La serie de fibonacci para n = 6 es 11235.

9. Retome el ejercicio Nro 6. Utilizando la solución propuesta, construya una biblioteca (subrutina global) en assembler para calcular a nivel de bits si un número es par. Luego utilice esta biblioteca desde un programa C para determinar si un número es par. **Nota: Revise el capítulo 4 del libro de Paul A. Carter**.

10. Realice una subrutina en assembler que calcule el mayor de dos números enteros. Utilice la misma para calcular el mayor valor de un arreglo de 10 elementos declarado en el segmento .data.

11. Construya una subrutina en assembler que dado un número (representado en un byte) determine si su representación binaria es palíndromo o no. Utilice la subrutina para determinar la cantidad de palíndromos (representación) de un arreglo de números.

12. Revise el archivo asm_io.asm:
    • Comprenda el código de las subrutinas print int y read int que esta biblioteca ofrece.
    • Agregue una subrutina print hex para imprimir en formato hexadecimal siguiendo el estilo de la biblioteca.
    • No olvide de declararla global para poder utilizarla desde otro código.
    • Recompile la biblioteca asm io.asm.
    • Retome algún ejercicio anterior o cree uno simple para utilizar la subrutina creada y poder imprimir en formato hexadecimal.

13. Codifique en assembler (NASM) el siguiente programa C: (Aclaración, recuerde que las variables locales deben alojarse en memoria)
    #include <stdio.h>
    void main ( ) {

```
        int n ;
        printf ( ”Sumar enteros hasta : ” ) ;
        scanf ( ”%d” ,&n ) ;
        int sum = calcSum ( n ) ;
        printf( ”Sum es %d\n” , sum ) ;
}
int calcSum ( int n) {
        int acum =0;
        int i ;
        for ( i =1; i<=n ; i ++){
                acum = acum + elevarCubo ( i )
        }
        return acum;
}
int elevarCubo ( int x ) {
        return x*x*x ;
}
```

14. Implemente en assembler mediante una subrutina el algoritmo de Euclides (Máximo Común Divisor). Utilice la misma, para calcular el MCD entre dos números enteros.