Organización del Procesador

año 2016

Guía de ejercicios prácticos 4

Assembler - Subrutinas

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*. Además, para simplificar la generación de archivos ejecutables y la Entrada/Salida, deberá descargar los archivos *driver.c, asm_io.asm y asm_io.inc* de la página de su autor *http://pacman128.github.io/pcasm/* o del aula virtual.

Evaluación: Esta práctica será evaluada individualmente en el laboratorio, el alumno deberá elegir un ejercicio y defender la solución propuesta.

1. Revise el archivo `asm_io.asm`:
   - Comprenda el código de las subrutinas `print_int` y `read_int` que esta librería ofrece.
   - Agregue una subrutina `print_hex` para imprimir en formato hexadecimal siguiendo el estilo de la librería.
   - No olvide de declararla `global` para poder utilizarla desde otro código.
   - Recompile la librería `asm_io.asm`.
   - Retome algún ejercicio anterior o cree uno simple para utilizar la subrutina creada y poder imprimir en formato hexadecimal.
2. Retome el ejercicio Nro.6 de la práctica anterior. Utilizando la solución propuesta, construya una librería (subrutina global) en assembler para calcular a nivel de bits si un número es par. Luego utilice esta librería desde un programa C para determinar si un número es par.
3. Realice un 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`.
4. 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.
5. 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 = calc_sum(n);
        printf("Sum is %d\n",sum);
}

int calc_sum(int n){
        int acum =0;
```

```
        int i;
        for(i=1;i<=n;i++){
                acum = acum + elevar_cubo(i)
        }
        return acum;
}

int elevar_cubo(int x){
        return x*x*x;
}
```

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