Programación en Assembly

El objetivo de esta práctica es realizar dos programas. Uno estará hecho en C, aunque el código a implementar estará hecho en ensamblador. El segundo programa estará hecho en ensamblador y no podrá utilizar ninguna librería de soporte como libc (la librería estándar de C) o cualquier otra librería externa que pueda servir de soporte.

Formato de entrega

Los dos programas estarán hechos en un solo directorio. En este directorio debe haber un Makefile capaz de compilar los dos programas al ejecutar **Make all**. El Makefile también debe permitir eliminar los dos archivos ejecutables de los programas al ejecutar **Make clean**.

Deben entregar un folder que contenga todos los archivos de los dos programas. No deben haber subfolders, solo un folder. El folder debe tener el formato prac_{# de practica}_{primer apellido}_{primer nombre}_{matricula}. Al momento de entregar deben hacer un zip file con el mismo formato prac_{# de practica}_{primer apellido}_{primer nombre}_{matricula}.

Programa #1

El nombre del ejecutable del primer programa es **cmix**. Este programa va recibir la operación que va realizar a través de argumentos provenientes de la consola, realizará la operación que se le pide y luego terminará. El programa será escrito una parte en C y otra en Assembly. La parte que será escrita en C será la parte que recibe los argumentos de la consola y decide que función llamar para procesar la operación. El código de las diferentes funciones que realizan las operaciones debe ser escrito en Assembly.

El program solo recibe argumentos a través de la consola, no se recibirá entrada de información de parte del usuario una vez este corriendo. El programa debe realizar la operación indicada, imprimir el resultado en pantalla y terminar.

Las funciones de Assembly deben estar en un archivo con nombre func.S, en este archivo deben estar las siguientes funciones de C pero implementadas en ensamblador. El archivo principal de C, desde donde se obtienen los argumentos de la consola y se llaman las funciones de ensamblador debe llamarse main.c

```
/* La función recibe un entero en data y debe intercambiar
 * los bytes de la posición pos1 y pos2 dentro de data. La función debe imprimir
 * el resultado en una línea en consola en hexadecimal con el
 * formato 0xFFFFFFFF\n.
 * por ejemplo switchBytes(200, 0, 2) debe retornar 0x00C80000
 */
void switchBytes(int data, int pos1, int pos2);
```

```
/*
 * La función recibe un arreglo de enteros sin signos y su longitud.
 * Luego debe imprimir en pantalla el número más grande que posee el
 * arreglo. La función debe imprimir el resultado en una línea en
 * consola en hexadecimal con el formato 0xFFFFFFFF\n.
 */
void printHigher(unsigned int *data, int len);

/*
 * La función debe imprimir la cantidad de bits en uno en data.
 * La función debe imprimir el resultado en una línea en  consola
 *  en hexadecimal con el formato 0xFFFFFFFF\n.
 */
void countOnes(int data);

/*
 * La función debe convertir todos los caracteres del String msg
 * a letra mayúscula. La función debe imprimir el resultado en una
 * línea en  consola en hexadecimal con el formato msg\n. Para esta función
 *no se pueden utilizar las funciones de string de la libreria estandar como por  *ejemplo strupr.
 */
void lowerToUpper(char *msg);
```

Estas son las funciones en ensamblador con sus prototipos en C. Ustedes deben implementar estas funciones completamente en ensamblador en el archivo func.S y deben poder llamarlas desde main.c como si fueran funciones de C.

Main.c debe debe obtener los argumentos desde la consola y debe llamar la función indicada de acuerdo a la operación que se le indique. Estas son las operaciones que debe realizar su programa.

-o {operación} La operación a realizar siempre estará indicada después de un -o, en este caso son cuatro operaciones, switch, higher, count y l2u

- El programa debe implementar las siguientes operaciones:
    - -o switch {pos1} {pos2} {dato}: los dos primeros números después del operador switch representan dos bytes del 0 al 3, el siguiente argumento representa un número unsigned de 32 bit a trabajar. Pos1 y pos2 son la posición de los bytes dentro de dato que se van a intercambiar. Si se específica por ejemplo, -o switch 1 2 1234555, quiere decir que del dato 1234555 se van intercambiar los valores del byte 1 y el 2, y esto producirá un nuevo valor. Este nuevo resultado debe imprimirse en la consola y terminar el programa.
    - -o higher 25 23 24 25 26 25 100: el programa debe imprimir el numero mayor de todos los enteros sin signos pasados por la consola. Debe imprimir el resultado en hexadecimal. Para poder capturar este arreglo en memoria, pueden definir un arreglo en main.c de 300 int, y asumir que luego de higher no se podrán procesar más de 300 int.

- -o count 12456812: esta función debe contar la cantidad de bits que son uno dentro del valor pasado como argumento. Se asume que el valor recibido es un unsigned 32 bit.
- -o l2u “Esto es una Prueba”: esta función debe convertir todos los caracteres de minúscula a mayúscula e imprimir el resultado en consola.

Programa #2

El nombre del ejecutable de este programa es **amix**. Este programa estará hecho en ensamblador y no podrá utilizar ninguna librería de soporte como libc (la librería estándar de C) o cualquier otra librería externa que pueda servir de soporte.

Este programa de imprimir los primeros n números de la serie de Fibonacci. La cantidad de números de la serie a imprimir será introducida por el usuario.

Introduzca el número de elementos:

El usuario puede introducir tres dígitos para introducir el número de elementos. Es decir se podrán ver elementos desde el 001 al 999.

Una vez los elementos son introducidos por el usuario el programa debe imprimir el siguiente mensaje seguido de la cantidad de elementos de la serie fibonacci que el usuario introdujo. Cada elemento debe estar en un línea. Este es un ejemplo de una corrida

Introduzca el número de elementos:
5
Los primeros 5 elementos de la serie son:
0
1
1
2
3

Después de imprimir los números el programa debe terminar.