Laboratorio de Sistemas Basados en Microprocesadores

Práctica 3: Diseño de programas utilizando C y ensamblador del 80x86.

Los códigos EAN 13 (European Article Number 13 dígitos) son códigos de barras utilizados en casi todo el mundo y en prácticamente todos los productos de consumo. El código de distribución comercial GTIN 13 comprenden 13 dígitos cuyo significado varía de acuerdo con el tipo de producto. Los primeros tres dígitos corresponden al país de origen del producto, o para una clase de productos estándar; los siguientes 4 son número de socio de la empresa participante en el sistema EAN; los siguientes 5 son el número de artículo del producto bien marcado y el trece es un dígito de control calculado de acuerdo con los anteriores doce.

La empresa NieRozumienTego ha fabricado un scanner de códigos de barra low-cost. Han reusado un porcesador 8086 antiguo y creado la comprobación de la base datos de productos para comparar el código de barras escaneado y devolver el precio correspondiente. Lo han implementado en código C pero se han dado cuenta que los tiempos de lectura y comprobación de los códigos de barra son muy lentos. Por ello han subcontratado a la empresa NuInteleg la optimización del lector. NuInteleg ha decidido implementar en ensamblador esta parte para optimizar su ejecución. Os acaban de contratar en la empresa, sois el equipo La'Afhumuha y os han encargado la simple tarea de verificar que el código de control del código de barras leído es correcto. Lo debéis realizar en ensamblador como funciones que puedan ser llamadas desde el código C del cliente original, NieRozumienTego. Los requisitos del cliente son que separéis cada uno de los campos del código de barras para poder usarlos en la base de datos, verifiquéis los datos de entrada, calculéis el dígito de control y comprobéis que es el mismo que se ha leído en el código de barras.

El código de barras GTIN-13 se compone de 13 dígitos decimales, de los cuales cada uno indica:

- Primeros 3 dígitos (los más significativos) corresponden al país
- Siguientes 4 dígitos identifican la empresa fabricante/vendedora
- Siguientes 5 dígitos el artículo/producto
- Último dígito (menos significativo) es el dígito de control.

Para comprobar el dígito de control se ha de realizar la siguiente operación:

- A1) Sumar los dígitos impares
- A2) Multiplicar por 3 todos los dígitos pares
- B) Sumar todos los resultados de las operaciones anteriores A1 y A2
- C) Se obtiene el número de la decena superior más cercana al valor anterior B.
- D) El dígito de control se obtiene restando C y B

Ejemplo para calcular el dígito de control del siguiente código:

- País → 845
- Empresa→ 6789
- Producto → 01234
- Dígito de Control → ?

A) El código es 845678901234?.
B) Sumamos los valores de los dígitos impares: 8+5+7+9+1+3.
C) Multiplicamos por 3 los valores 4,6,8,0,2,4.
D) Sumamos todos los resultados anteriores y obtenemos el valor 105.
E) La decena superior más cercana es el 110.
F) Restamos 110 – 105 = 5. El dígito de control es el valor 5.
G) **El código de barras resultado es el 8456789012345.**

Los pasos realizados se muestran en el siguiente dibujo:

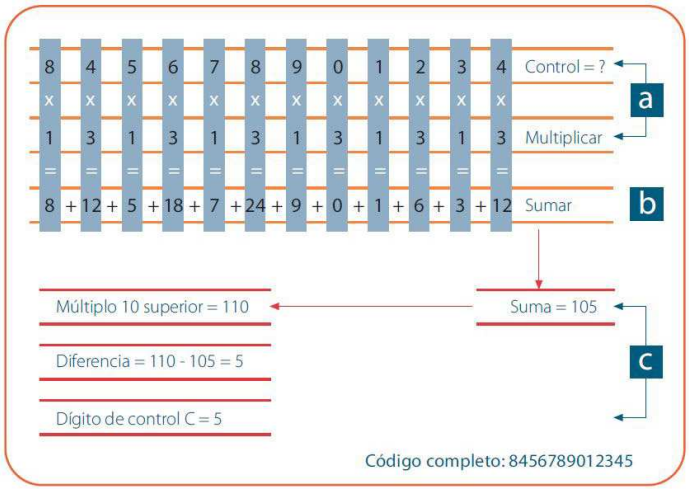

Figure 1 Ejemplo cálculo dígito de control GTIN-13 (fuente: https://www.gs1es.org/calculo-de-digito-de-control-gtin/)

En esta práctica se proporcionará al alumno con **un programa principal escrito en C** (pract3.c) que hará llamadas a las funciones que deben ser **desarrolladas por los alumnos en ensamblador en dos ficheros** (módulos) diferentes (pract3a.asm y pract3b.asm). El alumno deberá realizar el correspondiente análisis de los requisitos expuestos y realizar su implementación. Cada una de las funciones desarrolladas tiene el un peso de 7 y 3 puntos respectivamente sobre la nota final de la práctica.

Programa pract3a.asm (7 ptos)

En el módulo pract3a.asm se deben desarrollar en ensamblador un conjunto de funciones, F1 y F2 de acuerdo a los prototipos de cabecera especificados abajo. Estas funciones están relacionadas con cálculos matemáticos y formato del código de barras, que serán llamadas desde el programa principal **pract3.c** que se proporciona con el material de la práctica.

F1: unsigned char computeControlDigit (char* barCodeASCII); **(4 ptos)**

La función realizará el cómputo automático del dígito de control GTIN-13 de la codificación GS1. Recibirá un puntero a la cadena completa de caracteres ASCII del código de barras y retornará el valor decimal del dígito de control.

F2: void decodeBarCode(char* in_barCodeASCII, unsigned int* countryCode, unsigned int* companyCode, unsigned long* productCode, unsigned char* controlDigit) **(3 ptos)**

Leerá los 13 caracteres de entrada del código de barras y obtendrá cada campo almacenando en los parámetros de entrada-salida correspondientes al código de país, el código de empresa, el código de producto y dígito de control.

Programa pract3b.asm (3 ptos)

En el módulo pract3b.asm se debe desarrollar en ensamblador una función F3 que será llamada también desde el programa principal **prac3.c** de acuerdo al siguiente prototipo:

F3: void createBarCode(unsigned int countryCode, unsigned int companyCode, long int productCode, unsigned char controlDigit, char* out_barCodeASCII); **(3 ptos)**

La función F3 deberá generar los 13 caracteres ASCII del código de barras, leyendo las variables de entrada que se corresponden con cada uno de los campos que lo forman.

Nota: unsigned char controlDigit no respresenta un carácter ASCII; es el decimal que se corresponde al dígito de control pero almacenado en una variable de tipo char para que ocupe menos en memoria (short integer).

Notas adicionales:

- El compilador precede todas las referencias externas con el carácter "_", por lo que es necesario que todas las funciones desarrolladas en ensamblador comiencen con el mismo (ejemplo: _computeControlDigit)
- Todas las funciones desarrolladas en ensamblador deben ser declaradas como PUBLIC para poder ser llamadas desde el programa principal
- No se permite la declaración de variables globales en ensamblador. De hecho, cada módulo desarrollado en ensamblador contendrá **exclusivamente** un solo segmento de código, que debe comenzar con las siguientes líneas:

```
<nombre módulo> SEGMENT BYTE PUBLIC 'CODE'
ASSUME CS: <nombre módulo>
```

- El programa a desarrollar en C (pract3.c) debe estar compilado en modelo LARGE. Las funciones desarrolladas en ensamblador serán FAR.

- El programa pract3.c pedirá al usuario diferentes valores necesarios para probar cada una de las funciones desarrolladas en ensamblador en los diferentes módulos, mostrando el resultado obtenido para cada operación en el formato adecuado de manera que facilite su comprensión.
- Todas las cadenas de caracteres en C acaban con un carácter NULL (último byte a cero).

ENTREGA DE LA PRÁCTICA: Fecha y contenido

Se deberá subir a Moodle un fichero zip que contenga los ficheros fuentes de los programas y el fichero makefile. Sólo podrá ser subido por uno de los miembros de la pareja.

Los ficheros a entregar deberán contener en la cabecera los nombres de los autores y el identificador de la pareja. Así mismo, el código de los ficheros entregados deberá estar correctamente tabulado y comentado. La falta de comentarios o la baja calidad de éstos, será calificada negativamente.

El límite de fecha de subida de los ficheros, para cada grupo es el siguiente:

Grupo Miércoles: 02 de Abril a las 23:55h
Grupo Viernes: 04 de Abril a las 23:55h

Anexo: Compilación de un proyecto desarrollado en C y ensamblador

En esta práctica disponemos de 3 ficheros, el programa principal escrito en C (pract3.c) y los 2 módulos que contienen las funciones escritos en ensamblador (pract3a.asm y pract3b.asm).

Para realizar la compilación del programa en C se utilizará el compilador del TurboC (tcc). Para conocer todas las opciones de compilación que ofrece podemos ejecutar tcc sin parámetros dentro del DosBox. Los módulos en ensamblador se compilarán como en las prácticas anteriores, utilizando el tasm.

El programa en C debe ser compilado con la opción *–ml* (memory model large), mientras que los módulos en ensamblador deben ser ensamblados con la opción */ml* (case sensitivity on all symbols).

Para generar el ejecutable final "pract3.exe" podemos utilizar el siguiente makefile (respetando los tabuladores):

```
all: pract3.exe

pract3.exe: pract3.obj pract3a.obj pract3b.obj

	tcc -v -ml -Lc:\compila\tc\lib pract3.obj pract3a.obj pract3b.obj

pract3.obj: pract3.c

	tcc -c -v -ml -Ic:\compila\tc\include pract3.c

pract3a.obj: pract3a.asm

	tasm /zi /ml pract3a,,pract3a

pract3b.obj: pract3b.asm

	tasm /zi /ml pract3b,,pract3b
```