Proyecto programado 1

1st Jonathan Quesada Salas
Tecnológico de Costa Rica
Sede Interuniversitaria de Alajuela
Alajuela
nuwidra@gmail.com

2nd Valeria Calderon Charpentier
Tecnológico de Costa Rica
Sede Interuniversitaria de Alajuela
Alajuela
valecharpentier29@gmail.com

3rd Alejandro José Loaisa Alvarado
Tecnológico de Costa Rica
Sede Interuniversitaria de Alajuela
Alajuela
ale.loaiza30@gmail.com

Palabras clave: IEEE, ASM, multiplicación, corrimientos, binario

***Abstract*—El programa en cuanto a su funcionamiento se basaría en la posición de los unos del número binario el cual nos permitirá hacer la suma de multiplicación que comunmente conocemos, para así llegar al resultado de dicha operación de dos dígitos de ocho bits sin signo.**

I. Introduction

Este documento se basará en el algoritmo de multiplicación binaria con corrimietos lógicos a la izquierda dando así una explicación detallada de la misma acerca de los procesos que se llevaron a cabo para realizar este proyecto.

II. Desarrollo

A. Idea inicial del programa

Primeramente con la idea protagonista de dicho programa, es hacer una multiplicación, pero sin usar las funciones de MULT o IMULT, para compensar se usa los corrimientos lógicos a la izquierda, por ejemplo el siguiente número binario 10001 se convierte en 00010, de está manera se debe llegar a la solución de la multiplicación.

B. Lógica de la multiplicación

Para un mayor entendimiento es mejor explciarlo con un ejemplo, pero antes de todo hay que tener en cuenta lo siguiente es sobre la lectura del número binario como ejemplo tenemos el 69 que en binario sería 1000101 el cual y para esto hay que expresarlo de manera exponencial de base 2 ya que estamos hablando de unos y ceros, para definir que exponente tendrá dicha base se tomará en cuenta los unos que tenga el número binario y se contará las posiciones en las cuales esten dichos unos en el número binario empezando de derecha a izquierda iniciando con el cero por ejemplo tomemos el binario que teniamos antes 1000101 las posiciones serían de está manera [6][5][4][3][2][1][0] teniendo en cuenta esto se puede sacar el vamos de 69 en de modo que es igual decir:

$$2^6 + 2^2 + 2^0 = 69 \tag{1}$$

C. Enmascaramiento

El enmascaramiento de datos consiste en agarrar un numero y utilizarlo como base, con ello se realizarán diversas pruebas y comparaciones que permitirán al sistema admitir datos con similitudes a la máscara añadida, la mascara debe tener información o estructura similar al dato que se desea admitir y comprobar, en el proyecto implementamos mascaras que verificaban si el numero era viable para realizar una multiplicación en el ya que se debía verificar que cada numero ingresado cumpliera con un ”1” al final.

D. Idea principal del menu

En cuanto al proyecto programado se solicitó realizar un menú interactivo con el usuario el cual en este trabajo se tomo basó en la idea de solamente ejecutar el program, primeramente se le solucitará al usuario ingresar el multiplicando y después poder ingresar el multiplicador, ya teniendo ambos datos en el programa se multiplicarian con los corrimientos anteriormente mencionados dando así un mensaje por consola sobre el resultado que daría dicha opecación.

E. Programación del menú

Ya siendo más concisos en cuanto a la elavoración del menú y sus validaciones iniciaremos explicando las funcionabilidades que tendrá el .data, este mismo contendrá los mensaje que saldrán en consola con el nombre de msgX este mismo mensaje debe terminar con ”,0”, se definen los número de tamaño de 8 bits (BYTE) de manera indefinida ”?” y por último la mascara que será uno. Continuamente el .code se inicalizará definiendo el reguistro EAX en 0 para que está manera se evite cualquier tipo de basura del registro. Se moverá el msg0 que es ”Digite el multiplicando” en el registro EDX, para luego hace un call el cual va hacer que lea el msg0 con WriteString el cual escribe una cadena con terminación nula en la ventana de consola para luego hacer otro call pero está vez será a ReadDec el cual lee un entero decimal sin signo de 32 bits del teclado y devuelve el valor en EAX, luego se mueve el número ingresado a AL. Se usa CMP para comparar lo que está en EAX, pero AL es un subregistro de EAX, o sea se tomará lo que hay en AL y lo compara con 0 para proceder a realizar un salto condicional el cual es JNGE el cual es que salta si no es mayor o igual de está manera haremos que rechace los números negatovos, o bien los dígitos con signo retornando un mensaje de ”Número inválido” el cual con ayuda de ”call WriteString” se puede mostrar dicho mensaje por consola y se terminaría el programa; si es el caso que se ingrese un número sin signo se compara el registro EAX con 255 ya que este número es lo máximo que puede alcanzar un dígito de 8 bits (1111 1111) ya comparando lo

anterior se procede hacer dos saltos lógicos el cual uno sería el JG el cual es que salta si es mayor, entonces en este caso sobrepasaría el 255 y hace un salto a ”NumeroInvalido” el cual se llama con call ”writeString” para leer el mensajer de error, para finalizar se ejecutaría con el salto de JLE el cual sirve para saltar si es menor o igual, de está manera sabemos que el número hasta ahora no es negativo y no es mayor a 255 por lo cual sabemos que es válido, por la presente se procede hacer un salto a ”valido” realizar el mismo procedimiento que se comentó para analizar el primer dígito ingresado, pero en este caso con el segundo dígito ingresado, para luego hacer el último salto que sería ya propiamente a realizar el algoritmo anteriormente explicado con los números correspondientes a las especificaciones.

III. Conclusión

El algoritmo brinda la funcionalidad deseada, ejecuta sus movimientos a la izquierda de manera correcta y con ello permite una multiplicación exacta, su código tiene una estructura legible, lo que permite tener una comprensión sencilla y más eficiente del trabajo realizado, se generan diversos tipos de saltos según su condición y necesidad que permiten y ejecutan la validez y eficiencia del código, las mascaras permiten un acceso directo y generalizado de los números que se deben de admitir, cada sistema fue realizado con irvine, que tiene una funcionalidad amplia, con una código entendible y con estructuras acopladas a cada uno de los casos, también genera abreviación de código lo cual permite que su escritura sea más sencilla.

References

[1] Irvine, Kip R., and Alfonso Vidal Romero Elizondo. Lenguaje ensamblador para computadoras basadas en Intel. Pearson Prentice Hall. vol. 5, pp. 145–174, 2008.