Implementación de algoritmo optimizado de multiplicación en 8 bits para lenguaje ensamblador

1st Joseph Valenciano Madrigal
Tecnológico de Costa Rica
jvalencianom08@yahoo.com

2nd Erick Blanco Fonseca
Tecnológico de Costa Rica
erickbf22@gmail.com

3rd Emmanuel Murillo Sánches
Tecnológico de Costa Rica
e-mursa@live.com

***Abstract*—El primer proyecto programado de arquitectura de computadores básicamente consta de dos secciones, las cuales son implementar una función o método de multiplicación entera sin signo de ocho bits, pero mediante la técnica de corrimientos lógicos a la izquierda, pero inicialmente debe ser programada en un lenguaje de prototipado rápido, seguidamente debe pasarse a código de ensamblador(MASM), pero a diferencia que este debe tomar los datos del usuario en la terminal y validar que sea un numero que únicamente contenga ocho bits.**

I. Introducción

Para este primer proyecto se buscará el aprendizaje y puesta en práctica de los distintos operandos de ensamblador como ”MOV”, ”ADD”, ”SUB”, etc. Además de la implementación de directivas como ”PROC”, ”PROTO”, e ”INVOKE”, junto con el uso de la biblioteca Irvine, que ayudará con la disposición de estas directivas en el código.

Primero se implementará el algoritmo en python, ya que es un lenguaje muy sencillo y cómodo para el programador. Con esta base será un poco más fácil, pasar o implementar el código en ensamblador, porque se tiene la base en un lenguaje de alto nivel y además ya se conoce el funcionamiento del algoritmo.

Por cierto, el algoritmo consiste en la multiplicación de dos números enteros, pero con una gran diferencia, que se efectuará con corrimientos de bits lógicos a la izquierda, para aprender y experimentar el uso del operando ”SHL” en ensamblador.

II. Algoritmo en Python

Se escogió python como lenguaje de prototipado rápido para este algoritmo de multiplicación binaria mediante corrimiento de bits, porque es un lenguaje de programación el cual conocemos y nos sentimos muy cómodos con él.

básicamente la función recibe como parámetros, dos números enteros los cuales se multiplicarán, seguidamente de esto, se convierten de decimal a binario, para poder trabajar con el corrimiento de bits.

Python toma los números binarios como un ”string”, por lo que es necesario revertir el orden del binario del multiplicador(segundo numero binario), para poder tomar sus índices en una posición correcta, para posteriormente hacer el corrimiento hacia el izquierda con el índice tomado.

Posteriormente se realiza un ciclo para que recorra el multiplicador y verifique en que posiciones se encuentran los bits activos, es decir los ”uno”. Con dichas posiciones(índices), luego se realiza el ”Shift left” al multiplicando(primer numero binario). Se suman todos estos resultados y se guardan en una variable, finalmente retornando el resultado de la multiplicación.

Este algoritmo hizo una ”simulación” de lo que haría el ensamblador, conociendo que en ensamblador no habría que hacer el proceso de pasar un numero de decimal a binario, porque ya lo guarda como tal.

III. Diagrama de lenguaje prototipado

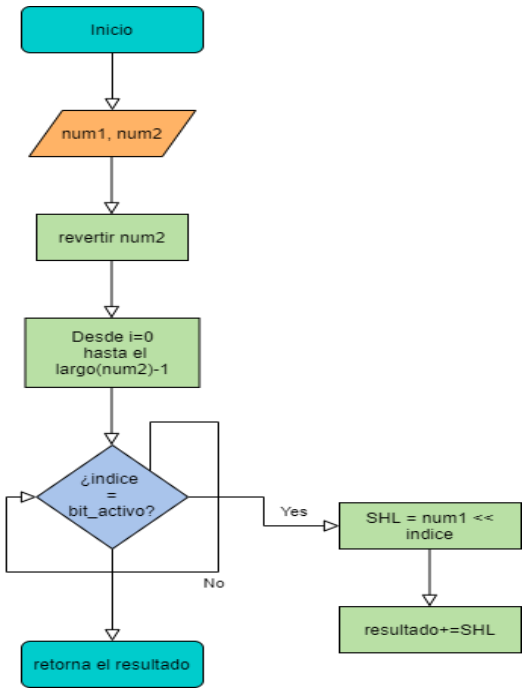

Fig. 1. Diagrama de flujo del algoritmo de python.