

ITEC Sede Interuniversitaria de Alajuela

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

Integrante 1: Steven Alvarado Aguilar*

Carnet: 2019044923

Integrante 2: Andres López Sánchez[†]

Carnet: 2019160378

Integrante 3: Fabian López Sánchez[‡]

Carnet: 2019064821

Profesor: Daniel Kohkemper

I Semestre 2020

Abstract—En el siguiente paper se encuentra la explicación de cómo funciona el código de Python para crear una función de multiplicación con la técnica de corrimientos lógicos a la izquierda, esta implementación fue hecha con base a técnicas que se pueden trasladar a código ensamblador, se explica la manera en cómo opera la técnica de corrimientos lógicos a la izquierda para agilizar la creación de códigos y como es una alternativa viable a la multiplicación normal.

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

1. Introducción

Se hace un algoritmo en el cuál implementamos lenguaje de alto nivel dónde debíamos demostrar que podíamos abstraer conceptos y funciones de bajo nivel e implementarlas en alto nivel con mayor facilidad y en menos líneas de código.

Con esto sabemos que el objetivo de este trabajo es lograr un mayor conocimiento de cómo fueron los avances del bajo nivel al alto nivel a través de los años y mejorando las horas de programación en un código simple o complejo.

Se obtiene el resultado de la investigación con base a los materiales oficiales del curso y por los vídeos de explicación que brinda el profesor en la lección 13 de la semana 7.

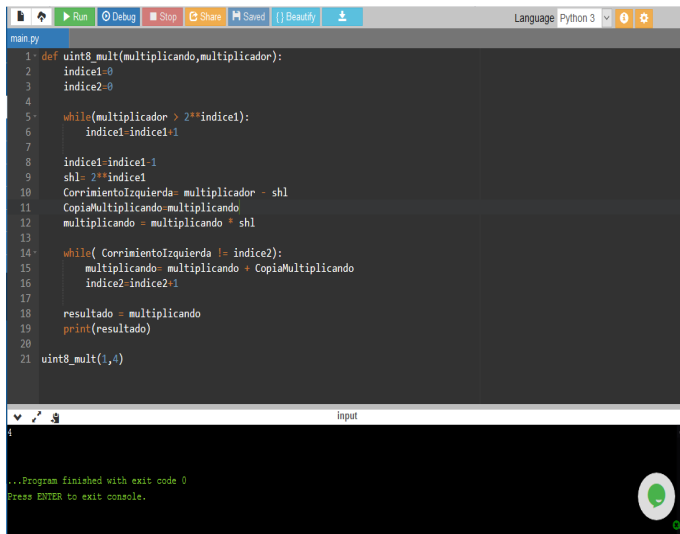
El concepto estudiado de ensamblador es SHL qué es el (desplazamiento a la izquierda) que realiza un desplazamiento lógico a la izquierda en el operando de destino, llenando los bits inferiores con 0. El bit superior se mueve hacia la bandera Acarreo, y el bit que estaba en la bandera Acarreo se pierde.

2. El Algoritmo

Utilizamos el lenguaje de Python ya que, no logramos hacerlo en Octave y por esta razón migramos a Python un lenguaje que sabemos utilizarlo mejor y que ya teníamos

experiencia y nos permitió mejorarlo de la mejor manera para implementar los corrimientos a la izquierda. El algoritmo tiene dos variables de entrada el multiplicando y el multiplicador, utiliza dos índices para llevar acabo sus funciones. Con el primer ciclo se compara el multiplicador con 2 elevado a la n (dónde n es la variable indice1) para identificar la potencia a la cual se va a elevar 2 a la n según lo ingresado en el multiplicador una vez hecho esto sale del while. Con el valor del indice1 debemos restarle uno y la guardamos en el indice1 esto porque debemos restarlo para que nos quite un valor del total para que la función de shl nos sirva de la manera en que necesitamos. La variable shl adquiere el formato 2 elevado al indice1 nuevo.

Siguiente al multiplicador se le resta su potencia de dos más cercana para encontrar el corrimiento a la izquierda esto nos permitirá encontrar cuantas veces se necesita sumarle el multiplicando al resultado final. Se le hace una copia a multiplicando llamada copiamultiplicando esto para utilizarla más adelante y al multiplicando se le aplica una multiplicación del shl esto porque en ese paso hacemos un nuevo valor de corrimiento a la izquierda. En el último ciclo while se compara el CorrimientoIzquierda distinto del indice2 esto para encontrar la cantidad de veces que se le sumará el multiplicando más la CopiaMultiplicando el indice2 aumentará en cada repetición hasta que el CorrimientoIzquierda sea igual al indice2 y una vez hecho esto la variable multiplicando se guardará en resultado y se imprimirá el resultado final en pantalla.



```
1: def uint8_mult(multiplicando, multiplicador):
2:     indice1 = 0
3:     indice2 = 0
4:
5:     while (multiplicador > 2**indice1):
6:         indice1 = indice1 + 1
7:
8:     indice1 = indice1 - 1
9:     shl = 2**indice1
10:    CorrimientoIzquierda = multiplicador - shl
11:    CopiaMultiplicando = multiplicando
12:    multiplicando = multiplicando + shl
13:
14:    while (CorrimientoIzquierda != indice2):
15:        multiplicando = multiplicando + CopiaMultiplicando
16:        indice2 = indice2 + 1
17:
18:    resultado = multiplicando
19:    print(resultado)
20:
21: uint8_mult(1, 4)
```

Input

...Program finished with exit code 0
Press ENTER to exit console.

Utilizamos este algoritmo ya que, representa el alto nivel de un lenguaje de programación para saber que se hace en bajo nivel.

Esto es importante para saber cuál es el procedimiento que se hace por ejemplo saber que la instrucción SHL agrega un 0 a la derecha según la cantidad que fue implementada en el multiplicador con base en nuestro algoritmo y hace que todos los valores se desplacen a la izquierda y produzca un nuevo valor que sería la multiplicación humana para con una intensión distinta que son desplazamientos a la izquierda cómo lo dice el material de clase.