LABORATORIO DE PROGRAMACIÓN EN LENGUAJE ENSAMBLADOR x86-16bits

Aritmética multiprecisión

Objetivo

El objetivo de esta práctica es presentar algunas nociones acerca de las operaciones aritméticas multiprecisión también conocidas como de ***precisión arbitraria*** o ***bignum*** (del inglés *big number* o número grande).

Para alcanzar este objetivo, se propone la programación de operaciones básicas con enteros de tamaño arbitrario. Concretamente, trabajaremos con operaciones de suma y resta sobre números representados en BCD y en binario. También se propone realizar desplazamientos unitarios a derecha e izquierda sobre números binarios.

Precisión arbitraria: concepto

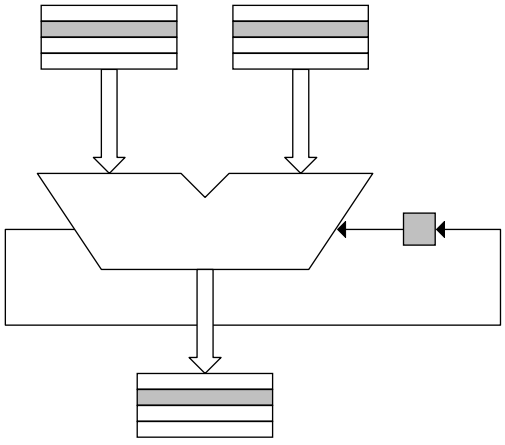
Con el nombre de precisión arbitraria designamos las representaciones numéricas que tienen tantos dígitos como se requiera. Normalmente, las representaciones numéricas en las máquinas se ajustan a un tamaño fijo (tamaño *palabra*) que es el tamaño característicos de los registros y operadores del procesador. Los datos de precisión arbitraria suelen ser de tamaño múltiplo de la palabra y se almacenan en memoria como *arrays*. Estos datos de tamaño arbitrario pueden estar representados en cualquiera de los sistemas de representación habituales: signo-magnitud, complemento a 2, BCD, etc.

Sobre los números de precisión arbitraria se realizan todas las operaciones aritméticas habituales. Existen algoritmos que las llevan a cabo con mayor o menor velocidad y uso de recursos. En general, estos algoritmos forman lo que llamamos aritmética multiprecisión o aritmética de precisión arbitraria. Hoy en día existen librerías de funciones *bignum* como por ejemplo GMP (*GNU Multiple Precision Arithmetic Library*).

La aritmética sobre números de precisión arbitraria tiene mucha importancia en criptografía.

Operaciones aritméticas

Puesto que los operadores de que dispone el procesador se ajustan al tamaño palabra, las operaciones aritméticas sobre números de precisión arbitraria se descomponen en una sucesión de operaciones de dicho tamaño teniendo cuidado de proporcionar a la etapa siguiente la información necesaria para mantener el significado numérico. Es decir, cada operación sobre una componente del operando debe suministrar el acarreo correspondiente a la siguiente.



En la figura, se ilustra el hecho de que la operación *i*-ésima entre componentes del número de precisión arbitraria debe tener en cuenta el acarreo de la operación *i*–1-ésima.

A tener en cuenta

Lógicamente, además de realizar la operación aritmética requerida, hemos de tener en cuenta algunos aspectos adicionales. El primero de ellos es que tanto los operandos como el resultado deben ser alojados siempre en memoria ya que no existe ningún registro de tamaño suficiente (salvo que algún operando no sea *bignum*).

Otro aspecto importante es la información que describe cómo es el resultado, es decir, si es cero, si tiene acarreo, qué signo tiene el resultado y si se ha producido desbordamiento.

La información acerca del acarreo, el signo y el desbordamiento se obtiene con la última iteración del procedimiento aritmético. La información de si el resultado es cero o no es consecuencia de evaluar si es cero o no en cada una de las iteraciones. En general, el resultado sólo es cero si lo es en todas y cada una de las iteraciones.

Cuando realizamos un programa para operar con números de precisión arbitraria, hemos de tener en cuenta que entre cada dos iteraciones consecutivas hemos de salvar los *flags* ya que el proceso de cargar componentes nuevas, actualizar punteros, etc., puedo provocar que perdamos dicha información alterando los resultados.

Prácticas

A) Escriba un programa que realice un desplazamiento unitario a la izquierda sobre un número de precisión arbitraria. Evalúe si el resultado es cero y si existe acarreo

Declare un número de precisión arbitraria constante en el área de datos y otro inicializado a cero del mismo tamaño. Declare también una variable de tamaño byte llamada ‘acarreo’ y otra igual llamada ‘cero’. Ambas estarán inicializadas con el carácter ‘0’ y ‘1’ respectivamente, indicando que no existe acarreo y que el resultado es nulo.

Programe la rutina que realiza el desplazamiento unitario a la izquierda sobre componentes de tamaño *word* comenzando por la de menor peso. En cada iteración salvamos el *flag* de acarreo para usarlo en la siguiente iteración. También evaluamos si el resultado es nulo en esa componente observando el *flag* de cero. Si no es nulo, escribimos el carácter ‘0’ en la variable ‘cero’ indicando que el resultado ya no será nulo.

En la última iteración evaluamos el acarreo. Si el *flag* de acarreo indica que existe, escribimos ‘1’ en la variable ‘acarreo’.

Una vez finalizada la operación, podemos presentar los operandos en pantalla así como las variables ‘acarreo’ y ‘cero’. Para presentar los operandos, nos limitamos a escribir su interpretación ASCII byte a byte no su valor numérico. Si el valor del operando es una secuencia de bytes con el mismo carácter ASCII, es fácil comprobar si funciona bien o no el programa. Por ejemplo, si introducimos el carácter ‘1’, es decir, 31h, después del cálculo deberíamos tener 62h, es decir, el carácter ‘b’.

¿Qué habría que cambiar si el desplazamiento fuera a la derecha? ¿Qué diferencias habría que introducir para trabajar con representaciones con signo y sin signo?

B) Suma y resta multiprecisión

Escriba un programa que realice la suma y resta de dos operandos de precisión arbitraria. Al igual que en el caso anterior, además del resultado hemos de obtener la información que describa el mismo, es decir, si es nulo y si tiene acarreo.

C) Suma de números de precisión arbitraria en BCD

Asumiendo que disponemos de 2 números representados en BCD desempaquetado (un dígito BCD por byte) de precisión arbitraria, escriba un programa que calcule la suma. Tenga en cuenta que hay que utilizar las instrucciones de ajuste ASCII. No olvide el acarreo y dar cuenta de si el resultado es nulo o no lo es.

Los números en BCD de precisión arbitraria se pueden declarar como constantes en el área de datos o bien se pueden capturar desde teclado.