CAPÍTULO 4

ARITMETICA Y FUNCIONAMIENTO DEL COMPUTADOR II

1. Sistemas de Representación II y Aritmética con enteros

El algoritmo de multiplicación de Booth para números con signo fue creado por Andrew Booth en 1950, Tras una observación sobre cómo trabajaban los calculadores humanos para hacer más rápida la multiplicación con sus equipos mecánicos. para ello empleaban el complemento a 10 de las cifras del multiplicando.

El algoritmo usa dos pasos. suma y desplazamiento, guiándose por cuatro posibles condiciones. veamos primeramente el algoritmo y luego haremos un ejemplo.

El algoritmo trabaja con los números en complemento a dos.

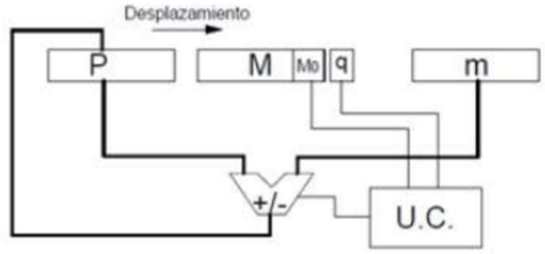
1. Se inicializan tres registros M, S y Q con el multiplicando, su complemento a dos y el multiplicador.
2. Se inicializa un registro A a 0 y un registro de un único bit Q -1 a 0.
3. Si Q0 y Q-1 no son iguales (0-1, 1-0):
4. Si 0,1: Se suma A + M y se almacena en A. El acarreo se pierde.
5. Si 1,0: Se suma A + S y se almacena en A. El acarreo se pierde.
6. Se desplaza un bit a la derecha A, Q, y Q -1. (Se rellena A extendiendo el signo)

Si Q0 y Q -1 son iguales (1-1, 0-0): sólo se realiza desplazamiento de A, Q, y Q -1. Se vuelve a 3 hasta que se agoten los bits del multiplicador original. El resultado del producto es la concatenación de A y Q.

De forma resumida se puede decir que las reglas son:

1. Q0Q1 =10 => sumar S a A y desplazar
2. Q0Q1 =00 ó 11 => Sólo desplazar
3. Q0Q1 =01 => sumar M a A y desplazar

El algoritmo de Booth reduce el número de sumas necesarias quitando las que son 0.
Llegados a este punto es conveniente que el alumno practique este algoritmo con algún ejercicio en el cual uno de los operandos sea de signo negativo para comprobar que se obtiene el signo correcto.



1. M<= multiplicando | M = multiplicador | P = q <= 0.
2. M_oq =00 ó M_oq =11 => DD_{arit}(P,M,q)
M_oq =10 => P <= P-m | DD_{arit}(P,M,q)
M_oq =01 => P <= P+m | DD_{arit}(P,M,q)

El paso 2 se realiza *n* veces, siendo n el número de bits del multiplicador.
Una vez finalizado el resultado se hallará en los registros P(parte más significativa) y M (parte menos significativa).

Por otro lado, tenemos el algoritmo de división con restauración o signo que es empleado para realizar la división de enteros con signo trabajando con los números en complemento a dos.

Su algoritmo es:

Primero se inicializan dos registros M y S con el divisor y su complemento a dos. Después se inicializan dos registros A y Q con el dividendo en complemento a dos extendido al tamaño 2n.

Desplazar A y Q un bit a la izquierda.

1. Si M y A tienen el mismo signo: Se suma A + S y se almacena en A. El acarreo se pierde.

2. Si M y A tienen distinto signo: Se suma A + M y se almacena en A. El acarreo se pierde.

La operación tiene éxito si el signo de A no cambia.

1. Si tiene éxito o (A = 0 y Q = 0), hacer Q0 = 1.

2. Si no tiene éxito y (A != 0 ó Q != 0), entonces hacer Q0 = 0 y restablecer el valor anterior de A.

Se vuelve a 3 hasta que se agoten los bits del dividendo original.

El resto está en A.

1. Si los signos del divisor y el dividendo eran iguales, el cociente está en Q.

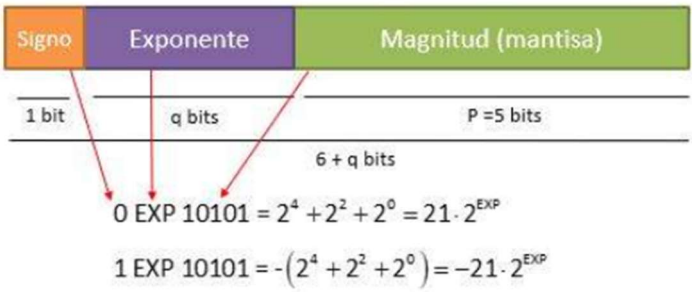
2. Si no, el cociente es el complemento a dos de Q

Conviene reforzar antes de ver un ejemplo el concepto de "éxito" empleado dentro del algoritmo. Se dice que una división parcial tiene éxito, si el signo del registro de acumulación no cambia.

- Las reglas se podrían resumir de la siguiente manera:
- Desplazar A y Q 1 bit a la izquierda
- Si M y A tienen el mismo signo, A= A+S y se ignora el acarreo
- Si signo(M) no es igual a signo(A), A= A+M y se ignora también el acarreo.
- La operación tiene éxito si el signo de A no cambia (actuar según proceda)
- Si signo dividendo =signo divisor al final => cociente = Q, en caso contrario cociente =Ca2(Q).

2. Representación en coma flotante

La coma flotante en signo se representa con un bit, siendo 1 negativo y 0 positivo:



Las magnitudes se deben presentar normalizadas. La coma se puede situar entre cualquier par de bits o en alguno de los extremos. En una magnitud normalizada, se sobreentiende que la coma se sitúa a la izquierda del bit más significativo (el más a la izquierda). En S-M, una magnitud está normalizada si el bit que acompaña al bit de signo es igual a 1.

S-M	Valor	Normalizado
1 01010	- 0,01010	NO
0 01010	+ 0,01010	NO
1 10101	- 0,10101	SÍ
0 10101	+ 0,10101	SÍ

S-M codificado	S-M representado
0 01011	0 101011
1 01011	1 101011

1 + 5 bits 1 + 6 bits Se gana un bit

Como el bit más significativo de la magnitud siempre debe ser 1, nunca se representa, sino que se entiende que está implícito. Dicho de otra forma, el primer bit de la mantisa se suprime, se dice que va "implícito" y se gana un bit de precisión. En nuestro ejemplo:

0,28125 en decimal se representaría como 0 001 0010. Donde el primer 0 indica que es positivo, el 001 es el exponente que explicaremos seguidamente y el 0010 corresponde al 1001 al cual se ha quitado el primer 1 por considerarlo implícito (por tanto no se almacena) y a su vez hemos ganado más precisión con un cero más al final.

		S-M	Valor
Negativos	Mayor	1 111111	$-\left(1-2^{-6}\right)$
	Menor	1 100000	-2^{-1}
Positivos	Menor	0 100000	2^{-1}
	Mayor	0 111111	$\left(1-2^{-6}\right)$

Bits implícitos

El exponente Está codificado en representación sesgada, también llamada exceso M:

- El exponente puede ser negativo, pero el signo no se codifica en ningún bit.
- El valor codificado en el exponente es igual al valor en binario más un valor fijo llamado sesgo. El valor típico del sesgo es 2q-1-1. En el ejemplo, el sesgo de 4 es:
- El sesgo permite que se codifiquen exponentes negativos. Con q = 4 se puede codificar desde 0000 hasta 1111 en binario puro.

Representación sesgada también es llamada exceso M. Para un valor genérico de q bits, se tiene que el rango del exponente en representación sesgada es: [-(2q-1 - 1), 2q-1].

IEE 754 es un estándar de representación de punto flotante de precisión simple con 32 bits. se utiliza 1 bit para el signo, 8 bits para el exponente y 23 para la magnitud. El sesgo vale 27-1 =127.

- Precisión simple (Float)

- Exponente en exceso 127 Exp = (valor - 127)
  - Exp 0 y Mantisa 0. Numero 0
  - Expo 0 y Mantisa distinta de 0. Número desnormalizados 0, pero con exponente -126 no -127
  - Exp 255 y Mantisa 0. + Infinito y –Infinito
  - Exp 255 y Mantisa distinto de 0. Not a number (NaNs)
  - Exp 1 a 254 Números normalizados con Exponente [-126,127]
- Mantisa normalizada a 1, y bit implícito

valores especiales que resumidos son:

- Exponente 0 y Mantisa 0 = Número 0 (ahora sí que lo podemos representar +0y -0)
- Exponente 255 y Mantisa 0 = + o – menos infinito
- Exponente 255 y Mantisa distinto 0 = NaN not a number)

- Precisión simple (Float)

- Máximo positivo representable $(2\text{-}2^{-23})x\ 2^{127} = 3.402823 \times 10^{38}$
- Mínimo positivo normalizado $(1,)x\ 2^{-126} = 1.18 \times 10^{-38}$
- Máximo positivo no normalizado $(1 - 2^{-23})x\ 2^{-126} = 1.175 \times 10^{-38}$
- Mínimo positivo no normalizado $(2^{-23})x\ 2^{-126} = 1.4 \times 10^{-45}$
- Dos representaciones para el 0 (-0 y +0)

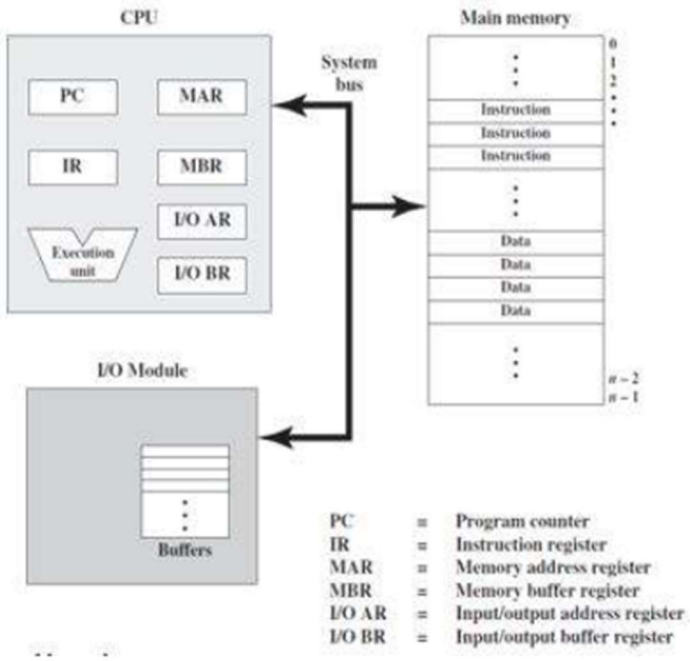
EXPONENTE	MANTISA	Numero
0	0	+0 [signo (0)] - 0 [signo (1)]
255	0	+ infinito [signo (0)] - infinito [signo (1)]
255	≠0	NaN (Not a Number) (ej. 0/0, √ - 1)
1 a 254 *(Sin codificar -126 a 127)*	Cualquier valor	Número normalizados 1, ------- (bit implícito) (Número normales)
0 (Sin codificar valor fijo de exponente -126)	Cualquier valor ≠0	Número (des)normalizados 0, ------- (no hay bit implícito) (Número muy pequeños) $< 1 \times 2^{-126}$

3. Funciones del computador

La estructura de un computador programable por software comprende varios componentes interconectados. La CPU (Unidad Central de Procesamiento) es el núcleo del sistema y se comunica con la memoria y los dispositivos de entrada/salida (E/S) a través de registros. Por ejemplo, el MAR (Registro de Dirección de Memoria) especifica la próxima dirección de lectura/escritura en la memoria, mientras que el MBR (Registro de Búfer de Memoria) almacena los datos a leer o escribir. Además, la CPU intercambia datos con la E/S mediante registros como el IOAR (Registro de Dirección de E/S) y el IOBR (Registro de Búfer de E/S).

El PC (Contador de Programa) es otro componente vital que contiene la dirección de la siguiente instrucción a ejecutar. El IR (Registro de Instrucción) almacena la instrucción actual que se está ejecutando. La memoria del sistema está formada por una serie de posiciones identificadas por direcciones, donde cada posición almacena un valor binario interpretable como instrucción o dato.

Las transferencias de datos entre los componentes se realizan a través del bus del sistema, que actúa como un canal de comunicación para los datos en el sistema. Este bus facilita la comunicación entre la CPU, la memoria y los dispositivos de E/S.

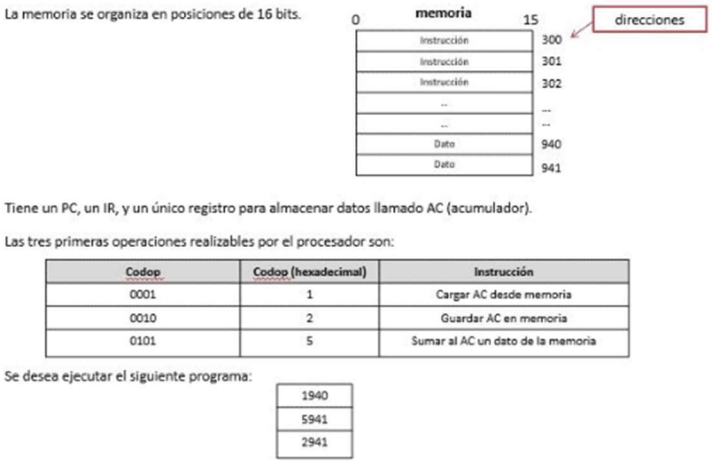


Ejecución de un programa

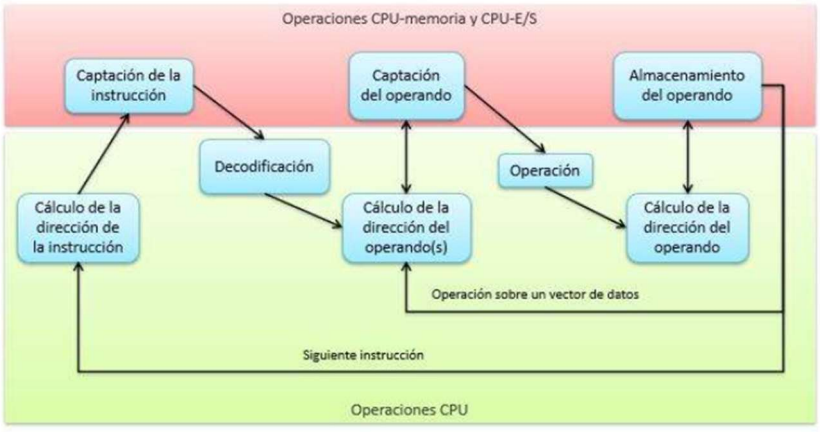
Un programa software se ejecuta mediante un proceso llamado ciclo de instrucción, que comprende dos fases esenciales: el ciclo de captación y el ciclo de ejecución. Durante el ciclo de captación, la CPU extrae la instrucción indicada en el Contador de Programa (PC) desde la memoria y la almacena en el Registro de Instrucción (IR), mientras el PC se incrementa para apuntar a la próxima instrucción.

En el ciclo de ejecución, la CPU interpreta la instrucción para determinar la acción a realizar, que puede ser transferir datos entre la CPU y la memoria o entre la CPU y los dispositivos de E/S, aplicar operaciones lógicas o aritméticas, o alterar el valor del PC para la siguiente instrucción.

Por ejemplo, consideremos un computador que maneja datos de 16 bits de longitud y cuyo formato de instrucciones y datos se muestra en la imagen adjunta.



El siguiente diagrama resume lo que se conoce como ciclo de instrucción. Muestra de forma esquemática los diferentes estados por los que va pasando la ejecución de una instrucción, los cuales han sido comentados en el video sin detenernos a nombrarlos.



Dentro del ciclo de instrucción, existe un ciclo adicional llamado ciclo de interrupción.

- Una interrupción es un mecanismo que detiene temporalmente el funcionamiento normal de la CPU para que pueda gestionar una incidencia, como una operación de entrada/salida (E/S), temporización, fallo de hardware o fallo de programa.

- Las interrupciones de E/S son las más comunes, ya que los dispositivos de E/S suelen operar más lentamente que la CPU.
- Es importante evitar que la CPU espere a que la E/S termine su operación para continuar trabajando, ya que esto sería ineficiente.
- Las interrupciones permiten que la CPU continúe ejecutando instrucciones mientras la E/S realiza su trabajo, lo que mejora la eficiencia del sistema.
- El tema de las interrupciones se profundizará en la asignatura de Sistemas Operativos.