Capítulo 5. Sistemas de numeración

¿Te interesa la historia y divulgación de la Informática?

Puedes acceder a más contenidos relacionados en:Informática: apuntes y divulgación.

Ya he leído, cierra la ventana

5.1 Introducción

Como se ha visto, un ordenador es una máquina que procesa información. La ejecución de una tarea implica la realización de unos tratamientos, según especifica un conjunto ordenado de instrucciones (es decir, un programa) sobre unos datos. Para que el ordenador ejecute un programa es necesario darle información de dos tipos:

- Instrucciones que forman el programa
- Los datos con los que debe operar ese programa

Uno de los aspectos más importantes relacionado con la información, es cómo representarla. Normalmente se le da al ordenador en la forma usual escrita que utilizan los humanos, es decir, con ayuda de un alfabeto o conjunto de símbolos, los caracteres.

Los caracteres que se utilizan para la representación externa son:

- Numéricos: Constituidos por las diez dígitos en el sistema decimal
- Alfabéticos: Letras mayúsculas y minúsculas
- Especiales: Son símbolos no incluidos en los grupos anteriores, como:), (, *, /, +, -, [,]...

Al conjunto de los dos primeros grupos se le denominan caracteres alfanuméricos.

Veremos cómo estos caracteres usados en la representación externa son representables en los ordenadores. Este paso de una representación a otra se denomina codificación y el proceso inverso decodificación.

Por lo tanto hay dos niveles en la representación de la información

- Nivel de representación externa: Usada por las personas e inadecuada para el ordenador
- Nivel de representación interna: Adecuada al ordenador y no inteligible directamente por el ser humano.

Las informaciones más complejas se reducirán a un conjunto de informaciones elementales por técnicas de codificación.

Los elementos básicos que constituyen un ordenador son de naturaleza binaria, ya que sólo pueden adoptar dos valores, 0 y 1 (corresponden a dos niveles de tensión, dos valores de corriente, dos situaciones de una lámpara...). Al tener que traducir toda la información suministrada a ceros y unos es necesario establecer una correspondencia entre el conjunto de todos los caracteres:

{A, B, C, D,...Z, a, b, c,...z, 0, 1,...9, /, +,...}

y el conjunto binario:

$$\{0, 1\}^n$$

de forma que a cada elemento del primero le corresponda un elemento distinto del segundo.

Estos códigos de transformación se denominan códigos entrada/salida (E/S) o externos y se pueden definir de forma arbitraria. Las operaciones aritméticas con datos numéricos se suelen realizar en una representación más adecuada para este objetivo que la del código de E/S. Por ello en el propio ordenador se efectúa una transformación entre códigos binarios, obteniéndose una representación fundamentada en el sistema de numeración en base dos, que al ser una representación numérica posicional es muy apta para realizar operaciones aritméticas.

5.2 Códigos de entrada/salida.

Los códigos de E/S o externos son códigos que asocian a cada carácter una combinación de bit. En otras palabras, un código de E/S es una correspondencia entre los conjuntos:

$$A = \{0, 1,...9, A, B,...Z, a, b,...z, *, +, /...\}$$
y
$$B = \{0, 1\}^n$$

Si se usa un número fijo, n, de bit para codificar los símbolos de A, el valor mínimo de n dependerá del número m de elementos de A. Así:

- Con 2 bit (n=2) podemos hacer 4 combinaciones distintas y se pueden codificar hasta 4 símbolos (m=4) distintos
- Con 3 bit (n=3) podemos hacer 8 combinaciones distintas y se pueden codificar hasta 8 símbolos (m=8) distintos
- Con 4 bit (n=4) podemos realizar 16 combinaciones dise pueden codificar hasta 16 símbolos (m=16) distintos
- ....
- Con n bit pueden codificarse $m = 2^n$ símbolos distintos.

Para codificar m símbolos distintos necesitamos n bit, siendo,

$$n = \log_2 m = 3.32 * \log m$$

Es decir, n debe ser el menor entero que verifique la relación anterior.

Ejemplo: Para codificar las 10 cifras decimales (0, 1,...,9) se necesitarán:

$$n = 3.32 * \log (m) = 3.32 * \log (10) = 3.32 \text{ bit}$$

es decir 4 bit (ya que con 3 sólo podremos codificar 8 símbolos).

Dos codificaciones posibles son las siguientes:

Símbolos	Código 1	Código 2
0	0000	0000
1	1000	0001
2	0100	1001
3	1100	1000
4	0010	0101
5	1010	0100
6	0110	1100
7	1110	1101

8	0001	0011
9	1001	0010

Pueden hacerse codificaciones con más bit de los necesarios; es decir, podríamos establecer códigos de E/S de forma totalmente aleatoria. Obviamente existen códigos normalizados que suelen ser utilizados por los constructores de ordenadores, son conocidos como:

- BCD de intercambio normalizado.
- EBCDIC
- ASCII
- ANSI

BCD DE INTERCAMBIO NORMALIZADO

Usualmente este código utiliza 6 bit, con lo que se pueden representar, m = 2^6 = 64 caracteres. A veces se añade a su izquierda un bit adicional para verificar posibles errores en la transmisión del código (tema que se verá más adelante) de forma que el carácter queda representado por n = 7 bit.

bit de verificación						
6	5	4	3	2	1	0

Los bit 4, 5 son conocidos como bit de zona. Los bit de zona indican el tipo de carácter representado. Ejemplo: 00 para los numéricos. Los bit 0, 1, 2, 3 son conocidos como bit de posición, que coinciden para los caracteres numéricos con la representación en binario natural y para el 0 con la representación del 10.

CODIGO EBCDIC (Extended Binary Coded Decimal Interchange).

Utiliza n = 8 para representar cada carácter, pudiendo codificar hasta m = 2^8 = 256 símbolos distintos.

CODIGO ASCII (American Standard Code for Information Interchange)

Utiliza 7 bit y es de los más utilizados. Normalmente se incluye un octavo bit para detectar posibles errores de transmisión o grabación. Si el octavo bit se emplea para representar más caracteres como letras griegas y símbolos semigráficos, se tiene el denominado ASCII extendido, usado en el PC de IBM y compatibles.

CODIGO ANSI (American National Standards Institute - Instituto Nacional Americano de Estándares)

El estándar ANSI especifica una serie de secuencias de escape, que hacen que el monitor se comporte de distintas formas. Por ejemplo, una secuencia de escape limpia la pantalla, mientras que otra causa que los siguientes caracteres se inviertan.

5.3 Sistemas de numeración más usuales

Los ordenadores suelen efectuar las operaciones aritméticas utilizando una representación para los datos numéricos basada en el sistema de numeración en base 2 (binario natural). También se utilizan los sistemas de numeración octal y hexadecimal, para obtener códigos intermedios. Un número expresado en uno de estos códigos puede transformarse a binario y viceversa.

LOS SISTEMAS DE NUMERACION A LO LARGO DE LA HISTORIA

Seguidamente se comentan los sistemas de numeración que distintas culturas han usado a lo largo de la Historia

· Introducción. El Concepto de Base
· Sistemas de Numeracion Aditivos

o Egipcio

o Griego
· Sistemas de Numeración Híbridos

o Chino
· Sistemas de Numeración Posicionales

o Babilónico

o Maya

Enlace recomendado: Sistema de numeración romano

Introducción. El concepto de base

Cuando los hombres empezaron a contar usaron los dedos, guigarros, marcas en bastones, nudos en una cuerda y algunas otras formas para ir pasando de un número al siguiente. A medida que la cantidad crece se hace necesario un sistema de representación más práctico.

En diferentes partes del mundo y en distintas épocas se llegó a la misma solución, cuando se alcanza un determinado número se hace una marca distinta que los representa a todos ellos. Este número es la base. Se sigue añadiendo unidades hasta que se vuelve a alcanzar por segunda vez el número anterior y se añade otra marca de la segunda clase. Cuando se alcanza un número determinado (que puede ser diferente del anterior constituyendo la base auxiliar) de estas unidades de segundo orden, las decenas en caso de base 10, se añade una de tercer orden y así sucesivamente.

La base que más se ha utilizado a lo largo de la historia es 10 según todas las apariencias por ser ese el número de dedos con los que contamos. Hay alguna excepción notable como son las numeración babilónica que usaba 10 y 60 como bases y la numeración maya que usaba 20 y 5 aunque con alguna irregularidad.

Desde hace 5000 años la gran mayoría de las civilizaciones han contado en unidades, decenas, centenas, millares etc. es decir de la misma forma que seguimos haciéndolo hoy. Sin embargo la forma de escribir los números ha sido muy diversa y muchos pueblos han visto impedido su avance científico por no disponer de un sistema eficaz que permitiese el cálculo.

Casi todos los sistemas utilizados representan con exactitud los números enteros, aunque en algunos pueden confundirse unos números con otros, pero muchos de ellos no son capaces de representar grandes cantidades, y otros requieren tal cantidad de símbolos que los hace poco prácticos. Pero sobre todo no permiten en general efectuar operaciones tan sencillas

como la multiplicación, requiriendo procedimientos muy complicados que sólo estaban al alcance de unos pocos iniciados. De hecho cuando se empezó a utilizar en Europa el sistema de numeración actual, los abaquistas, los profesionales del cálculo se opusieron con las más peregrinas razones, entre ellas la de que siendo el cálculo algo complicado en sí mismo, tendría que ser un metodo diabólico aquel que permitiese efectuar las operaciones de forma tan sencilla.

El sistema actual fue inventado por los hindús y transmitido a Europa por los árabes. Del origen hindú del sistema hay pruebas documentales más que suficientes, entre ellas la opinión de Leonardo de Pisa (Fibonacci) que fue uno de los indroductores del nuevo sistema en la Europa de 1200. El gran mérito fue la introducción del concepto y símbolo del cero, lo que permite un sistema en el que sólo diez simbolos puedan representar cualquier número por grande que sea y simplificar la forma de efectuar las operaciones.

Sistemas de numeracion aditivos

Para ver cómo es la forma de representación aditiva consideremos el sistema geroglífico egipcio. Por cada unidad se escribe un trazo vertical, por cada decena un símbolo en forma de arco y por cada centena, millar, decena y centena de millar y millón un geroglífico específico. Así para escribir 754 usaban siete geroglíficos de centenas cinco de decenas y 4 trazos. De alguna forma todas las unidades están fisicamente presentes.

Los sistemas aditivos son aquellos que acumulan los simbolos de todas las unidades, decenas... como sean necesarios hasta completar el número. Una de sus características es por tanto que se pueden poner los símbolos en cualquier orden, aunque en general se ha preferido una determinada disposición. Han sido de este tipo las numeraciones egipcia, sumeria (de base 60), hitita, cretense, azteca (de base 20), romana y las alfabéticas de los griegos, armenios, judios y árabes.

El sistema de numeración egipcio

Desde el tercer milenio a.n.e. los egipcios usaron un sistema deescribir los números en base diez utilizando los geroglíficos para representar los distintos órdenes de unidades. Se usaban tantos de cada uno cómo fuera necesario y se podian escribir indistintamente de izquierda a derecha, al revés o de arriba abajo, cambiando la orientación de las figuras según el caso. Al ser indiferente el orden se escribían a veces según criterios estéticos, y solían ir acompañados de los geroglíficos correspondientes al tipo de objeto (animales, prisioneros, vasijas etc.) cuyo número indicaban. Estos signos fueron utilizados hasta la incorporación de Egipto al imperio romano. Pero su uso quedó reservado a las inscripciones monumentales, en el uso diario fue sustituido por la escritura hierática y demótica, formas más simples que permitian mayor rapidez y comodidad a los escribas

En estos sistemas de escritura los grupos de signos

adquirieron una forma propia, y asi se introdujeron símbolos particulares para 20, 30....90....200, 300.....900, 2000, 3000..... con lo que disminuye el número de signos necesarios para escribir una cifra.

El sistema de numeración griego

El primer sitema de numeración griego se desarrolló hacia el 600 a.n.e. Era un sistema de base decimal que usaba unos símbolos para representar esas cantidades. Se utilizaban tantas de ellas como fuera necesario según el principio de las numeraciones aditivas. Para representar la unidad y los números hasta el cuatro se usaban trazos verticales. Para el cinco, 10 y 100 las letras correspondientes a la inicial de la palabra cinco (pente), diez (deka) y mil (khiloi). Por este motivo se llama a este sistema acrofónico.

Los símbolos de 50, 500 y 5000 se obtienen añadiendo el signo de 10, 100 y 1000 al de cinco, usando un principio multiplicativo. Progresivamente este sistema ático fue reemplazado por el jónico, que empleaba las 24 letras del alfabeto griego junto con algunos otros símbolos. De esta forma los números parecen palabras, ya que están compuestos por letras, y a su vez a

Es un sistema decimal estricto que usa las unidades y los distintas potencias de 10. Utiliza ideogramas y usa la combinación de los números hasta el diez con la decena, centena, millar y decena de millar para según el principio multiplicativo representar 50, 700 ó 3000. El orden de escritura se hace fundamental,ya que 5 10 7 igual podría representar 57 que 75.

Tradicionalmente se ha escrito de arriba abajo aunque también se hace de izquierda a derecha. No es necesario un símbolo para el cero siempre y cuando se pongan todos los ideogramas, pero aún así a veces se suprimían los correspondientes a las potencias de 10.

Aparte de esta forma que podríamos llamar canónica se usaron otras. Para los documento importantes se usaba una grafía más complicada con objeto de evitar falsificaciones y errores. En los sellos se escribía de forma más estilizada y lineal y aún se usaban hasta dos grafías diferentes en usos domésticos y comerciales, aparte de las variantes regionales. Los eruditos chinos por su parte desarrollaron un sistema posicional muy parecido al actual que desde que incorporó el cero por influencia india en siglo VIII en nada se diferencia de este.

Sistemas de numeración posicionales

Mucho más efectivos que los sitemas anteriores son los posicionales. En ellos la posición de una cifra nos dice si son decenas, centenas ... o en general la potencia de la base correspondiente. Sólo tres culturas además de la india lograron desarrollar un sistema de este tipo. Babilonios, chinos y mayas en distintas épocas llegaron al mismo principio. La ausencia del cero impidió a los chinos un desarrollo completo hasta la intraducción del mismo. Los sistemas babilónico y maya no eran prácticos para operar porque no disponían de simbolos particulares para los dígitos, usando para representarlos una acumulación del signo de la unidad y la decena. El hecho que sus bases fuese 60 y 20 respectivamente no hubiese representado en principio nigún obstáculo. Los mayas por su parte cometían una irregularidad a partir de las unidades de tercer orden, ya que detrás de las veintenas no usaban 20x20=400 sino 20x18=360 para adecuar los números al calendario, una de sus mayores preocupaciones culturales.

Fueron los hindúes antes del siglo VII los que idearon el sistema tal y como hoy lo conocemos, sin mas que un cambio en la forma en la que escribimos los nueve dígitos y el cero. Aunque con frecuencia nos referimos a nuestro sistema de numeración cómo árabe, las pruebas arqueológicas y documentales demuestran el uso del cero tanto en posiciones intermedias como finales se originó en India. Los árabes transmitieron esta forma de representar los números y sobre todo el cálculo asociado a ellas, aunque tardaron siglos en ser usadas y aceptadas. Una vez más se produjo una gran resistencia a algo por el mero hecho de ser nuevo o ajeno, aunque sus ventajas eran evidentes. Sin esta forma eficaz de numerar y efectuar cálculos dificilmente la ciencia hubiese podido avanzar.

El sistema de numeración Babilónico

Entre la muchas civilizaciones que florecieron en la antigua Mesopotamia se desarrollaron distintos sistemas de numeración. Inventaron un sistema de base 10, aditivo hasta el 60 y posicional para números superiores. Para la unidad se usaba la marca vertical que se hacía con el punzón en forma de cuña. Se ponían tantos como fuera preciso hasta llegar a 10, que tenía su propio signo. De este se usaban los que fuera necesario completando con las unidades hasta llegar a 60. A partir de ahí se usaba un sistema posicional en el que los grupos de signos iban representando sucesivamente el número de unidades, 60, 60x60, 60x60x60 y asi sucesivamente como en los ejemplos que se acompañan.

El sistema de numeración Maya

Los mayas idearon un sistema de base 20 con el 5 cómo base auxiliar. La unidad se representaba por un punto. Dos, tres, y cuatro puntos servían para 2, 3 y 4. El 5 era una raya horizontal, a la que seañadían los puntos necesarios para representar 6, 7, 8 y 9. Para el 10 se usaban dos rayas, y de la misma forma se continúa hasta el 20, con cuatro rayas. Hasta aquí parece ser un sistema de base 5 aditivo, pero en realidad, considerados cada uno un solo signo, estos símbolos constituyen las cífras de un sistema de base 20, en el que hay que multiplicar el valor de cada cifra por 1, 20, 20x20, 20x20x20 ... según el lugar que ocupe, y sumar el resultado. Es por tanto un sistema posicional que se escribe a arriba abajo, empezando por el orden de magnitud mayor.

Al tener cada cifra un valor relativo según el lugar que ocupa, la presencia de un signo para el cero, con el que indicar la ausencia de unidades de algún orden, se hace imprescindible y los mayas lo usaron, aunque no parece haberles interesado el concepto de cantidad nula. Cómo los babilonios lo usaron simplemente para indicar la ausencia de otro número.

Pero los científicos mayas eran a la vez sacerdotes ocupados en la observación astronómica y para expresar los número correspondientes a las fechas usaron unas unidades de tercer orden irregulares para la base 20. Así la cifra que ocupaba el tercer lugar desde abajo se multiplicaba por 20x18=360 para completar una cifra muy próxima a la duración de un año.

El año lo consideraban dividido en 18 uinal que constaba cada uno de 20 días. Se añadían algunos festivos (uayeb) y de esta forma se conseguía que durara justo lo que una de las unidades de tercer orden del sistema numérico. Además de éste calendario solar, usaron otro de carater religioso en el que el año se divide en 20 ciclos de 13 días.

Al romperse la unidad del sistema éste se hace poco práctico para el cálculo y aunque los conocimiento astronómicos y de otro tipo fueron notables los mayas no desarrollaron una matemática más allá del calendario

5.3.1 Representación posicional de los números

Se define un sistema de numeración: como el conjunto de símbolos y reglas que se utilizan para la representación de cantidades. En ellos existe un elemento característico que define el sistema y se denomina base, siendo ésta el número de símbolos que se utilizan para la representación.

Un sistema de numeración en base "b" utiliza para representar los números un alfabeto compuesto por b símbolos o cifras. Así todo número se expresa por un conjunto de cifras, teniendo cada una de ellas dentro del número un valor que depende:

- De la cifra en sí
- De la posición que ocupe dentro del número

En el sistema de numeración decimal (base 10), que habitualmente se utiliza, b = 10 y el alfabeto por tanto, está constituido por 10 símbolos: {0, 1, 2..., 9}

Por ejemplo, el número 3278.52 puede obtenerse como suma de:

3000
200
70
8
0.5
0.02

3278.52

por tanto se verifica que:

$3278.52 = 3 * 10^3 + 2 * 10^2 + 7 * 10^1 + 8 * 10^0 + 5 * 10^{-1} + 2 * 10^{-2}$

Cada posición, por tanto, tiene un peso:

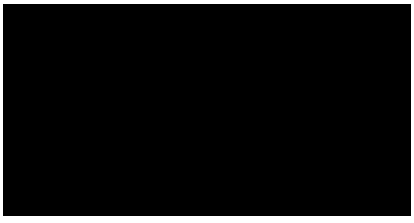
Posición 0 Peso b^0
Posición 1 Peso b^1
Posición 2 Peso b^2
Posición 3 Peso b^3
....
Posición -1 Peso b^{-1}
Posición -2 Peso b^{-2}
.....

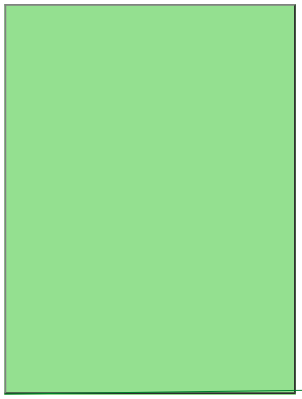
Generalizando se tiene que la representación de un número en una base b :

$$N = ...n_4\ n_3\ n_2\ n_1\ n_0\ n_{-1}\ n_{-2}\ ...$$

es una forma abreviada de expresar su valor, que es:

$$N = n_4\ b^4 + n_3\ b^3 + ... + n_{-1}\ b^{-1} + n_{-2}\ b^{-2}$$





$10100.001_{)2} = 1*2^4 + 1*2^2 + 1*2^{-3} = 20.125_{)10}$

Para transformar un número decimal a binario:

a) La parte entera del nuevo número (binario) se obtiene efectuando divisiones enteras (sin obtener decimales) por dos, de la parte entera del número decimal de partida y de los cocientes que sucesivamente se vayan obteniendo. Los restos de estas divisiones y el último cociente (que serán siempre ceros y unos) son las cifras binarias. El último cociente será el bit más significativo y el primer resto el bit menos significativo (más a la derecha).

Ejemplo:

$26_{)10}$ es en binario:

```
26 | 2
 0 13 | 2
      1  6 | 2
           0  3 | 2
                1  1
```

$26_{)10} = 11010_{)2}$

b) La parte fraccionaria del número binario se obtiene multiplicando por 2 sucesivamente la parte fraccionaria del número decimal de partida y las partes fraccionarias que se van obteniendo en los productos sucesivos. El número binario se forma con las partes enteras (que serán ceros y unos) de los productos obtenidos.

Ejemplo:

Transformar a binario natural el número decimal 0.1875

```
0.1875      0.3750      0.7500      0.5000
   * 2         * 2         *2          * 2
---------   ---------   ---------   ---------
0.3750      0.7500      1.5000      1.0000
```

$0.1875_{)10} = 0.0011_{)2}$

Ejemplo:

Transformar a binario el número decimal 74.423

a) Parte entera:

```
74  | 2
 0  37 | 2
      1  18 | 2
           0  9 | 2
                1  4  | 2
                   0   2  | 2
                       0   1
```

b) Parte fraccionaria:

0.423	0.846	0.692	0.384	0.768
*2	* 2	* 2	* 2	* 2
0.846	1.692	1.384	0.768	1.536

Es decir:

$74.423_{)10} = 1001010.01101..._{)2}$

Ejemplo, programa en C, para convertir de base 10 a base 2

1) Se divide el numero entre la base y se va guardando el residuo, el resultado de la división se vuelve a dividir entre la base y se guarda el residuo; esto se efectúa tantas veces hasta que el resultado de la división sea cero.

Ejemplo: convertir 14 en base 10 a base 2

14/ 2= 7 sobran 0
7 / 2= 3 sobran 1
3 / 2= 1 sobran 1
1 / 2 =0 sobran 1

Ahora lo vamos a poner en una tabla con subindices

```
c0 b c0 r0
 14 2 7 0

c0 b c1 r1
  7 2 3 1

c1 b c2 r2
  3 2 1 1

c2 b c3 r3
  1 2 0 1
```

¿Qué observamos? Es posible utilizar un arreglo de vectores de:

c0,c1,c2,c3 r0,r1,r2,r3 b se mantiene constante. En la primera columna se observa que en el tercer renglón se repite un valor de c0 porque es el resultado de la división, que luego se debe utilizar, nuevamente para hacer la siguiente división.

¿Hasta donde dejar de hacer divisiones? Hasta que c[i] se haga cero, en este caso c3=0. Y por lo tanto ya terminamos.

¿Ahora como hacemos para que vayan cambiando los subíndices?
primera iteración c0 b c0 r0 Haríamos un "*for* del índice de *0 a 3*"

segunda iteración c0 b c1 r1 Sin embargo los que estan obscuros perjudican

tercera iteración c1 b c2 r2 nuestro for, lo que podemos mejorarlo de este

tercera iteración c2 b c3 r3 modo: c[0]= c[1]- 1. Y con esto quedaria asi:

co b co ro

c1-1 b c1 r1

c2-1 b c2 r2

c3-1 b c3 r3 si hago el numero= c0;

Que así con un solo for (i=0;i<=3;i++){

c[i]=c[i-1]/b;

r[i]=fmod(c[i-1],b)

El fmod saca el residuo de la division(a,b) a/b= residuo.

.requiere #include <math.h>

}

¿Cómo lo mandamos a imprimir el resultado?: r4r3r2r1= 1110 base2 = 14 base10

hacemos otro for para el subindice de r[i] desde i=4 hasta i=0

for (i=4; i<=0;i--){

printf("%d", r[i]);

}

```
#include <stdio.h>

#include <conio.h>
#include <math.h>
int c[50], r[50]; /* es mejor poner más de lo necesario */

int i,b;
main()
{


clrscr();

/* Datos */
b=2;
c[0]= 14;
i=1;

/* conversion */
for(;;)

{ c[i]= c[i-1]/b;
r[i]= fmod(c[i-1],b);
if (c[i]==0) break;
i=i+1;

}
for(;;) /* impresión de resultado de atrás para adelante */

{
if (i==0) break;
```

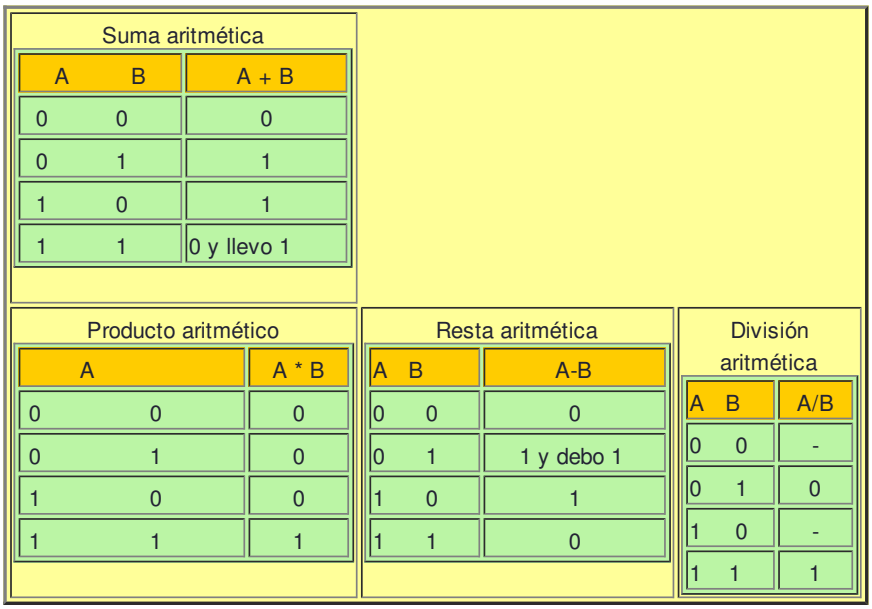
```
printf(" %d",r[i]);
i=i-1;
}

}
```

C) Operaciones aritméticas y lógicas con variables binarias

Una variable binaria puede representar, una cifra de un número en el sistema de numeración en base dos. Las operaciones aritméticas básicas con variables binarias naturales son la suma, resta, multiplicación y división. Estas operaciones son análogas a las realizadas en decimal pero usando ceros y unos.

Tabla de operaciones aritméticas:



Suma aritmética

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	0 y llevo 1

Producto aritmético

A	B	A * B
0	0	0
0	1	0
1	0	0
1	1	1

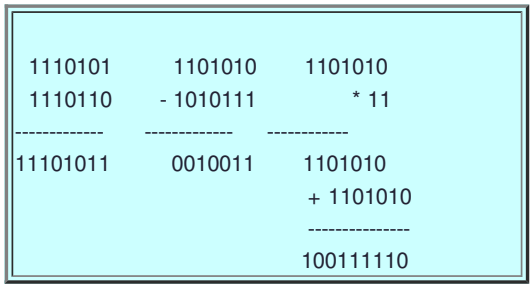
Resta aritmética

A	B	A-B
0	0	0
0	1	1 y debo 1
1	0	1
1	1	0

División aritmética

A	B	A/B
0	0	-
0	1	0
1	0	-
1	1	1

Ejemplo:



```
 1110101       1101010      1101010
 1110110     - 1010111            * 11
-------------  -------------  ------------
11101011       0010011      1101010
                             + 1101010
                             ----------
                            100111110
```

Las operaciones lógicas o booleanas con variables binarias son la suma lógica (+), llamada también función OR, el producto lógico (llamado también AND y la complementación (-) o negación o NOT.

Un tutorial sobre lógica se puede ver en el siguiente vídeo:

Las tablas son las siguientes:

Suma lógica (OR)

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Producto lógico (AND)

A	B	A * B
0	0	0
0	1	0
1	0	0
1	1	1

Complementación (NOT)

A	-A
0	1
1	0

Es frecuente también la utilización de las operaciones combinadas como NAND (AND y NOT) y NOR (OR y NOT).
Las tablas son las siguientes:

NAND

A B	A * B	- (A*B)
0 0	0	1
0 1	0	1
1 0	0	1
1 1	1	0

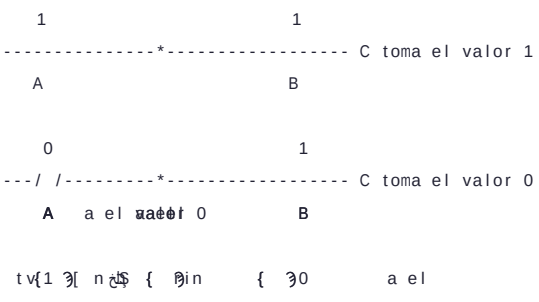
NOR

A	B	A + B	- (A + B)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Veamos ahora por qué resulta más sencillo operar con el sistema binario que con el decimal.

Se ha visto la simplicidad de las operaciones aritméticas en binario. Los ordenadores funcionan con circuitos que pueden representar los dos estados del sistema binario (0 y 1) abriéndose o cerrándose el circuito.

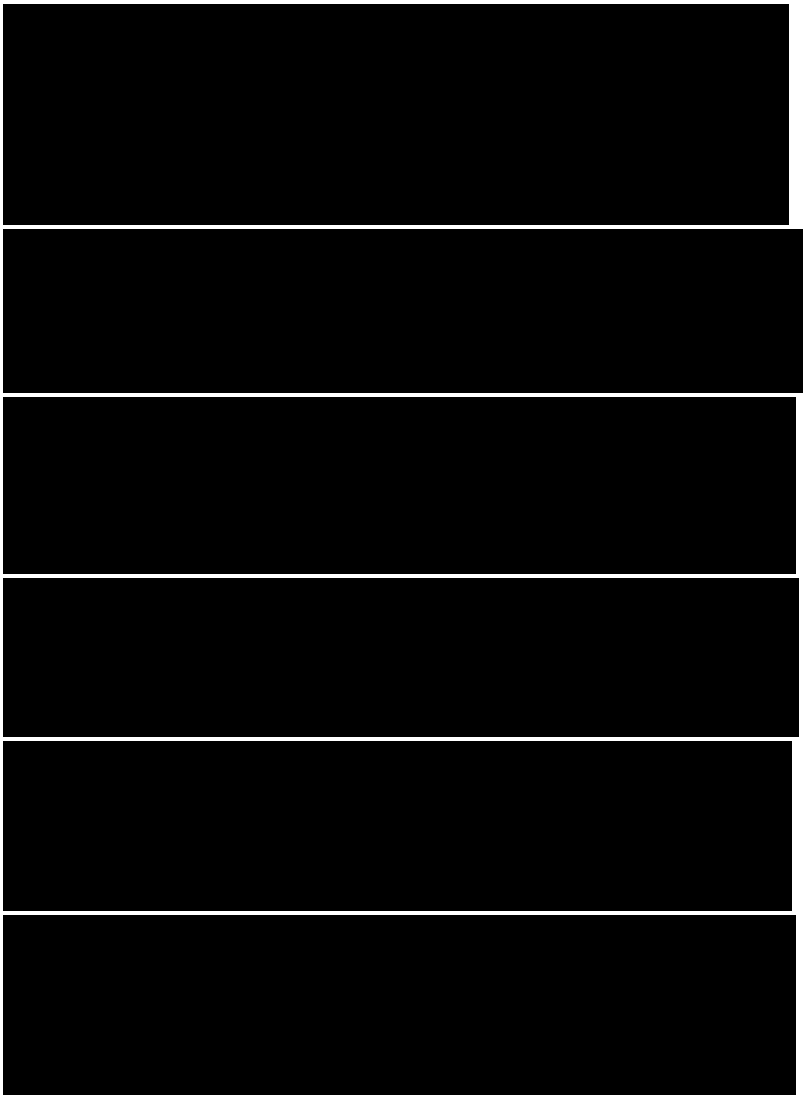
Al conectar dos circuitos según el esquema, y hacer pasar una corriente eléctrica por A se puede conseguir que el estado del conductor C represente el resultado de la operación AND lógica.



```
                    |------------------- C=0
----/     /----------
B = 0
```

Estos ejemplos ilustran cómo basándose en circuitos e impulsos eléctricos un ordenador puede realizar operaciones aritméticas y lógicas. El motivo por el que los ordenadores electrónicos trabajan en el sistema binario se puede deducir de los ejemplos vistos. Queda imaginar la complejidad que supondría trabajar en el sistema decimal (con circuitos 10 veces más complejos para representar operaciones de 2 dígitos).

Una puerta es un circuíto electrónico que produce una señal de salida que es una operación booleana sencilla de las señales de entrada. Las puertas básicas usadas en lógica digital son AND, OR, NOT, XOR, NAND y NOR. cada puerta se define de tres formas: símbolo gráfico, notación algebraica y tabla de verdad, como se muestra seguidamente:



D) Representación en complementos

Para representar un número negativo se puede utilizar el complemento de ese número a la base. De esta forma las sumas y restas quedan reducidas a sumas. Este sistema de representación es de sumo interés en el caso de los ordenadores ya que al usarlo se reduce la complejidad de los circuitos.

El complemento a la base de un número, es el número que resulta de restar a cada una de las cifras del número N a la base menos uno del sistema que se esté utilizando y posteriormente sumar uno a la diferencia obtenida.

Ejemplo:

- En base 10:
- Base menos uno del sistema: 9
- Representar el número 63 en complemento a la base.

```
N = 63     99         36
            -63        + 1
           ------     ------
            36         37
```

Es decir, el complemento a 10 (base)del número 63 es 37.

En base 2:

Base menos uno: 1

Complemento a 2 del número 10010 es 01110

```
    11111            01101
  - 10010             +    1
-------------      -------------
    01101            01110
```

Complemento a 2 del número 101010 es 010110

```
111111           010101
101010             +     1
-----------       ------------
010101           010110
```

Observamos que para transformar un número binario N a complemento a 2 basta con cambiar los 0 por 1 y los 1 por 0 de N y sumar 1 al resultado.

Veremos ahora que la utilidad de esta representación es para convertir la realización de las restas a sumas, lo cual simplifica el diseño del procesador.

Ejemplo: Base 10

Supongamos que se ha de realizar la siguiente operación: 77 - 63

Se puede hacer de dos formas diferentes:

a) Directamente: 77 - 63 = 14
b) Utilizando el complemento a 10 del substraendo:

- Complemento a 10 del substraendo 63 es 37

```
 99  36
-63  +1
---- -----
36  37

  77
+37
-----
114
- (No se considera)

El resultado es 14
```

Es decir, para restar basta con sumar el minuendo con el complemento a la base del substraendo y sin considerar el acarreo.

Ejemplo: Base 2

- Supongamos se ha de efectuar la siguiente resta:

11001 - 10010

Se puede hacer de dos formas:

a) Directamente:

```
 11001
-10010
--------
 00111
```

b) Usando el complemento a 2 del substraendo:

El substraendo es 10010. Su complemento a 2 se obtiene cambiando 0 por 1 y 1 por 0.
01101
y sumándole 1

```
 01101
+     1
--------
 01110
```

- Ahora sumamos al minuendo el complemento a 2 del substraendo :

```
 11001
+01110
---------
 100111
- (No se considera)
```

E) Códigos intermedios

Los códigos intermedios se basan en la facilidad de transformar un número en base 2 a otra base que sea potencia de 2 y viceversa. Usualmente se usan como códigos intermedios los sistemas de numeración en base 8 y en base 16 (conocidos como octal y hexadecimal).

- a) OCTAL.

En la base octal, b = 8 y el conjunto de símbolos utilizados es:
{0, 1,..., 7}

Para convertir un número octal a binario sólo debemos sustituir cada dígito octal por su equivalente binario.

Equivalencias

OCTAL	BINARIO
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Ejemplo:

6 se sustituye por 110

2 se sustituye por 010

$537.24_{)8} = 101\ 011\ 111\ .\ 010\ 100_{)2}$

que equivale según la tabla, a: 5 3 7 . 2 4

La conversión de binario a octal se realiza juntando en grupos de tres dígitos binarios, comenzando por la izquierda desde el punto decimal y sustituyendo cada grupo por el correspondiente dígito octal.

Ejemplo:

El número binario $10001101100.11010_{)2}$ es en octal

$10\ 001\ 101\ 100\ .\ 110\ 10 = 2154.64_{)8}$

Para pasar un número de octal a decimal aplicamos la expresión:

$$N)_8 = \ldots n_4\, b^4 + n_3\, b^3 + \ldots + n_{-1}\, b^{-1} + n_{-2}\, b^{-2} \ldots)_{10}$$

con b = 8.

Ejemplo:

Para pasar el número octal $1367.25_{)8}$ a decimal:

$1367.25 = 1*8^3 + 3*8^2 + 6*8^1 + 7*8^0 + 2*8^{-1} + 5*8^{-2} = 759.328125_{)10}$

Para pasar un número entero decimal a octal se hacen sucesivas divisiones enteras del número y los subsiguientes cocientes por 8 (al igual que en binario). Para transformar la parte fraccionaria de un número decimal a octal se hacen sucesivas multiplicaciones por 8 (de la misma forma que en binario).

Ejemplo:

Para pasar el número decimal $760.33_{)10}$ a octal:

760 | 8
40
0 95 | 8
15
7 11 | 8

```
              3  1

0.33     0.64    0.12    0.96
  *8      * 8     * 8     * 8
------  -------  -------  -------
2.64     5.12    0.96    7.68

El número en octal es 1370.2507
```

b) HEXADECIMAL

Para representar un número en base hexadecimal (b = 16) es necesario disponer de un conjunto o alfabeto de 16 símbolos. Se suele usar el conjunto:

{0, 1, 2,..., 9, A, B, C, D, E, F}

Podemos hacer las conversiones de binario a hexadecimal y viceversa en forma análoga al sistema octal. Ahora bien, aquí utilizaremos grupos de 4 bit en lugar de grupos de 3 bit.

Equivalencias

HEX	BINARIO	DECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0110	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

Ejemplo:

Pasar el número binario $010010111011111.1011101_{)2}$ a hexadecimal.

$010\ 0101\ 1101\ 1111\ .\ 1011\ 101_{)2} = 25DF.BA_{)16}$

Ejemplo:

Pasar el número $1ABC701.C4_{)16}$ a binario:

$0001\ 1010\ 1011\ 1100\ 0111\ 0000\ 0001\ .\ 1100\ 0100_{)2}$

1 A B C 7 0 1 C 4

El número binario resultante es:

$0001101010111100011100000001.11000100_{)2}$

Para transformar un número hexadecimal a decimal aplicamos la

expresión siguiente con b=16.

$$N_{)16} = ...n_4\, b^4 + n_3\, b^3 + ... + n_{-1}\, b^{-1} + n_{-2}\, b^{-2} ...)_{10}$$

Ejemplo:

Pasar el número hexadecimal A798C.1E)$_{16}$ a decimal.

$10*16^4 + 7*16^3 + 9*16^2 + 8*16^1 + 12*16^0 + 1*16^{-1} + 14*16^{-2} = = 686476.1171_{)10}$

Para pasar un número de decimal a hexadecimal se hace de forma análoga a los casos binario y octal: la parte entera se divide por 16, así como los cocientes enteros sucesivos, y la parte fraccionaria se multiplica por 16, así como las partes fraccionarias de los productos sucesivos.

Ejemplo:

El número 4573.79)$_{10}$ se corresponde en hexadecimal:

```
4573 |16
137
 093  285 |16
   13  125
         13  17  |16
                1  1

0.79     0.64      0.24
* 16     * 16     * 16
------   ----- -   -----
474      384       144
+79      +64       +24
-----    ----      -----
12.64 10.24     3.84
  C     A      3
```

El número en hexadecimal es 11DD.CA3)$_{16}$

5.4 Representación interna de la información.

En la memoria y el procesador central la información se transmite y procesa en unidades denominadas palabras. La organización de las palabras depende del ordenador, siendo usuales las longitudes: 8, 16, 32, 36, 60 y 64 bit, aunque hay hasta de 512 bit.

La memoria principal se encuentra organizada en palabras, cada una de las cuales tiene asignada una dirección. Los intercambios de información entre el procesador y la memoria se hacen en unidades denominadas palabras y no en caracteres (octetos) o en bit.

Normalmente para aprovechar la memoria, la longitud de la palabra debe ser un múltiplo entero del número de bit usados para representar un carácter.

Los datos se introducen inicialmente en el ordenador según un código de entrada/salida (que ya hemos visto), tanto si éstos son de tipo alfabético como de tipo numérico.

Los datos de tipo numérico se utilizan normalmente para operar aritméticamente con ellos, y la representación simbólica obtenida con el código de E/S no resulta adecuada para realizar este tipo de operaciones. Resulta más adecuado operar en un sistema de numeración que en un código de E/S.

Por los motivos anteriores, y teniendo en cuenta que la ALU opera con palabras, se realiza una conversión de notaciones pasando de la representación simbólica de E/S a otra notación que denominamos representación interna.

TIPOS DE INFORMACION.

En un sistema de procesamiento de la información es necesaria la codificación de tres clases de información:

- 1 Información numérica:

  Enteros
  Reales
  Complejos
  Lógicos

- 2 Información no numérica (o alfanumérica):

  Caracteres
- 3 Instrucciones del programa

A) Datos de tipo complejo

Los datos de tipo complejo se representan por parejas de números reales almacenados en posiciones consecutivas de memoria. Es decir, pueden considerarse como un caso particular de números reales.

B) Datos de tipo lógico

Representan un valor del Algebra de Boole binaria, es decir, 0 (falso) ó 1 (verdad).

C) Representación en punto fijo

El nombre de esta representación surge al considerar el punto fraccional, situado en una posición fija. El punto fijo es utilizado para la representación de números enteros, suponiéndose el punto fraccional ubicado a la derecha de los bit. Cualquiera de los sistemas de representación de enteros es una representación de punto fijo. Tambien, se puede utilizar la representación en punto fijo para representar fracciones binarias escalando los números, de modo que el punto fraccional quede ubicado implícitamente en otra posición entre los bit, y en el caso límite a la izquierda de todos ellos describiendo un número fraccional binario puro (menor a 1).

- a) El signo se representa en el bit situado más a la izquierda de la palabra. Este bit es 0 si el número es positivo ó 1 si el número es negativo.

- b) El valor absoluto:
  - b1) Números positivos: Se almacenan directamente el número en binario natural.
  - b2) Números negativos: Dependiendo del ordenador se almacena el complemento a 2 del número binario natural o la magnitud del número en binario natural.

Ejemplo de representación interna de datos de tipo entero en un ordenador

de palabras de 4 bit:

DECIMAL	SIGNO Y MAGNITUD	COMPLEMENTO A 2
7	0111	0111
6	0110	0110
5	0101	0101
4	0100	0100
3	0011	0011
2	0010	0010
1	0001	0001
+0	0000	0000
-0	1000	----
-1	1001	1111
-2	1010	1110
-3	1011	1101
-4	1100	1100
-5	1101	1011
-6	1110	1010
-7	1111	1001
-8	-----	1000

D) Representación en punto flotante

El punto flotante surge de la necesidad de representar números reales y enteros con un rango de representación mayor que el que ofrece la representación en punto fijo y posibilitar al ordenador el tratamiento de números muy grandes y muy pequeños. Estas ventajas que ofrece el punto flotante traen como contraprestación una disminución en la precisión de los números representados.

En su representación se utiliza la notación científica o exponencial matemática en la que una cantidad se representa de la siguiente forma:

n° = mantisa * base de exponenciación $^{\text{exponente}}$

Un número en esta notación tiene infinitas representaciones, de las que se toma como estándar la denominada normalizada, que consiste en que la mantisa no tiene parte entera y el primer dígito o cifra a la derecha del punto decimal es significativo (distinto de 0), salvo en la representación del número 0.

Ejemplo (^ significa elevado a):

835.4 = 8354*10^-1 = 835.4 * 10^0 = 83.54 * 10^1 = 8.354 * 10^2 = .8354 * 10^3

Representación del número decimal 835.4 con base de exponenciación 10. siendo está última expresión la que corresponde al número normalizado.

En este sistema de codificación, se dividen los bit disponibles en la palabra o doble palabra del ordenador entre la mantisa y el exponente, teniendo una base de exponenciación determinada (2 o potencia de 2). Normalmente la definición de la coma flotante sigue las siguientes reglas:

- El exponente se representa en uno de los siguientes sistemas de codificación: módulo y signo o exceso a 2^{n-1}, siendo siempre un

número entero. En este sistema de codificación el exponente también recibe el nombre de característica.
- La mantisa es un número real con el punto decimal implícito a la izquierda de sus bit, representada normalmente en uno de los siguientes sistemas de codificación: módulo y signo, complemento a 1 o complemento a 2.
- La base de exponenciación es una potencia de 2 determinada por el fabricante del equipo (2, 8 o 16).

Existen muchas formas de representación en punto flotante, variando la longitud de la palabra del ordenador, la base de la exponenciación, el número de bit reservados para la mantisa y para el exponente, el sistema utilizado para representar la mantisa y el exponente, etc.. El punto flotante se define particularmente en cada caso. Las definiciones más comunes son las siguientes:

a) para simple precisión (32 bit)

signo exponente mantisa
31 30 23 22 0

b) para doble precisión (64 bit)

signo exponente mantisa
63 62 52 51 0

El rango de representación en la coma flotante debe ser analizado teniendo en cuenta los máximos y mínimos valores representables tanto con signo positivo como negativo:

mínimo número negativo = -(mantisa máxima) * $base^{\text{máximo exponente}}$

máximo número negativo = -(mantisa mínima) * $base^{\text{-maximo exponente}}$

mínimo número positivo = mantisa mínima * $base^{\text{-máximo exponente}}$

máximo número positivo = mantisa máxima * $base^{\text{máximo exponente}}$

Conviene observar que existen cuatro zonas de números que no pueden ser representados mediante un determinado formato de coma flotante. Estas zonas están ubicadas cercanas al 0, tanto para valores positivos como negativos (subdesbordamiento positivo o negativo), como para valores grandes (positivos) o pequeños(negativos) que exceden el rango de representación.

Ejemplo

Un ordenador utiliza el siguiente formato para registrar números en punto flotante:

- los bit del 23 al 30 se utilizan para representar el exponente en exceso a 128 (2^7)
- los bit del 0 al 22 se utilizan para representar la mantisa normalizada para el sistema Módulo y signo
- el bit 31 se utiliza para representar el signo de la mantisa (0 para el +)
- la base de exponenciación es 2
- el 0 se representa con todos los bit en 0.

Representar en este formato el número 12:

- 12 en notación normalizada de base 2 es $0.75 * 2^4$
- el exponente de valor 4 en exceso a 128 es: 10000100
- la mantisa 0.75 en binario es 0.11

de donde la representación del número 12 quedará como:

0 10000100 11000000000000000000000
signo (+) exponente 4 mantisa 0.75

Representar en el formato definido el 12. En este caso la notación normalizada solo sufre cambio en el signo de la mantisa ($-0.75 * 2^4$), la expresión quedará ntonces:

1 10000100 11000000000000000000000
signo (-) exponente 4 mantisa 0.75

El rango de representación de este formato en coma flotante será:

mínimo negativo = $-(1 - 2^{23})*2^{127} = -2^{127} = 1.701411834605 * 10^{38}$

máximo negativo = $-0.5 * 2^{-128} = -2^{-129} = -1.469367938528 * 10^{-39}$

mínimo positivo = $0.5 * 2^{-128} = 2^{-129} = 1.469367938528 * 10^{-39}$

máximo positivo = $(1 - 2^{23})*2^{127} = 2^{127} = 1.701411834605 * 10^{38}$

E) Datos de tipo carácter

Los datos de tipo carácter, representan sencillamente cadenas de caracteres representados según el código de E/S.

A las representaciones de los caracteres se les imponen las siguientes condiciones:

- Deben englobar las 26 letras del alfabeto latino, los 10 dígitos y un cierto número de caracteres gráficos (operadores) y signos de puntuación.
- Deben permitir añadir nuevos caracteres específicos.
- Deben incluir un sistema de redundancia que permita la detección de errores en el curso de la transmisión.
- Los subconjuntos de letras y números deben estar ordenados y ser coherentes. Estarán en dos grupos diferentes y ordenados.

F) Codificación de instrucciones

Las instrucciones llevan cierto número de informaciones:

- Código de operación.
- Dirección de operandos/resultados.
- Condiciones de direccionamiento, etc.

A cada una de estas informaciones se le asocia una zona formada por un número de bit suficientes para codificar los diferentes estados posibles de la instrucción.

Así una zona de código de operación de 6 bit permite codificar $2^6 = 64$ operaciones diferentes, y si una zona de direcciones es de 16 bit permitirá direccionar una memoria de 2^{16} direcciones.

5.5 Detección de errores en la [redacted]

información codificada

Hemos visto anteriormente que si representamos cada carácter por un número fijo de bit, para representar m símbolos distintos necesitamos al menos n bit, siendo n el menor número entero que verifica la relación

$$n = \log_2 m = 3.32 * \log m$$

También hemos observado que a veces no es necesario utilizar todas las combinaciones posibles de los n bit. Cuantas menos combinaciones se desperdicien decimos que el código es más eficiente.

Un código que es poco eficiente se dice que es redundante. La eficiencia de un código se expresa como el cociente entre el número de símbolos que representa y el número total posible,

$$\text{Eficiencia} = m/m' = m/2^n$$

así se tiene para el ASCII una eficiencia de $95/2^7$, que es 0.742, con R=25.8% y para el ASCII extendido la eficiencia es $95/2^8$, que es 0.371, con R=62.9%, donde R es la redundancia, que se calcula, R=(1-þ)*100%

A veces las redundancias se introducen deliberadamente para poder detectar posibles errores en la transmisión o grabación de información.

Así por ejemplo, si necesitamos transmitir 8 símbolos (A, B, C,...,H) y si se hace con un código sin redundancias, necesitamos n = 3 bit, y un código posible puede ser:

ALFABETO	CÓDIGO
A	000
B	001
C	010
D	011
E	100
F	101
G	110
H	111

En el caso de que por algún error uno de los bit varíe obtenemos otro símbolo del alfabeto, que considerado aisladamente no puede ser detectado como erróneo. Si se usase un código redundante, tal como el siguiente:

ALFABETO	CÓDIGO
A	0000
B	0001
C	0010
D	0011
E	0100
F	0101
G	0110
H	0111

existirían algunas posibilidades de detectar errores. Así por ejemplo, si se

transmite el símbolo H, esto es 0111, y por un error la transmisión cambiara el primer bit, esto es se recibiese 1111, podría detectarse el error ya que 1111 no corresponde a ninguno de los símbolos posibles.

Usualmente las redundancias se introducen deliberadamente y de acuerdo con algún algoritmo predeterminado. Uno de estos algoritmos añade al código inicial da cada carácter un nuevo bit denominado bit de paridad. Existen dos criterios para introducir este bit:

- Bit de paridad, criterio par: se añade un bit (0 ó 1) de forma tal que el número total de unos del código que resulte sea par.
- Bit de paridad, criterio impar: Se añade un bit (0 ó 1) de forma tal que el número total de unos del código que resulte sea impar.

Ejemplo:

Código inicial	Código con bit de paridad par
100 0001	0100 0001
101 1011	1101 1011
101 0000	0101 0000
110 1000	1110 1000

- Código con bit de paridad impar

100 0001	1100 0001
110 0101	1110 0101
010 0000	0010 0000
000 0000	1000 0000

El bit de paridad se introduce antes de transmitir o grabar la información. Por ruido o interferencias en la transmisión o defecto del soporte de la información puede eventualmente cambiar un bit (de 1 a 0 ó de 0 a 1). Si en el receptor o al leer la información se comprueba la paridad, se detectaría el error, ya que el número de unos dejaría de ser par (en el criterio par) o impar (en el criterio impar).

En el caso de transmisión de datos, automáticamente se podría provocar una nueva transmisión del carácter erróneo. Obviamente si se produjese el cambio simultáneo de dos bit distintos no se detectaría el error de paridad, ahora bien, esta eventualidad es mucho menos probable que la de que cambie un sólo bit.

Códigos de barras

Otro caso habitual donde aparece información redundante con la finalidad de verificar errores, es en los códigos de baras, habituales en cualquier producto que se comercialice masivamente. En 1974 los 12 países que entonces formaban la Unión Europea decidieron adoptar un sistema de codificación para los productos, similar al sistema UPC de Estados Unidos de Norteamérica. Así surgió el código EAN (European Article Numbering), sistema que han adoptado más de 100 países y cerca de un millón de empresas.

El más usual es EAN 13, formado por 13 dígitos agrupados en cuatro partes: prefijo, código empresa, código producto y dígito de control. El prefijo asignado por EAN internacional a AECOC es el 84, de modo que la

mayoría de las empresas que forman parte del sistema EAN a través de AECOC utilizan este número. El código de empresa (fabricante o cadena de distribución) está formado por un número de entre 5 y 8 dígitos. El código del producto completa los primeros 12 dígitos y el último dígito es de control. Por ejemplo, un tarro con garbanzos cocidos marca Eroski, lleva el siguiente código 8480010021967, donde 84 significa España, 80010 es el número que tiene asignado la cooperativa Eroski, 02196 corresponde a ese tipo de garbanzos, siendo una clasificación interna del distribuidor y 7 es el código de control.

En el momento de la venta, el terminal punto de venta (TPV), realiza las siguientes operaciones:

1. Suma los dígitos de las posiciones pares: 4+0+1+0+1+6 = 12
2. Multiplica el resultado por 3: 12x3= 36
3. Le añade los dígitos de las posiciones impares: 36+8+8+2+9=63
4. Resta la suma obtenida del siguiente múltiplo de 10: 70-63=7

Si el resultado coincide con el dígito de control, como es el caso, el ordenador enviará el precio al TPV.

Rafael Menéndez-Barzanallana Asensio
Departamento Informática y Sistemas. Universidad de Murcia

Actualizado 2009/03/30