UD01. REPRESENTACIÓN DE INFORMACIÓN

1. INTRODUCCIÓN

1.1 Pieza de información e información

Las computadoras (o más correctamente sistemas de información) son máquinas diseñadas para

procesar información o, en otras palabras, obtener resultados de la aplicación de

operaciones en un conjunto de datos. Pero, ¿qué es información? Que es un pedazo de

¿información? ¿Y qué es una operación ?. Tome un ejemplo:

La temperatura es de 30º

• Pieza de información: representación formal de un concepto, en este caso: “30”

• Información: el resultado de la interpretación de los datos: "Hace calor"

• Operación: regla aplicada para obtener información: “A medida que la temperatura es más alta

de 23, hace calor "

1.2 Representación interna de datos

Por lo tanto, necesitamos almacenar y manejar en computadoras datos y operaciones.

Y para eso, necesitan usar el código binario.

Todo tipo de datos, tanto números como letras, se almacenan utilizando este

sistema.

Este sistema se basa en el uso de solo dos dígitos, 0 y 1, a diferencia del decimal

sistema que utiliza diez (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Esto se debe a que solo las computadoras

conocer estos dos valores numéricos resultantes de la detección o no de alguna

potencial, de varios voltios. Por tanto, una computadora sabe que hay un 0 cuando el

El potencial medido en un miembro interno tiene un valor cercano a 0 voltios. De otra manera,

detecta un 1.

En términos eléctricos, el potencial podría asimilarse al

fuerza en la que la corriente eléctrica pasa a través de un cable.

En general, los valores de 1 suelen corresponder a un potencial de alrededor de 3

o 5 voltios.

Todos los elementos informáticos manejan este sistema de numeración e interpretación de

información. Se podría decir que las computadoras en realidad no saben nada en absoluto. Ellos

sólo sé acerca de 0 y 1 y cómo realizar algunas operaciones básicas con

ellos (+, -, \* ...), aunque más rápido.

CGFS. DESARROLLO DE APLICACIONES WEB 1.4

SISTEMAS INFORMÁTICOS UD01. REPRESENTACIÓN DE INFORMACIÓN

2. SISTEMAS NUMERALES

Un sistema de numeración es un conjunto de símbolos clasificados que se utilizan para representar cantidades.

El número de símbolos se denomina base del sistema.

En el mundo real, estamos acostumbrados a usar el sistema decimal (base 10), cuyo conjunto de

los símbolos ordenados son 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Cualquier número, representado en cualquier sistema numérico, se puede dividir en dígitos. por

ejemplo, 128 se puede dividir en 1, 2, 8 o 34,76 en 3, 4, 7, 6. De estos dígitos y

con su posición y la base del sistema es posible obtener nuevamente el número:

128 = 1 \* 102

+ 2 \* 101

+ 8 \* 100

34,76 = 3 \* 101

+ 4 \* 100

+ 7 \* 10-1 + 6 \* 10-2

Podemos ver que un número decimal se puede representar como sumas de potencias.

de 10 (la base del sistema decimal).

Si generalizamos, un número N expresado en un sistema numérico B sería como:

N = an-1 an-2… a1 a0, a-1 a-2…. a-p + 1 a-p

dónde:

N: número a representar

a: los símbolos que incluye nuestro sistema numérico (números enteros de 0 a B-1)

Los dígitos antes de la coma (,) 1

son la parte entera.

Los dígitos después de la coma (,) son la parte fraccionaria.

2.1 Código binario

El código binario es un sistema numérico cuya base del sistema es 2 y sus símbolos

son 0 y 1.

 Cada dígito de un número binario se llama bit y es el más pequeño

unidad de información, en otras palabras, es lo mínimo que se puede

representado

 Para evitar confusiones, es habitual indicar el sistema base

número que se representará con un subíndice a la derecha. Por ejemplo

101 (10 o 101 (2

2.1.1 Cómo convertir un número decimal en un número binario

En general, para convertir un número decimal en otra base, tenemos que realizar

divisiones sucesivas del número por la base. Al final, tenemos que conseguir el

1 En la cultura inglesa, la separación entre la parte decimal y la parte fraccionaria es un punto decimal (.)

residuos y el último cociente y los ordenó en la dirección opuesta.

Considere el caso de convertir un decimal en binario con algunos ejemplos:

\*ilustraciones\*

En números con parte fraccionaria, el proceso es el mismo para la parte entera,

pero la parte fraccionaria se calcula multiplicando por 2 sucesivamente y para tomar el

parte entera (en este caso en el orden correcto).

El bit más a la izquierda se llama bit más significativo (MSB) y el

El bit más a la derecha se denomina bit menos significativo (LSB).

2.1.2 Cómo convertir un número binario en un número decimal

En este caso el proceso es muy sencillo. Como se explicó anteriormente, un número decimal

se puede representar como sumas de potencias de diez.

En general, puede convertir el valor de un número representado en un numeral

sistema B2

en el sistema decimal usando la siguiente fórmula:

n-1

N = an-1B

n-1 + an-2B

n-2 + ... + a1B

1 + a0B

0 + a-1B

-1 + ... + a-pB

-p = ∑ ai Bi

i = -p

Lo vamos a utilizar para convertir en base 2

El proceso consta de cuatro pasos

1. Escribir las cifras de números binarios multiplicadas por 2.

2. Escribir un signo más (+) entre cada uno de los productos.

3. Escribir un exponente en cada 2, comenzando desde cero y desde el último

número de la parte entera (en el extremo derecho si no hay parte fraccionaria)

y aumentando uno a uno hacia la izquierda y disminuyendo hacia la derecha.

4. Para realizar la operación

2.1.3 Número máximo de valores a representar

Una de las preguntas típicas al manejar un número binario es saber cuál es el máximo

valor decimal que se puede representar mediante un determinado número de bits. La respuesta es fácil: 2n, donde n

es el número de bit. Por ejemplo, con 4 bits podemos representar 16 valores, de 0 a 15 (0000-1111)

2.1.4 Operaciones con números binarios

La suma y resta binaria siguen las siguientes reglas:

Adición:

0 + 0 = 0

1 + 0 = 1

0 + 1 = 1

1 + 1 = 0 (llevar 1)

Sustracción:

0-0 = 0

1 - 0 = 1

0-1 = 1 (llevar 1 al sustraendo)

1 - 1 = 0

El resultado de las operaciones es el mismo que sus operaciones decimales relacionadas, excepto

para los casos en los que el resultado no tiene un valor en el sistema binario, es decir

1 + 1, que no se puede representar con 2 y 0-1, que no se puede representar con -1.

Aquí es donde la transferencia es importante.

Algunos ejemplos:

\*\*

Si queremos sumar dos números binarios cuál suma es mayor

que el número máximo para representar la computadora arroja un

advertencia de desbordamiento. Por ejemplo, si tenemos una computadora que funciona

con 8 bits, puede representar de 0 (10 a 255 (10. Si queremos agregar

10000000 (2 (128 (10)) más 10000000 (2 (128 (10)) tenemos un problema

porque el resultado es 100000000 (2 (256 (10) mayor que 255. Así que un

se produce un desbordamiento.

En la resta, la transferencia no se suma al minuendo, pero

sustraendo.

Multiplicación:

0 \* 0 = 0

1 \* 0 = 0

0 \* 1 = 0

1 \* 1 = 1

División:

0/0 = Indefinido

1/0 = ilimitado

1/1 = 1

0/1 = 0

Tanto la multiplicación como la división no presentan diferencias con respecto a los

operaciones en decimal, a menos que las operaciones auxiliares se realicen en binario.

 En la multiplicación, cuando sumamos, puede ser que tengamos en el

misma columna más de dos unos. En este caso, realizamos el

adiciones en grupos de dos y van a llevar los 1 en el próximo

columna.

 En la división, comenzamos a obtener dividendos y divisores iguales

número de figuras. Si no se puede dividir, intentamos conseguir uno

figura más en el dividendo.

Si la división es posible, entonces, el divisor solo puede estar contenido

una vez en el dividendo, es decir, el primer cociente es 1. En este

caso, el resultado de multiplicar el divisor por 1 es el divisor mismo

(el valor que restaremos).

2.1.5 Números negativos

Cuando necesitamos representar un número binario negativo, tenemos varias opciones

CGFS. DESARROLLO DE APLICACIONES WEB 1.9

SISTEMAS INFORMÁTICOS UD01. REPRESENTACIÓN DE INFORMACIÓN

aunque tres son los más importantes. Este rango indica que la forma de

Expresar debe ser un acuerdo entre dos partes: la que genera

el número y uno para leerlo. En caso contrario, el valor real a expresar sería

incorrecto.

Magnitud firmada

Quizás sea el enfoque más fácil de entender. La idea es mantener el MSB

para indicar el signo del número: 0 positivo, 1 negativo. Los bits restantes

indicar el valor numérico en valor absoluto. Por ejemplo:

Como puede verse, necesitamos un poco para indicar el signo, de modo que lo que en normal

Los valores de representación serían de 0 a 15 en este caso, para usar el signo, se vuelve de

-7 a +7 (1111-0111).

Este sistema es simple de entender pero complejo de usar al realizar

Operaciones matemáticas. Además tiene un problema: hay dos formas de definir

el 0 (10: 0000 (2 y 1000 (2

Complemento de unos

La segunda opción también usa el primer bit como indicador de signo, pero en este caso el

se logra un número negativo complementado número positivo (cambiando unos

por ceros y viceversa).

En esta opción se requiere dar el número de bits a codificar, de tal manera

que si en el ejemplo anterior usamos 8 bits para codificar:

Este método tiene el mismo problema que la magnitud con signo: hay dos formas de definir

0 (10: 0000 (2 y 1111 (2

Complemento a dos

Aunque el complemento de unos simplifica las operaciones matemáticas, lo hacen

mucho más con el uso del complemento a dos. Por eso es el más usado

método.

El complemento a dos consiste en aplicar un complemento a unos y luego sumar 1.

Por ejemplo, el complemento a dos de 5 codificado con 8 bits es:

5 (10 → 101 (2 → (codificado en 8 bits) 00000101 (2 → (complemento de 1) 1111010 (2 →

(+1) 11111011

¿Qué número decimal representa un número en complemento a dos? Fácil. Nosotros

tiene que realizar el mismo proceso:

11111011 (2 → (complemento de 1) → 00000100 (2 → (+1) 00000101 (2 → 5 (10

La gran ventaja del método del complemento a dos es que permite

resta como si fueran sumas. Esto se debe a que restar dos números binarios es

lo mismo que sumar al minuendo el complemento del sustraendo.

101101 (2 (45 (10) - 010101 (2 (21 (10)  010101 (2 (complemento de 1) → 101010 (2 →

→ (+1) 101011  101101 + 101011

101101

+ 101011

1011000 (2 (24 (10) el último arrastre 1 se rechaza

Exceso de K o binario de compensación

Dependiendo de la cantidad de bits disponibles, el rango medio está dedicado para negativos.

números y la otra mitad (menos 1) a los positivos (el valor cero está en el

medio). El nuevo rango será [-K, K-1], donde podemos calcular por K = 2n-1 3

.

Una vez que tenemos el rango permitido, el número más pequeño es quién tiene todos sus bits

a 0. Veamos un ejemplo:

Tenemos 3 bits para representar el número, por lo que podemos representar 23

números,

el rango [0,7]. En este caso, K será 23-1 = 22

= 4, entonces el rango con negativo

los números serán [-4,3]. El número más pequeño -4 será 000 y los 3 más grandes serán

ser 111. El tablero completo será:

2.1.6 Números reales

Cuando escribimos un número real en un papel, usamos una coma decimal (o decimal

punto, depende de la cultura) para distinguir entre la parte entera y

parte fraccional. En una computadora, el espacio para representar este tipo de números es

3 Existe otra versión de este método con K = 2n-1-1

CGFS. DESARROLLO DE APLICACIONES WEB 1.11

SISTEMAS INFORMÁTICOS UD01. REPRESENTACIÓN DE INFORMACIÓN

dividido en dos áreas: una para la parte entera y otra para la parte fraccionaria.

Hay dos formas de denotar el tamaño de estas áreas (campos) y, por lo tanto, la

posición de la coma: coma fija y coma flotante.

Punto fijo

En esta notación, asignamos un tamaño fijo a la parte entera y la fracción

parte del número, en otras palabras, un lugar fijo a la coma.

La ventaja es que el proceso para realizar operaciones básicas es el mismo que

números enteros. Sin embargo, este método no aprovecha las

capacidad del formato de representación utilizado. Por ejemplo, una computadora con 8 bits para

representar números, podría usar 5 bits para la parte entera y 3 para la parte fraccionaria

b7b6b5b4b3, b2b1b0.

En este caso el número máximo a representar será 01111,111 y el

mínimo (positivo) 00000,001. Si la coma decimal estuviera en forma flotante

posición, el rango de números positivos que tu representa sería 011111111 -

0,0000001

Punto flotante

El rango de números que se pueden representar en el formato de punto fijo es

insuficiente para muchas aplicaciones, particularmente para aplicaciones científicas que

a menudo se utilizan números muy grandes y muy pequeños. Para representar una amplia gama de

números que utilizan relativamente pocos dígitos, es bien conocido en el sistema decimal, el

representación científica o notación exponencial. Por ejemplo, 0,00000025 =

2,5 \* 10-7. en general, para cualquier sistema de numeración, se puede expresar un número real

Como

N = M \* BE o N = (M; B; E)

dónde:

M: mantisa

B: base

E: exponente

La representación interna que hace este formato en las computadoras se conoce como

punto flotante.

Por ejemplo, en decimal (B = 10) 259,75 (10 = 0,25975 \* 103

o (0,25975; 10; 3) o,

en código binario (B = 2)

259,75 (10 → 100000011,11 (2 → 0,10000001111 \* 29

(2 → 0,10000001111 \* 21001

(2

→ (0,10000001111; 1001)

El rango de números representables para un valor dado de B, está fijado por el número

de bits del exponente E, mientras que la precisión está determinada por el número de

bits de M.

Normalización

El mismo valor real se puede representar de infinitas formas mediante exponencial

notación. Por ejemplo, 2,5 se puede representar como 0,25 \* 101

, 0,025 \* 102

, 250 \* 10-

2

, ... Para evitar confusiones, debemos elegir uno de estos formatos como estándar

para la representación en coma flotante de un número real. La forma elegida se llama

Forma normalizada y es aquella que mantiene la mayor precisión en el

representación de números. Esto se logra cuando se ubica el punto binario

inmediatamente a la izquierda del primer dígito significativo, de modo que no se desperdicie espacio

que no representan dígitos significativos.

Por ejemplo:

- 2,5 representado en forma normalizada es 0,25 \* 101

- (0,000011101; 2; 0111) → (0,11101; 0011) 4

→ Exponente de exceso-k (0,11101; 1011)

En general, para representar exponentes negativos se utiliza el método de exceso-k. En

por otro lado, para representar mantisas negativas como método de magnitud con signo

IEEE754

El formato más popular para representar puntos flotantes en binario fue

desarrollado por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) y es

llamado IEEE754. Este formato puede representar casos especiales como infinito

valores y resultados indefinidos, resultados NaN o Not a Number. Propone 3

formatos:

Media precisión. Utiliza 16 bits

Utiliza 16 bits: un bit para el signo, 5 para el exponente, 10 para la mantisa

Precisión simple.

Utiliza 32 bits: un bit para el signo, 8 para el exponente, 23 para la mantisa

Precisión doble.

Utiliza 64 bits: un bit para el signo, 11 para el exponente, 55 para la mantisa

Los tres formatos usan mantisa normalizada, por lo que el primer bit en

la mano izquierda de la mantisa será un 1 (el primer dígito significativo

tiene que ser un 1). Debido a esto, tres formatos no codifican este 1

en la mantisa, aunque se tiene en cuenta al operar

con el numero. En otras palabras, en estos formatos el MSB está en el

a la izquierda de la coma decimal y solo guardan los bits del lado derecho.

Para representar el exponente, el estándar usa el método Exceso-K

con K = 2n-1 -1

Para comprender mejor esta operación se recomienda encarecidamente

mira la pastilla 02

Además de las operaciones matemáticas (+, -, \* /), en números binarios se puede aplicar booleano

u operaciones lógicas: y, o, xor, no ...

NO:

Se puede representar de varias formas: NOT, ¬

NO 0 = 1

NO 1 = 0

Y:

Se puede representar de varias formas: AND, Y, ^, \*

0 Y 0 = 0

1 Y 0 = 0

0 Y 1 = 0

1 Y 1 = 1

En otras palabras, el resultado será verdadero (1) solo cuando ambos dígitos sean verdaderos. Como

se puede ver, el resultado es el mismo que la multiplicación.

10011010 1011

Y 01001100 Y 111101

00001000 001001

O:

Se puede representar de varias formas: OR, OOR:

Se puede representar de varias formas: OR, O, v, v

0 O 0 = 0

1 O 0 = 1

0 O 1 = 1

1 O 1 = 1

En este caso, el resultado será verdadero tan pronto como uno de los dígitos sea verdadero.

XOR:

0 XOR 0 = 0

1 XOR 0 = 1

0 XOR 1 = 1

1 XOR 1 = 0

En este caso, el resultado será verdadero cuando uno y solo uno de los dígitos fueron

cierto.

2.2 Octal

Además del binario, hay otros dos sistemas numéricos interesantes cuando se trabaja

sobre temas relacionados con las tecnologías de la información: octal y hexadecimal. Esto es

porque a partir de ellos son fáciles de convertir a binarios.

El octal es un sistema numérico con una base de sistema igual a 8 (símbolos 0,1, 2, 3, 4, 5, 6, 7). Sus

base es una potencia exacta del sistema binario base 23 = 8 o, en otras palabras, con tres dígitos binarios

(con tres bits) podemos representar todos los dígitos octales.

\*\*

2.2.1 Cómo convertir un número binario en octal

El proceso consiste en crear grupos de tres bits, comenzando por la mano derecha,

y reemplazarlos por el valor octal relacionado

1101011 (2 => 1101011 => 153 (8

2.2.2 Cómo convertir un número octal en binario

2.2.2 Cómo convertir un número octal en binario

El proceso se invierte al anterior: se convierte a binario cada uno de los

números de número octal

7402 (8 => 11110000010 (2 = 111100000010 (2

2.3 Hexadecimal

Su base del sistema es 16. Como el número de símbolos usados ​​en el sistema es mayor

de 10, se deben utilizar 6 caracteres, en este caso de la A a la F.

conjunto de símbolos es: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

2.3.1 Cómo convertir números binarios en hexadecimales

El proceso es similar al proceso binario-octal, excepto en este caso grupos

están en cuatro.

1101011 (2 => 110 1011 => 6B (16

2.3.2 Cómo convertir números hexadecimales en binarios

El proceso se invierte al anterior: se convierte a binario cada uno de los números de octal

número

7F0A (16 => 0111 1111 0000 1010 (2 = 111111100001010 (2

2.3.3 Cómo convertir números hexadecimales en números octales

Convertimos a binario y agrupamos los bits en cuatro o tres, cualquiera que sea el sistema numérico

destino

6B (16 => 110 1011 => 1101011 (2 => 1101011 => 153 (8

La conversión entre octal o hexadecimal y decimal o

viceversa, se puede realizar siguiendo los métodos para convertir

entre binario y decimal, pero multiplicando por potencia de 8 o 16 o

dividir por estos números y obtener el resto (en

hexadecimal si el resto es mayor que 9 obtenemos su relación

valores A..F).

Sin embargo, suele ser más práctico realizar directamente

conversión a binario y luego convertir al sistema solicitado.

3. REPRESENTACIÓN ALFANUMÉRICA

3.1 Datos numéricos y alfanuméricos.

Un dato es numérico si es posible realizar operaciones matemáticas. En

Por el contrario, un dato es alfanumérico Si NO puede realizar

operaciones en él.

numérico: ¿cuántos años tienes? 45

alfanumérico: ¿Cuál es tu nombre? "Roberto"

 Para diferenciar claramente entre los dos tipos de datos,

Es común usar comillas simples o dobles para indicar que los datos son

alfanumérico.

Es habitual pensar que los datos numéricos son números y los alfanuméricos son

solo cartas. Pero esto no es correcto. Por ejemplo:

¿Cual es tu dirección? "Avenida de las Palmeras 34"

¿Cuál es su número de teléfono móvil? "555341273"

En el primer caso, Avenida de las Palmeras 34, se compone de letras y

números, y en el segundo, 555341273, solo por números, pero no operables

(no tiene sentido sumar o multiplicar dos números de teléfono).

3.2 Representación interna

Los caracteres alfanuméricos para representar computadoras se basan en tablas, de modo que cada

de las entradas de la tabla (cada número) corresponde a un símbolo alfanumérico.

A lo largo de la historia de la informática, ha habido varias tablas que han

siempre se ha caracterizado por el número de bits utilizados para representar cada

personaje. Uno de los mejores ejemplos en la tabla ASCII. El número de bits es 7,

que dejó espacio para 128 caracteres (27 = 128)

Como se puede ver en la siguiente tabla, cada número está relacionado con un carácter.

Por ejemplo, 73 (10 es una "I", 105 (10 es una "i" o 50 (10 es un "2"). Las primeras entradas son

reservado para caracteres no imprimibles, aquellos que no son visibles, como

tabulador (9 (10) o retorno de carro (15 (10).

 El espacio también es un carácter: 32 (10

El problema de esta mesa es su espacio limitado. Como puede ver, tiene espacio para

todas las grafías latinas utilizadas en los idiomas anglosajones, pero no podemos encontrar

ortografías como ñ, ç o vocales acentuadas. Por lo tanto, la tabla ASCII extendida de

Se crearon 8 bits (256 caracteres). Esta nueva mesa puede incorporar todo el latín

ortografía más algunos símbolos gráficos.

Hoy las tablas ASCII están casi obsoletas. La expansión de

Internet y la globalización hacen necesarias tablas que

incorporar no solo caracteres latinos, sino también chino, árabe, coreano,

Ruso, hebreo ...

4. SISTEMA DE UNIDADES

Como se discutió anteriormente, el bit es la unidad de información más pequeña. Hoy no

funcionan a nivel de bits, pero en grupos de bits (en la sección anterior hemos visto que

un carácter se codifica con 7 u 8 bits).

Porque dentro de la computadora todo está en código binario, una manera fácil de

manejar grupos es usar algún valor que sea una potencia de 2, siendo el más básico 23

= 8. Un grupo de 8 bits se denomina byte.

Hoy en día no es habitual usar el grupo de bytes, pero algunos múltiplos de él: Kilobyte (kB),

Megabyte (MB), Gigabyte (GB), Terabyte (TB) .... En el sistema internacional

estos múltiplos son potencias de 10 (se basan en el sistema decimal), pero en

se utilizan potencias de cálculo de 2. Sin embargo, la tendencia es utilizar el International

Sistema, aunque cabe señalar que los valores son similares pero no el

mismo.

Usamos dos tipos diferentes de palabras para diferenciar entre las unidades del sistema.

Cuando hablamos de kilobyte nos referimos al sistema decimal y cuando hablamos de

kibibytes al sistema binario.

Las equivalencias se pueden ver en la siguiente tabla:

Aunque se suele utilizar de forma indiferente e, incluso hoy en día es más

sistema internacional común, los valores que se representan son

diferente: 1 MB son 1,000,000 de bytes (un millón de bytes), mientras que 1 MiB

son 1.048.576 bytes

 Es importante diferenciar entre kB y kb. El primero

se refiere al kilobyte, mientras que el segundo kilobit, 8 veces menos.

 Aunque la mayoría de los nombres y abreviaturas de múltiplos

tienen letras mayúsculas, el kilo se define con minúsculas.