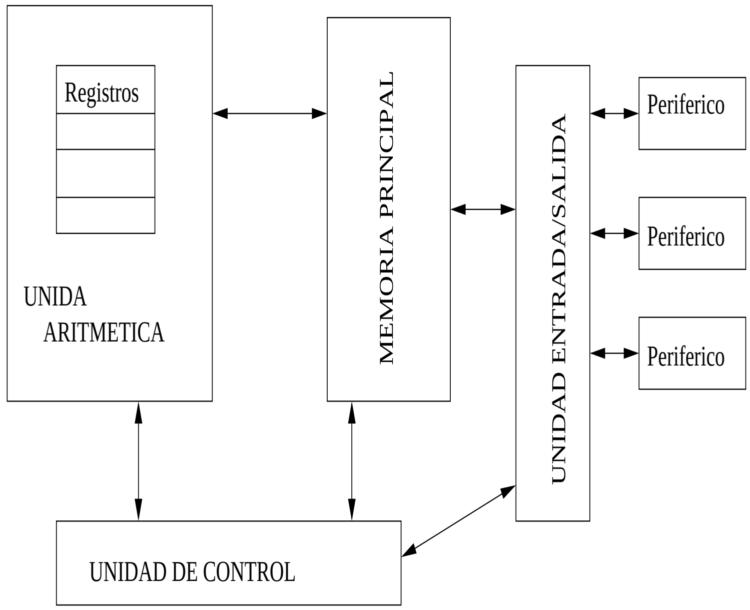

Figure 1: *Arquitectura Von Neumann del Ordenador Digital.*

Transistor tiene dos estados: pasa o no pasa corriente.

Tambien disco duro tiene dos estados: polarización izquierda o polarización derecha.

Por tanto, la unidad básica de información en el ordenador es el bit, con dos estados posibles, que llamaremos: 0 y 1.

- Sistema decimal (Base 10) $\longrightarrow$ 10 símbolos (0,1,2,3,4,5,6,7,8,9)

  Ejemplo:

  $342 = 3 \cdot 10^2 + 4 \cdot 10^1 + 2 \cdot 10^0$

- Sistema binario (Base 2) $\longrightarrow$ 2 símbolos (0,1)

  - Conversiones (decimal-binario) para números enteros.

    * Conversión de binario a decimal. Sean $b_0, \ldots, b_n$ los n bits de un número en binario. Entonces el número en decimal será:

      $$\sum_{i=0}^{n} b_i 2^i$$

      Ejemplo:
      $1001 = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 9$

    * Conversión de decimal a binario. Sea D un número decimal conocido. Entonces se podrá expresar como:

      $$D = \sum_{i=0}^{n} b_i 2^i$$

      donde $b_0, \ldots, b_n$ representan los bits de su representación binaria. Llamemos R al resto de una división y llamemos C al conciente de una división. Entonces:

      $$\begin{array}{rcr} D = \sum_{i=0}^{n} b_i 2^i & \Rightarrow & R(D/2) = b_0 \\ C(D/2) = \sum_{i=1}^{n} b_i 2^{i-1} & \Rightarrow & R(\frac{D/2}{2}) = b_1 \\ C(\frac{D/2}{2}) = \sum_{i=2}^{n} b_i 2^{i-2} & \Rightarrow & R((\frac{D/2}{2})/2) = b_2 \\ \vdots & \vdots & \vdots \end{array}$$

      Realizar hasta obtener cociente cero

      Ejemplo:

      $$\begin{gathered} R(C(9/2)) = b_0 \\ R(C(C(9/2)/2)) = b_1 \\ etc \end{gathered}$$

      El algoritmo para sumar y multiplicar es similar a base 10.

    * Números binarios con decimales.

      $$D = \sum_{i=0}^{n} b_i 2^i + \sum_{i=1}^{n} c_i 2^{-i}$$

      Example: 10.101 = 2.625

    * Conversion de decimal con decimales a binario: multiplicar por 2 sucesivamente hasta terminar en 1 exacto. Ejemplo: Convertir 0.625 decimal a binario.

$$0.625 * 2 = 1.25 \rightarrow 1$$
$$0.25 * 2 = 0.5 \rightarrow 0$$
$$0.5 * 2 = 1.0 \rightarrow 1$$

Solución: 0.101

Ejercicio: Convertir 0.1 decimal a binario. Observar que hay infinitos decimales en la representación binaria del número.

- Operaciones algebraicas con binarios. Ejemplo, sumar 11+01

- Sistema hexadecimal (Base 16) $\longrightarrow$ 16 símbolos (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)

  Ejemplo:

  Conversión a decimal:

  $2C8 = 2 \cdot 16^2 + 12 \cdot 16^1 + 8 \cdot 16^0 = 712$

  Conversión de hexadecimal a decimal: similar a binario pero dividiendo por 16 en lugar de por 2.

REPRESENTACION DE DATOS.

- Carácter

  Es un campo de 8 bits (1 Byte). El código ASCII asocia 8 bits a un carácter. Ver por ejemplo la siguiente direccion web:

  http://www.ascii-code.com/

  Algunos ejemplos de esta correspondecia ASCII:

  M $\rightarrow$ 01001101

) $\rightarrow$ 00101001

  t $\rightarrow$ 01110100

  Ejercicio:

  Cuántos caracteres se pueden representar con 1 byte?

- Número Entero

  Se utilizan "n" bits donde n=16 o n=32. Normalmente por defecto son 32 bits (numerados de derecha a izquierda de 0 a n-1).

  - Representación con bit de signo.
    El bit n-1 se suele reservar para el signo del número:
    $0 \rightarrow +$
    $1 \rightarrow -$
    Ejemplos:
    $00\cdots010 \;=\; 2$
    $10\cdots011 \;=\; -3$
    Ejercicio:
    Obtener el subconjunto de enteros que se pueden representar con 4 bits.
    Encontramos dos problemas:
    - la suma de positivo y negativo no da cero (por ejemplo, 0010 + 1010 = 1100), puesto que el bit de signo es un convenio. Sí podemos aplicar la ley de suma en los numeros positivos (o en la parte positiva del numero).
    - hay dos ceros
  - Representación complemento a dos.
    El número negativo se obtiene del positivo cambiando 0 por 1 y 1 por 0 y sumando 1. Por ejemplo: 0101 = 5; 1011 = -5.
    Ejercicio:
    Obtener el subconjunto de enteros que se pueden representar con 4 bits.

- Número Real (o de coma flotante).

  Se utilizan "n" bits donde típicamente n=32 (precisión simple) o n=64 (precisión doble).. Normalmente por defecto son 32 bits.

Los n bits se dividen en 3 campos, comenzando por la derecha (menos significativa): "p" bits para la mantisa o fracción (m), "q" bits para el exponente (e), un bit para el signo.

El número x se representa entonces como:

$$x = s \cdot m \cdot B^e$$

donde B es la base. Normalmente $B = 2$.

El exponente se suele obtener sumando $2^{q-1} - 1$. Por ejemplo, si $q = 8$, para representar el exponente 14 escribiremos $14 + 127 = 141$ (en binario) en los 8 bits del exponente.

Para precisión simple la representación suele ser:

$q = 8$, $p = 23$

lo cual nos dá un rango de exponentes de -126 a 127 y unos 7 decimales exactos.

Para precisión doble la representación suele ser:

$q = 11$, $p = 52$

lo cual nos dá un rango de exponentes de -1022 a 1023 y unos 16 decimales exactos.

Ejemplos:

En precisión simple.

1 10000101 11011010100000000000000 = -118.625

Ejercicio:

Supongamos $q = 3$, $p = 4$. Convertir el siguiente contenido de memoria a decimal sabiendo que es un número en coma flotante (número real): 11100111.