Estructura de Datos

Unidad I

Tipos de Datos

Conceptos Básicos

Algoritmo: es una secuencia finita de pasos o instrucciones ordenadas crono-lógicamente que describen un método para resolver un problema específico.

Programa: es la codificación de un algoritmo en un lenguaje de programación.

Lenguaje de programación:es un conjunto limitado de palabras y símbolos que representan procedimientos, cálculos, decisiones y otras operaciones, como control de procesos, que puede ejecutar una computadora.

Clasificación de lenguajes

Por su nivel:

- Alto: instrucciones semejantes a lenguaje natural
- Medio
- Bajo: instrucciones semejantes a lenguaje máquina

Por su traducción:

- Compiladores: lee un programa escrito en un lenguaje fuente y lo traduce a un programa equivalente en otro lenguaje, el lenguaje objeto.
- Intérpretes: un intérprete es un programa que, como su nombre lo indica, interpreta símbolos en un programa y los traduce a lenguaje máquina conforme deban ser ejecutados.

Elementos de un programa

Estructura de datos: tipos de datos simples y estructurados (colecciones).

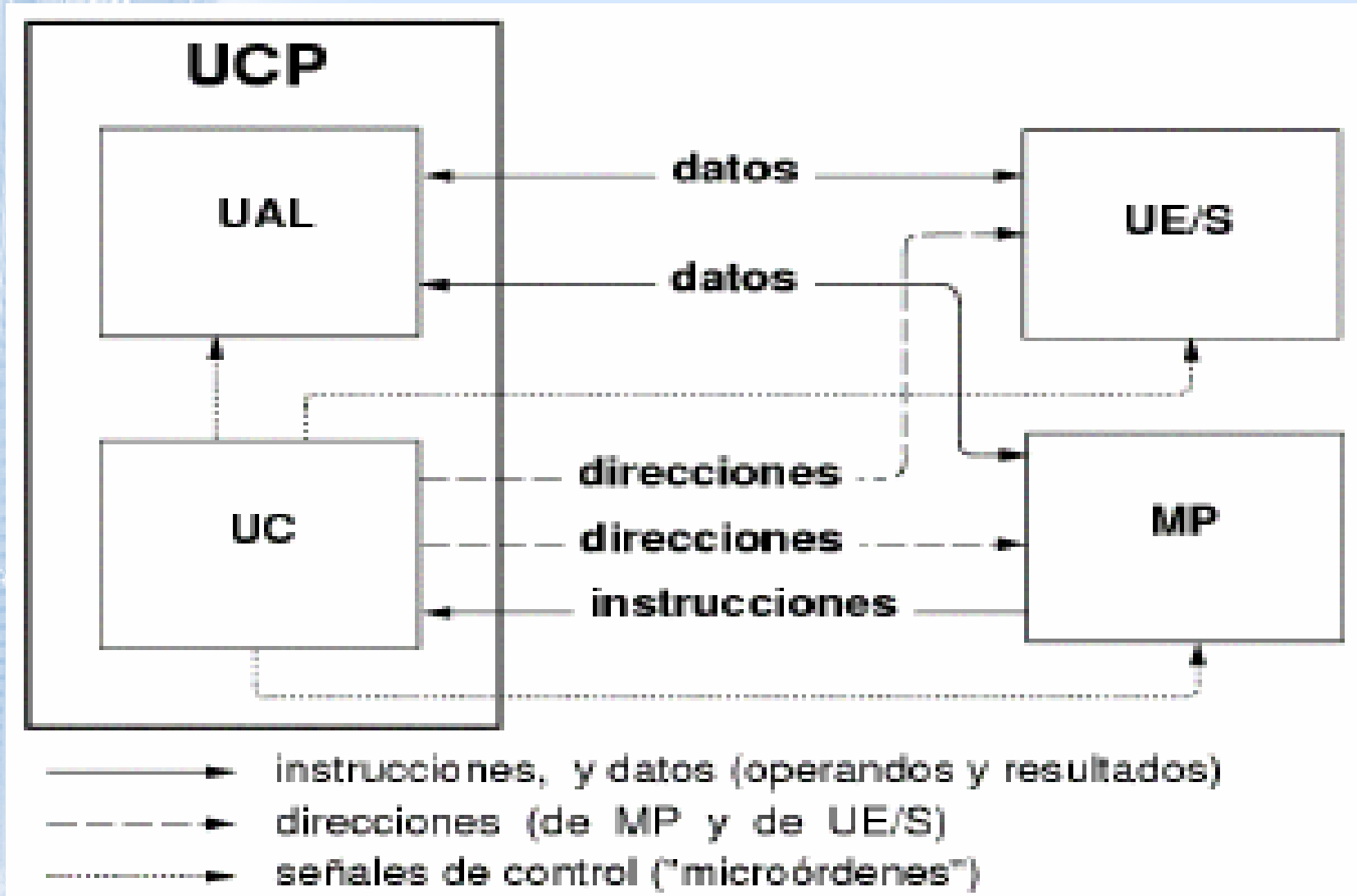
Operaciones primitivas elementales:

- Aritméticas:+,-,/,*,%
- Relaciónales: >,<,>=,<=,!=,==
- Lógicas: &&,||,!

Estructuras de control:

- Secuencia
- Selección: simple, doble, múltiple
- Repetición: predefinida (for)
  condicionada (while)

Arquitectura de la computadora



Definición de bit, byte, dir. memoria

- bit= Unidad mínima de memoria (celda) que significa dígito binario (**bi**nary digi**t**) = (0,1)

- 1 Byte = 8 bits y es identificado a través de una dirección de memoria.

- Dirección de memoria: número que identifica de manera única un espacio de memoria (byte o palabra). Generalmente esta conformado por segmento (segment) y corrimiento (offset) y por facilidad se trabaja en hexadecimal.

Ejemplo para un direccionamiento de 16 bits:

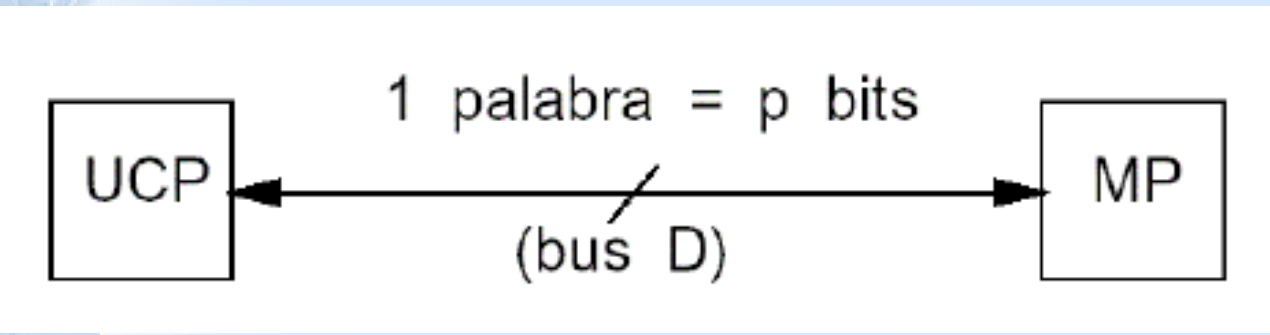
01A7:FFFF

Definición de carácter y palabra

- Carácter: es representado por medio de tablas de codificación, las mas comunes son:
  - ASCII (8 bits = 256 caracteres)
  - UNICODE (16 bits = 65,535 caracteres)
- Palabra: un procesador trabaja con un tamaño de palabra N, porque sus registros tienen tamaño N y generalmente las transacciones de datos o instrucciones se hacen en bloques de N bits.

  Normalmente, el tamaño de palabra de las pc´s modernas es de 32 bits; es decir, el bus del sistema puede transmitir 32 bits (4 bytes de 8 bits) a la vez entre el procesador, la RAM y los periféricos.

Palabra

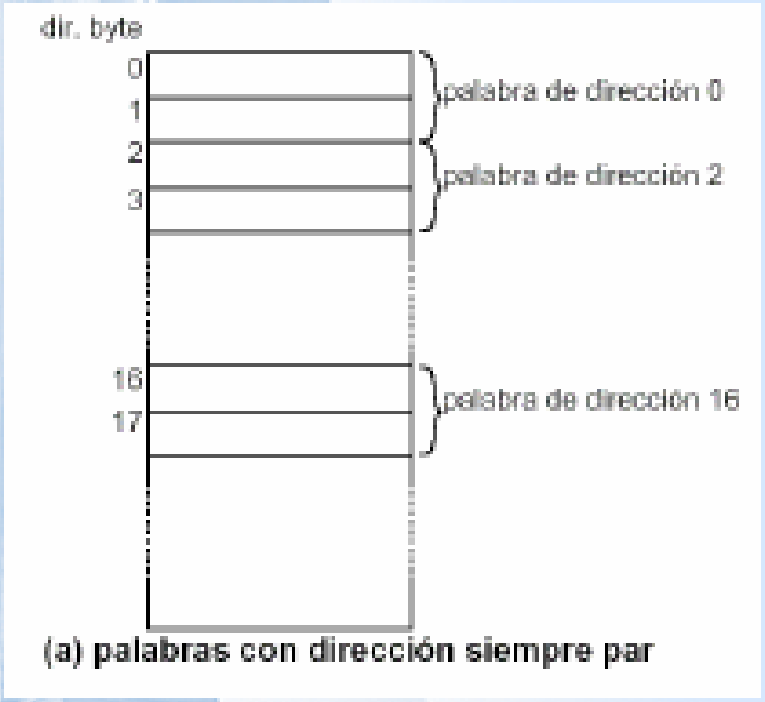


Palabra = p bits (con p = 8, 16, 32, 64, … bits)

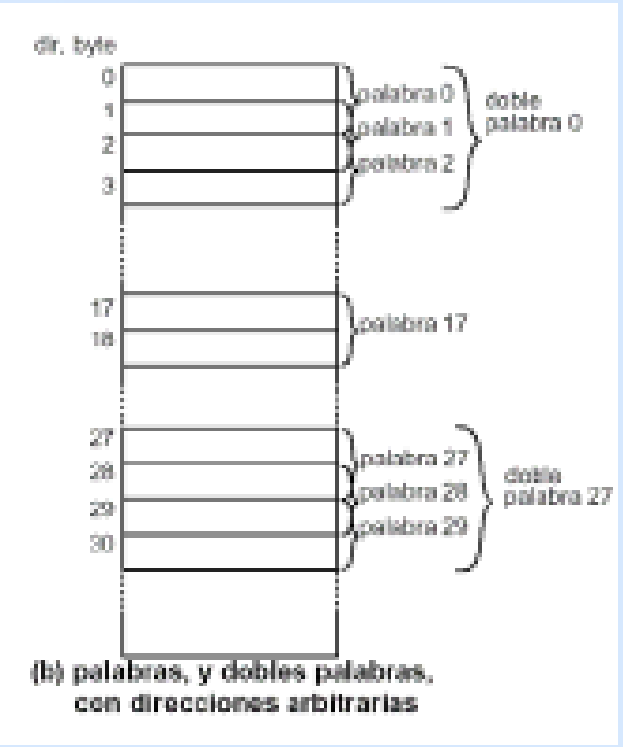
- Si necesitamos n bits para representar un dato, son:
  - n / 8 bytes
  - n / p palabras (1/4, 1/2, 1, 2, 4, … palabras)
- En la UCP, los registros suelen tener p bits
- En MP, normalmente se puede acceder a palabras, medias palabras, … y bytes

Acceso a datos en la Memoria Principal

- Direccionable por bytes.
- Palabras de 1, 2, 4, 8 bytes:
  - 1 KB = 2^{10} Bytes = 1024 Bytes
  - 1 MB = 2^{20} Bytes = 1024 KB
  - 1 GB = 2^{30}, 1 TB=2^{40}
- Direcciones alineadas o no alineadas.



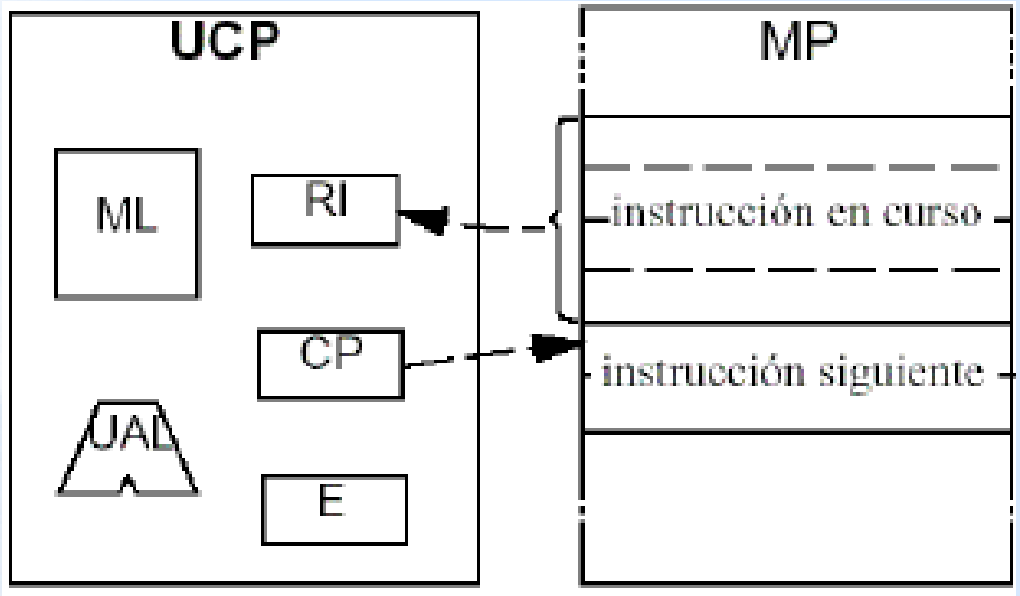
(a) palabras con dirección siempre par



(b) palabras, y dobles palabras, con direcciones arbitrarias

Registros en la UCP

- Aritméticos(antiguos acumuladores)
- De direccionamiento (índice,base, etc.)
- De propósito general (normalmente, organizados como una ML)
- De estado (incluyen indicadores Z, V, N, C)
- Contador de programa
- Otros, transparentes en el nivel de máquina convencional



Conversión de decimal a otro sistema numérico

- Se divide repetidamente el número decimal entre la base del otro sistema numérico y se escriben los residuos después de cada división, hasta obtener un cociente de ceros.
  - Binario (base 2, dígitos del 0 al 1)
  - Octal (base 8, dígitos del 0 al 7)
  - Hexadecimal (base 16, dígitos del 0 al 9 y letras A-F)
- El primer residuo es el bit o dígito menos significativo (bms) y el último el Bit o dígito Más Significativo (BMS).
- La parte fraccionaria del número se convierte a binario multiplicándola repetidamente por la base y anotando cualquier acarreo en la posición de los enteros.
- Las multiplicaciones continúan hasta obtener un producto con parte fraccionaria de cero (la mayoría de las veces esto no ocurre y el proceso se termina hasta alcanzar el número de bits o dígitos deseado).

Conversión de cualquier sistema numérico a decimal

- Se multiplica cada dígito del número por su correspondiente peso según su posición y se suman los resultados. El peso del dígito es una potencia de la base del sistema numérico, comenzando el dígito menos significativo de la parte entera en una potencia de cero.

$$\ldots\ldots b^5, b^4, b^3, b^2, b^1, b^0 . b^{-1}, b^{-2}, b^{-3}, b^{-4}, b^{-5}, \ldots\ldots$$

$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$1\ 1\ 0\ 1\ 0\ 0\ 1\ 1 = 128+64+0+16+0+0+2+1 = 211_{10}$

$2^2\ 2^1\ 2^0 \quad 2^{-1}\ 2^{-2}\ 2^{-3}$

$1\ 0\ 1\ .\ 1\ \ 0\ \ 1 = 4+0+1+0.5+0+0.125 = 5.625_{10}$

Otras conversiones

Conversión de Binario a Hexadecimal: se forman grupos de 4 bits hacia la izquierda y hacia la derecha del punto que indica las fracciones, hasta cubrir la totalidad del número binario. Enseguida se convierte cada grupo de número binario de 4 bits a su equivalente hexadecimal. Ejemplo:

$$010011101010=0100\text{-}1110\text{-}1010=4EA_{16}$$

Conversión de Binario a Octal: se hacen grupos de 3 bits hacia la izquierda y hacia la derecha del punto que indica las fracciones, hasta cubrir la totalidad del número binario. Enseguida se convierte cada grupo de número binario de 3 bits a su equivalente octal. Ejemplo:

$$1010101=001\text{-}010\text{-}101=125_8$$

Conversión de octal a binario: se convierte cada dígito octal por separado a binario (1 dígito octal equivale a 3 dígitos binarios).Ejemplo:

$$715_8 = (111001101)_2$$

Conversión de hexadecimal a binario: se convierte cada dígito hexadecimal por separado a binario (1 dígito hexadecimal equivale a 4 dígitos binarios).

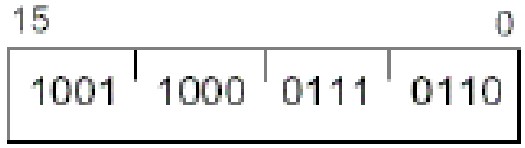
$$1F0C_{16} = 1111100001100_2$$

Representación de datos simples

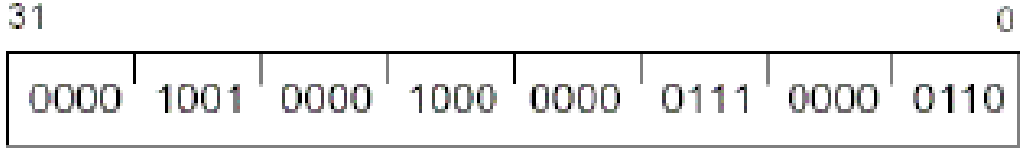
- Direcciones o «punteros» y enteros sin signo:
  - d bits para 2^d direcciones
- Números enteros (con signo):
  - binario (o decimal, BCD)
  - n bits para una extensión de -2^{n-1} a $2^{n-1}-1$ (formatos de punto fijo)
- *Números reales:*
  - formatos de punto flotante
  - *n* bits que determinan la extensión y la precisión
- *Caracteres:*
  - ASCII, o ISO Latin9 (8 bits) o Unicode (uno o varios bytes)

Representación de enteros

BCD (poco usado):

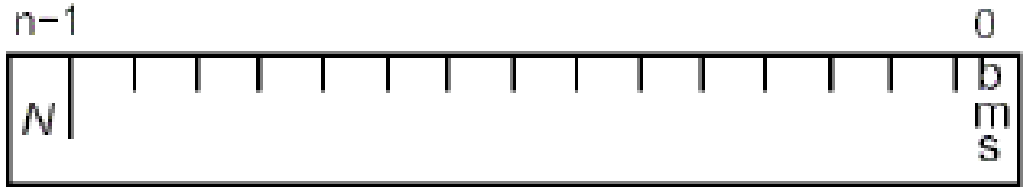


decimal empaquetado



decimal desempaquetado

Formato de coma fija:



Números negativos:

Signo y módulo, o complemento a 1, o complemento a 2

Representación de enteros sin signo

– En binario, con 16 bits

– Rango de valores: de 0 a $2^{16} – 1 = 65535$

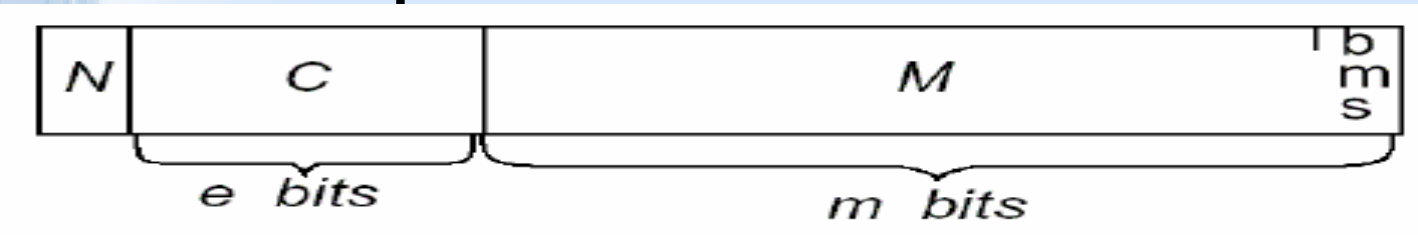
– Ejemplos:

- D’4 = B’0000 0000 0000 0100 = H’0004
- D’10 = B’0000 0000 0000 1010 = H’000A
- D’1022 = B’0000 0011 1111 1110 = H’03FE
- D’12365 = B’0011 0000 0100 1101 = H’304D

Representación de enteros con signo

* Complemento a 1. En binario, con 16 bits
– Rango de valores: de $-(2^{15} - 1)$ a $2^{15} - 1$ = -32767 a 32767
– Ejemplos:
  - D’4 = B’0000 0000 0000 0100 = H’0004
  - D’-4 = B’1111 1111 1111 1011 = H’FFFB
  - D’-12365 = B’1100 0000 1011 0010 = H’C0D2
  - D’12365 = B’0011 0000 0100 1101 = H’304D
* Complemento a 2. En binario, con 16 bits
– Rango de valores: de -2^{15} a $2^{15} - 1$ = -32768 a 32767
– Ejemplos:
  - D’-12365 = B’1100 0000 1011 0011 = H’C0D3
  - D’-4 = B’1111 1111 1111 1100 = H’FFFC

Representación de números reales: punto flotante



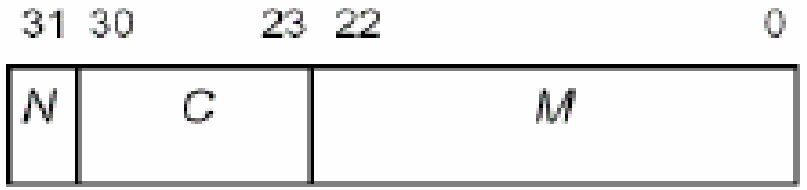
$X = \pm M \times bE$ (normalización entera), o bien

$X = \pm 0.M \times bE$ (normalización fraccionaria)

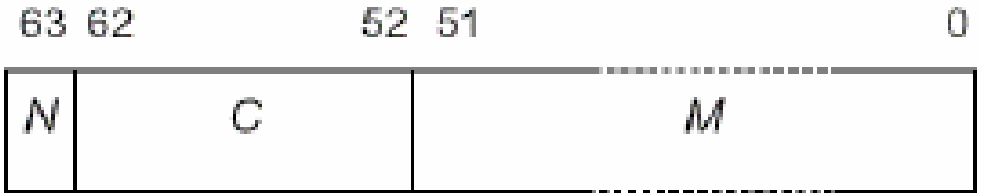
$X = \pm 1.M \times bE$ (bit fantasma)

- Base implícita: b (normalmente, b = 2)
- Números negativos: signo y módulo o complemento (normalmente, sólo la mantisa)
- Exponente: E (tradicionalmente, en exceso de 2^{e-1}, es decir, $E = C - 2^{e-1}$)

Norma IEEE 754



(a) precisión sencilla



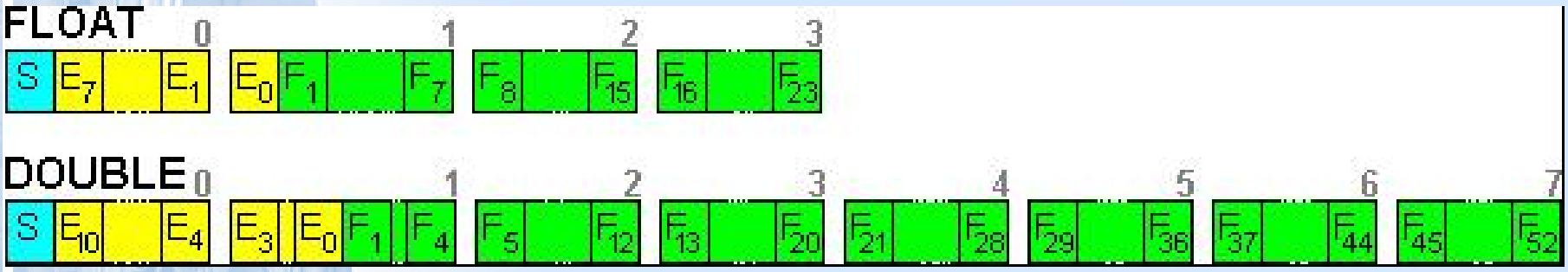
(b) precisión doble

$$X = \pm 1, M \times b^E$$

- *N: Negativos:* signo y módulo
- M: normalización fraccionaria, omitiendo el bit más significativo (= 1)
- *Base*: b = 2
- E en exceso de $2^{e-1}-1$ ($E = C - 2^{e-1} + 1$)

Estándar IEEE 754

Se especifican 2 tipos básicos que corresponden a los tipos de datos "float" y "double" en lenguaje C, que utilizan 4 y 8 bytes respectivamente, con la siguiente asignación de bits:



Donde "S" es el signo (1 si es negativo, 0 si es positivo), "E" es el exponente con un "BIAS" (o corrimiento sumado) para no utilizar signo en el exponente y "F" es la parte fraccionaria.

Representación de caracteres

Ejemplo de representación de información textual (caracteres):

– Código ASCII en los 8 bits menos significativos de una palabra:

- ‘a’ = B’0110 0001 = Q’141 = H’61 = 97
- ‘ret’ = B’0000 1101 = Q’015 = H’0D = 13
- ‘ ‘ = B’0010 0000 = Q’040 = H’20 = 32

Interpretación de los contenidos de la Memoria Principal

dir. MP	contenido (binario)	cont. (hexadecimal)
[0]	1010 0001 0010 0001	A121
[1]	0111 0111 1111 1010	77FA
… … …		
[5]	0100 0001 0110 1000	4168
	… … …	

¿Qué hay en estas palabras?

¿Cómo se interpreta el contenido?

Depende…

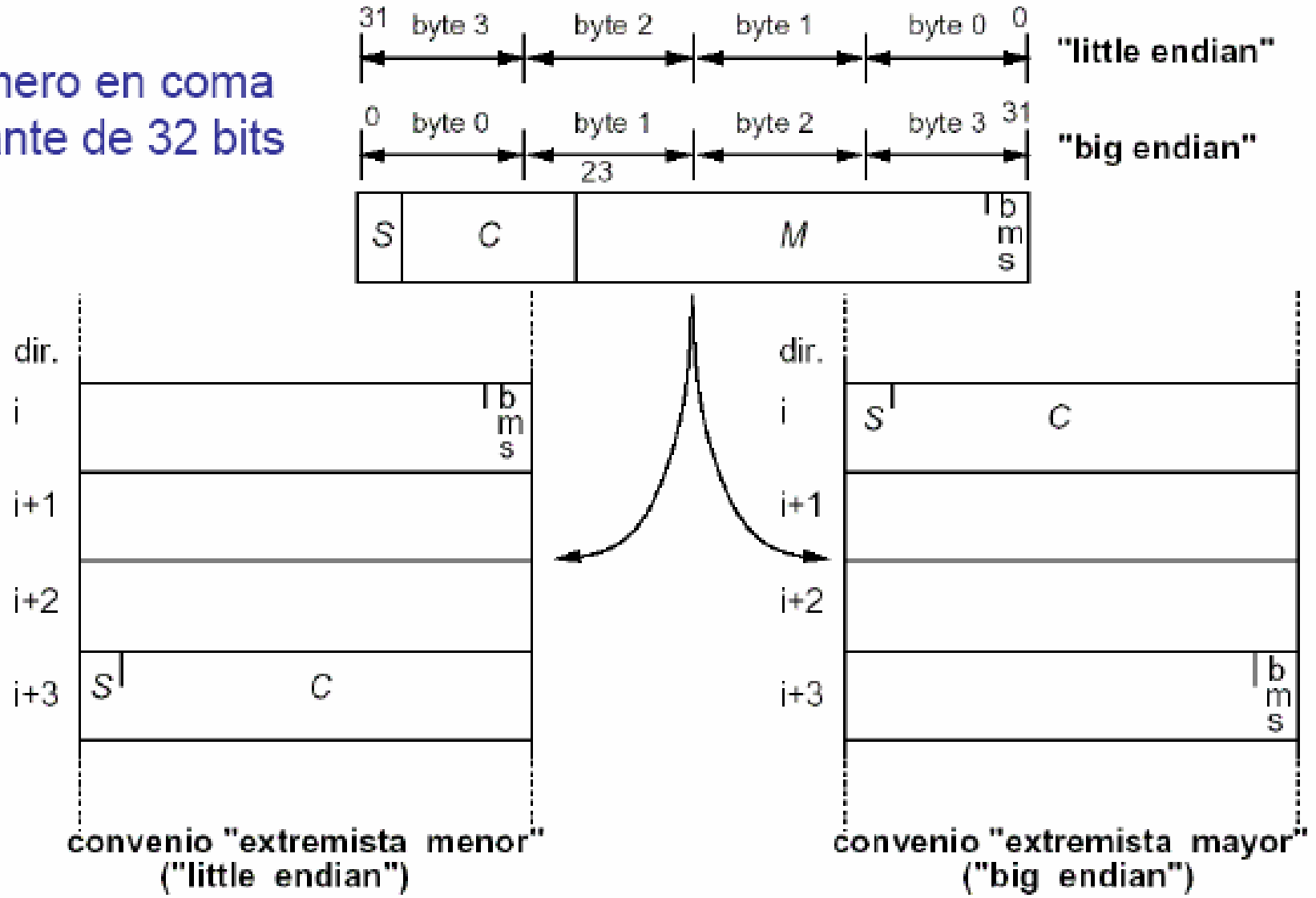
Interpretación de los contenidos

Dirección de Memoria	Interpretación como		
	Instrucción	Entero con signo (C a 2)	ASCII (2 caracteres)
0	Sumar 2 bytes	-24287	¡!
1	Desplazar bits a la izquierda con signo	30714	wú
5	Desplazar bits a la izquierda	16744	Ah

En el nivel de máquina convencional no hay “tipos” (salvo en algunas máquinas)

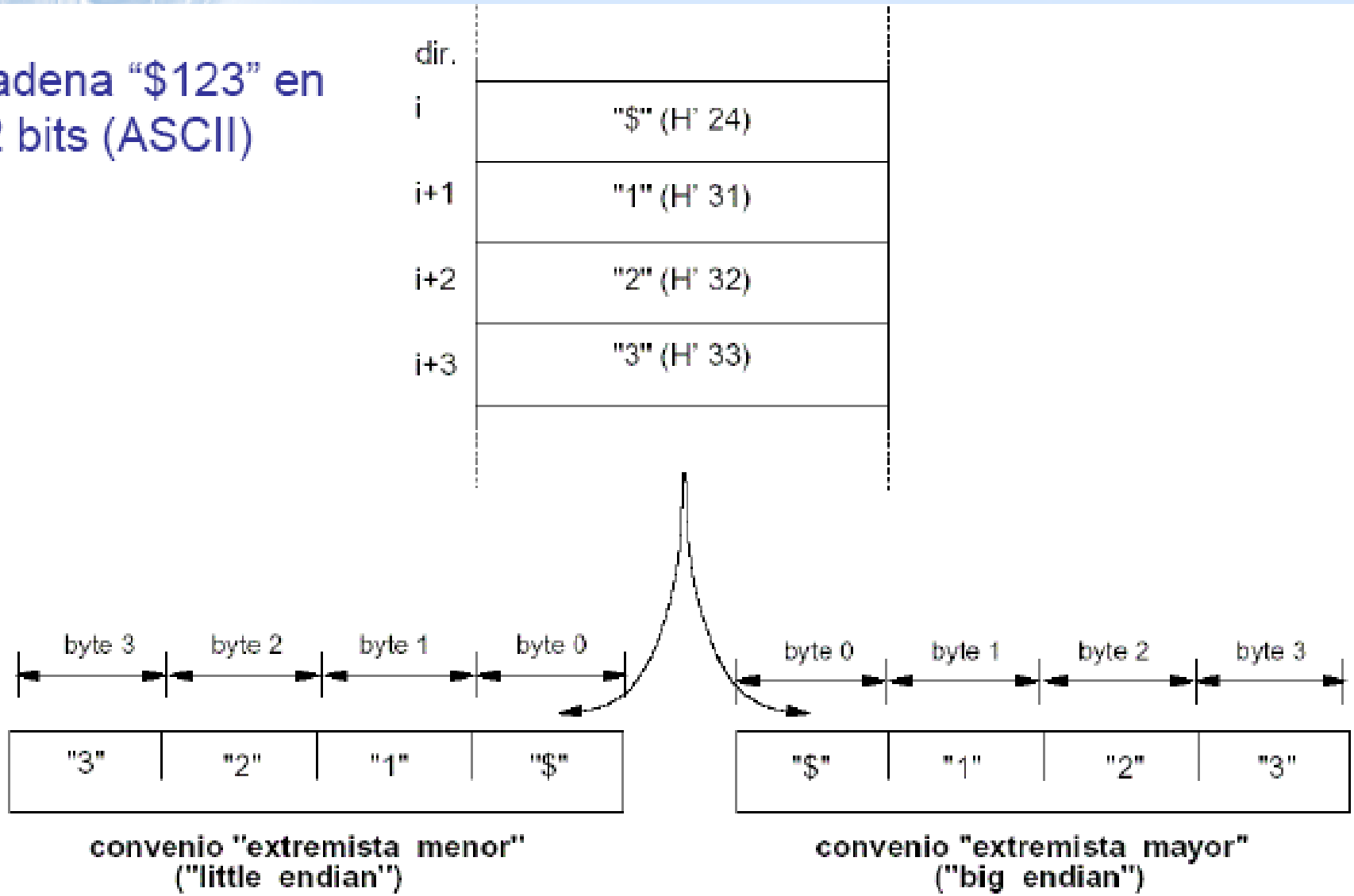
Extremistas mayores y menores: ejemplo 1

Número en coma flotante de 32 bits



Extremistas mayores y menores: ejemplo 2

Cadena “$123” en 32 bits (ASCII)



Tipos Abstractos de Datos (TAD)

Abstracción:La abstracción es la capacidad para encapsular y aislar la información, del diseño y ejecución. Se refiere a ignorar los detalles no esenciales, tratando en su lugar con el modelo ideal del objeto y centrándonos en el estudio de sus aspectos esenciales.

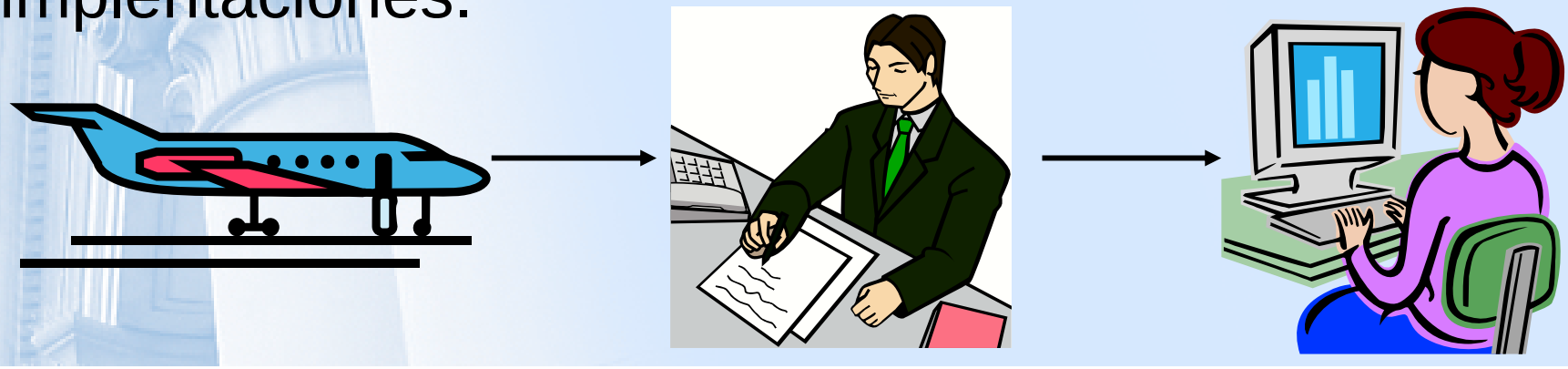
Ejemplo: Una abstracción de un alumno para un sistema de inscripción sería:

- Propiedades: nombre, noctrl., carrera, semestre…
- Métodos: Carga_materia, Eliminar_materia, ….

¿Que es un TAD?

• Cuando se escribe un programa para resolver un problema, con el enfoque tradicional se pasa directamente de la realidad a una implementación en el lenguaje de programación.

• Los TAD establecen un nivel intermedio, donde se quiere moderar lo esencial de la realidad sin comprometerse con detalles de implementación. De hecho, es posible considerar diferentes implentaciones.



Definición de un TAD

Un TAD es una estructura algebraica, o sea, un conjunto de objetos con ciertas operaciones definidas sobre ellos.

Ejemplos:

Una calculadora: los elementos que maneja son cantidades numéricas y las operaciones que tiene definidas sobre éstas son las operaciones aritméticas.

Una matriz: los elementos que maneja son las matrices y las operaciones son aquellas que nos permiten crearlas, sumarlas, invertirlas, etc.

Elementos de un TAD

Con los TAD se identifican ciertas operaciones o partes del algoritmo que manipulan los datos:

Programa = Datos + Algoritmo de Datos + Algoritmos de Control

Algoritmo de datos: es la parte del algoritmo encargada de manipular las estructuras de datos del problema.

Algoritmo de control: es la parte restante, la que representa en sí el método de solución del problema, independiente hasta cierto punto de las estructuras de datos seleccionadas.

La Implementación de TAD son:

Datos + Algoritmo de Datos

El enfoque de desarrollo con TAD define programa como:

Implementación del TAD + Algoritmo de Control

Tipo de operaciones en TAD

Creación de objetos:

- Iniciales: Se utilizan para crear objetos del TAD, en cuya creación no se requieren ningunos objetos abstractos del mismo tipo.
- Constructores: Utilizadas para crear objetos del TAD cuya creación depende de objetos del mismo tipo.

Transformación de objetos:

- Simplificadoras: Son operaciones cuyo codominio es el TAD que se define, pero que dan como resultado objetos que pueden ser descritos utilizando únicamente operaciones iniciales y constructoras.

Análisis de los elementos del TAD:

- Analizadoras: Son operaciones cuyo codominio no es el TAD que se define, sino otro ya conocido. El propósito de este tipo de operaciones es obtener información concerniente a cualquiera de los objetos abstractos de tipo.

Implementación de un TAD

• Un tipo abstracto de datos (TAD) es un tipo de dato definido por el programador que se puede manipular de un modo similar a los tipos de datos definidos por el sistema.

• Un tipo de dato abstracto corresponde a un conjunto (puede ser de tamaño indefinido) de valores legales de datos y un número de operaciones primitivas que se pueden realizar sobre esos valores.

• Los usuarios pueden crear variables con valores que están en el rango de valores legales y pueden operar sobre esos valores utilizando las operaciones definidas.

• Escoger unas estructuras de datos para representar cada uno de los objetos abstractos y escribir una rutina (Procedimiento o función) en un lenguaje de programación, que simule el funcionamiento de cada una de las operaciones de acuerdo con su especificación.

Estructura de datos(1)

- Se trata de un conjunto de variables de un determinado tipo agrupadas y organizadas de alguna manera para representar un comportamiento.

- Lo que se pretende con las estructuras de datos es facilitar un esquema lógico para manipular los datos en función del problema que haya que tratar y el algoritmo para resolverlo. En algunos casos la dificultad para resolver un problema radica en escoger la estructura de datos adecuada. Y, en general, la elección del algoritmo y de las estructuras de datos que manipulará estarán muy relacionadas.

Estructura de datos(2)

- Una estructura de datos se refiere a la definición y organización de la estructura (espacio de memoria) donde se almacena el valor o valores que toma un dato en determinado instante.

- Cuando se habla de estructura de datos generalmente se asocia con los tipos de datos estructurados, por lo cual se dice que una estructura de datos es:

- "La organización de espacios de memoria necesarios para almacenar una colección de valores de diferente o igual tipo de dato y que describe el comportamiento de los datos abstraídos de la realidad".

- Una estructura de datos es referenciada a través de un identificador o nombre de variable.

Clasificación de Estructura de datos

Existen tipos de datos simples y estructurados:

- Datos de tipo simple almacenan un solo valor a la vez.
- Datos de tipo estructurado almacenan una colección de valores de un mismo tipo de datos o de diferentes tipos.

Con respecto al manejo de memoria se clasifican en:

- Estáticas: su tamaño se define en tiempo de compilación y durante la ejecución del programa el tamaño de la estructura no cambia.
- Dinámicas: durante la ejecución del programa el tamaño de la estructura puede cambiar.

Con respecto a su organización se clasifican en:

- Lineales: se organizan como una lista secuencial de elementos (unidimensional). Ejemplo: pilas, colas, listas enlazadas, arreglos unidimensionales.
- No lineales: su organización es no lineal (multidimensional). Ejemplo: arreglos multidimensionales, árboles (organización jeraquica) y grafos.