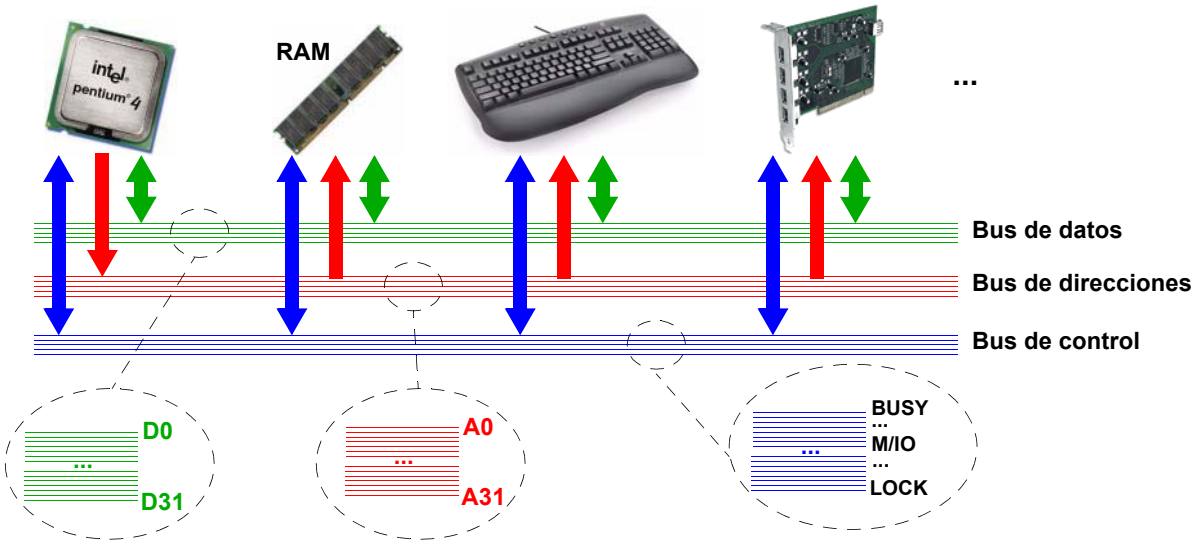
Instrumentación de Tiempo Real

UNIVERSIDAD DE CANTABRIA

Mapas de memoria y mapas de entrada/salida

UNIVERSIDAD DE CANTABRIA

El acceso a la memoria y a los dispositivos de E/S se realiza mediante los buses de datos, direcciones y control:

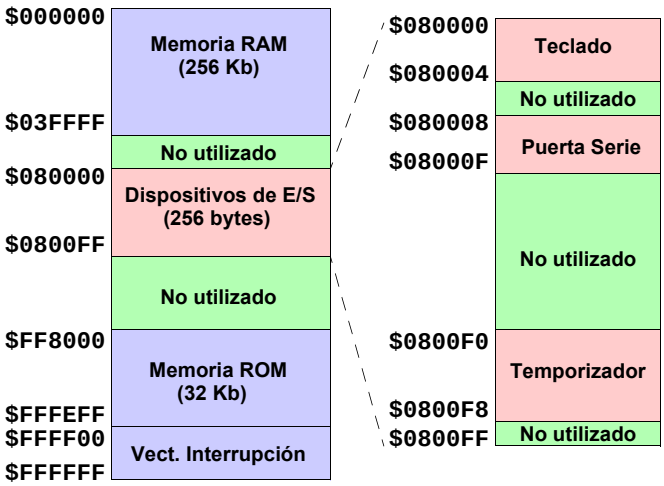


Mapas de memoria y mapas de entrada/salida (cont.)

UNIVERSIDAD DE CANTABRIA

En algunas arquitecturas no hay diferencia entre registros de E/S y registros de memoria (p.e. familia 68K de Motorola)

- los registros de E/S se encuentran "mapeados" en posiciones de memoria
- se accede a ellos utilizando las mismas instrucciones que para acceder a la memoria

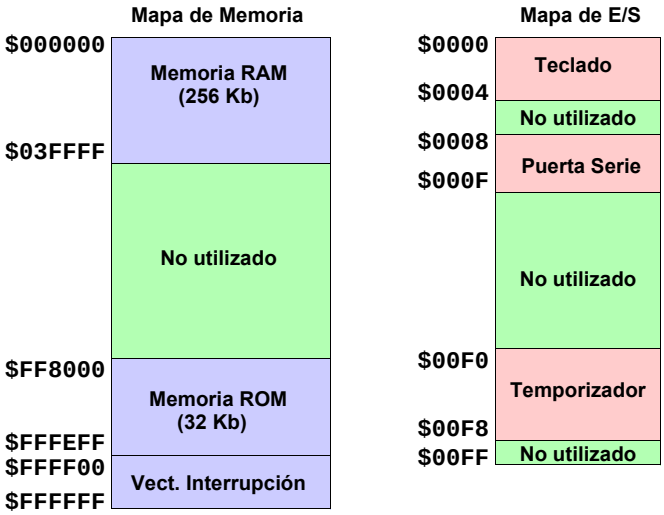


Mapas de memoria y mapas de entrada/salida (cont.)

UNIVERSIDAD DE CANTABRIA

En otras arquitecturas los mapas de memoria y de E/S son independientes (p.e. familia x86 de Intel)

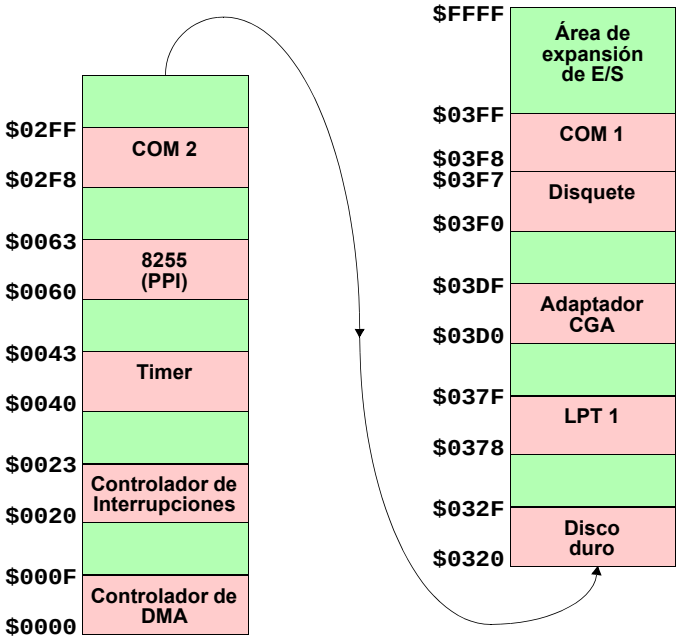
- se utilizan instrucciones diferentes para acceder a la memoria (`MOV`, `XCHG`, etc.) o a los registros de E/S (`IN`, `OUT`, etc.)
- con la línea `MEM/IO` del bus de control se selecciona el mapa



Mapas de memoria y mapas de entrada/salida (cont.)

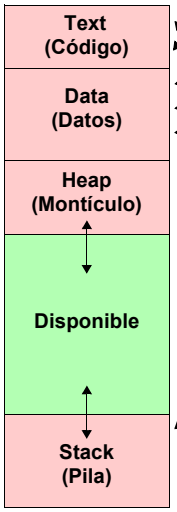
UNIVERSIDAD DE CANTABRIA

Mapa de E/S de un PC



Instanciación física de variables

UNIVERSIDAD DE CANTABRIA



Estructura de un programa en memoria

```
procedure Ejemplo_Memoria is
  type Int_10 is range 1 .. 10;
  D1 : Int_10;
  D2 : Float := 1.0;
  D3 : access all Integer;

  procedure Un_Proc is
    S : Integer;
  begin
    ...;
  end Un_Proc;

begin
  D3 := new Integer;
  D1 := 2 * Int_10 (D2);
  ...;
end Ejemplo_Memoria;
```

no tiene representación en memoria

Acceso a direcciones de memoria física

UNIVERSIDAD DE CANTABRIA

En ocasiones es necesario acceder a una posición determinada del mapa memoria para acceder a los dispositivos de E/S:

- También en la arquitectura Intel x86
  - dispositivos PCI mapeados en memoria
  - acceso directo a memoria de video

Para ubicar una variable en una posición de memoria:

```
N : Integer;
for N'address use 16#EF5A01#;
```


Acceso a direcciones de memoria física (cont.)

UNIVERSIDAD DE CANTABRIA

Para asignar una variable puntero a una dirección absoluta:

```
with MaRTE_OS;
with Ada.Unchecked_Conversion;
with Basic_Integer_Types; use Basic_Integer_Types;

procedure Asigna_Punteros is
  type Mi_Entero is range 1 .. 100;
  type Puntero_A_Mi_Entero is access all Mi_Entero;

  function Asigna_Direccion is
    new Ada.Unchecked_Conversion (Unsigned_32,
                                  Puntero_A_Mi_Entero);
  Un_Puntero : Puntero_A_Mi_Entero;

begin
  Un_Puntero := Asigna_Direccion (16#F7B45#);
end Asigna_Punteros;
```

Acceso a los registros de E/S

UNIVERSIDAD DE CANTABRIA

En la arquitectura Intel x86 el acceso a las direcciones de E/S se realiza mediante instrucciones ensamblador especiales:

- `inb`, `outb`, `inw`, `outw`, `inl`, `outl`

Para acceder a las direcciones de E/S desde el lenguaje Ada, MaRTE OS proporciona el paquete `IO_Interface`

- **con funciones y procedimientos para leer y escribir bytes, words (2 bytes) y long words (4 bytes)**
- **para cada función y procedimiento existe una versión que realiza un retraso después de la lectura o escritura del dato**
  - **necesarias para acceder a dispositivos lentos o cuando se realizan muchos accesos seguidos**
  - **son las operaciones terminadas en "`_P`"**

Acceso a los registros de E/S (cont.)

UNIVERSIDAD DE CANTABRIA

```
package IO_Interface is

   --  Registros de E/S (también llamados "Ports")
   type IO_Port is range 0 .. 16#FFFF#;

   --  Lee un byte de un registro de E/S
   function Inb (Port : in IO_Port) return Unsigned_8;
   function Inb_P (Port : in IO_Port) return Unsigned_8;

   --  Lee un word (2 bytes) de 2 registros consecutivos
   function Inw (Port : in IO_Port) return Unsigned_16;
   function Inw_P (Port : in IO_Port) return Unsigned_16;

   --  Lee un long word (4 bytes) de 4 reg. consecutivos
   function Inl (Port : in IO_Port) return Unsigned_32;
   function Inl_P (Port : in IO_Port) return Unsigned_32;
```

Acceso a los registros de E/S (cont.)

UNIVERSIDAD DE CANTABRIA

```
   --  Escribe un byte en un registro de E/S
   procedure Outb (Port : in IO_Port; Val : in Unsigned_8);
   procedure Outb_P (Port : in IO_Port;
                     Val  : in Unsigned_8);

   --  Escribe un word (2 bytes) 2 registros consecutivos
   procedure Outw (Port : in IO_Port;
                   Val  : in Unsigned_16);
   procedure Outw_P (Port : in IO_Port;
                     Val  : in Unsigned_16);

   -- Escribe un long word (4 bytes) en 4 reg. consecutivos
   procedure Outl (Port : in IO_Port;
                   Val  : in Unsigned_32);
   procedure Outl_P (Port : in IO_Port;
                     Val  : in Unsigned_32);
end IO_Interface;
```

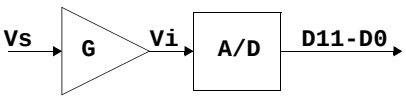

Ejemplo: Lectura de un convertidor A/D

UNIVERSIDAD DE CANTABRIA

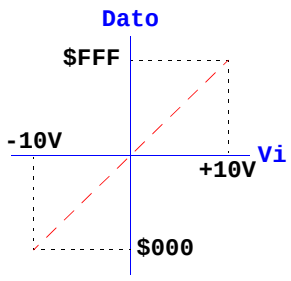
Un convertidor Analógico a Digital convierte una tensión de entrada (V_i) a un valor numérico proporcional al valor de V_i

Convertidor A/D de 12 bits de resolución y ganancia regulable:

LoDato	($240)	D3	D2	D1	D0	X	X	X	X
HiDato	($241)	D11	D10	D9	D8	D7	D6	D5	D4
Ganancia	($244)	X	X	X	X	R3	R2	R1	R0
Estado	($246)	B	X	X	X	X	X	X	X



Ganancia	R3	R2	R1	R0
1	0	0	0	1
2	0	0	1	0
4	0	1	0	0
8	1	0	0	0
16	0	0	0	0

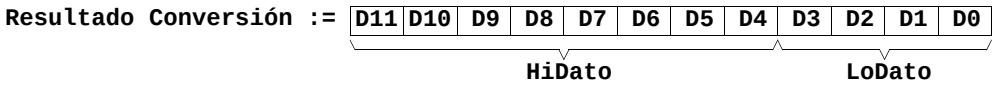


Ejemplo: Convertidor A/D (cont.)

UNIVERSIDAD DE CANTABRIA

Modo de operación del convertidor A/D:

- La ganancia se configura escribiendo en el registro `Ganancia`
- La conversión se inicia escribiendo cualquier valor en el registro `Estado`
- El fin de la conversión se detecta por el cambio de 1 a 0 del bit `B` del registro `Estado`
- El resultado de la conversión se lee de `HiDato` y `LoDato`
- El valor resultante de la conversión se obtiene "uniendo" ambos registros:



Ejemplo: Convertidor A/D (cont.)

UNIVERSIDAD DE CANTABRIA

La fórmula que relaciona el dato leido del convertidor con el valor real de la magnitud medida es:

$$ValorReal = \frac{-10 + \frac{20}{4095} \times Dato}{Ganancia}$$

Ejemplo: Convertidor A/D (cont.)

UNIVERSIDAD DE CANTABRIA

```
with MaRTE_OS;
with Basic_Integer_Types; use Basic_Integer_Types;
with IO_Interface;
with Ada.Text_IO; use Ada.Text_IO;

procedure Lee_Convertidor is

   --  Direcciones de los registros de E/S del convertidor
   Reg_Lo_Dato  : constant IO_Interface.IO_Port := 16#240#;
   Reg_Hi_Dato  : constant IO_Interface.IO_Port := 16#241#;
   Reg_Ganancia : constant IO_Interface.IO_Port := 16#244#;
   Reg_Estado   : constant IO_Interface.IO_Port := 16#246#;

   --  Valores de la ganancia
   type Valores_Ganancia is (G1, G2, G4, G8, G16);
```


Ejemplo: Convertidor A/D (cont.)

UNIVERSIDAD DE CANTABRIA

```
   --  Conversión entre los valores de la ganancia y
   --  el código binario que les representa
   Codigo_Ganancia : constant array (Valores_Ganancia)
     of Basic_Integer_Types.Unsigned_8 := (G1  => 16#01#,
                                           G2  => 16#02#,
                                           G4  => 16#04#,
                                           G8  => 16#08#,
                                           G16 => 16#00#);

   --  Conversión entre los valores de ganancia y el
   --  factor de ganancia
   Factor_Ganancia : constant array (Valores_Ganancia)
     of Float := (G1  => 1.0,
                  G2  => 2.0,
                  G4  => 4.0,
                  G8  => 8.0,
                  G16 => 16.0);
```

Ejemplo: Convertidor A/D (cont.)

UNIVERSIDAD DE CANTABRIA

```
function Lee_AD (Ganancia : Valores_Ganancia)
                 return Float is
   Hi, Lo : Basic_Integer_Types.Unsigned_8;
   Codigo : Basic_Integer_Types.Unsigned_16;
begin

   --  Configura la ganancia del convertidor
   IO_Interface.Outb (Reg_Ganancia,
                      Codigo_Ganancia (Ganancia));

   --  Inicia la conversión escribiendo un valor
   --  cualquiera en el registro de estado
   IO_Interface.Outb (Reg_Estado, 0);
```


Ejemplo: Convertidor A/D (cont.)

UNIVERSIDAD DE CANTABRIA

```
   --  Espera a que finalice la conversión
   loop
      exit when (IO_Interface.Inb (Reg_Estado)
                 and 16#80#) = 0;
   end loop;

   --  Lee el código de salida del convertidor
   Hi := IO_Interface.Inb (Reg_Hi_Dato);
   Lo := IO_Interface.Inb (Reg_Lo_Dato);
   Codigo := Basic_Integer_Types.Unsigned_16 (Hi) * 16
       + Basic_Integer_Types.Unsigned_16 (Lo) / 16;

   --  Obtiene el valor de la magnitud de entrada
   return (-10.0 + 20.0 / 4095.0 * Float (Codigo))
          / Factor_Ganancia (Ganancia);
end Lee_AD;
```

Ejemplo: Convertidor A/D (cont.)

UNIVERSIDAD DE CANTABRIA

```
   Car : Character;

--  Cuerpo del procedimiento principal
begin
   --  Lee valores de tensión
   Put_Line ("Pulsa cualquier tecla para iniciar " &
             "una conversión ('q' para salir)...");
   loop
      Get_Immediate (Car);
      exit when Car = 'q';  --  dejar el lazo?
      Put_Line ("Valor leido por el convertidor: " &
                Float'Image (Lee_AD (G1)));
   end loop;

end Lee_Convertidor;
```


Representación física de los tipos de datos

UNIVERSIDAD DE CANTABRIA

En la memoria del computador la información se almacena en grupos lineales de circuitos binarios

- pueden tomar dos estados que denominamos “0” y “1”

Los circuitos se organizan en las siguientes agrupaciones:

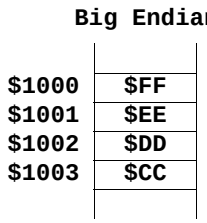
- Bit: unidad básica de memoria, puede tomar los valores “0” y “1”
- Byte: agrupación de 8 bits
- Word: agrupación de 2 bytes
  - este término también se utiliza para denotar a una agrupación de bytes con tantos bits como líneas tiene el bus de datos
- Long word: agrupación de 4 bytes

Representación física de los tipos de datos

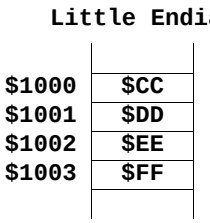
UNIVERSIDAD DE CANTABRIA

Existen dos criterios para mapear una palabra en registros de memoria:

```
P : Unsigned_32 := 16#FFEEDDCC#;
for P'address use 16#1000#;
```



Arquitecturas Motorola 68k



Arquitecturas Intel x86

Representación alfanumérica de la información

UNIVERSIDAD DE CANTABRIA

Modelo general de codificación de la información a través de su representación mediante secuencias de caracteres alfanuméricos (*strings*)

- **carácter alfanumérico: letras (mayúsculas y minúsculas), dígitos, símbolos matemáticos, símbolos ortográficos, códigos de control, otros caracteres**

Cada carácter alfanumérico se representa mediante el valor numérico indicado en la tabla ASCII

- **ocupa un byte de memoria**

Tabla ASCII

UNIVERSIDAD DE CANTABRIA

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Sistema de numeración posicional

UNIVERSIDAD DE CANTABRIA

$$\underbrace{d_p d_{p-1} \ldots d_1 d_0}_{\text{Número en base b escrito con p+1 dígitos}} = d_p \cdot b^p + d_{p-1} \cdot b^{p-1} + \ldots + d_1 \cdot b^1 + d_0 \cdot b^0$$

Base decimal:

$$6851_{10} = 6 \cdot 10^3 + 8 \cdot 10^2 + 5 \cdot 10^1 + 1 \cdot 10^0$$

Base binaria:

$$10011_2 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 19_{10}$$

Base hexadecimal:

$$1BE8_{16} = 1 \cdot 16^3 + B \cdot 16^2 + E \cdot 16^1 + 8 \cdot 16^0 = 7144_{10}$$

Conversión entre diferentes bases

UNIVERSIDAD DE CANTABRIA

Algoritmo general de conversión de base 10 a otra base:

- realizar sucesivas divisiones entre la base y quedarse con los restos

Ejemplo: escribir 23_{10} en base 2

```
23
÷2 = 11 ──────────► resto 1 ◄──── bit menos significativo
   ÷2 = 5 ────────► resto 1
      ÷2 = 2 ─────► resto 1
         ÷2 = 1 ──► resto 0
            ÷2 = 0► resto 1 ◄──── bit mas significativo
```

$23_{10} = 10111_2$

Conversiones entre las bases hexadecimal y binaria

- cada dígito hexadecimal corresponde a 4 cifras binarias

	2	A	4	E
$2A4E =	0010	1010	0100	1110

Representación binaria de números enteros con signo

UNIVERSIDAD DE CANTABRIA

Representación *signo-magnitud*

s	magnitud (n-1 bits)

- El signo (s) se codifica con un bit:
  - 0 => números positivos,
  - 1 => números negativos
- La magnitud con n-1 bits

Inconvenientes:

- dos representaciones del 0 (`00000000` y `10000000`)
- requiere tratar de forma independiente el signo y la magnitud en las operaciones aritméticas

Representación binaria de números enteros con signo (cont.)

UNIVERSIDAD DE CANTABRIA

Representación *complemento a 2*

- números positivos (`D in 0..`$2^{n-1}-1$) se expresan directamente mediante su código binario
- números negativos (`D in -1..-`2^{n-1}) se expresan como su complementario respecto a 2^n, es decir, como 2^n-D
- el bit más significativo es siempre 0 para los números positivos y 1 para los negativos

Cálculo del "complemento a 2" de un número:

`complemento_a_2(D) =` 2^n`-D = complemento_bit_a_bit(D)+1`

- Ejemplo: representar -45 con 8 bits en "complemento a 2"

  $2^8 - 45 = 256 - 45 = 211_{10} = 11010011_2$

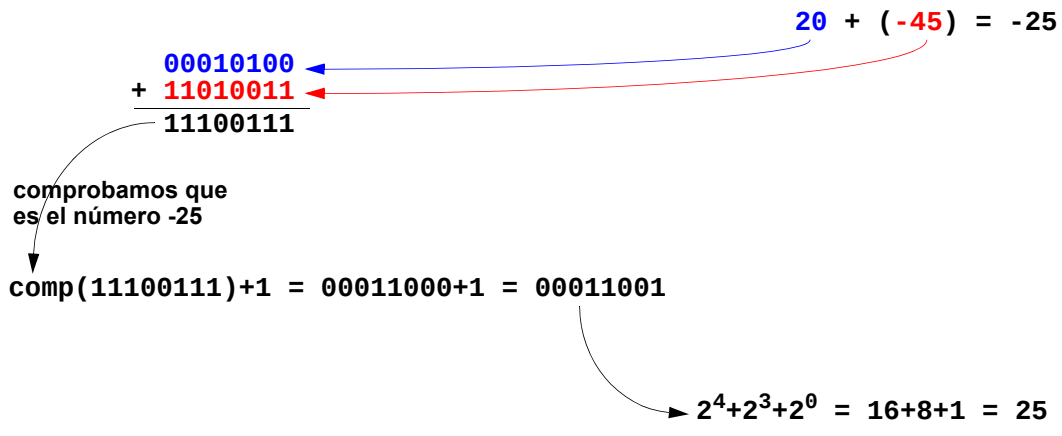
  $comp(45)+1 = comp(00101101)+1 = 11010010+1 = 11010011_2$

Representación binaria de números enteros con signo (cont.)

UNIVERSIDAD DE CANTABRIA

Las operaciones aritméticas producen resultados correctos para los número escritos en "complemento a 2"

Ejemplo:



Representación binaria de números enteros con signo (cont.)

UNIVERSIDAD DE CANTABRIA

Representación *exceso* 2^{n-1}

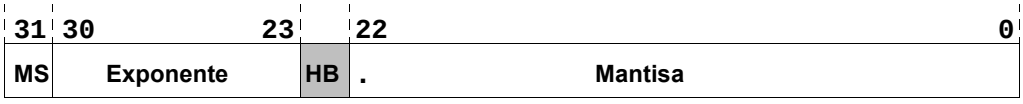
- cada número (positivo o negativo) se representa mediante el valor que resulta al sumarle 2^{n-1}
- los códigos están "ordenados": facilita las comparaciones
- los códigos de "exceso 2^{n-1}" coinciden con los de "complemento a 2", salvo en el bit más significativo

decimal	exceso 128	complemento a 2
-128	00000000	10000000
-1	01111111	11111111
0	10000000	00000000
1	10000001	00000001
127	11111111	01111111

Representación de números reales en punto flotante

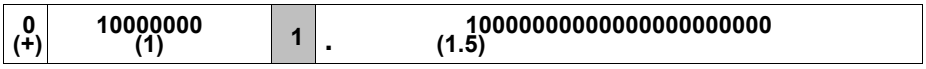
UNIVERSIDAD DE CANTABRIA

Formato propuesto por IEEE:



- Mantisa: código de 24 bits (incluye HB que siempre vale 1)
- Exponente: palabra de 8 bits codificada en "exceso 127" con dos valores reservados:
  - E = 00000000: indica que el número es el 0
  - E = 11111111: indica un valor numérico incorrecto
- Bit de signo de la mantisa (MS): positivo (0) o negativo (1)

Ejemplo: Codificación del número 3.0_{10}



$= 1.5 \cdot 2^1 = 3.0$

Representación física de datos en Ada y GNAT

UNIVERSIDAD DE CANTABRIA

Tipo *Character*:

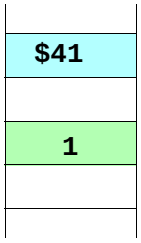
- Las variables de este tipo permiten almacenar el código ASCII de un carácter y ocupan un byte

Tipo *Boolean*:

- Las variables booleanas se almacenan en un byte de memoria, codificando los dos posibles valores como:

```
True => 1                    False => 0
```

```
C : Character;
B : Boolean;
...
C := 'A';

B := True;
```



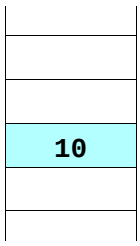
Representación física de datos en Ada y GNAT (cont.)

UNIVERSIDAD DE CANTABRIA

Tipos *enumerados*:

- ocupan 1, 2 o 4 bytes dependiendo del número de valores que definen
- cada valor se representa por un código numérico (por defecto se empieza a numerar desde 0, pero se puede cambiar)

```
type Semaforo is (Rojo, Amarillo, Verde);
for Semaforo use (Rojo => 4,
                  Amarillo => 10,
                  Verde => 12);
S : Semaforo;
...
S := Amarillo;
```



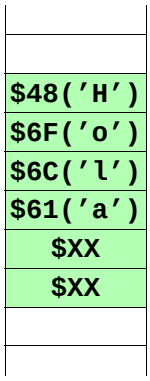
Representación física de datos en Ada y GNAT (cont.)

UNIVERSIDAD DE CANTABRIA

Tipos *String*:

- se representan como un array de caracteres sin incluir un byte para el tamaño o un carácter de fin de string
- por consiguiente ocupan tantos bytes como caracteres pueda contener el string

```
Str : String (1 .. 6);
...
Str (1 .. 4) := "Hola";
```



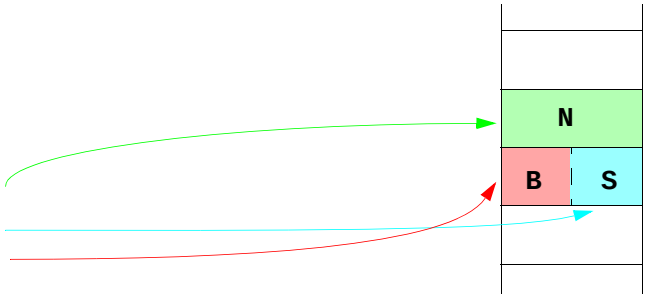
Representación física de datos en Ada y GNAT (cont.)

UNIVERSIDAD DE CANTABRIA

Tipos *registro*:

- de no indicar lo contrario, el compilador pueden decidir que ocupen más que la suma de los tamaños de sus campos
- pero si se desea, puede definirse exactamente los bits que ocupa cada uno de los campos y el tamaño total del registro

```
type Reg is record
  N : Unsigned_8;
  S : Semaforo;
  B : Boolean;
end record;
type Reg is record
  N at 0 range 0 .. 7;
  S at 1 range 0 .. 3;
  B at 1 range 4 .. 7;
end record;
for Reg'Size use 16;
```



Representación física de datos en Ada y GNAT (cont.)

UNIVERSIDAD DE CANTABRIA

Tipos *Enteros*:

- Es posible ajustar el tamaño de un tipo entero creado por el programador
  - El compilador avisa si el tamaño no es suficiente para ese rango de valores

```
type Mi_Entero is range 0 .. 255;
for Mi_Entero'Size use 8;  -- Tamaño en bits
```

- También se pueden utilizar los tipos definidos en el paquete `Basic_Integer_Types`:

```
Integer_8          Unsigned_8
Integer_16         Unsigned_16
Integer_32         Unsigned_32
Integer_64         Unsigned_64
```


Ejemplo: gestión del registro de control de un dispositivo

UNIVERSIDAD DE CANTABRIA

Registro de control de un dispositivo de E/S

Dirección E/S $222

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Ch3	Ch2	Ch1	Ch0	M1	M0	C	H
Puerta				Modo		Código	Habilita

Puerta	Ch3	Ch2	Ch1	Ch0
Canal 0	0	0	0	1
Canal 1	0	0	1	0
Canal 2	0	1	0	0
Canal 3	1	0	0	0

Modo	M1	M0
Reset	0	0
Escritura	0	1
Lectura	1	0
No def.	1	1

Código	C
Binario	0
BCD	1

Habilita	H
Habilitado	0
Deshabilitado	1

- es un registro de lectura y escritura
- el valor leído corresponde con el último valor escrito

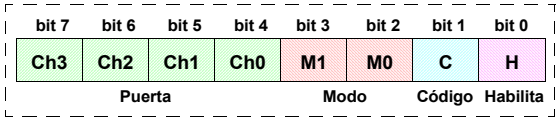
Ejemplo: gestión del registro de control de un dispositivo (cont.)

UNIVERSIDAD DE CANTABRIA

```
with MaRTE_OS;
with IO_Interface; use IO_Interface;
with Basic_Integer_Types; use Basic_Integer_Types;
with Ada.Unchecked_Conversion;

procedure Ejemplo_Registro_Control is
  REG_CTRL : constant IO_Port := 16#222#;

  type Puertas is (Canal_0, Canal_1, Canal_2, Canal_3);
  for Puertas use (Canal_0 => 2#0001#,
                   Canal_1 => 2#0010#,
                   Canal_2 => 2#0100#,
                   Canal_3 => 2#1000#);

  type Modos is (Reset, Escritura, Lectura);

  type Codigos is (Binario, BCD);
```


Ejemplo: gestión del registro de control de un dispositivo (cont.)

UNIVERSIDAD DE CANTABRIA




```
  type Registro_Control is record
    Puerta   : Puertas;
    Modo     : Modos;
    Codigo   : Codigos;
    Habilita : Boolean;
  end record;
  for Registro_Control use record
    Puerta   at 0 range 4 .. 7;
    Modo     at 0 range 2 .. 3;
    Codigo   at 0 range 1 .. 1;
    Habilita at 0 range 0 .. 0;
  end record;
```

Ejemplo: gestión del registro de control de un dispositivo (cont.)

UNIVERSIDAD DE CANTABRIA

```
function Lee_Registro_Control return Registro_Control is
  function RC is
    new Ada.Unchecked_Conversion (Unsigned_8,
                                  Registro_Control);
begin
  return RC (Inb (REG_CTRL));
end Lee_Registro_Control;


procedure Escribe_Registro_Control
                   (Ctrl : Registro_Control) is
  function U8 is
    new Ada.Unchecked_Conversion (Registro_Control,
                                  Unsigned_8);
begin
  Outb (REG_CTRL, U8 (Ctrl));
end Escribe_Registro_Control;
```


Ejemplo: gestión del registro de control de un dispositivo (cont.)

UNIVERSIDAD DE CANTABRIA

```
procedure Configura_Canal (P : Puertas) is
   Reg_Ctrl : Registro_Control;
begin
   Reg_Ctrl := Lee_Registro_Control;
   Reg_Ctrl.Puerta := P;
   Escribe_Registro_Control (Reg_Ctrl);
end Configura_Canal;

function Canal_Activo return Puertas is
   Reg_Ctrl : Registro_Control;
begin
   Reg_Ctrl := Lee_Registro_Control;
   return Reg_Ctrl.Puerta;
end Canal_Activo;

procedure Configura_Modo (M : Modos) is
   ...
```

Ejemplo: gestión del registro de control de un dispositivo (cont.)

UNIVERSIDAD DE CANTABRIA

```
--  Versión utilizando máscaras de bits
procedure Configura_Canal (P : Puertas) is
      Ctrl : Unsigned_8;
begin
  case P is
    when Canal_0 =>
      Ctrl := Inb (REG_CTRL);
      Ctrl := (Reg_Ctrl and 16#0F#) or 16#10#;
      Outb (REG_CTRL, Ctrl);
    when Canal_1 =>
      Outb(REG_CTRL,(Inb(REG_CTRL) and 16#0F#) or 16#20#);
    when Canal_2 =>
      Outb(REG_CTRL,(Inb(REG_CTRL) and 16#0F#) or 16#40#);
    when Canal_3 =>
      Outb(REG_CTRL,(Inb(REG_CTRL) and 16#0F#) or 16#80#);
  end case;
end Configura_Canal;
```


Ejemplo: gestión del registro de control de un dispositivo (cont.)

UNIVERSIDAD DE CANTABRIA

```
--  Versión utilizando máscaras de bits
function Canal_Activo return Puertas is
begin
  case Inb (REG_CONTROL) and 16#F0# is
    when 16#10# =>
      return Canal_0;
    when 16#20# =>
      return Canal_1;
    when 16#40# =>
      return Canal_2;
    when 16#80# =>
      return Canal_3;
    when others =>
      Put_Line ("Error leyendo registro de control");
      return Canal_1;
  end case;
end Canal_Activo;
```