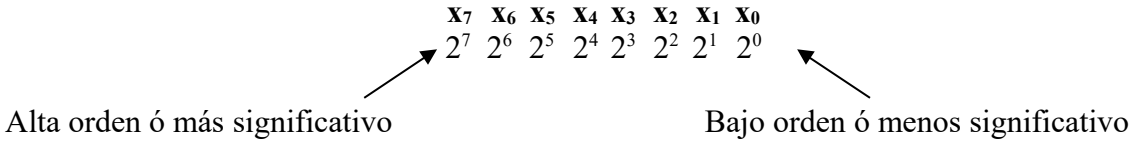
Representación de datos en formato Binario y Hexadecimal

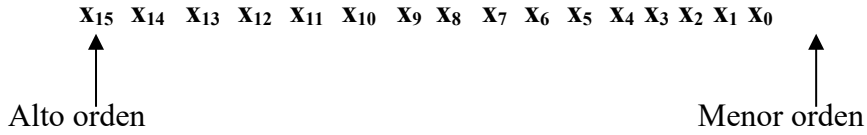
En esta lección se tratará el formato binario, así como la organización de los datos, además se revisarán las operaciones aritméticas básica en base binarias y hexadecimal para su tratamiento y uso en el Lenguaje Ensamblador. Finalmente se explicarán operaciones lógicas a nivel de bits.

Formatos Binarios (80x86)

A 8 bits :



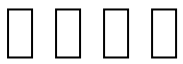
A 16 bits:



Organización de los datos (80 x 86)

1. 1 bit es la unidad de datos más pequeña.
2. 4 bits (nibble)
3. 8 bits (bytes)
4. 16 bits (palabra)

Nibble (4 bits) Sirve para representar números hexadecimales. (0-9 A F), además también se utiliza para representar números BCD (0...9)



Nibble= 4 bits

Byte (8 bits) En un 80 x 86 es el dato más pequeño direccionable. Se usa para direcciones de memoria y E/S

2^8 = 256 valores $2^8 - 1 = 0 \ldots 255$ (sin signo)
-128 ... 127 (con signo)

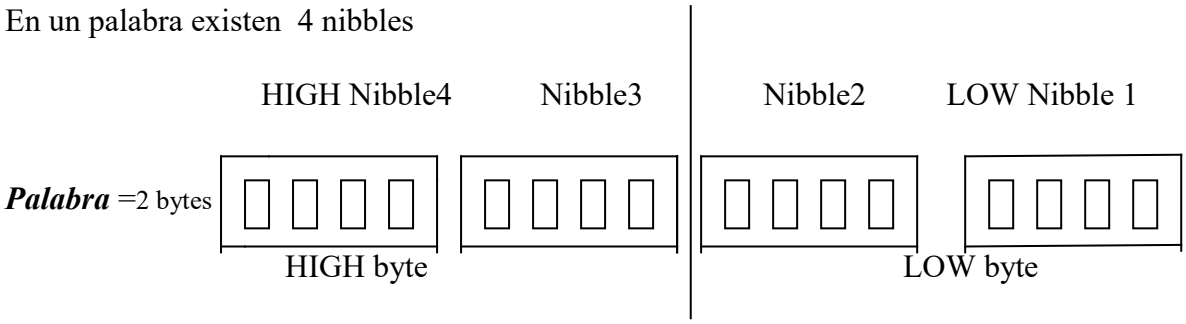
En un byte existen 2 nibbles

HIGH Nibble LOW Nibble

Palabra (16 bits) Se utilizan para valores enteros, offset, valores de segmentos los cuales constituyen el paragraph addres de un segmento de código extra o stack en memoria.

Representa 2^{16} =65, 536 valores 2^{16}-1= 0...65, 535 (sin signo)
-32,768 ... + 32, 767 (con singo)

En un palabra existen 4 nibbles



Doble palabra (32 bits) Se utilizan para representar segmentos de direcciones 8 nibbles, 4 bytes , 2 palabras.

Representa 2^{32} =4,294,467, 296 valores

Se utilizan para enteros de 32 bits, 2^{32}-1= 0... 4,294,467, 295 (sin signo)
–2,147483,648...+2,147,483,647 (con signo)

y valores de punto flotante.

Conversión decimal a binario

Notación:

Decimal $\longrightarrow$ 120_d ó 120_t
Binario $\longrightarrow$ 110_b ó 110_2
Hexadecimal $\longrightarrow$ $12\ F_H$

$X_t \longrightarrow X_2$ Divisiones sucesivas hasta que el cociente sea 0

$120_t \longrightarrow$ 01111000

60	30	15	7	3	1	0
2 | 120	2 | 60	2 | 30	2 | 15	2 | 7	2 | 3	2 | 1
2 | 00	00	10	1	1	1	1
0	0	0				

Notese que para formar el número binario se colocan los digitos a partir del cociente cero de izquierda a derecha, es decir el cociente cero es el bit más alto.

Conversión binario a decimal

$X_b \longrightarrow x_d$

$X_b \longrightarrow x_d$

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

$0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0 = 2^6 + 2^5 + 2^4 + 2^3 = 120$

Para hacer esta convesrisón solo basta obtener su valor de acuerdo a la posición de cada digito binario.

Conversión hexadecimal a binario / Binario a Hexadecimal derecha

Cada número binario representado a 4 bits es asociado con el correspondiente digito hexadecimal.

Binario	Hexadecimal	Binario	Hexadecimal
0000	0	0111	7
0001	1	1000	8
0010	2	1001	9
0011	3	1010	A
0100	4	1011	B
0101	5	1100	C
0110	6		

Binario	Hexadecimal
1101	D
1110	E
1111	F

El sistema hexadecimal es compacto y facilita la conversión de binario a hexadecimal y viceversa.

$1\ 2\ 7\ F_H \longrightarrow 0001\ \ 0010\ \ 0111\ \ 1111_b$

Operaciones aritméticas : Suma y Resta

Suma Binaria	***Suma Hexadecimal***
0 0 1 1 +0 +1 +0 +1 0 1 1 10 5d ⟶ 00000101_b +4d ⟶ 00000100_b 9d 00001001_b	0,1,2,3,4,5,6,7,8,9,A, B, C, D, E, F 9_h $+F_h$ 18_h

Decimal	Binario	Hexadecimal
89 d	01011001_b	59_h
+ 100 d	$+01100100_b$	64_h
	10111101_b	BD_h

89	44	22	11	5	2	1	0
2 | 89	2 | 44	2 | 22	2 | 11	2 | 5	2 | 2	2 | 1	
09	0	07	1	1	0	1	
1		0					

100	50	25	12	6	3	1	0
2 | 100	2 | 50	2 | 25	2 | 12	2 | 6	2 | 3	2 | 1	
0	0	05	0	0	1	1	
		1					

Decimal	Binario	Hexa
120d	$0111\ 1000_b$	78_h
+ 149d	$1001\ 0101_b$	$+ 95_h$
269d	$10000\ 0001_b$	$10D_h$

Resta Binaria	*Resta Hexadecimal*
0 0 1 1 -0 -1 -0 -1 0 1| 1 1 0 5_d 00000101 $_b$ - 4_d - 00000100 $_b$ 1_d 00000001 $_b$	0,1,2,3,4,5,6,7,8,9,A, B, C, D, E, F

Decimal	Binario	Hexa
99_d	- 01100011	63_h
- 50_d	00110010	- 32_h
49	00110001	31_h

Decimal	Hexa	Binario
1024d	400_h	
- 820 d	- 334_h	0000 1100 1100
	$0CC_h$	

Operaciones lógicas en bits

And	
00	0
01	0
10	0
11	1

or	
00	0
01	1
10	1
11	1

xor	
00	0
01	1
10	1
11	0

NOT	
0	1
1	0

NOTA:

El uso de AND Y OR nos per miten cambiar un bit a cero o a uno.
El operador XOR nos permite invertir bits.
El operador NOT niega

Las Operaciones lógicas son utilizadas en el manejo de strings y para la generación de mascaras.

Para realizar máscaras se utilizan operadores lógicos and, or, xor, not.

Números negativos
Bit mas alto es 1.

Ejemplos 16-bits

8000_h es negativo 1000 0000 0000 0000 $= -32768_d$
100_h es positivo 0000 0001 0000 0000
7FFFh es positivo 0 111 1111 1111 1111 $= +32767_d$
$0FFFF_h$ es negativo 1111 1111 1111 1111 $= -1$
$0FFF_h$ es positivo 0000 1111 1111 1111

Para convertir un número positivo a su correspondiente negativo se utiliza la operación complemento a 2.

Algoritmo
1. Invertir todos los números con operador lógico NOT
2 .Agregar uno al resultado invertido

Ejemplo 1:a 8 bits el número 5 a –5

	0000	0101	$\rightarrow$ 05_h	$+5_d$
P1.	1111	1010		
P2.	0000	0001		
	1111	1011	$\rightarrow$ FB_h	-5_d

Convertir un negativo a un positivo utilizar complemento a 2

Ejemplo 1: Convertir -5 a +5 con 8 bits

	1111	1011	$\rightarrow$	-5_d
P1.	0000	0100		
P2.	0000	0001		
	0000	0101	$\rightarrow$	$+5_d$

Esto es un error del tipo “signed arithmetic over flow”

8000_h es negativo 1000 0000 0000 0000 $= -32768_d$
Complemento a 2: 0111 1111 1111 1111
\+ 1
1000 0000 0000 0000 $= +32768_d$

No es posible que -32768=-(-32768). No se puede representar a 16 bits para números con signo.

Conclusión: No se puede negar el valor negativo más pequeño si lo intenta el microprocesador demanda un error de “aritmética de signo overflow”.

Signo y Extensión cero

Extensión cero permite convertir un número pequeño con signo a un largo con signo.
Regla: Si es un número negativo en la parte HIGH byte debe contener OFF H.
Si es un número positivo en la parte HIGH byte debe contener 000H.

* *Extensión con signo de un número de 8 bits a 16 bits*: Copiar los 8 bits a los 8 bits en las posiciones más bajas de los 16 bits y aplicar una extensión cero.

Ejemplo: considere -64_d a 8 bits número.

Número con signo

$64_d = 0100\ 0000$
$-64_d = 1100\ 0000$

Convertir de 8 bits a 16 bits

$-64_d \longrightarrow CO_H \longrightarrow FFC0_H$

$64_d \longrightarrow 40_H \longrightarrow 0040_H$

* Extensión con signo de un número de 16 bits a 32 bits: Copiar los 16 bits a la posición más bajas y aplicar extensión cero.

Ejemplos: Números con signo

8 bits	16 bits	32 bits
(-) 80_H	$FF80_H$	$FFFFFF80_H$
(+) 28_H	0028_H	00000028_H
(-) $9A_H$	$FF9A_H$	$FFFFFF9A_H$
(+) $7F_H$	$007F_H$	$0000007F_H$
	8 bits	16 bits

Números sin signo

8 bits	16 bits	32 bits
80 H	0080_H	00000080_H
28H	0028_H	00000028_H
$9A_H$	$009A_H$	$0000009A_H$
$7F_H$	$007F_H$	$0000007F_H$

¿Se puede realizar una conversión de un número 16 bits a 8 bits o de 32 bits a 16 bits.?

Resp:= No siempre es posible

Ejemplos:

$BAC1_H$ ⟶ no se puede porque los bits más altos y más bajos están ocupados.
16 bit 8 bit

-448_d a 16 bits $0FE40_h$ No se puede

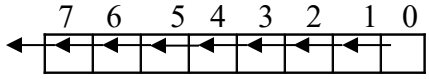
Regla: Para contraer signo de un valor solo hay que remover los bits más altos si contienen 0's o 0FF o 0FFFF, si no los contiene se produce un overflow

Corrimientos y Rotaciones

Es otro tipo de operaciones lógicas la cual se aplica a cadenas de bits, existen hacia la :

* Izquierda
* Derecha

Corrimiento a la izquierda



Colocar un cero en el bit más bajo y el valor del bit más alto es un accareo

Este tipo de corrimiento permite generar una multiplicación por 2 (radix base 2). En general si corres un valor a la izquierda n veces se multiplica por 2^n .

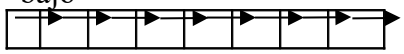
Ejemplo:

			Acarreo	Resultado
$00101010 = 2A_h$	por $2^1 = 54_h$	1 Corrimiento	0	01010100
$00101010 = 2A_h$	por $2^2 = A8_h$	2 Corrimiento	0	10101000
$00101010 = 2A_h$	por $2^3 = 150_h$	3 Corrimiento	1	01010000

**** Corrimiento a la derecha***

7 6 5 4 3 2 1 0

bajo



Coloco un cero en el bit más alto y el bit más bajo es un accareo

Este tipo de corrimiento permite generar una división por 2 (radix base 2). En general si corres un valor a la derecha n veces se divide por 2^n . (Solo es válido para divisiones de números sin signo)

Ejemplo:

			Resultado	Acarreo
11111110 =FE_h	entre $2^1 = 7F_h$	1 Corrimiento	01111111	0

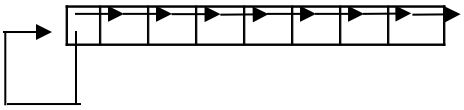
Notese que para valores con signo este tipo de operación no resulta, por ejemplo:

-2= $0FE_h$, al realizar un corrimiento, se obtiene $07F_h=127_h$, esto no es correcto.

El problema se originó al insertar el 0 en el bit del signo, cambiándolo a positivo.

Para este tipo de operaciones se define el Corrimiento aritmético a la derecha.

**** Corrimiento aritmético a la derecha***



No modifica el bit 7 durante el corrimiento

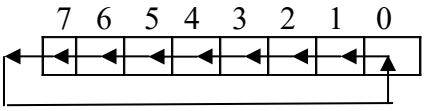
Ejemplo:

			Resultado
-2=11111110 =FE_h	entre $2^1 = FF_h$	1 Corrimiento	11111111
-100=10011100=$9C_h$	entre $2^1=CE_h$	1 Corrimiento	11001110

Esta operación redondea el número al entero más cercano, el cual es “menor o igual al actual resultado”. Por ejemplo:

-1=11111111 =FF_h	entre $2^1 = FF_h$	1 Corrimiento	11111111 = -1 < 0

**** Rotación por la izquierda***

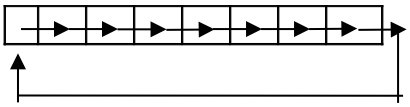


El bit más alto es corrido al más bajo

ejemplo: 01101011

11010110

**** Rotación a la derecha***



El bit más bajo es corrido al más alto

Ejercicios de Repaso - Formatos binarios.
(no se enetregan)

Intrucciones:
Lea detenidamente y realice lo que se indica en cada ejercicio.

1. Convertir

$32{,}768_d \longrightarrow x_b$ $888d \longrightarrow x_b$ $1001_b \longrightarrow x_d$

2. Complementar a dos (considere que son números con signo).
Primeramente determine si es positivo o negativo el número de acuerdo a su longitud, en caso de ser positivo obtenga su negativo, si es negativo encuentre su representación positiva. Los valores hexadecimal conviertelos a binario.

$1001\ 0111_b \longrightarrow x_d$ $\longrightarrow$
$0ABCD_H \quad X_b$
$0FEBA_H \longrightarrow x_b$

3. Realizar las sig. operaciones

$1232_H + 9876_H$
$0FF_H - 0F34_H$
$100_b - 1b$
$0FFE_H - 1_H$

4. Realizar la extensión de signo de 8-16 bits a los números listados en cada columna (considere que son números con signo)

FF	DE
BF	F7
0E	DE
8F	FD
54	AD

5. Aplicar los operadores lógicos a todos los números hexadecimales del ejercicio 4. El operador not solo a la columna 1, para las demas operaciones defina como operandos a cada una de las columnas de la siguiente forma operando 1= columna 1, operando 2= columna 2.

6. Contraer el signo para los siguientes valores listados en cada columna de 16 a 8 bits. Si no lo puede realizar explique por qué.

$FF0C_H$	$7F_H$	12_H
8080_H	$FF98_H$	98_H
67_H	$F98_H$	

7. Aplicar dos rotaciones a la izquierda a los números de la primera columna del ejercicio 4, considere que son números de 16 bits

8. Aplicar un corrimiento a la derecha o corrimiento aritmético a la derecha según sea el caso a los números de la segunda columna del ejercicio 4, considere que son números de 16 bits (recuerde que son números con signo).