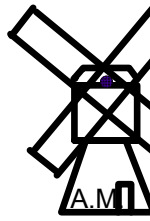
Data and Data Representation

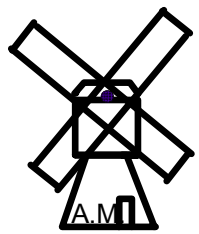
Ir. Andries Mulder
Mulder Training & Consultancy
the Netherlands
gsm. +31 651 71 40 00
tel. +31 544 37 40 37
internet driesmulder@kpnmail.nl



There are 10 types of people:

Those who can read binary and those who can't

A. Mulder
driesmulder@kpnmail.nl



Bits and Bytes

- The unit of data in computer systems is called a **BYTE**
- In most systems a byte represents 1 character
  eg. 'A' or 'b' or '9' or '&' or '%' etc.
- Not all bytes can de displayed as a regular character
- A Byte consist of 8 **BITS**
- A **BIT** can have the value 0 (OFF) or 1 (ON)

A. Mulder
driesmulder@kpnmail.nl

Binary Representation of Bytes

- As a byte consist of 8 bits. We can write out all the bits in order to show the value of a byte.

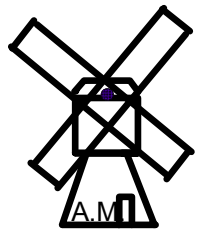
  eg. **01000001**

  We call this the **Binary Representation.**

- As you can imagine, writing out all the bits is very im-practical and therefore is not used very much.

- There are potential 256 unique combinations possible in 1 byte.
  From **00000000** to **11111111**

A. Mulder
driesmulder@kpnmail.nl



Hexa-Decimal Representation of Bytes (1)

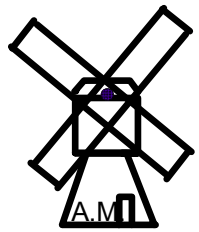
- Another way to represent bytes is the so called

  Hexa-Decimal representation

- This representation takes **two characters per byte** and it can show every byte between 00000000 and 11111111

- So if you want to show 20 bytes in "Hex" it will take 40 bytes

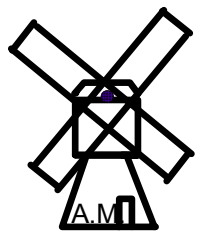
A. Mulder
driesmulder@kpnmail.nl



Hexa-Decimal Representation of Bytes (2)

- A byte consist of 8 bits.
- We are looking for a better way to represent bytes (better than the "ugly" bits representation)
- We divide the byte in 2 halves (Left, Right).
  We call those halves "**Nibbles**"
  So a Nibble consist of 4 bits
- Here is an example of a Nibble: **"0100"**
- A Nibble can have 16 different values

A. Mulder
driesmulder@kpnmail.nl



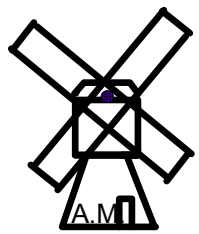
Hexa-Decimal Representation of Bytes (3)

- We have divided the byte in 2 Nibbles
- We now assign a position based value to each of the bits in a nibble

Rightmost bit has a value of	1
One bit to the left has value	2
One more bit to the left has value	4
Leftmost bit of the Nibble has value	8

- Example: "0011" ---> 1 + 2 ---> 3
  "0111" ---> 1 + 2 + 4 ---> 7

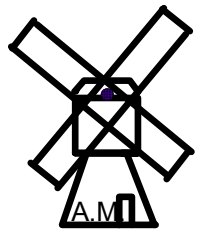
A. Mulder
driesmulder@kpnmail.nl



Hexa-Decimal Representation of Bytes (4)

- For the Nibble value, we use the characters **'0'..'1'...'9' 'A'...'F'**
- Nibble Values from 0 ..9 will become '0' to '9'
- Nibble Values from 10 .. 15 will become 'A' to 'F'
- A byte in Hex can run vary from 00 to FF

A. Mulder
driesmulder@kpnmail.nl



Hexa-Decimal Representation of Bytes (5)

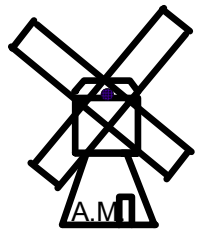
- By convention, a number prefixed by 0X
  will denote a hexadecimal number
  Also you can use 0x as a prefix for hexadecimal

- We indicate that it is "HEX" by writing 0XCC
  or 0xcc where c or C is a char between 0 and F
  and the 0X or 0x tells it is hex.

- So 10100101b is written as 0XA5 in hex

  and 11111111b is written as 0XFF in hex (or as 0xff)

A. Mulder
driesmulder@kpnmail.nl



Hexa-Decimal Representation of Bytes (6)

A few examples

10100101 ---> 0XA5
11111111 ---> 0xff (or 0XFF)
01000001 ---> 0X41
01000010 ---> 0X42
10100111 ---> 0xa7

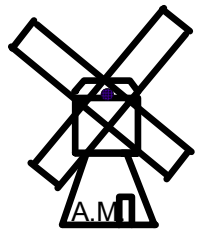
As you can see we don't make difference between lowercase and UPPERCASE.

A. Mulder
driesmulder@kpnmail.nl

Nibbles in Decimal, Hex and Binary

Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

A. Mulder
driesmulder@kpnmail.nl



Hexa-Decimal Representation of Bytes (7)

Representing larger groups of bytes

Byte strings are just prefixed once by 0X (or 0x)
eg. 1010 1011 0001 0010b ---> 0xAB12

A. Mulder
driesmulder@kpnmail.nl

Hexa-Decimal Representation of Bytes (8)

Hex dumps

See next example of a hex dump of 64 bytes

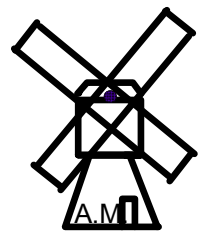
```
3ØD4E5DD B6B0527C 69D7816E 46D5C391 [        R|i   nF
588CD2BD 9C879BE1 6F03FBDØ 6A23A44A [ X        o    j# J
DBA9D729 Ø83B5B2Ø 247F2751 2EA6FA29 [     ) ;[ $ 'Q.   )
DA839BD2 25B3786A BDD845BC 84E7CFCB [      % xj  E
```

A. Mulder
driesmulder@kpnmail.nl

Hexa-Decimal Representation of Bytes (9)

There are many utility programs that support hex dumping

- Total Commander (Win)
- Midnight Commander (Linux)
- UltraEdit (Win + Linux)
- hd (Linux)
- hexdump (Linux)
- hexdump (Windows)



A. Mulder
driesmulder@kpnmail.nl

What is the meaning of a byte ?

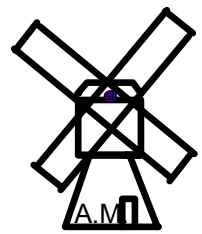
ASCII = American Standard Code for Information Interchange
ASCII is used on PC's and some UNIX systems

EBCDIC = Extended Binary-Coded Decimal Interchange Code
EBCDIC is used on IBM mainframes and on IBM system-i and on IBM system-p

Both are systems that give a meaning to a byte

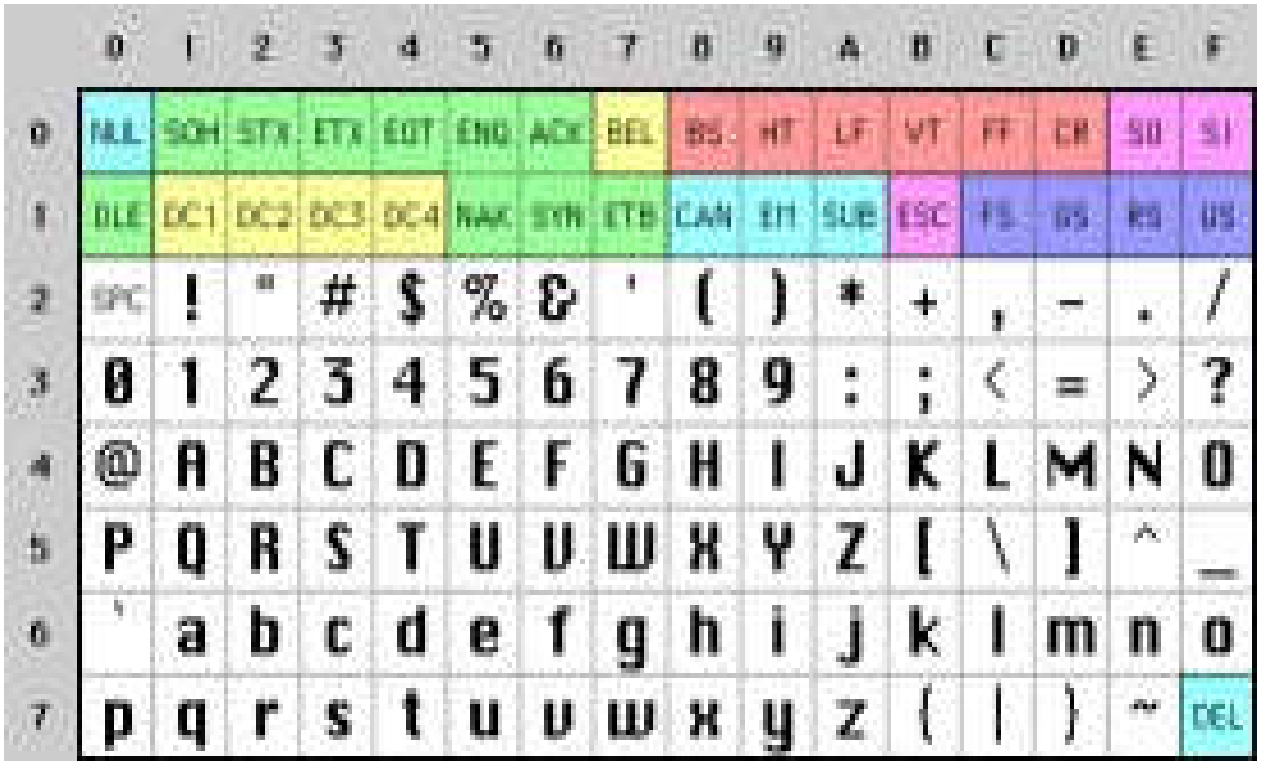
```
eg.   In ASCII     ØX41 ---> "A"
      in EBCDIC    ØXC1 ---> "A"
```

A. Mulder
driesmulder@kpnmail.nl



ASCII and EBCDIC (1)

Original ASCII character table (7 bits)



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SPC	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

A. Mulder
driesmulder@kpnmail.nl

ASCII and EBCDIC (2)

ASCII
CP 437
(8 bits)

ASCII
CP 850
(8 bits)

A. Mulder
driesmulder@kpnmail.nl

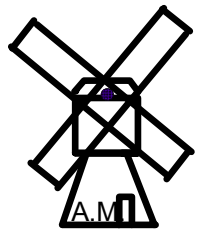
ASCII and EBCDIC (3)

EBCDIC = Extended Binary-Coded Decimal Interchange Code
EBCDIC was invented by IBM an is (unfortunately) still used on many systems

For an EBCDIC table, see the Internet.

On Internet, you can also find conversion tables from ASCII to EBCDIC v.v.

A. Mulder
driesmulder@kpnmail.nl



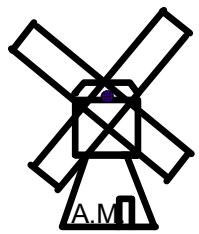
Numbers and how they are stored (1)

When we write "1234" in **DECIMAL** we mean:

4 x 1	+
3 x 10	+
2 x 100	+
1 x 1000	= 1234

So we use powers of 10 and the Leftmost number is the most significant.

A. Mulder
driesmulder@kpnmail.nl



Numbers and how they are stored (2)

When a computer stores **0X 12 34** it means

4 x 1 +
3 x 16 +
2 x 256 +
1 x 4096 = 14660 (decimal)

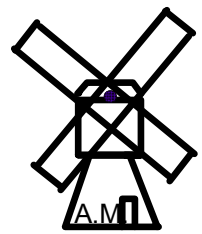
BIG ENDIAN

OR 2 x 1 +
1 x 16 +
4 x 256 +
3 x 4096 = 13330 (decimal)

LITTLE ENDIAN

It uses powers of 16

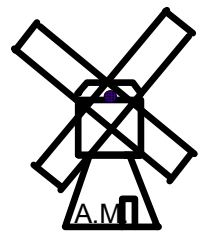
A. Mulder
driesmulder@kpnmail.nl



Numbers and how they are stored (3)

- Numbers can be 8 bits, 16, 32 or 64 bits.
- We can have signed numbers and unsigned numbers.
- For signed numbers the leftmost bit is the sign-bit
  When set, it indicates a negative number.
- You as a programmer decide whether a number
  is signed or unsigned.

A. Mulder
driesmulder@kpnmail.nl



Numbers and how they are stored (4)

How does the computer store a Signed Number ?

- Construct the number as positive number
- Invert all the bits
- Add 1

```
eg. We want to code  -93
We write 93          01011101b
Invert it            10100010b
Add 1                00000001b
Results in -93       10100011b
```

A. Mulder
driesmulder@kpnmail.nl

Base64 Encoding and Decoding

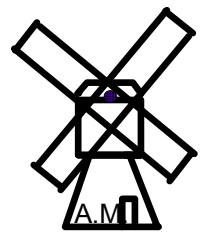
- Hex coding takes twice as much bytes as binary
- A more (space) efficient way of encoding is Base64

ase64 uses ABCDEFGHIJKLMNOPQRSTUVWXYZabcdef
ghijklmnopqrstuvwxyz0123456789+/

nd (as a special) the "=" character

Please check the 64 chars

A. Mulder
driesmulder@kpnmail.nl



An Example of Base64

-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDag84L+pT6OlFZ8kav08ixPdug
o6bsqOICJrl8TzGY87Iek9K1dgbz2sr7r0OBl2/ndz0CoxiyKoiLc5YY6wy5Qg1q
NRCc4C71r2A7N+vjhfkSSoS7RQekuhKSMJ1Wp8RfXB/AccWPdqb0Mm1TjklSizpt
JAANppC12fijFVTocQIDAQAB
-----END PUBLIC KEY-----

A. Mulder
driesmulder@kpnmail.nl

Base64 usage

- Base64 is used to represent PGP keys and in email protocols like Mime
- It is more space efficient than Hexadecimal
- It is much more difficult to read than Hexadecimal
- You can find online converters for Base64
- You can find example implementations of Base64 Encoding / Decoding

A. Mulder
driesmulder@kpnmail.nl