Il file binary `"compito.dat"` contiene record da 16 byte il cui tracciato è il seguente:

```
struct Rec {
   int              x;          //--- 4 byte
   float            y;          //--- 4 byte
   char             W[4];       //--- 4 byte
   unsigned short int z;        //--- 2 byte
   short int        t;          //--- 2 byte
};
```

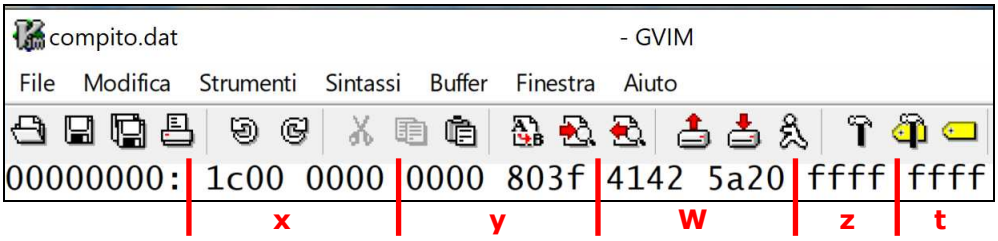
Sapendo che la codifica ASCII di

'**A**' è **41_H** (in esadecimale)

e di

'**a**' è **61_H**

Quali valori sono memorizzati nel record riportato qui in basso?



Si procede alla **inversione** dei campi **numerici** multi-byte (memorizzati dai processori Intel con il sistema ***little endian***) e poi allo spacchettamento in binario di ogni cifra esadecimale:

x	y	W	z	t
00 00 00 **1C**	3F 80 00 00	41 42 5A 20	FF FF	FF FF
0000 … 0001 1100	**0**011 1111 **1**000 0000 …	A B Z spazio	1111 … 1111	**1**111 … 1111
valore **positivo**	valore **positivo**	**"ABZ "**	valore più alto	val. **negativo**

x = 11100_2 = $2^2 + 2^3 + 2^4$ = 4 + 8 + 16 = **28_{10}** (un valore positivo va solo convertito in base 10)

x = +28

y è un numero **reale positivo**; il segno è **+** perchè il primo bit è **0**,

l'esponente in eccesso 127 è *01111111* = 1 + 2+ 4 + 8 + 16 + 32 + 64 = 7 + 40 + 80 = **127_{10}**

l'esponente quindi è **0** =**127 -** 127

la mantissa è **1,**0000000… (va inserito 1 prima della virgola, NON è memorizzato in IEEE-P574-SP)

il valore rappresentato nei 4 byte relativi al dato **y** è: **$+1,0 \times 2^0 = +1,0_2 = +1,0_{10}$**

y = +1,0

z è un valore senza segno (unsigned), ed è il valore più alto rappresentabile con 2 byte (=16 bit), cioè

z = 65.535 = $2^{16} - 1$ = 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128

(16 bit a 1) + 256 + 512 + 1024 + 2048 + 4096 + 8192 + 16384 + 32768

t è un valore negativo, va quindi determinato il suo complemento a 2:

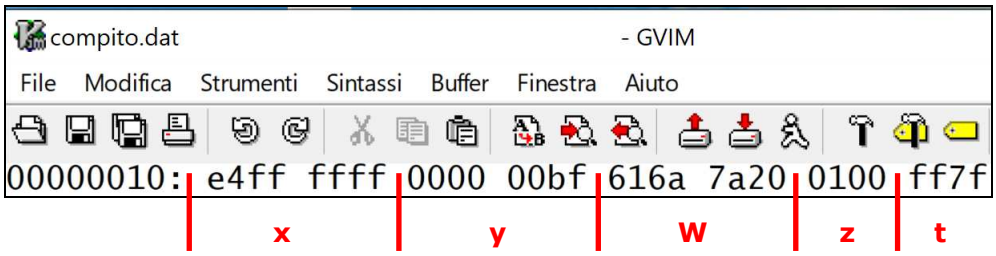
1111 1111 1111 1111 determinare la rappresentazione del suo opposto in CA2:

0000 0000 0000 $0001_2 = 1_{10}$ convertire in base 10; quindi:

t = -1

W contiene i caratteri **"ABZ "** (NOTA: dopo la **Z** c'è uno **spazio**, non il terminatore di stringa – NULL - '\0' - perchè si sono valorizzate le singole posizioni dell'array di char)

Analizziamo il secondo record:



Inversione dei campi **numerici** multi-byte memorizzati dai processori Intel con il sistema ***little endian***:

x	y	W	z	t
FF FF FF **E4**	BF 00 00 00	61 6A 7A 20	00 01	7F FF
1111 … 1110 0100	**1***011 1111* ***0***000 0000 …	a j z spazio	0000 … 0001	**0**111 … 1111
valore **negativo**	valore **negativo**	**"ajz "**	**1_{10}**	val. **positivo** più alto

x è un valore negativo, va quindi determinato il suo complemento a 2

1111 …… **1110 0100** determinare la rappresentazione del suo opposto in CA2:

0000 …… **0001 1100_2** = convertire in base 10:

= 4 + 8 + 16 = **28_{10}** quindi:

x = -28

y è un numero **reale negativo** : il segno è **-** perchè il primo bit è **1**,

l'esponente in eccesso 127 è *0111111**0*** = 2+ 4 + 8 + 16 + 32 + 64 = 6 + 40 + 80 = **126_{10}**

l'esponente quindi è **-1** = **126 -** 127

la mantissa è **1,**0000000… (va inserito 1 prima della virgola)

il dato **y** è: **-1,0 x 2^{-1} = -$0,1_2$** = -1 x 2^{-1} = -1 x ½ = **-$0,5_{10}$**

y = -0,5

z = 1

t è un valore positivo, va quindi solo convertito in base 10

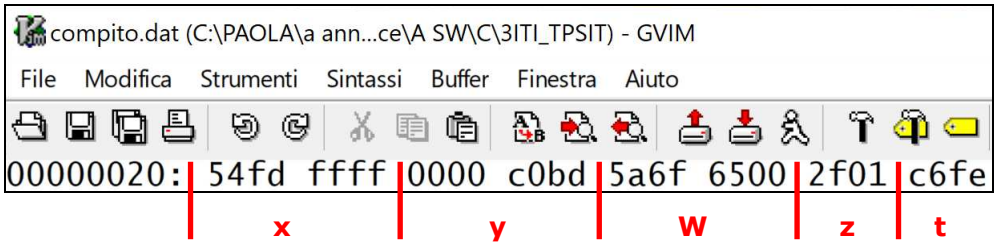
0111 1111 1111 1111 = 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128

(15 bit a 1) + 256 + 512 + 1024 + 2048 + 4096 + 8192 + 16384 = **32.767_{10}** = 2^{15} –1

t = +32.767

W contiene i caratteri **"ajz "** (NOTA: dopo la **z** c'è uno **spazio**, non il terminatore di stringa – NULL - '\0' - perchè si sono valorizzate le singole posizioni dell'array di char)

Analizziamo il terzo record:



Inversione dei campi **numerici** multi-byte memorizzati dai processori Intel con il sistema ***little endian***:

x	y	W	z	t
FF FF FD 54	BD C0 00 00	5A 6F 65 00	01 2F	FE C6
1111 … 0101 0100	1011 1101 1100 0000 …	Z o e NULL	0000 … 1111	1111 … 0110
valore **negativo**	valore **negativo**	**"Zoe"**		val. **negativo**

x è un valore **negativo**, va quindi determinato il suo complemento a 2

1111 …… 1111 **1101 0101 0100** determinare il suo opposto in CA2:

0000 …… 0000 **0010 1010 1100** = convertire in base 10:

= 4 + 8 + 32 + 128 + 512 = 4 + 40 + 640 = $\mathbf{684_{10}}$ quindi:

x = -684

y è un numero **reale negativo** : il segno è **-** perchè il primo bit è **1**,

l'esponente in eccesso 127 è *01111011* = 1 + 2 + 8 + 16 + 32 + 64 = 3 + 40 + 80 = $\mathbf{123_{10}}$

l'esponente quindi è **-4** = **123 -** 127

la mantissa è **1,** 1000000… (va inserito 1 prima della virgola)

il valore del dato **y** è: $\mathbf{-1{,}1 \times 2^{-4} = -0{,}00011_2}$ = -(1 x 1/16 + 1 x 1/32) = -(0,0625 + 0,03125)

y = -0,09375

z è un valore senza segno, va convertito in base 10 (anche direttamente da esadecimale)

$12F_H = 15 \times 16^0 + 2 \times 16^1 + 1 \times 16^2 = 15 + 32 + 256 = 303$

z = 303

t è un valore **negativo**, va quindi determinato il suo complemento a 2

1111 1110 1100 0110 determinare la rappresentazione del suo opposto in CA2:

0000 0001 0011 1010 = 1_{10} convertire in base 10; quindi:

= 2 + 8 + 16 + 32 + 256 = 10 + 48 + 256 = 314

t = -314

W contiene il nome **"Zoe"** (NOTA: array delimitato dal terminatore di stringa - NULL - **'\0'**)