Binary Coded Decimals

Binary Coded Decimals

BCD	DECIMAL
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

Represent each decimal digit using a 4 bit code

923_{10}
$(9)\ (2)\ (3)_{10}$
(1001) (0010) (0011)
100100100011_{BCD}

Encoding

Uncompressed - 8 bits used. 4 for BCD, other 4 bits unused

- i.e. Byte pattern for '25'
  00000010 00000101

Packed - 2 BCD numbers packed into a byte

- i.e. Byte pattern for '25'
  00100101

Signs

BCD	HEXADECIMAL
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Credit: +
Debit: -

Signs are placed at the end of the number:

-327 is encoded as 0011 0010 0111 1101 = (3 2 7 -)

Signs

BCD	HEXADECIMAL
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Exercises:

What values are represented by the following BCD?

1) 0111 0010 1100
2) 0010 0101 1101
3) 0111 0010

Range

What is the largest number that can be represented by a 32 bit BCD?

Compare with Unsigned Integer which can represent
0 to 4,294,967,295

Fractions

Implied Fixed Point - Decimal point is not stored with data - BCD format does not have value representing decimal point. Decimal points are "implied" by programmers/programs and can be placed anywhere.

i.e. -32.5 is stored as 0011 0010 0101 1101. It is up to the program to translate that value as -32.5 instead of -325

Fractions

Encode 25 into BCD:

0010 0101

Encode 2.5 into BCD:

0010 0101

Encode .25 into BCD:

0010 0101

Why BCD?

Easy translation of a value to characters for display.

i.e. 234 $(0010\ 0011\ 0100\ 1100)_{BCD}$ can be translated quickly because it is a one to one relationship between the numerals 0-9.

Corresponding two's complement encoding: 11101010_2 takes much longer.

Fractions

Can represent fractions easily.

Exercises:

Encode 0.2 into Unsigned Binary.

Encode 0.2 into BCD

Downsides

Arithmetic is more difficult:

1001 (9)
+ 1000 (8)
0001 0001 (11)

Requires +6 correction

Downsides

Requires more space to store values.
i.e.
255

= $1111\ 1111_2$ or $(0010\ 0101\ 0101)_{BCD}$