Review: Data representation in Digital systems

I: Bits and words

Models and basic principles

- Two Voltage levels:
  - High: V> Vhmin ➔ Symbol 1
  - Low: V < Vlmax ➔ Symbol 0
- Power Supplies (rails):
  - VCC: is also usually maximum value for High
  - VSS (may be ground): is also usually minimum value for low
- We represent actions using relationships between “1”’s and “0”’s

Basic definitions (1/2)

- Bit: 1 or 0
  - A bit variable can only have a value 1 or a value 0
- n-bit word: an ordered sequence of n bits
  - Examples: 00101, 1100, 11011100
  - General Representation: $b_{n-1}\ b_{n-2} \ldots b_1\ b_0$
- Most significant bit (msb or MSB): Leftmost bit, b_{n-1}
- Least significant bit (lsb or LSB): Rightmost bit b_0

Basic definitions (2/2)

- Specific word names:
  - Nibble: 4-bit word
  - Byte: 8-bit word
  - Word: 16-bit word
  - Double word (long): 32-bit word
  - Quad: 64-bit word
- The terms least significant byte (LSB), most significant byte (MSB), least significant nibble (LSN), etc. are used similar to previous definition
- Notice ambiguity in use of "word". Must be specific when necessary.

Representing with words (1/2)

- We can only use 0’s and 1’s to represent all sort of data
- Meaning of an n-bit word is context dependent
  - The word may be interpreted as a whole, bit by bit or by groups of bits.
- When necessary, we use more than one n-bit word

Representing with words (2/2)

Basic principle

With n-bit words we can represent at most

$$2\text{\^{}}n = 2^n$$

different elements.

Special cases

- Nibbles: 2^4 = 16 cases
- Bytes: 2^8 = 256 cases
- Words: 2^16 = 65,536
- Double word: 2^32 = 4,294,967,296
- Quad: 2^64 =

18,446,744,073,708,271,616

(> 1.844 x 10^{19})

Powers of two and definitions (1/2)

N	2^N	N	2^N	N	2^N
0	1	11	2048	21	2097152
1	2	12	4096	22	4194304
2	4	13	8192	23	8388608
3	8	14	16384	24	16777216
4	16	15	32768	25	33554432
5	32	16	65536	26	67108864
6	64	17	131072	27	134217728
7	128	18	262144	28	268435456
8	256	19	524288	29	536870912
9	512	20	1048576	30	1073741824
10	1024	21	2097152	31	2147483648

Powers of two and definitions (2/2)

- 1 Kilo (1K) = 2^10 = 1 024
  - Example: 16K = (2^4)(2^10)=2^14
- 1 Mega (1M) = 2^20 = 1 048 576
  - Example: 4M = (2^2)(2^20) = 2^22
- 1 Giga (1G) = 2^30 = 1 073 741 824
  - Example: 8G = (2^3)(2^30)=2^33
- 1 Tera (1T) = 2^40 = 1 099 511 627 776
- <u>NOTE:</u> When speaking of hard drives, the powers are of ten (1K=10^3, 1M=10^6, etc.)

Review: Data representation in Digital systems

I: Binary Numbers

Positional System Base *r* : Principles (1 /4)

- It has *r* digits: 0, 1, 2, …. “*r* – 1”, ordered in value:
  - **Binary (base 2): 0, 1**
  - **Decimal (base 10): 0, 1, 2, 3 , 4, 5, 6, 7, 8, 9**
  - **Hexadecimal or Hex (base 16): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**
  - **Octal (Base 8): 0, 1, 2, 3, 4, 5, 6, 7**
  - Base 5: 0, 1, 2, 3, 4
- Digits 0, 1, .. 9 have same meaning. A = ten, B= eleven, …..

Positional System Base *r* : Principles (2/4)

- Any number written as an ordered sequence of the *r* digits including a decimal or radix point
  - Differentiate system with subscript:
    - 1011.01_2, $A251.F2_{16}$, 697.45_{10}, etc.
  - Left of the point: **Integer part**;
  - Right of the point: **fractional part**
- Numbers generated as usual
  - Decimal 0, 1, .. 9, 10, 11, .. 19, 20,…. 99, 100, 101
  - Binary: 0, 1, 10, 11, 100, 101, …
  - Hex: 0, 1, .. F, 10, 11, .. 19, 1A,….FF, 100, 101, … FFF

Positional System Base *r* : Principles (3/4)

- Suffix or prefix instead of subscript for main systems used in digital systems:
  - Decimal: No suffix ➔ 697.45
  - Binary: Suffix **b** or **B** ➔ 1011.01b, 1011.01B
  - Hex: Suffix **h**, **H**, or prefix **0x** ➔ A251.F2h, 0xA251.F2
  - Octal: Suffix **q, Q** ➔ 217.36q, 217.36Q

Positional System Base r : Principles (4/4)

- General representation

$$a_{n-1}a_{n-2}....a_1a_0.a_{-1}a_{-2}....a_{-m}$$

- Most significant digit: leftmost digit a_{n-1}
- Least significant digit: rightmost digit a_{-m}
- Integer part (left of point) $a_{n-1}a_{n-2}...a_1a_0$
- Fractional part (right of point) $a_{-1}a_{-2}....a_{-m}$

Positional System base *r* .

Power expansion

$$(a_{n-1}a_{n-2}...a_1a0.a_{-1}a_{-2}..a_{-m})_r =$$

$$a_{n-1}\,r^{n-1} + a_{n-2}\,r^{n-2} + ...+ a_1\,r^1 + a_0\,r^0 + a_{-1}\,r^{-1} + a_{-2}\,r^{-2}...a_{-m}\,r^{-m}$$

$$(a_{n-1}a_{n-2}...a_1a0.a_{-1}a_{-2}..a_{-m})_r = \sum_{j=-m}^{n-1} a_j r^j$$

CONVERSION BETWEEN BASES (1):

- **<u>Method 1: Use power expansion in target base</u>**
  - Since we know how to multiply and add in base 10, this is usually done by this method to convert to base 10
  - Examples:

1011.101B = $(1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}) = 11.625$

A2F.Bh = $(10 \times 16^2 + 2 \times 16 + 15 + 11 \times 16^{-1}) = 2607.6875$

${r^n}_{10} = 1000......00_r$: A power n of r in base 10 is equivalent to 1 followed by n 0's in base r

- $r^n{}_{10} = 1000\ldots\ldots00_r$: A power n of r in base 10 is equivalent to 1 followed by n 0's in base r
- 5^2=100(base 5)

Convenient shortcuts

Binary :	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	b-1	b-2	b-3	b-4	b-5
Power of 2:	512	256	128	64	32	16	8	4	2	1	0.5	0.25	0.125	0.0625	0.03125
Power of 16:		16^2				16^1				1				16^(-1)	

1011011.01B = 64 + 16 + 8 + 2 + 1 + .25 = 91.25

Conversion between bases (1/6) - Base 10 to base r – Integer part

- **Method 2**: **Successive division for integer part; successive multiplication for fractional part.**
- <u>Integer part:</u> To convert N_{10} to $(a_{n-1}a_{n-2} \ldots a_1a_0)_r$, extract digits with the following algorithm:
  - STEP 1: $M = N_{10}$ $k = 0$
  - STEP 2: While $M > 0$
    - Divide M/r: Quotient Q, a_k= residue
    - $M = Q$, $k = k+1$

Examples (2/6)

- 109 = ?? Base 7
  - 109/7 → Q=15, a0 = 4
  - 15/7 → Q=2, a1 = 1
  - 2/7 → Q=0, a2 = 2
  - Hence 109 = 214_7
- 109 = ?? Base 16
  - 109/16 → Q=6, a0 = 13 (D)
  - 6/16 → Q=0, a1 = 6
  - Hence 109 = 0x6D

- 215 = ?? Base 2
  - 215/2 → Q=107, a0 = 1
  - 107/2 → Q = 53, a1 = 1
  - 53/2 → Q = 26, a2 = 1
  - 26/2 → Q = 13. a3 = 0
  - 13/2 → Q = 6 , a4 = 1
  - 6/2 → Q = 3, a5 = 0
  - 3/2 → Q = 1, a6 = 1
  - 1/2 → Q=0 , a7 =1
  - Hence 215 = 11010111b

Conversion between bases (3/6) - Base 10 to base r – Fractional Part

- <u>Fractional part:</u>
  - To convert $N_{10} < 1$ to $(0.a_{-1}a_{-2}\,a_{-3}....)_r$, extract digits with the following algorithm:
  - Step 1: $M=N_{10}$, $k=-1$
  - Step 2: REPEAT
    - Multiply Mxr: a_k = Integer Part
    - M = Fractional part, $k = k-1$
  - UNTIL **STOP** CRITERION

STOP CRITERIA FOR FRACTIONAL CONVERSION: (4/6)

- We stop the repeat loop when one of the following happens:

1. **Fractional part = 0 or**

2. **Fractional part repeats (Periodical) or**

3. **Predetermined number of digits is reached**

Examples (5/6)

0.625 = ?? Base 2:

0.625 x 2 = 1.25 a(-1) =1
0.25 x 2 = 0.5 a(-2) = 0
0.5 x 2 = 1.0 a(-3) = 1
STOP: Frac. = 0

0.625 = 0.101B

Converting 0.05:

$$
\begin{aligned}
2\times 0.05 &= 0.1 \rightarrow a_{-1} = 0 \\
2\times 0.10 &= 0.2 \rightarrow a_{-2} = 0 \\
2\times 0.2 &= 0.4 \rightarrow a_{-3} = 0 \\
2\times 0.4 &= 0.8 \rightarrow a_{-4} = 0 \\
2\times 0.8 &= 1.6 \rightarrow a_{-5} = 1 \\
2\times 0.6 &= 1.2 \rightarrow a_{-6} = 1 \quad \text{Repeating fractional part 0.2. Stop}
\end{aligned}
$$

0.05 = 00001100110011……B

Example: Stop with 8 digits (6/6)

0.67 = ?? Base 2 with maximum 8 digits:

0.67 x 2 = 1.34	a(-1) = 1
0.34 x 2 = 0.68	a(-2) = 0
0.68 x 2 = 1.36	a(-3) = 1
0.36 x 2 = 0.72	a(-4) = 0
0.72 x 2 = 1.44	a(-5) = 1
0.44 x 2 = 0.88	a(-6) = 0
0.88 x 2 = 1.76	a(-7) = 1
0.76 x 2 = 1.52	a(-8) = 1

0.67 ≈ 0.10101011 B

Very Important equivalencies:

Decimal	Binary	Hex	Octal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17

Hex 🡐 🡒 Binary

- Hex 🡒Binary: expand each hex digit by its binary equivalent

  92A.32h = 100100101010.00110010B

  1F24h = 0001 1111 0010 0100B

- Binary 🡒Hex: Group binary digits in sets of four digits and convert to Hex digits. Integer from right to left, fractional part from left to right. Add zeros if necessary.

  1011011.101101B = 5B.B4h

  [0101 1011. 1011 0100]

HEX NOTATION

Since there is a one to one correspondence between hex expressions and binary expressions, <u>any n-bit word may be expressed as a sequence of hex digits</u> <u>irrespectively of its actual meaning.</u>

110 0110 1010 1101 → 66AD

15-bit word expressed with a 4-digit hex

(16, 15, 14, 13-bit words expressed with 4-digit hex)