Name: Sylvia Le

Course: COM219

**Homework 3**

**Question 3**

|  |  |
| --- | --- |
| **Value in base 10** | **Binary (IEEE-754 Floating Point)** |
| * First bit = 0 → positive number * The exponent part: 10000011 (excess-127) to decimal   100000112 (pure binary) = 131  131 – 127 = 410  → The exponent is 4   * The fraction: 1.101001   Mantissa: 2-1 + 2-3 + 2-6 = 0.640625  → 1.1010012 = 1.64062510   * X = sign x significand x 2exponent   = 1.640625 x 24 = 26.25 | 0 10000011 10100100000000000000000 |
| * First bit = 1 → negative number * The exponent part: 10000111 (excess-127) to decimal   100001112 (pure binary) = 135  135 – 127 = 810  → The exponent is 8   * ­ The fraction: 1.01010100001   Mantissa: 2-2 + 2-4 + 2-6 + 2-11 = 0.3286132813 (or 673/2048, I don’t know if my calculator displayed all the digits after decimal point)  → 1.010101000012 = 1.328613281310   * X = sign x significand x 2exponent   = -1 x 1.3286132813 x 28 = -340.125 | 1 10000111 01010100001000000000000 |
| -75.875 | * Number is negative → first bit = 1 * 75 to binary is 1001011 ( = 64 + 8 + 2 + 1) * The decimal part:   + 0.875 x 2 = 1.75 (1)  + 0.75 x 2 = 1.5 (1)  + 0.5 x 2 = 1.0 (1)  → 0.875 to binary is 0.111  → 75.87510 = 1001011.1112  → Normalized: 1.001011111 x 26  → Significand = 001011111   * The exponent: 6 + 127 = 133   133 to binary is: 10000101   * Combine the sign, the exponent and the significand we have:   1 10000101 00101111100000000000000 |
| 19.71875 | * Number is positive → first bit = * 19 to binary is 10011 ( = 16 + 2 + 1) * The decimal part:   + 0.71875 x 2 = 1.4375 (1)  + 0.4375 x 2 = 0.875 (0)  + 0.875 x 2 = 1.75 (1)  + 0.75 x 2 = 1.5 (1)  + 0.5 x 2 = 1.0 (1)  → 0.71875 to binary is 0.10111  → 75.87510 = 10011.101112  → Normalized: 1.001110111 x 24  → Significand = 001110111   * The exponent: 4 + 127 = 131   131 to binary is: 10000011   * Combine the sign, the exponent and the significand we have:   0 10000011 00111011100000000000000 |

**Question 3**

1. (-102)10 = -215 + 214 + 213 + 212 + 211 + 210 + 29 + 28 + 27 + 24 + 23 + 2 = 1111 1111 1001 1010

|  |  |
| --- | --- |
| address p + 1 |  |
| address p | 1111 1111 1001 1010 |
| address p - 1 |  |

1. 8-bit cell and 16-bit word, little endian

|  |  |
| --- | --- |
|  |  |
| address p + 1 | 1111 1111 |
| address p | 1001 1010 |
| address p - 1 |  |
|  |  |

* For part a: since it’s 16-bit cell storing 16-bit words, the byte ordering is the same for big-endian and little-endian → no changes, it’s still -102
* For part b: the binary string if mistakenly read as big endian: 1001 1010 1111 1111

-215 + 212 + 211 + 29 + 27 + 26 + 25 + 24 + 23 + 22 + 2 + 1 = -2585710

**Question 4**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Address range of memory chip**  **(hex)** | **Cell size**  **(bits)** | **Size of CPU**  **address space**  **(locations)** | **Number of**  **cells on**  **memory chip** | **Percentage of chip size in address space** | **Capacity (bytes)** |
| **Ex** | 00H – 3FH | 8 bits | 256 locs | 64 cells | 26/28 = 0.25 | 64 bytes |
| **A** | 200H – 3FFH | 8 bits | Address is in hex, has three hex digit → 12 binary digits  → add. space = 212 | 20016 = 0010000000002  3FF16 = 0011111111112  (changes in 9 bits)  → 29 locations of physical memory  = 512/1 = 512 cells | % = #cells/add. space  29/212 = 0.125 | Each memory location is a byte  → 29 x 1 = 512 bytes |
| **B** | 0000H – ?  2# bits change = 2#memory cells  → #bits change = 11  0000H = 00000000000000002  → The other address:  00000111111111112 = 07FFH | 32 bits | Address is in hex, has four hex digit → 16 binary digits  → add. space = 216 | 2048 cells  (= 211) | % = #cells/add. space  211/216 = 0.03125 | Capacity = #cells x cell size  = 2048 (211) x 4 (22)  = 213 = 23 x 210 = 8 KB |
| **C** | 80000H – ?  2# bits change = 2#memory cells  → #bits change = 18  80000H = 100000000000000000002  → The other address:  101111111111111111112 = BFFFFH | 16 bits | Address is in hex, has five hex digit → 20 binary digits  → add. space = 220 | Number of cells  = capacity/cell size  = 219/2 = 218 cells | % = #cells/add. space  218/220 = 0.25 | 512 KB  (kilobytes)  (= 29 x 210= 219 bytes) |
| **D** | ? - ?  Address has 4 hex digits  Let the beginning address be 0000H  2#bits change x cell size = capacity  → 2#bits change = 218/22 = 216  → #bits change = 16  0000H = 00000000000000002  → The other address:  11111111111111112 = FFFFH | 32 bits | % = #cells/add. space  → add. Space = 216/1 = 216  (→ address has four hex digits) | Number of cells  = capacity/cell size  = 218/22 = 216 cells | 1 | 256 KB  (= 28 x 210 = 218 bytes) |
| **E** | 400000H - 6FFFFFH | 64 bits | Address is in hex, has six hex digit → 24 binary digits  → add. space = 224 | 400000H =  0100000000000000000000002  6FFFFFH =  0111111111111111111111112  (changes in 22 bits)  → 222 locations of physical memory | % = #cells/add. space  = 222/224 = 0.25 | Each memory location is 8 byte  → 222 x 8 (23) = 225 = 25 x 220 = 32 MB |