**Отчет по лабораторной работе №8**

**Цель работы:**

* Изучение алгоритма представления целых чисел со знаком и без знака.
* Построение формальной модели оперативной памяти.

**Выполнение работы:**

**1. Перевод чисел в указанные системы счисления**

**1.1. Перевод 509(10) в двоичную систему и обратно**

509 / 2 = 254 (1)  
254 / 2 = 127 (0)  
127 / 2 = 63 (1)  
63 / 2 = 31 (1)  
31 / 2 = 15 (1)  
15 / 2 = 7 (1)  
7 / 2 = 3 (1)  
3 / 2 = 1 (1)  
1 / 2 = 0 (1)

Записываем остатки снизу вверх: 509(10) = 111111101(2)

Выполняем разложение по степеням двойки:  
1\*2^8 + 1\*2^7 + 1\*2^6 + 1\*2^5 + 1\*2^4 + 1\*2^3 + 1\*2^2 + 0\*2^1 + 1\*2^0 = 256 + 128 + 64 + 32 + 16 + 8 + 4 + 0 + 1 = 509(10)

**1.2. Перевод 187(10) в шестнадцатеричную систему и обратно**

187 / 16 = 11 (11 = B)  
11 / 16 = 0 (11 = B)  
Записываем остатки снизу вверх: 187(10) = BB(16)

Выполняем разложение по степеням числа 16:  
B\*16^1 + B\*16^0 = 11\*16 + 11\*1 = 176 + 11 = 187(10)

**1.3. Перевод 10111010(2) в шестнадцатеричную систему и обратно**

Разбиваем на тетрады (группы по 4 бита) справа налево: 1011 1010  
Заменяем каждую тетраду hex-цифрой:  
1011(2) = B(16), 1010(2) = A(16)  
10111010(2) = BA(16)

Заменяем каждую hex-цифру на 4 бита:  
B(16) = 1011(2), A(16) = 1010(2)  
BA(16) = 10111010(2)

**1.4. Перевод 9FA(16) в двоичную систему и обратно**

Заменяем каждую hex-цифру на 4 бита:  
9(16) = 1001(2), F(16) = 1111(2), A(16) = 1010(2)  
9FA(16) = 100111111010(2)

Разбиваем на тетрады справа налево: 1001 1111 1010  
Заменяем каждую тетраду hex-цифрой:  
1001(2) = 9(16), 1111(2) = F(16), 1010(2) = A(16)  
100111111010(2) = 9FA(16)

**2. Минимальное число бит и байт для хранения чисел**

34: 2^5 = 32 < 34 < 64 = 2^6 = 6 бит (1 байт)  
65: 2^6 = 64 < 65 < 128 = 2^7 = 7 бит (1 байт)  
309: 2^8 = 256 < 309 < 512 = 2^9 = 9 бит (2 байта)  
101(2): 101(2) = 5(10) = 2^2 = 4 < 5 < 8 = 2^3 = 3 бита (1 байт)  
1100001(2): 1100001(2) = 97(10) = 2^6 = 64 < 97 < 128 = 2^7 = 7 бит (1 байт)  
11100011110110011(2): количество разрядов - 17 = 17 бит (3 байта)  
0(16): 0(16) = 0(10) = 1 бит (1 байт)  
56(16): 56(16) = 86(10) = 2^6 = 64 < 86 < 128 = 2^7 = 7 бит (1 байт)  
1F67A56(16): в hex 7 цифр = 7 \* 4 = 28 бит (4 байта)

**3. Дополнительный код чисел в двухбайтовой ячейке (16 бит)**

**3.1. –6**

Модуль: 6 = 0000000000000110(2)  
Инверсия: 1111111111111001(2)  
+1: 1111111111111010(2)  
Доп. код (–6) = 0xFFFA

**3.2. –1023**

Модуль: 1023 = 0000001111111111(2)  
Инверсия: 1111110000000000(2)  
+1: 1111110000000001(2)  
Доп. код (–1023) = 0xFC01

**3.3. –101001(2)**

Модуль: 101001(2) = 41(10) = 0000000000101001(2)  
Инверсия: 1111111111010110(2)  
+1: 1111111111010111(2)  
Доп. код (–101001(2)) = 0xFFD7

**3.4. –101111100111(2)**

Модуль: 101111100111(2) = 3047(10) = 0000101111100111(2)  
Инверсия: 1111010000011000(2)  
+1: 1111010000011001(2)  
Доп. код (–101111100111(2)) = 0xF419

**3.5. –6E(16)**

Модуль: 6E(16) = 110(10) = 0000000001101110(2)  
Инверсия: 1111111110010001(2)  
+1: 1111111110010010(2)  
Доп. код (–6E(16)) = 0xFF92

**3.6. –1F09(16)**

Модуль: 1F09(16) = 7945(10) = 0001111100001001(2)  
Инверсия: 1110000011110110(2)  
+1: 1110000011110111(2)  
Доп. код (–1F09(16)) = 0xE0F7

**4. Дополнительный код нуля**

Модуль: 0 = 00000000(2)  
Инверсия: 11111111(2)  
+1: 100000000(2) → старший бит отбрасывается  
Результат: 00000000(2) = 0  
Дополнительный код нуля является нулем

**5. Дополнительный код числа –128 в рамках одного байта**

Модуль: 128 = 10000000(2)  
Инверсия: 01111111(2)  
+1: 10000000(2)  
Доп. код (–128) = 0x80 (10000000(2))  
Вывод: в байтовой ячейке (–128) представляется как 10000000(2)

**6. Корректность сложения на основе дополнительного кода**

**6.1. Два положительных: 5 + 3 = 8**

5 = 00000101(2)

3 = 00000011(2)

Складываем:

00000101

+ 00000011

----------------

00001000

Результат: 00001000(2) = 8(10)

**6.2. Положительное и отрицательное (отриц. меньше по модулю): 10 + (–3) = 7**

Находим дополнительный код для -3:

Модуль 3 = 00000011(2)

Инверсия: 11111100(2)

Прибавляем 1: 11111101(2) - это -3

10 = 00001010(2)

-3 = 11111101(2)

Складываем:

00001010

+ 11111101

---------------

100000111

Отбрасываем старший бит: 00000111(2) = 7(10)

**6.3. Положительное и отрицательное (отриц. больше по модулю): 5 + (–10) = –5**

Находим дополнительный код для -10:

Модуль 10 = 00001010(2)

Инверсия: 11110101(2)

Прибавляем 1: 11110110(2) - это -10

5 = 00000101(2)

-10 = 11110110(2)

Складываем:

00000101

+ 11110110

--------------

11111011

Получили отрицательное число 11111011(2)

Находим его значение через обратный перевод:

Инверсия: 00000100(2)

Прибавляем 1: 00000101(2) = 5(10)

Значит исходное число было -5

**6.4. Два отрицательных: (–4) + (–5) = –9**

Находим дополнительные коды:

Для -4: модуль 4 = 00000100(2) → инверсия 11111011(2) → +1 = 11111100(2)

Для -5: модуль 5 = 00000101(2) → инверсия 11111010(2) → +1 = 11111011(2)

-4 = 11111100(2)

-5 = 11111011(2)

Складываем:

11111100

+ 11111011

---------------

111110111

Отбрасываем старший бит: 11110111(2)

Находим значение результата:

Инверсия: 00001000(2)

Прибавляем 1: 00001001(2) = 9(10)

Значит исходное число было -9