ДЗ 2. Гусев Егор Дмитриевич

Задача 1. Перевод десятичных чисел в 6-битные двоичные числа

(a) Беззнаковые числа: 0, 13, 24, 63

- **0:** $0_{10} = 000000_2$.
- **13:** Делим 13 на 2: $13 = 8 + 4 + 1$, то есть $13_{10} = 001101_2$.
- **24:** $24 = 16 + 8$, т.е. $24_{10} = 011000_2$.
- **63:** Максимальное число для 6 бит ($2^6 - 1 = 63$), $63_{10} = 111111_2$.

(b) Знаковые числа (представление в дополнительном коде): 16, -2, 31, -32

- **16:** В 6-битном диапазоне (от -32 до 31) положительное число. $16_{10} = 010000_2$.
- **-2:** Для отрицательного числа находим двоичное представление для 2: 000010_2, затем инвертируем и прибавляем 1: инверсия 111101_2, $+1 \to 111110_2$.
- **31:** Максимальное положительное число: $31_{10} = 011111_2$.
- **-32:** Минимальное число в 6-битном диапазоне: $-32_{10} = 100000_2$.

Замечание: Для знаковых чисел используется метод дополнительного кода, где для отрицательного числа значение получается как $2^n - |x|$, $n = 6$.

Задача 2. Перевод 6-битных значений в десятичные числа

Даны значения: `000101`, `101011`, `111111`, `100000`.

Формулы

- **Беззнаковое представление:**

$$D = \sum_{i=0}^{5} b_i \cdot 2^i.$$

- **Представление в дополнительном коде:** Если MSB (b_5) равен 1, то

$$D = \left(\sum_{i=0}^{4} b_i \cdot 2^i \right) - 2^5,$$

иначе D определяется как беззнаковое число.

Решения

1. `000101`:
   - Беззнаковое: $0 \cdot 32 + 0 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 4 + 1 = 5$.
   - Дополнительный код: MSB $= 0 \to 5$.
2. `101011`:
   - Беззнаковое: $1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 = 32 + 8 + 2 + 1 = 43$.
   - Дополнительный код: MSB $= 1$, значит: $43 - 64 = -21$.
3. `111111`:
   - Беззнаковое: $32 + 16 + 8 + 4 + 2 + 1 = 63$.
   - Дополнительный код: $63 - 64 = -1$.
4. `100000`:
   - Беззнаковое: $1 \cdot 32 = 32$.
   - Дополнительный код: $32 - 64 = -32$.

Задача 3. Перевод десятичных значений в 8-битные шестнадцатеричные числа

Даны значения: 7, 240, 171, 126.

Вычисления:

- 7_{10}**:**
  Делим 7 на 16:
  $$7 \div 16 = 0 \quad \text{(частное)}, \quad \text{остаток} = 7.$$
  Остаток 7 соответствует 7_{16}. Так как для 8-битного представления требуется 2 шестнадцатеричных знака, добавляем ведущий 0: $\mathbf{07_{16}}$ (0x07).

- 240_{10}**:**
  Делим 240 на 16:
  $$240 \div 16 = 15 \quad (\text{частное}), \quad \text{остаток} = 0.$$
  Число 15 в шестнадцатеричной системе обозначается как F_{16}, а остаток 0 соответствует 0_{16}. Получаем $\mathbf{F0_{16}}$ (0xF0).

- 171_{10}**:**
  Делим 171 на 16:
  $$171 \div 16 = 10 \quad (\text{частное}), \quad \text{остаток} = 11.$$
  Число 10 обозначается как A_{16}, а 11 — как B_{16}. Получаем $\mathbf{AB_{16}}$ (0xAB).

- 126_{10}**:**
  Делим 126 на 16:
  $$126 \div 16 = 7 \quad (\text{частное}), \quad \text{остаток} = 14.$$
  Число 7 остаётся 7_{16}, а 14 обозначается как E_{16}. Таким образом, $126_{10} = \mathbf{7E_{16}}$ (0x7E).

Задача 4. Перевод шестнадцатеричных чисел в 8-битные двоичные значения

Даны значения: `0x3C`, `0x7E`, `0xFF`, `0xA5`.

Вычисления:

- `0x3C`**:**
  Запишем каждую цифру в двоичном представлении по 4 бита:
  $3_{16} = 0011_2, \quad C_{16} = 12_{10} = 1100_2.$
  Таким образом, $\mathbf{0x3C = 0011\,1100_2}$.

- `0x7E`**:**
  $7_{16} = 0111_2, \quad E_{16} = 14_{10} = 1110_2.$
  Получаем $\mathbf{0x7E = 0111\,1110_2}$.

- `0xFF`**:**
  $F_{16} = 15_{10} = 1111_2$. Так как обе цифры равны F, то
  $\mathbf{0xFF = 1111\,1111_2}$.

- `0xA5`**:**
  $A_{16} = 10_{10} = 1010_2, \quad 5_{16} = 0101_2.$
  Следовательно, $\mathbf{0xA5 = 1010\,0101_2}$.

Задача 5. Сменить знак полученных выше чисел

Используем уже знакомый метод (то есть, для числа x находим $-x$ как инверсию битов и прибавление 1). Пусть исходные значения рассматриваются в знаковом представлении.

1. Для `00111100` (0x3C, что есть +60):

$$-60 \Rightarrow 256 - 60 = 196 \Rightarrow 196_{10} = 1100\,0100_2.$$

2. Для `01111110` (0x7E, +126):

$$-126 \Rightarrow 256 - 126 = 130 \Rightarrow 130_{10} = 1000\,0010_2.$$

3. Для `11111111` (0xFF, представляется как -1):

$$-(-1) = 1 \Rightarrow 0000\,0001_2.$$

4. Для `10100101` (0xA5, представляется как -91, т.к. $256 - 165 = 91$):

$$-(-91) = 91 \Rightarrow 91_{10} = 0101\,1011_2.$$

Задача 6. Расположение байтов 0xDEADBEEF в памяти

Запишем значение в виде байтов:

`DE  AD  BE  EF`

- **Big-endian:** байты хранятся в порядке: `DE AD BE EF` (от младшего к старшему адресам — в том же порядке, что и запись числа).
- **Little-endian:** байты располагаются в обратном порядке: `EF BE AD DE`.

Задача 7. Перевод десятичных значений в 5-битные двоичные числа и расширение до 8 бит

Даны значения: 7, 15, -16, -5.

Шаг 1. Представление в 5 битах

- 7_{10}: в 5 битах $7 = 00111_2$.
- 15_{10}: $15 = 01111_2$.
- -16_{10}: В 5-битном дополнительном коде минимальное значение: $-16 = 10000_2$.
- -5_{10}: Для 5 бит: сначала $5 = 00101_2$. Инвертируем: 11010_2, прибавляем 1: 11011_2.

Шаг 2. Расширение до 8 бит

- **Знаковое расширение** (копируем старший бит):
  - 7 (00111): MSB $= 0$, расширение: $0000\,0111_2$.
  - 15 (01111): MSB $= 0$, расширение: $0000\,1111_2$.
  - -16 (10000): MSB $= 1$, расширение: $1111\,0000_2$.
  - -5 (11011): MSB $= 1$, расширение: $1110\,1011_2$.
- **Нулевое расширение** (просто добавляем нули слева):
  - 7 (00111): $\to 0000\,0111_2$.
  - 15 (01111): $\to 0000\,1111_2$.
  - -16 (10000): $\to 0001\,0000_2$ (но это будет не совсем корректно для представления отрицательных чисел, ведь нулевое расширение не сохранит знак).
  - -5 (11011): $\to 0001\,1011_2$ (аналогично вышенаписанному).

Задача 8. Представление пар десятичных чисел в 4-битных двоичных и их сложение

Даны две пары.

(a) Беззнаковая: (7, 9)

- $7_{10} = 0111_2$ (в 4 битах).
- $9_{10} = 1001_2$.
- Сумма: $7+9 = 16_{10}$. Но в 4 битах максимальное число — 15, поэтому происходит переполнение.

  При сложении в 4 битах:

  $$0111 + 1001 = 0000 \quad \text{(с переносом 1)}$$

  Итоговое значение по модулю 16: 0_{10}.

(b) Знаковая (в дополнительном коде): (4, -5)

- $4_{10} = 0100_2$ (4 бита).
- -5_{10}: Для 4 бит диапазон: $-8\ldots7$. Сначала $5 = 0101_2$; инвертируем: 1010_2; прибавляем 1: 1011_2.
- Сложение:

  $$0100 + 1011 = 1111_2.$$

  Интерпретация: 1111_2 в 4-битном дополнительном коде равно -1_{10} (так как $16 - 1 = 15$).

Задача 12*. Объяснение трюка обмена значениями без временной переменной

Дан следующий алгоритм для обмена значениями переменных `x` и `y` без использования временной переменной:

```
x = x ^ y;
y = x ^ y;
x = x ^ y;
```

Объяснение: Пусть, для примера, исходные значения:

$$x = 0011_2, \quad y = 1100_2.$$

1. **Первый шаг:**
$$x = x \oplus y = 0011_2 \oplus 1100_2 = 1111_2.$$
Теперь `x` содержит значение 1111_2.

2. **Второй шаг:**
$$y = x \oplus y = 1111_2 \oplus 1100_2.$$
Побитово:
$$1 \oplus 1 = 0, \quad 1 \oplus 1 = 0, \quad 1 \oplus 0 = 1, \quad 1 \oplus 0 = 1,$$
таким образом, $y = 0011_2$ (исходное значение x).

3. **Третий шаг:**
$$x = x \oplus y = 1111_2 \oplus 0011_2.$$
Побитово:
$$1 \oplus 0 = 1, \quad 1 \oplus 0 = 1, \quad 1 \oplus 1 = 0, \quad 1 \oplus 1 = 0,$$
таким образом, $x = 1100_2$ (исходное значение y).

В результате обмена получаем:

$$\text{Начальные: } x = 0011_2,\ y = 1100_2 \quad \Longrightarrow \quad \text{После: } x = 1100_2,\ y = 0011_2.$$

Как я понимаю, основная идея трюка заключается в том, что операция XOR обладает обратимым свойством, позволяющим «смешать» оба значения, а затем восстановить каждое из них без использования дополнительной памяти.

Задача 16*. Объяснение битовых трюков

Рассмотрим следующие битовые операции и объясним их действие.

1. Выключение самой младшей единицы: $x \,\&\, (x-1)$

Эта операция обнуляет самый младший бит, равный 1, оставляя все остальные биты без изменений.

Пример: Пусть

$$x = 0101\,1000_2.$$

Тогда:

$$x - 1 = 0101\,0111_2.$$

Для наглядности приведём побитовое представление в виде таблицы:

x:	0	1	0	1	1	0	0	0
x-1:	0	1	0	1	0	1	1	1
$x \,\&\, (x-1):$	0	1	0	1	0	0	0	0

Таким образом, $0101\,1000_2$ превращается в $0101\,0000_2$ — то есть самая младшая единица обнулена.

2. Включение самой младшей нулевой позиции: $x \,|\, (x+1)$

Эта операция устанавливает в 1 самый правый нулевой бит, который непосредственно следует за группой единиц.

Пример: Пусть

$$x = 1010\,0111_2.$$

Тогда:

$$x + 1 = 1010\,1000_2.$$

Покажем побитово:

x:	1	0	1	0	0	1	1	1
x+1:	1	0	1	0	1	0	0	0
$x \,|\, (x+1):$	1	0	1	0	1	1	1	1

В результате получается $1010\,1111_2$ — то есть включён самый правый нулевой бит.

3. Включение всех младших битов: $x \,|\, (x-1)$

Эта операция приводит к тому, что все биты, находящиеся правее самой младшей единицы, принимабт новое значение - 1.

Пример: Пусть

$$x = 1010\,1000_2.$$

Тогда:

$$x - 1 = 1010\,0111_2.$$

Покажем вычисление:

x:	1	0	1	0	1	0	0	0
x-1:	1	0	1	0	0	1	1	1
$x \,|\, (x-1):$	1	0	1	0	1	1	1	1

Таким образом, $x \mid (x-1)$ превращает $1010\,1000_2$ в $1010\,1111_2$, то есть все биты правее первой единицы становятся единицами.