ЛАБОРАТОРНАЯ РАБОТА 5

УПАКОВКА БИТОВЫХ ГРУПП

Рассмотрим решение следующей задачи.

1. **Задание.** Даны четыре битовые группы, хранящие независимую информацию: поля xxx, yyy, zzz и www. Каждое поле хранится в виде целого беззнакового числа в переменной минимальной длины. Старшие биты могут содержать несущественную информацию, отличную от нуля.

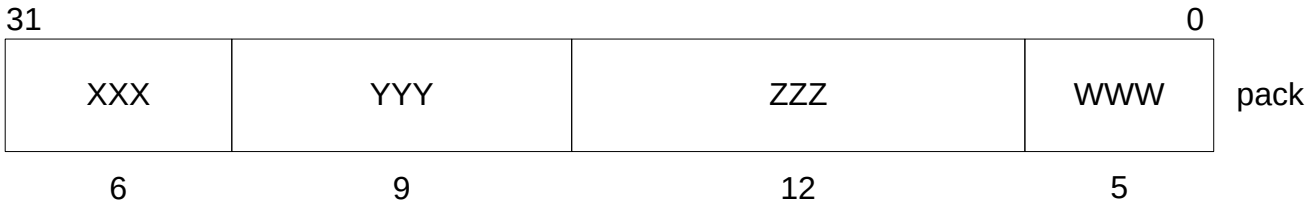

Рисунок 1 – Число pack, содержащие битовые группы

Написать программу на C и на ассемблере, которая *упаковывает* группы битов, содержащиеся в изолированных переменных x, y, z и w, в одно 32-битовое целое беззнаковое число `pack`.

2. Определим исходные данные, необходимые для решения поставленной задачи. Для этого рассмотрим идентификаторы, указанные в задании.

Исходное число содержит четыре битовые группы, каждая из которых храниться изолировано в своей переменной, имеющей минимально возможную длину. Поэтому определим идентификаторы x, y, z и w, каждый из которых будет использоваться для хранения поля xxx, yyy, zzz и www соответственно (рис.2). Исходные переменные содержат битовые группы в младших разрядах, а старшие неиспользуемые биты могут иметь произвольные значения. Поле www в переменной w имеет 5 битов, поэтому для хранения этой битовой группы достаточно байтовой переменной типа char. Поле xxx в переменной x имеет 6 битов и для его хранения достаточно байтовой переменной тира char. Поле zzz в переменной z имеет 12 битов, для хранения которых достаточно 16 битовой переменной типа short. Для хранения 9-битового поля yyy в переменной y также достаточно 16-битовой переменной типа short.

Исходные данные.

x, w – переменные, типа unsigned char, длиной байт;

y, z - переменные, типа unsigned short, длиной слово.

Формат исходных данных.

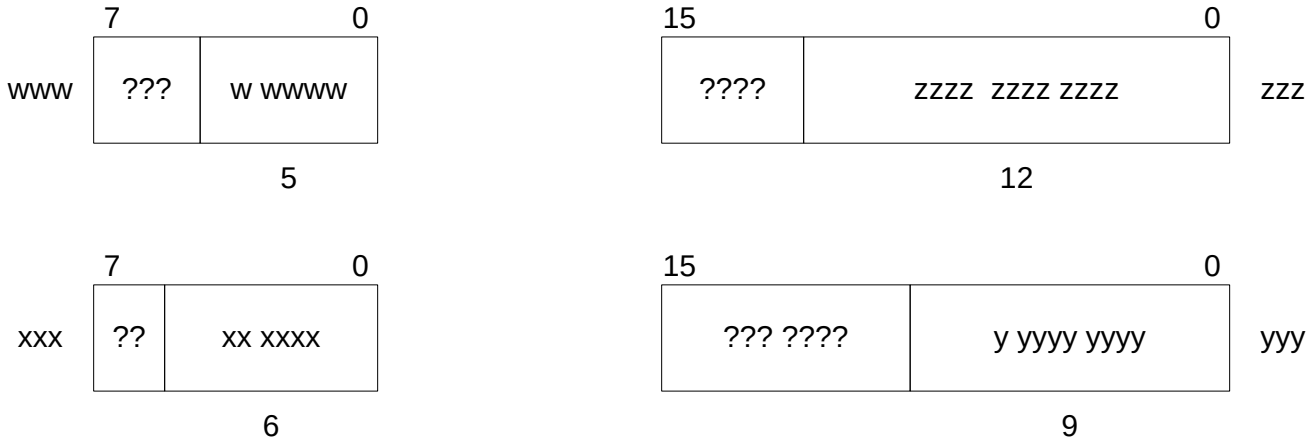

Рисунок 2 – Исходные переменные, хранящие только одну битовую группу

3. Поскольку решается задача упаковки, то идентификатор числа `pack`, указанного в задании и будет требуемым результатом. По условию задачи требуемый результат является 32-разрядной беззнаковой величиной, то эта переменная будет определен как `unsigned long`.

Формат требуемого результата.

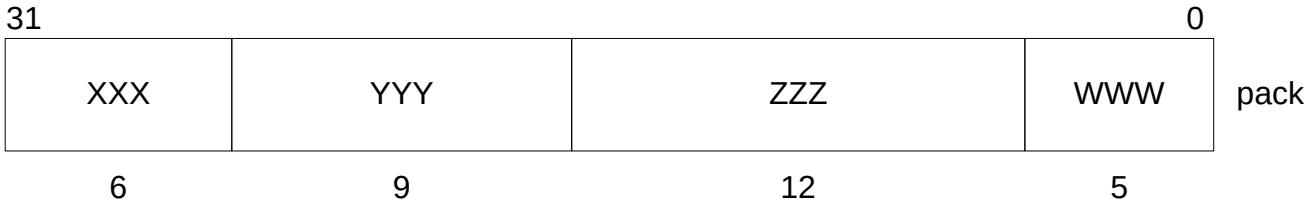

Рисунок 3 – Результирующая переменная pack, хранящая четыре битовых группы

Требуемого результат.

pack – переменная, длинное целое беззнаковое число.

4. Разработка алгоритма.

Алгоритм упаковки объединяет битовые поля, каждое из которых размещается в собственной переменной, в единой переменной `pack`. Упаковка начинается со старшей битовой группы, т.е. с той, которая расположена с левого края числа `pack`..

```
Ввести исходные переменные xxx, yyy, zzz, www;
Очистить старшие биты в каждой исходной переменной на C;
Объединить поле xxx и число pack на C;
Объединить поле yyy и число pack на C;
Объединить поле zzz и число pack на C;
Объединить поле www и число pack на C;
Очистить старшие биты в каждой исходной переменной на ассемблере;
Очистить результат на ассемблере;
Объединить поле xxx и число pack на ассемблере;
```

```
Объединить поле yyy и число pack на ассемблере;
Объединить поле zzz и число pack на ассемблере;
Объединить поле www и число pack на ассемблере;
Вывести значения переменных на C;
Вывести значения переменных на ассемблере.
```

Окончательная версия алгоритма приводится ниже.

Описание алгоритма на псевдокоде.

Детальный алгоритм решения задачи представлен ниже (рис.4). В одном блоке указываются те операции, которые можно выполнять последовательно слева направо. Для выполнения промежуточных вычислений используется отдельная вершина в схеме алгоритма. Чтобы ссылаться на отдельные блоки в схеме алгоритма удобно их пронумеровать, как показано на рисунке.

Описание схемы алгоритма.

Блок 1 – "Приглашение к вводу данных". Данный блок выдает на дисплей приглашение к вводу данных с клавиатуры.

Блок 2 – "Ввод переменных". Данный блок выполняет ввод чисел с клавиатуры и размещение их в памяти, выделенной для переменных. В данном случае вводятся переменные `xxx`, `yyy`, `zzz` и `www`. Пользователь может вводить различные числа.

Блок 3 – "Вычисление выражения на С данных". Данный блок объединяет битовые группы www, zzz, xxx, yyy в единой переменной `pack` на С.

Блок 4 - "Очистить результат". Данный блок обнуляет результирующую переменную pack.

Блок 5 – "Очистить старшие биты в каждой переменной". Данный блок при помощи битовой операции И сбрасывает в ноль старшие биты каждой исходной переменной. До выполнения этого блока старшие биты могли содержать единичные или нулевые биты, а после него – только несущественные нули.

Блок 6 – "Объединить поле xxx и результат". Данный блок при помощи битовой операции ИЛИ объединяет битовую группу xxx и результирующе число pack.

Блок 7 – "Объединить поле yyy и результат". Данный блок при помощи битовой операции ИЛИ объединяет битовую группу yyy и результирующе число pack.

Блок 8 – "Объединить поле zzz и результат". Данный блок при помощи битовой операции ИЛИ объединяет битовую группу zzz и результирующе число pack.

Блок 9 – "Объединить поле www и результат". Данный блок при помощи битовой операции ИЛИ объединяет битовую группу www и результирующе число pack.

Блок 7 – "Вывод результатов на дисплей". Данный блок выводит на дисплей значение переменных pack, pack_a, хранящих результаты вычислений на языке С и на ассемблере.

Рисунок 4 – Схема алгоритма упаковки битовых групп в переменной pack

5. Кодирование алгоритма.

Теперь можно выполнить кодирование алгоритма распаковки в виде программы на языке ассемблер. Программа представляет собой бесконечный цикл, который начинается в строках 47-48, а заканчивается – в строке 100. В этом цикле выполняется вывод приглашения к вводу данных, вычисление выражения на С, распаковка битовых групп на ассемблере и вывод полученных результатов. Поскольку в данной работе мы имеем дело с битовыми полями, то ввод и вывод будем выполнять в шестнадцатеричном коде, который наиболее нагляден для работы с битами.

Исходные переменные объявлены в строках 39-40. Байтовые переменные xxx и yyy объявлены как unsigned char в строке 39 для фрагмента на С (xxx, www) и на ассемблере (xxx_a, www_a). В строке 40 как unsigned short объявлены 16-битовые переменные для хранения результата на С – переменные yyy, zzz, и переменные для хранения результата на ассемблере - переменные yyy_a, zzz_a. В строке 42 объявлена временная переменная tmp, которая используется при вводе битовых групп в байтовые переменные.

Строки 46-49 выводят заставку, а строки 50-51 начинают бесконечный цикл.

Блоки 1 – "Приглашение к вводу данных" - реализуют операторы вывода языка С++, расположенные в строках 52, 54, 56, 58.

Блок 2 – " Ввод xxx, yyy, zzz, www " реализует операторы ввода языка С++, расположенные в строках 53, 55, 57,59. Исходные числа вводятся в 16-м коде, поэтому в операторе ввода используется манипулятор hex. Без каких-либо трудностей в строках 55 и 57 вводятся 16-битные переменные, которые сохраняются в переменных типа short.

Сложнее выполнить ввод байтовых переменных, которые должны сохраняться в переменных типа char, предназначенных для хранения символов. При попытке ввести несколько цифр, обозначающих одно шестнадцатеричное число, вводится первый символ, код которого помещается в заданную переменную. Поэтому поступаем так: в шестнадцатеричном коде в переменную tmp вводим 16–битовое число. Переменная tmp определена как unsigned short, поэтому ввод числа выполняется без проблем. После этого, младшая часть введенного числа копируется в байтовую переменную. Именно так в строках 53 и 59 вводятся байтовые переменные.

В строке 61 создаются дубликаты исходных переменных для ассемблера.

Блок 3 – "Упаковка битовых групп на С" – реализуют команды, расположенные в строках 64-72.

Блок 4 – "Очистить результат " – реализует команда, расположенная в строке 81. Логическая операция Исключающее ИЛИ с двумя одинаковыми операндами в результате дает ноль.

Блок 5 – " Очистить старшие биты в каждой переменной" – реализуют команды, расположенные в строках 77-80. Чтобы в числе оставить неизменными младшие биты и сбросить в ноль старшие используется логическая операция И, выполняемая при помощи команды and. Логическая команда and выполняемая применительно к двоичной маске и исходному числу сбрасывает в ноль

те биты результата, которые в маске равны нулю, и оставляет неизменными, те биты, которые в маске равны единице. Чтобы сбросить в 5 битов в байте xxx_a в строке 77 используется логическая команда and для числа в ячейке памяти xxx_a и маски, имеющей 5 единиц в младших битах 11111_2 (0x1F). Команда and в строке 78 применяется к числу в ячейке памяти yyy_a и маске 1 1111 1111_2 (0x1FF), что приводит к сбросу 7 старших битов числа, оставляя неизменными 9 младших. Аналогичным образом маска 1111 1111 1111_2 (0xFFF) в строке 79 примененная к числу zzz_a сбрасывает в нем все старшие биты, кроме 12 младших. Команда and в строке 80 оставляет только 5 младших битов в байте, сбрасывая все старшие.

Блоки 6-9 выполняют упаковку битовых групп, содержащихся в исходных числах, и реализуются командами на ассемблере в строках 83-93. Упакованное число будет размещаться в аккумуляторе eax. Операция распаковки каждого поля выполняется однотипно для всех битовых групп и включает следующие этапы:

– крайнее справа битовое поле, имеющее разрядность до 8 или до 16 битов пересылается в результирующую переменную, соответственно длиной 8 или 16 битов. Если битовая группа имеет длину байт, то в результирующую переменную передается содержимое регистра al, а в случае 16-разрядной битовой группы – содержимое регистра ax;

– при помощи битовой операции ИЛИ в 8 или 16 младших битов результата в аккумуляторе eax объединяются с одной из исходных переменной;

– при помощи команд сдвига содержимое *всего* исходного числа сдвигается влево на столько битов, сколько содержит следующее упаковываемое поле, расположенное справа от заданного. В результате текущее поле в результате перемещается влево, а в младших разрядах результата располагаются столько нулевых битов, сколько значащих разрядов в следующем поле.

Данная реализация алгоритма предполагает, что первой упаковывается самая старшая битовая группа, то есть та которая расположена, начиная со старшего бита. Для сдвига можно использовать любую операцию сдвига, но удобнее всего сдвиг логический, так как освобождаемые биты заполняются нулями.

Блок 6 - "Объединить поле xxx и результат" – реализуют команды, расположенные в строках 83-84. Команда or в строке 83 объединяет младший байт результата, расположенный в регистр al и байтовую переменную xxx_a с 5-ю информационными битами. Следующая команда логического сдвига shl сдвигает весь 32-битовый результат влево на 9 битов. Именно столько содержит поле yyy_a, которое будет упаковываться следующим.

Блок 7 - "Объединить поле yyy и результат" – реализуют команды, расположенные в строках 86-87. Команда or в строке 86 объединяет младшее слово результата, расположенное в регистре ax, и 16-разрядную переменную yyy_a, имеющую 9 информационных битов. Следующая команда логического сдвига shl сдвигает весь 32-битовый результат влево на 12 битов, поскольку следующее упаковываемое поле zzz_a, имеет именно столько информационных битов.

Блок 8 - "Объединить поле zzz и результат" – реализуют команды, расположенные в строках 89-90. Команда or в строке 89 объединяет младшее слово результата, расположенное в регистре ax, и 16-разрядную переменную zzz_a, имеющую 12 информационных битов. Следующая команда логического сдвига shl сдвигает весь 32-битовый результат влево на 5 битов, поскольку следующее упаковываемое поле www_a содержит ровно 5 информационных битов.

Блок 9 - "Объединить поле www и результат" – реализуют команды, расположенные в строках 92-93. Команда or в строке 92 объединяет младший байт результата, расположенный в регистре al, и байтовую переменную www_a, имеющую 5 информационных битов. Процесс упаковки завершен, и команда в строке 93 сохраняет его в памяти.

Блок 8 – "Вывод результатов на дисплей" – выполняют операторы ввода языка C++, расположенные в строках 96-99. 32-битный результат, выводится в шестнадцатеричном коде. Для вывода используются операторы вывода языка C++. Для того, чтобы выполнять вывод в 16-м коде в операторе вставки (<<) в поток используется манипулятор hex.

Текст программы с комментариями.

```
 1:  //+==================================================================
 2:  // File pack.cc
 3:  // Упаковка битовых групп
 4:  //
 5:  // Эта программа упаковывает битовые группы в динное беззнаковое число
 6:  //
 7:  // (C) Дужий В.И., 2012
 8:  //
 9:  // Входные данные:
10:  //           7----5------0
11:  //           ! ?? !  xxx !  xxx
12:  //           +----+------+
13:  //                    6
14:  //           15---8------0
15:  //           ! ?? !  yyy !  yyy
16:  //           +----+------+
17:  //                    9
18:  //           15---11-----0
19:  //           ! ?? !  zzz !  zzz
20:  //           +----+------+
21:  //                    12
22:  //           7----4------0
23:  //           ! ?? !  www !  www
24:  //           +----+------+
25:  //                    5
26:  // Выходные данные:
27:  // Длинное целое беззнаковое число, которое содержит указанные битовые
28:  // группы
29:  //         31----+-----+-----+-----0
30:  //         ! xxx ! yyy ! zzz ! www !  value
31:  //         +-----+-----+-----+-----+
32:  //            6     9     12    5
33:  //
34:  //-==================================================================
35:  #include <iostream>
```

```
36:  #include <iomanip>
37:  using namespace std;
38:
39:  unsigned char  xxx, www, xxx_a, www_a;
40:  unsigned short yyy, zzz, yyy_a, zzz_a;
41:  unsigned long  value, value_a;
42:  unsigned short tmp;
43:
44:  int main()
45:  {
46:    cout<<"\n\t\t(C) Дужий В.И., 2012"
47:        <<"\n\t\tРаспаковка битовых групп"
48:        << "\n\tУпаковать битовые группы, содержащиеся в целых числах,"
49:           << "\n\t\tв 32-битовое целое число Value";
50:    for (;;)
51:    {
52:     cout<<"\nПожалуйста, введите 2 16-ые цифры для XXX (например,5a):";
53:     cin  >> hex >> tmp; xxx = tmp;
54:     cout<<"Пожалуйста, введите 3 16-х цифры для YYY (например,  9db):";
55:     cin  >> yyy;
56:     cout<<"Пожалуйста, введите 4 16-х цифры для ZZZ (например, 8e7f):";
57:     cin  >> zzz;
58:     cout<<"Пожалуйста, введите 2 16-х цифры для WWW (например,   14):";
59:     cin  >> tmp; www = tmp;
60:  // Сделать копии исходных чисел для фагмента на ассемблере
61:     xxx_a = xxx; yyy_a = yyy; zzz_a = zzz; www_a = www;
62:  //============================== C ================================
63:  // Очистить старшие биты в каждом исходном числе
64:     www &=  0x1f; // очистить все биты в числе, кроме 5 младших (4-0)
65:     zzz &= 0xfff; // очистить все биты в числе, кроме 12 младших (16-5)
66:     yyy &= 0x1ff; // очистить все биты в числе, кроме 9 младших (25-17)
67:     xxx &=  0x3f; // очистить все биты в числе, кроме 6 младших (31-26)
68:  // Объединить каждое битовое поле с результирующим числом Value
69:     value = xxx;  // объединить поле xxx с value
70:     value = (value << 9) | yyy; // объединить поле yyy с value
71:     value = (value <<12) | zzz; // объединить поле zzz с value
72:     value = (value << 5) | www; // объединить поле www с value
73:  //============================ Assembler ============================
74:  // Упаковать битовые группы
75:  __asm{
76:  // Очистить старшие биты в каждом исходном числе
77:           andb    xxx_a,0x3f
78:           andw    yyy_a,0x1ff
79:           andw    zzz_a,0xfff
80:           andb    www_a,0x1f
81:           xorl    eax,eax
82:  // объединить поле xxx с value
83:           orb     al,xxx_a
84:           shll    eax,9
85:  // объединить поле yyy с value
86:           orw     ax,yyy_a
87:           shll    eax,12
88:  // объединить поле zzz с value
89:           orw     ax,zzz_a
90:           shll    ax,5
91:  // объединить поле www с value
92:           orb     al,www_a
93:           movl    value_a,eax
94:  };
```

```
 95:  // Форматный вывод результатов
 96:    cout << hex
 97:         << "Результирующее упакованное число (C++): " << value
 98:         << "\nРезультирующее упакованное число (Asm): " << value_a
 99:         << endl;
100:  };
101:     return 0;
102:  }
```

6. Тестирование программы.

Составим тестовые примеры, которые позволят найти ошибки в программе. Главное требование для правильного выбора тестов: исходные данные при распаковке битовых групп являются ожидаемыми результатами при упаковке, а ожидаемые результаты при распаковке битовых групп являются исходными данными при упаковке. Кроме того нужно выбрать такую группу тестовых случает которые бы в старших несущественных битах содержали биты, отличные от нуля. Это позволит проверить, очищаются ли старшие биты в каждой переменной перед упаковкой. С учетом этих соображений несложно выбрать значения исходных данных:

– битовые группы в исходных числах должны содержать все биты равные единице. В приведенной ниже таблице это тест 1.

– битовые группы в исходных числах должны содержать чередующиеся нулевые и единичные биты. В приведенной ниже таблице это тесты 2 и 3.

– каждое исходное число должно содержать единичные биты в своей битовой группе и нулевые биты в остальных разрядах. Всего у нас четыре исходных числа, поэтому таких тестов должно быть четыре. В приведенной ниже таблице это тесты 4-7.

– исходные числа должны содержать произвольные битовые значения в своей битовой группе, так чтобы в каждой битовой группе была хотя бы одна единица. Остальные разряды в исходных числах должны содержать нулевые значения. Таких тестов должно быть несколько. В приведенной ниже таблице это тесты 8-9.

– любые тесты, приведенные выше, но в несущественных битах должны быть не нули, а единицы. Так можно проверить реакцию на несущественные данные. В приведенной ниже таблице это тесты 10-11.

Тестовые примеры.

Номер	Исходные данные				Ожидаемый результат	Полученный результат	Цель теста
	xxx	yyy	zzz	www			
1	3f	1ff	fff	1f	FFFF FFFF		Все биты равны 1
2	2a	155	555	a	AAAA AAAA		Чередующиеся 1 и 0
3	15	aa	aaa	15	5555 5555		Чередующиеся 0 и 1
4	3f	0	0	0	FC000000		Все 1 в битовой группе xxx
5	0	0	0	1f	1F		Все 1 в битовой группе www

6	0	1ff	0	0	3FE0000		Все 1 в битовой группе yyy
7	0	0	fff	0	1FFE0		Все 1 в битовой группе zzz
8	02	11a	2b3	18	1234 5678		Произвольные биты
9	06	115	9e2	0d	1A2B 3C4D		Произвольные биты
10	ea	ff55	f555	ea	AAAA AAAA		Влияние несущественных битов
11	15	aa	aaa	15	5555 5555		Влияние несущественных битов

Протокол тестирования программы приводится ниже.

```
                (C) Дужий В.И., 2012
                Распаковка битовых групп

        Упаковать битовые группы, содержащиеся в целых числах,
                в 32-битовое целое число Value
Пожалуйста, введите 2 16-ые цифры для XXX (например, 5a): 3f
Пожалуйста, введите 3 16-х цифры для YYY (например,  9db): 0
Пожалуйста, введите 4 16-х цифры для ZZZ (например, 8e7f): 0
Пожалуйста, введите 2 16-х цифры для WWW (например,   14): 0
Результирующее упакованное число (C++): fc000000
Результирующее упакованное число (Asm): fc000000

Пожалуйста, введите 2 16-ые цифры для XXX (например, 5a): 0
Пожалуйста, введите 3 16-х цифры для YYY (например,  9db): 1ff
Пожалуйста, введите 4 16-х цифры для ZZZ (например, 8e7f): 0
Пожалуйста, введите 2 16-х цифры для WWW (например,   14): 0
Результирующее упакованное число (C++): 3fe0000
Результирующее упакованное число (Asm): 3fe0000

Пожалуйста, введите 2 16-ые цифры для XXX (например, 5a): 0
Пожалуйста, введите 3 16-х цифры для YYY (например,  9db): 0
Пожалуйста, введите 4 16-х цифры для ZZZ (например, 8e7f): fff
Пожалуйста, введите 2 16-х цифры для WWW (например,   14): 0
Результирующее упакованное число (C++): 1ffe0
Результирующее упакованное число (Asm): 1ffe0

Пожалуйста, введите 2 16-ые цифры для XXX (например, 5a): 0
Пожалуйста, введите 3 16-х цифры для YYY (например,  9db): 0
Пожалуйста, введите 4 16-х цифры для ZZZ (например, 8e7f): 0
Пожалуйста, введите 2 16-х цифры для WWW (например,   14): 1f
Результирующее упакованное число (C++): 1f
Результирующее упакованное число (Asm): 1f

Пожалуйста, введите 2 16-ые цифры для XXX (например, 5a): 3f
Пожалуйста, введите 3 16-х цифры для YYY (например,  9db): 1ff
Пожалуйста, введите 4 16-х цифры для ZZZ (например, 8e7f): fff
Пожалуйста, введите 2 16-х цифры для WWW (например,   14): 1f
Результирующее упакованное число (C++): ffffffff
Результирующее упакованное число (Asm): ffffffff

Пожалуйста, введите 2 16-ые цифры для XXX (например, 5a): 2a
Пожалуйста, введите 3 16-х цифры для YYY (например,  9db): 155
Пожалуйста, введите 4 16-х цифры для ZZZ (например, 8e7f): 555
Пожалуйста, введите 2 16-х цифры для WWW (например,   14): a
Результирующее упакованное число (C++): aaaaaaaa
Результирующее упакованное число (Asm): aaaaaaaa
```

```
Пожалуйста, введите 2 16-ые цифры для XXX (например, 5a): 15
Пожалуйста, введите 3 16-х цифры для YYY (например,  9db): aa
Пожалуйста, введите 4 16-х цифры для ZZZ (например, 8e7f): aaa
Пожалуйста, введите 2 16-х цифры для WWW (например,   14): 15
Результирующее упакованное число (C++): 55555555
Результирующее упакованное число (Asm): 55555555

Пожалуйста, введите 2 16-ые цифры для XXX (например, 5a):^C
```