**Индивидуальное задание №3**

**1. Задание**

Необходимо реализовать на языке ассемблера задание, предложенное в варианте. Программа оформляется в виде ассемблерной вставки языка C. Программный код не должен содержать комментариев. Обучающийся должен быть способен пояснить каждую строку программы по просьбе преподавателя. Индивидуальное задание засчитывается, если программный код успешно компилируется, программа выполняется без ошибок и обучающийся отвечает на все вопросы преподавателя по программному коду.

**2. Справочные данные**

В некоторых задачах программирования необходимо отслеживать или изменять состояние битов в переменных или регистрах. Для решения этой задачи необходимо использовать логические операции и битовые сдвиги.

or – команда логического сложения (логическое «ИЛИ»). Содержит два операнда. Выполняет поразрядную операцию логического сложения над битами операндов. Результат записывается на место первого операнда, второй операнд не меняется.

and – команда логического умножения (логическое «И»). Содержит два операнда. Выполняет поразрядную операцию логического умножения над битами операндов. Результат записывается на место первого операнда, второй операнд не меняется.

xor – команда логического вычитания (логическое «исключающее ИЛИ»). Содержит два операнда. Выполняет поразрядную операцию логического вычитания над битами операндов. Результат записывается на место первого операнда, второй операнд не меняется.

not – команда логического отрицания (логическое «НЕ»). Содержит один операнд. Выполняет поразрядное инвертирование битов операнда. Результат записывается в сам операнд.

test – команда проверки методом логического умножения. Содержит два операнда. Выполняет поразрядную операцию логического умножения над битами операндов. Операнды не меняются, но изменяется состояние флагов zf, sf и pf.

bsf – команда сканирования битов вперед. Содержит два операнда. Сканирует биты второго операнда от младшего к старшему, в поисках первого бита, установленного в 1. Если такой бит присутствует во втором операнде, то в первый операнд заносится номер (позиция) этого бита в виде целочисленного значения. Если во втором операнде все биты равны 0, то первый операнд не меняется и устанавливается флаг нуля (zf=1).

bsr – команда сканирования битов в обратном порядке. Содержит два операнда. Сканирует биты второго операнда от старшего к младшему, в поисках первого бита, установленного в 1. Если такой бит присутствует во втором операнде, то в первый операнд заносится номер (позиция) этого бита в виде целочисленного значения. Если во втором операнде все биты равны 0, то первый операнд не меняется и устанавливается флаг нуля (zf=1).

bt – команда проверки бита. Принимает операнд и смещение бита. Переносит значение бита операнда с номером (позицией) «смещение бита» в флаг cf. Доступ к флагуможно получить через команды условного перехода.

bts – команда проверки и установки бита. Принимает операнд и смещение бита. Переносит значение бита операнда с номером (позицией) «смещение бита» в флаг cf и устанавливает указанный в операнде бит в 1.

btr - команда проверки и сброса бита. Принимает операнд и смещение бита. Переносит значение бита операнда с номером (позицией) «смещение бита» в флаг cf и сбрасывает указанный в операнде бит в 0.

btc – команда проверки и инверсии бита. Принимает операнд и смещение бита. Переносит значение бита операнда с номером (позицией) «смещение бита» в флаг cf и меняет значение указанного в операнде бита на противоположное.

Команды сдвига имеют одинаковую структуру. Каждая команда сдвига принимает операнд, в котором выполняется сдвиг и счетчик сдвигов, который показывает, на сколько бит нужно выполнить сдвиг. Счетчик сдвигов варьируется в диапазоне от 0 до 31, поэтому в качестве счетчика использовать можно только регистр-счетчик, размером в 1 байт, т.е. регистр cl (можно в явном виде указать число в качестве счетчика сдвигов). Все команды сдвига очередной выдвигаемый бит заносят во флаг cf.

shl – логический сдвиг влево. Содержимое операнда сдвигается влево на количество бит, указанных в счетчике сдвигов. При этом справа, в позицию младшего бита на каждом шаге заносятся нули.

shr – логический сдвиг вправо. Содержимое операнда сдвигается вправо на количество бит, указанных в счетчике сдвигов. При этом слева, в позицию старшего бита на каждом шаге заносятся нули.

sal – арифметический сдвиг влево. Аналогична команде shl, но устанавливает флаг of, если очередной выдвигаемый бит меняет знак операнда.

sar – арифметический сдвиг вправо. Содержимое операнда сдвигается вправо на количество бит, указанных в счетчике сдвигов. При этом слева, в позицию старшего бита на каждом шаге заносятся нули. При этом команда сохраняет знак операнда, восстанавливая его после каждого очередного бита.

rol – циклический сдвиг влево. Содержимое операнда сдвигается влево на количество бит, указанных в счетчике сдвигов. Выдвигаемые влево биты заносятся в этот же операнд справа.

ror – циклический сдвиг вправо. Содержимое операнда сдвигается вправо на количество бит, указанных в счетчике сдвигов. Выдвигаемые вправо биты заносятся в этот же операнд слева.

rcl – циклический сдвиг влево через перенос. Содержимое операнда сдвигается влево на количество бит, указанных в счетчике сдвигов. Выдвигаемые влево биты поочередно становятся значениями флага cf. Выдвигаемые из флага cf заносятся в операнд справа.

rcr – циклический сдвиг вправо через перенос. Содержимое операнда сдвигается вправо на количество бит, указанных в счетчике сдвигов. Выдвигаемые вправо биты поочередно становятся значениями флага cf. Выдвигаемые из флага cf заносятся в операнд слева.

**3. Варианты заданий**

**3-1.** С помощью логических операций реализуйте запись битов в данной ячейки памяти, размером в 2 байта, в обратном порядке.

**3-2.** Реализуйте логическую арифметику, при которой в одном байте хранилось бы 8 логических переменных.

**3-3.** Реализуйте логическую арифметику, при которой в одном слове хранилось бы 16 логических переменных.

3-**4.** Реализуйте с помощью логических операций вставку заданного значения (0 или 1) в любой бит байта со сдвигом вправо и потерей крайнего бита.

**3-5.** Реализуйте с помощью логических операций вставку заданного значения (0 или 1) в любой бит байта со сдвигом влево и потерей крайнего бита.

**3-6.** Реализуйте с помощью логических операций вставку заданного значения (0 или 1) в любой бита байта с потерей исходного значения.

**3-7.** Реализуйте с помощью логических симметричное отображение младшего байта слова на старший байт с потерей исходных значений.

**3-8.** Реализуйте с помощью логических симметричное отображение старшего байта слова на младший байт с потерей исходных значений.

**3-9.** Реализуйте с помощью логических операцию, при которой старший байт слова логически складывается с младшим байтом, записанным в обратном порядке.

**3-10.** Реализуйте с помощью логических операцию, при которой старший байт слова логически умножается на младший байт, записанный в обратном порядке.

**3-11.** Реализуйте с помощью логических операцию, при которой младший байт слова логически складывается со старшим байтом записанным в обратном порядке.

**3-12.** Реализуйте с помощью логических операцию, при которой младший байт слова логически умножается на младшим байтом, записанный в обратном порядке.

**3-13.** Одномерный вектор имеет восемь координат, каждая из которых может принимать значения 0 или 1. Реализуйте с помощью битовых операций векторную алгебру (сложение, вычитание и скалярное произведение векторов).

**3-14.** С помощью битовых операций реализуйте перестановку двух любых битов в данном слове местами. Остальные биты остаются неизменными.

**3-15.** Заданы два числа ≤15. Найти их сумму и с помощью битовых операций записать оба слагаемых и их сумму одно за другим в ячейку памяти размером в два байта без пропуска битов.

**3-16.** Заданы два числа ≤15. Найти их произведение и с помощью битовых операций записать оба множителя и их произведение одно за другим в ячейку памяти размером в два байта без пропуска битов.

**3-17.** Заданы два числа ≤ 255. Найти целую часть от деления первого числа на второе и остаток деления. Записать две полученные величины одно за другим без пропуска битов в в ячейку памяти размером в два байта.

**3-18.** Дано слово из трех букв (например FCS). С помощью битовых операций запишите последовательность ASCII кодов его букв в ячейку памяти минимального размера. Слово может быть и другим, программа должна его запрашивать. Запрашивать можно на языке C.

**3-19.** Дано пятизначное число. (Может быть например такое 00001). С помощью битовых операций запишите его в ячейку памяти минимального размера. Запрашивать число можно на языке C.

**3-20.** В двухбайтной ячейке памяти записана последовательность нулей и единиц. С помощью битовых операций переставьте биты так чтобы сначала шли все единицы, а затем все нули.