Машины-2 (массивы – дополнительные задачи)

РАБОТА С 64-БИТНЫМИ ЧИСЛАМИ:

Все четыре предлагаемых далее задачи – сложные (и поэтому **необязательные**), но будет замечательно, если вы с ними справитесь, это будет ваша заслуженная победа
(со сдачей этих задач можно не торопиться, их надо делать тщательно и без спешки).

Общие замечания по выполнению этих задач.

Во всех этих задачах можно (при необходимости и желании) использовать работу со **стеком,** вводить свои **процедуры** или **макросы** (если выполнение этих задач будет происходить в конце семестра). Если решение задач будет дано до прохождения тем **"стек", "процедуры"** и **"макросы"**, то тогда пользуйтесь теми знаниями, которые вы уже успели получить (после изучения темы **"массивы"** знаний вполне достаточно, чтобы решать предлагаемые ниже задачи).

"Большие числа-1" **(15 очков)**

Описать в программе массив из десяти восьмибайтовых чисел (б/зн) с некоторыми начальными значениями:
`D dQ 1,502,40034,1234567,.....`

Числа должны быть упорядочены по возрастанию, последним должно стоять самое большое беззнаковое восьмибайтовое число (Какое число самое большое? Разберитесь самостоятельно!)

Задачей программы будет **распечатать каждое из этих больших чисел в десятичном виде** (*замечание*: по макрокоманде **outword** можно выводить на экран в настоящий момент 8-ми, 16-ти, 32-х или 64-х битовые числа; наша задача состоит в том, чтобы понять принцип, по которому можно выводить на экран любые длинные числа, не только 64-битовые; поэтому вывод 64-битовых чисел мы реализуем по нижеизложенному алгоритму, без использования макрокоманды **outword**). *Внимание*: в качестве подтверждения правильности вашего вывода рядом с каждым числом напечатать это же число, но с помощью макрокоманды **outword**. **Числа должны совпасть**.

Рекомендуется завести **вспомогательную переменную Buf размером dq** и использовать её в качестве буфера, куда будет копироваться очередной обрабатываемый элемент массива **D** и которая далее будет использоваться для хранения промежуточных результатов деления (чтобы не портить исходные данные):
`Buf dq ?`
Следует также завести **байтовый массив в расчете на N возможных цифр**, завершить описание числом **0** (для возможности вывода ответа по макрокоманде **outstr**) и настроиться модификатором (например **EBX**) на последний элемент этого массива:
`Ans db N dup(?),0`
Чему должно быть равно значение константы N? Разберитесь самостоятельно.
Идея решения. Нужно последовательно делить число на **10** и выделять остатки (получая тем самым искомые цифры, начиная с самой младшей, и отбрасывая в числе найденные цифры). Сложность состоит в том, что при делении на **10** 8-байтового числа целая часть может не поместиться в формат двойного слова (то есть сверхдлинное деление – не подойдёт!). Поэтому предлагается применить следующий **алгоритм**: Деление выполнять в **два этапа.** *1 этап*. Сначала обнуляем **EDX**, а в **EAX** загружаем старшее двойное слово из **Buf** и делим на **10** (сверхдлинным делением). Целая часть оказывается в **EAX**, а в **EDX** – остаток. Сохраняем **EAX** на место старшего двойного слова (т.е. по адресу **Buf+4**). *2 этап*. Загружаем в **EAX** младшее двойное слово из **Buf** (а в **EDX** при этом находится остаток от деления на 1 этапе) и снова делим на **10** (сверхдлинным делением). Сохраняем полученную целую часть (из **EAX**) на место младшего двойного слова (т.е. по адресу **Buf**). А в **EDX** (точнее в **DL**) – **младшая цифра искомой десятичной записи**. Помещаем найденную цифру в массив **Ans** по текущему положению модификатора **EBX** (не забываем преобразовать попутно цифру-число в цифру-символ). Сдвигаем модификатор в массиве **Ans** на байт влево. И повторяем процесс по получению (от конца к началу) десятичных цифр (в цикле выполняем последовательно *по два этапа*). Конец процесса – получение после *2-го этапа* на **EAX** нулевой целой части, значит на **EDX** (точнее на **DL**) – найдена **самая старшая десятичная цифра в записи числа.** С учётом текущего значения в **EBX** выводим сформированный в **Ans** ответ по макрокоманде **outstr** (вывод должен быть *без незначащих нулей*).
Этот алгоритм нужно применить по отношению к каждому элементу массива **D** (в цикле 10 раз).
В результате работы программы на экране должно быть распечатано 10 заданных вами восьмибайтовых слов (каждое – в отдельной строке).

“ Большие числа -2” (15 очков)

Выполняется после успешного завершения работы над задачей **“Большие числа -1”**

Постановка задачи прежняя, но большие числа (б/зн) нужно предварительно **ввести с клавиатуры (посимвольно в десятичном виде) – это и является целью данной задачи** (*замечание*: макрокоманда **inint** не умеет вводить 64-битные числа). Максимально допускается ввести 10 чисел. Числа отделяются друг от друга запятой, за последним числом – точка. Программа должна выдавать ***сообщение об “ошибке ввода”*** и завершать свою работу в том случае, если очередное вводимое число записано некорректно (встретился неожиданный символ) или не укладывается в 64-битовый формат.

Вводимые числа следует размещать в качестве элементов в массиве **D** с описанием:

```
D dq 10 dup(?);
```

Далее введённые числа распечатываются, как и в предыдущей задаче. Разрешается распечатывать числа с помощью макрокоманды **outword** (так как вручную распечатывать вы уже научились в первой задаче).

Основная сложность: правильно ввести с клавиатуры очередное длинное число (б/зн), пользуясь **только посимвольным вводом**. Предлагается для ввода использовать *схему Горнера*, считывая одна за другой десятичные цифры длинного числа. Но т.к. работаем с восьмибайтовым числом (т.е. вводим 64-битовую величину), то умножение на **10** восьмибайтового числа придётся разбить на 2 этапа. *1 этап*. Умножение (сверхдлинное) младшей половины формируемого восьмибайтового числа на **10** и сохранение результата (в виде восьмибайтового числа) во временном месте (например, в: **Buf dq ? ;** или лучше на паре свободных регистров). *2 этап*. Умножение (сверхдлинное) старшей половины формируемого восьмибайтового числа на **10** и соответствующая корректировка старшей половины двойного слова **Buf.** Добавление к полученному восьмибайтовому числу (в **Buf**) прочитанной цифры (не забыть предварительно преобразовать её из цифры-символа в цифру-число). В процессе корректировок (сложений) возможны “переносы”, которые следует учесть. Не забыть поместить полученный результат из **Buf** в массив **D** после очередного шага работы *по схеме Горнера*.

Как отловить (после *2-го этапа*) выход из диапазона представимости восьмибайтового числа (при вводе числа):
1) После умножения старшей половины формируемого слова на **10** результат (т.е. произведение) занимает удвоенный формат (т.е. **CF=1**): **jC err**
2) После сложения старшей половины **Buf** и полученного в **EAX** произведения (одинарного формата) произошёл “перенос”: **jC err**
3) После добавления к полученному двойному слову (в **Buf**) прочитанной цифры произошёл “перенос”: **jC err**

“ Большие числа -3” (15 очков)

Выполняется после успешного завершения работы над задачей **“Большие числа-2”.**

Ввести с клавиатуры три больших (укладывающихся в 64-битовый формат) беззнаковых числа (записанных десятичными цифрами) и разместить их последовательно в массиве (с описанием: **D dq 3 dup(?)**). Между вводимыми числами запятая, в конце – точка. Требуется **сравнить эти три числа** и распечатать их (каждое – с новой строки) в порядке **неубывания (≤).** Распечатывать числа разрешается с помощью макрокоманды **outword.**

Рекомендация: сначала сравнить 1-ое число со 2-ым и меньшее из них поместить на 1-е место (т.е. на место 1-го элемента массива), а большее – на 2-е (т.е. на место 2-го элемента массива). Затем сравнить 1-ое с 3-им и меньшее их них поместить на 1-е место, а большее – на 3-е. В итоге самое маленькое число окажется на 1-ом месте. А далее следует сравнить и переупорядочить числа на 2-м и 3-ем местах. Остаётся только распечатать все три элемента массива **D.** (*сравнение 64-битных чисел* – см. задачу **4.8**).

“ Большие числа -4” (15 очков)

Выполняется после успешного завершения работы над задачей **“ Большие числа -3”.**

Ввести с клавиатуры два больших числа (записанных десятичными цифрами и укладывающихся в 64-битовый формат). Между числами запятая, в конце – точка. Перемножить эти два числа и вывести ответ в виде: ***число1 * число2 = результат перемножения***.

Программа должна выдавать ***сообщение об "ошибке ввода"*** и завершать свою работу в том случае, если очередное вводимое число записано некорректно (встретился неожиданный символ) или не укладывается в 64-битовый формат.

Формировать ответ в массиве **`Ans db N dup(?),0;`** в ответе не будет больше десятичных N цифр (самостоятельно определить, чему равно N)

Идея получения этих цифр аналогична идее, сформулированной в задаче **" Большие числа -1".**

Отличие: т.к. работаем (выводим в десятичном виде) с 16-байтовым числом (результатом перемножения двух восьмибайтовых чисел), то деление на **`10`** выполняем *в 4 этапа* (от старшего двойного слова к младшему). 4 этапа – 4 шага цикла. На каждом шаге с помощью модификатора сдвигаемся к следующему обрабатываемому двойному слову.

Здесь приведена схема умножения 8-байтовых слов

(*см. ранее заданную домашнюю задачу, к ней был ответ*):

Умножение над длинными числами".

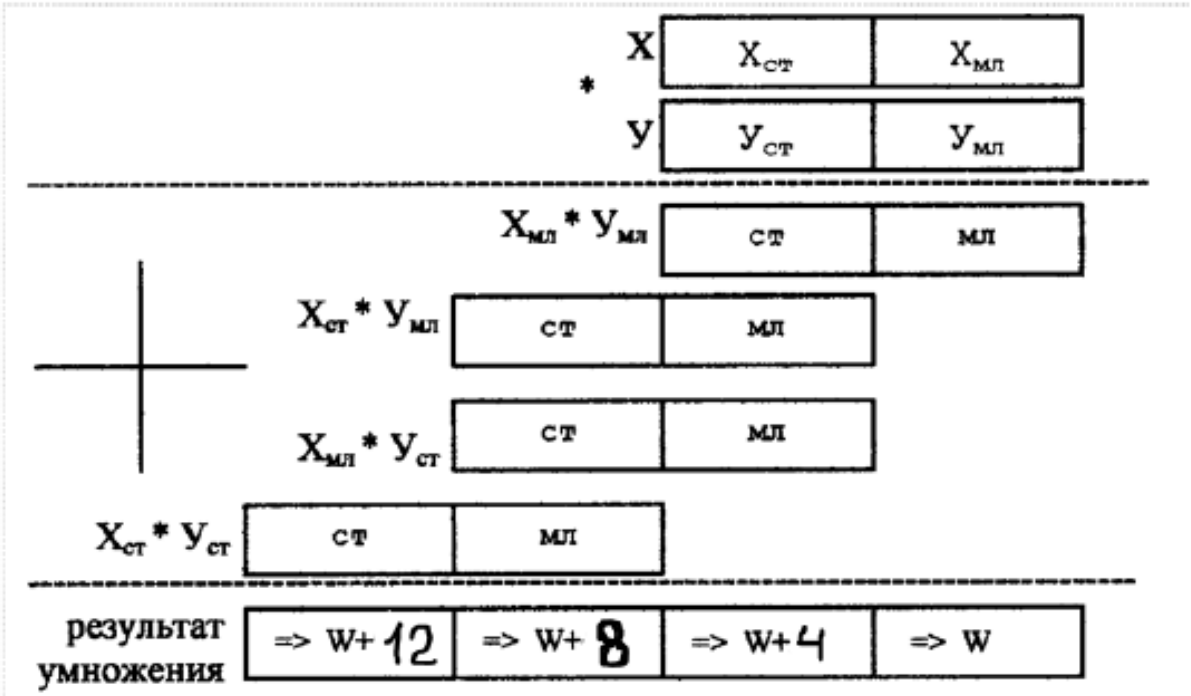
```
X DQ ?    ; 64 бита
Y DQ ?    ; 64 бита
W DQ ?,?  ; 128 битов
```

Реализовать операцию умножения над «длинными» числами: W:= X*Y

(считать, что X и Y – целые без знака).

Рекомендация по решению дополнительной задачи:

использовать следующую схему решения:



Пояснения по предложенной схеме: Можно не экономить на числе обращений к памяти (*иначе будет путаница с регистрами – на каком из них какая часть результата лежит*)!!! Как можно раньше заполнить ячейки W, W+4, W+8, W+12 появляющимися в процессе умножения промежуточными значениями (не стоит обнулять их с самого начала). Ответ накапливать постепенно, корректируя значения в ячейках W+4, W+8, W+12 (в связи с возможными переносами при сложении отдельных частей).

Желаю удачи и вдохновения!