Машины-6 МОДУЛИ (две обязательных задачи)

Задача 1 (обязательная) «Ввод-вывод четверичного числа»

В задаче рассматривается работа с упакованными четверичными числами.
Для представления таких чисел будем использовать двойные слова (занимающие в памяти, как известно, 32 разряда). Каждая цифра четверичного числа в этом случае кодируется парой соответствующих двоичных цифр: 0 → 00, 1 → 01, 2 → 10, 3 → 11. *Например*, 32-битное двоичное число **00**00**10**11**01**00**00**11**01**10**00**11**10**10**00**01 является упакованным представлением четверичного числа **2310031203220 1** (незначащие нули в четверичной записи отброшены). Заметим, что это же 32-битное двоичное число (**00**00**10**11**01**00**00**11**01**10**00**11**10**10**00**01) является также внутренним машинным представлением беззнакового десятичного числа **188965793**.

Формулировка задания.

Написать программу из *трёх модулей*.

В *головном модуле* описать 32-битную переменную с именем **X** (содержимое которой трактуется как упакованное представление некоторого четверичного числа). В *1-ом вспомогательном модуле* описать две **общедоступные** процедуры со стандартными соглашениями о связях stdcall: процедуру **In4(X)** и процедуру **Out4(X)** (их имена будут **внешними** для *головного модуля*).

Процедура **In4(X)** – для посимвольного ввода цифр четверичного числа (*конец записи четверичного числа* - ***пробел***) и формирования 32-битного упакованного представления введённого числа. Полученное представление возвращается через параметр процедуры (следовательно, параметр необходимо передать ***по ссылке***). Считать, что величина вводимого числа укладывается в формат двойного слова.

Процедура **Out4(X)** – для вывода содержимого двойного слова **X** в четверичном виде (цифра за цифрой). Параметр передать в процедуру ***по значению***.

Из *головного модуля* сначала вызывается процедура **In4(X)**, с помощью которой вводится четверичное число и присваивается параметру процедуры. Затем из *головного модуля* вызывается процедура **Out4(X)**, по которой введённое число распечатывается в четверичном виде.

Введённое и напечатанное числа должны, естественно, совпасть.

Затем из *головного модуля* управление передаётся (по внешней метке) во *2-ой вспомогательный модуль*, где печатается значение переменной **X** в десятичном виде б/зн (по **outword X**), на этом работа программы завершается (без возврата *в головной модуль*). Из *2-го вспомогательного модуля* доступ к переменной **X** осуществляется напрямую по её имени (то есть имя X во *2-ом вспомогательном модуле* следует объявить как *внешнее*, а *в головном модуле* – как

общедоступное). Продумать также вопрос о **точке входа** во *2-ой вспомогательный модуль*.

Требование: пользоваться командами умножения и деления при решении задачи – запрещено.

Замечания по реализации процедуры **In4(X).**
Пусть, например, ответ накапливается на **EAX** по схеме Горнера:
EAX := EAX*4 + (*очередная четверичная цифра***)**
Так как **4=2²**, то каждый раз достаточно сдвигать влево на 2 позиции содержимое **EAX** и затем складывать (с помощью **or**) младшую часть **EAX** с новой четверичной цифрой. Получим, с одной стороны, упакованное представление введённого четверичного числа, а с другой стороны – 32-битовое двоичное представление некоторой беззнаковой величины.

Замечания по реализации процедуры **Out4(X).**
Для вывода числа в четверичном виде следует поступить так. Запустить цикл из 16 шагов (максимальное количество цифр в четверичной записи числа). На каждом шаге – сдвигать циклически (**rol**) содержимое **EAX** сразу на 2 разряда влево. В результате – в двух младших разрядах **EAX** окажется пара двоичных цифр, соответствующая очередной четверичной цифре. Выделяем эту цифру, используя маску **11b** (содержимое **EAX** при этом не портим!) и выводим цифру на печать (*незначащие* нули – тоже выводить).

Требования к оформлению.
Головной модуль назвать **main.asm**
Вспомогательные модули назвать **unit1.asm** и **unit2.asm**
Останов (завершение) программы – в модуле **unit2** !

Сдача задания: прислать в архиве три исходных модуля (**main.asm**, **unit1.asm**, **unit2.asm**) и исполняемый файл **main.exe**. Просьба перед **exit** ставить **pause** (я запускаю ваши программы непосредственно из архива, мне важно успеть увидеть результат работы программы).

Задача 2 (обязательная) «Сверхдлинное умножение»

Пусть **A**, **B** – двойные слова, **Z** – учетверённое слово, числа без знака.

Присваивание **Z := A*B** можно реализовать, используя следующую идею.

Пусть $B = b_{31}*2^{31} + b_{30}*2^{30} + \dots + b_2*2^2 + b_1*2 + b_0$, $b_i \in \{0,1\}$.

(т.е. $b_{31}\ b_{30}\ \dots\ b_2\ b_1\ b_0$ – цифры двоичного представления числа B)

Тогда $A*B = A*(b_{31}*2^{31} + b_{30}*2^{30} + \dots + b_2*2^2 + b_1*2 + b_0) =$

$$= A*b_{31}*2^{31} + A*b_{30}*2^{30} + \dots + A*b_2*2^2 + A*b_1*2 + A*b_0 =$$

$$= ((\dots((0*2 + A*b_{31})*2 + A*b_{30})*2 + \dots + A*b_2)*2 + A*b_1)*2 + A*b_0$$

Умножение на **2** реализуется сдвигом влево на один разряд. Действие **+ A*b_i** означает, что нужно прибавить **A**, если **b_i=1**.

Приходим к такому **алгоритму**:

```
{ Здесь знак << обозначает логический сдвиг содержимого первого
операнда влево}
Z:=0;
for i:=31 downto 0 do begin
     Z<<1; {Z*2}
     B<<1; {CF=bi}
     if CF=1 then Z:=Z+A
end

{Величины, используемые в цикле, следует хранить на регистрах}
```

Формулировка задания. Написать программу из *трёх модулей*.

В *головном модуле* описать:
A и **B** - 32-битные переменные (б/зн), **Z** – 64-битную переменную.

В *1-ом вспомогательном модуле* описать **общедоступную** процедуру **Mult(A,B,Z)** со стандартными соглашениями о связях stdcall. Процедура реализует умножение **Z := A*B** согласно вышеприведённому алгоритму. Параметры **A** и **B** (двойные слова) – передаются по значению, **Z** (учетверённое слово) – по ссылке.

Головной модуль вводит (по **inint**) значения переменных **A** и **B**, а затем вызывает процедуру **Mult(A,B,Z)** для вычисления произведения **Z := A*B**. Полученный (в переменной **Z**) ответ головной модуль выводит (по **outword**) на экран, после чего передаёт управление по *2-ой вспомогательный модуль*.

2-ой вспомогательный модуль реализует (через команду **mul**) умножение **Z := A*B**, работая с переменными **A**, **B** и **Z** напрямую по их именам. Полученный (в переменной **Z**) ответ выводит (по **outword**) на экран. Ответы (в идеале) должны совпасть. Работа трёхмодульной программы завершается во *2-ом вспомогательном модуле*.

Требования к оформлению.
Головной модуль назвать **main.asm** . *Вспомогательные модули* назвать **unit1.asm** и **unit2.asm** . Останов – в модуле **unit2** !

Сдача задания: прислать в архиве три исходных модуля (**main.asm**, **unit1.asm**, **unit2.asm**) и исполняемый файл **main.exe**. Просьба перед **exit** ставить **pause** (ваши программы запускаются непосредственно из архива, важно до закрытия консольного окна успеть увидеть результаты работы программы).