Машины-5 БИТОВЫЕ ОПЕРАЦИИ

(Биты-1, Биты-2, Биты-3 – обязательные задачи)

Во всех трёх задачах процедуры использовать не надо

Биты - 1 "Ввод-вывод двоичного числа" (обязательная !)

Условие. Ввести (посимвольно с помощью макрокоманды **inchar**) число, записанное в двоичном виде (считать, что число записано корректно и содержит не более 32 двоичных цифр, за числом следует **пробел**). Сохранить прочитанное число в регистре **EAX.** Вывести содержимое регистра **EAX** в двоичном виде без незначащих нулей (при правильном решении **ввод и вывод должны совпасть**!). *Требование*: командами умножения и деления пользоваться в решении запрещено.

Рекомендации по решению задачи.

Ввод

ИДЕЯ. Обнуляем **EAX** (**xor**). *В цикле*: читаем (по макрокоманде **inchar**), например, в **DL**, очередной символ (`'0'` или `'1'`, пробел – выход из цикла); переводим прочитанную символ-цифру в число (`0` или `1`) вычитанием из кода прочитанного символа кода цифры ноль, оставляя результат в **DL**. Готовим **EAX** к приему следующей цифры (линейным сдвигом содержимого **EAX** влево на **1** разряд); логически складываем содержимое **AL** и **DL.**

Вывод За основу решения можно выбрать любую идею (см. рекомендации ниже) или свою.

ИДЕЯ - 1.

1) Сдвигаем (**shl**) содержимое **EAX** на **1** разряд влево (в результате чего уходящий влево бит попадет в **CF**), извлекаем значение из **CF** в какой-нибудь регистр, например в **DL,** с помощью команды **adc DL,0** (предварительно обнулив этот регистр перед очередным сдвигом влево); делаем это в цикле максимально 32 раза при **ECX=32;** как только в **DL** попадет *первая значащая единица* (т.е. старшая значащая двоичная цифра), то выходим из цикла досрочно (здесь удобна команда цикла **loopz –** см. про эту команду в *примечании* ниже) если в числе нет единиц, то цикл проработает до конца, после чего **ECX=0.**

2) Основываясь на полученном значении **ECX**, входим (не забыть про команду **jECXZ …**) в новый цикл: вывод (по **outword**) содержимого **DL** (там находится двоичная цифра, полученная в результате последнего сдвига **EAX** на *1 разряд влево*), а далее - очистка **DL** и новый сдвиг **EAX** влево с занесением содержимого **CF** в **DL**. Не забудьте вывести последнюю двоичную цифру после выхода из 2-го цикла!
Замечание: можно и не использовать команду **adc** …, а работать с командами перехода **jc** … или **jnc** …, проверяющими **CF)**

ИДЕЯ - 2.

1) Циклически (**rol**) сдвигаем содержимое **EAX** *влево на 1 разряд*. После каждого сдвига копируем полученный на **AL** результат на другой регистр, например, **DL**. С помощью **and DL,1** "гасим" все биты, кроме младшего (т.е. правого, интересующего нас). Заметим, что после каждого обращения к команде **and** устанавливается **ZF** (флаг нуля). Делаем это в цикле максимально 32 раза при **ECX=32;** как только найдем *первую значащую единицу*, то сразу досрочно выходим из цикла (здесь удобен цикл **loopz** – см. про эту команду в *примечании* ниже)**;** если в числе нет единиц, то цикл проработает до конца, после чего **ECX=0**.

2) Основываясь на полученном значении **ECX**, входим (не забыть про команду **jECXZ …**) в новый цикл: вывод (по **outword**) содержимого **DL** (там находится двоичная цифра, полученная в результате последнего циклического сдвига **EAX** на *1 разряд влево* и переноса этой цифры в младший разряд **DL**), и новый циклический сдвиг влево и т.п. Не забудьте вывести последнюю двоичную цифру после выхода из 2-го цикла!
Примечание. Действие команды **loopz L**:

ECX := ECX-1; if (ECX<>0) and (ZF=1) then goto L

Биты - 2 **“Ввод-вывод шестнадцатеричного числа”** **(обязательная !)**

Условие. Ввести (посимвольно) число, записанное *в шестнадцатеричном виде* (считать, что число записано корректно и содержит от 1 до 8 шестнадцатеричных цифр, за числом следует **пробел**). Сохранить прочитанное число в регистре **EAX.** Вывести содержимое регистра **EAX** в виде 8-значного шестнадцатеричного числа. Замечание: в качестве *буквенных цифр* использовать *заглавные латинские буквы* (как при вводе, так и при выводе). *Требование*: командами умножения и деления пользоваться в решении запрещено.

Рекомендации по решению задачи.

Ввод
ИДЕЯ. Обнулить **EAX** до входа в основной цикл. Далее в цикле (пока не пробел):
1) Чтение новой 16-ой цифры (например, в **BL**) и получение (сообразить самостоятельно, как это сделать) числовой величины этой цифры *в младших 4-х битах* **DL** ;
2) подготовка **EAX** к учёту прочитанной 16-ой цифры (линейным сдвигом **EAX** на 4 бита влево);
3) добавление в освободившиеся 4 младших бита **AL** новой 16-ой цифры (**or AL,DL**)

Вывод *см.* **задачу_3** ролика **bits-3.mp4,** слайды **bits-3(rotate-2_1).png**, **bits-3(rotate-2_2).png, bits-3(rotate-2_3).png**)

Биты - 3 **“Анализ и преобразование двоичного числа”** **(обязательная !)**

Условие. Ввести (посимвольно) число, записанное в двоичном виде (считать, что это число записано корректно и содержит ровно 32 двоичных цифры), поместить прочитанное число в регистр **EAX**. Проверить, получилось ли симметричным битовое представление регистра **EAX** (нужны все 32 двоичных цифр!).
Если **ДА**, то напечатать “СИММЕТРИЧНО” и **обнулить два ближайших к середине (центру) единичных бита этого регистра (если такие есть)**.
Если **НЕТ**, то напечатать слово “НЕСИММЕТРИЧНО” и **поменять местами два крайних (левый и правый) бита в EAX**.
Вывести результат преобразований (над содержимым **EAX**) в двоичном виде (обязательно все 32 двоичных цифры, в т.ч. и незначащие нули). *Требование к решению*. При вводе после каждой четвёрки двоичных цифр **ставить обязательно один пробел**, при выводе – аналогично (для удобства восприятия двоичной информации), иначе проверять задачу не буду.

Подсказка как проверить на симметричность: записать в **EBX** перевёрнутое значение **EAX** (см. **задача_1** ролика **bits-3.mp4,** слайд **bits-3(carry-1).png**), затем сравнить **EAX** и **EBX** с помощью команды **xor**.

Остальные идеи решения: ваши собственные. Желательно их кратко сформулировать в качестве комментария к программе. Тогда будет проще проверять ваше “произведение”.
