МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФГБОУ ВО «ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ И.С.ТУРГЕНЕВА»

Кафедра «Программная инженерия»

**Отчет**

По лабораторной работе №3  
на тему:  
“Команды условного и безусловного переходов. Организация ветвлений и циклов в программе ”

Работу выполнили:  
Студенты группы 01-ИТ

Колчев Д.Н.  
Галькеев А. Ш.

Проверил:  
Доцент кафедры  
программной инженерии

Конюхова О.В.

Орёл, 2020

1. **Структура программы:**

d\_s segment

num1 db 1Eh; ; 1-е число в 16-й системе (30)  
num2 db 12h; ; 2-е число в 16-й системе (18)  
NOD db 0 ; Ячейка, где храни

d\_s ends

c\_s segment  
assume ds:d\_s, cs:c\_s  
begin:  
mov ax, d\_s  
mov ds, ax

mov bh, num1 ; помещаем числа в регистры.  
mov bl, num2

again: ; метка начачала алгоритма Евклида  
cmp bh, 0   
jnz bhnz  
jmp exit ; выход если bh = 0

bhnz: ; while bh != 0 and bl != 0  
cmp bl, 0  
jnz blnz ; переход к условию сравнения  
jmp exit ; выход если bl = 0

blnz:  
cmp bh, bl ; if bh > bl  
jg bhmore ; переход при условии bh > bl

mov ax, 0   
mov al, bl ; if bh < bl  
div bh ; ax / bh = ax  
mov bl, ah ; ah (целое) ah (остаток)  
jmp again

bhmore: ; начало метки - if bh > bl  
mov ax, 0 ; ax / bl = ax  
mov al, bh ; ah (целое) ah (остаток)  
div bl  
mov bh, ah  
jmp again ; повтор цикла Евклида

exit:  
add NOD, bl ;сохраняем НОД в ячейку памяти NOD  
add NOD, bh

mov cl, 8 ; (int i = 0; i < 8; i++)  
mov ax, 0   
mov al, NOD ; передаём в al число, биты которого будем сдв.  
mov dx, 0  
mov bx, 0  
cycle: ; начало цикла loop  
rcr al, 1 ; сдвигаем бит нашего NOD  
rcl bl, 1 ; из FC сохраняем рез. сдвига в регистр bl  
add dl, bl ; сохраняем число единиц  
mov bx, 0  
loop cycle ; конец цикла loop (cl - 1)

mov ah, 4ch  
int 21h ; завершение программы, прерывание 21h  
c\_s ends  
end begin

1. **Контрольные вопросы:**
2. **Ветвления в алгоритмах. Реализация ветвлений на языке ассемблера.** Ветвление (условный переход) происходит только при соблюдении определённого условия, в противном случае выполняется следующая по порядку команда программы. Условием, на основании которого осуществляется переход, чаще всего выступают признаки результата выполнения предшествующей арифметической или логической команды (без флага AF). Каждый из признаков фиксируется в своём разряде регистра флагов PSW. Возможен и такой подход, когда решение о переходе принимается в зависимости от состояния одного из регистров общего назначения, куда предварительно помещается результат операции сравнения CMP.
3. **Команды условного и безусловного переходов. Каким образом вычисляются адреса переходов?** Команды передачи управления используются для изменения последовательности выполнения команд при наличии программных ветвлений: команд условных и безусловного (JMP) переходов, обращении к подпрограммам (CALL) и выхода из них (RETURN). Команды условных переходов реализуют передачи управления в зависимости от значения флагов в регистре PSW. С их помощью процессор выполняет одну из возможных ветвей продолжения программы. Обычно в системе команд имеется несколько команд условных переходов.  
    Команда CMP устанавливает статусные флаги в зависимости от результата сравнения операндов, не меняя самих операндов.  
    Команды условного перехода позволяют выполнять только короткие переходы (внутрисегментные прямые переходы в диапазоне от -128 байт до +127 байт).  
    Команда безусловного перехода обеспечивает переход по заданному адресу без проверки каких-либо условий.  
    Формат команды: *JMP Модификатор адрес\_перехода*  
    Этой командой задаются внутрисегментные и межсегментные переходы.  
    Модификатор указывает вид перехода (внутрисегментный прямой, внутрисегментный косвенный, межсегментный прямой, межсегментный косвенный) и принцип изменения содержимого регистров CS и IP. Модификатор не всегда указывается в команде JMP.  
    При выполнении команд перехода в счётчик команд IP принудительно записывается новое значение - новый адрес в памяти, начиная с которого будут выполняться команды.
4. **Циклы в алгоритмах. Организация циклов на языке ассемблера. Особенности и ограничения цикла LOOP.** Организация циклов. Циклы организуются для многократного повторения одной или нескольких команд программы или процедуры. Цикл можно организовать, используя команды условного и безусловного переходов, рассмотренные выше, а можно с помощью специальных команд.  
    Команда LOOP и её расширения позволяет организовывать циклы, подобные циклам for в языках высокого уровня с автоматическим уменьшением счётчика цикла. Количество повторений содержится в регистре CX, который в командах управления циклами выполняет функции счётчика цикла.
5. **В каком регистре находится во время выполнения программы смещение кода? Каким образом вычисляется адрес команды?** Во время выполнения программы смещение кода находится в регистре IP. Чтобы вычислить адрес команды, нужно сложить адрес начала сегмента (CS) и смещение (IP).  
    Смещение следующей команды подлежащей выполнению содержится в регистре EIP (указатель команд).  
    Прямая адресация:  
    Это простейший вид адресации операнда в памяти, так как эффективный адрес содержится в самой команде и для его формирования не используется никаких дополнительных источников или регистров. Эффективный адрес берется непосредственно из поля смещения машинной команды, которое может иметь размер 8, 16, 32 бит. Это значение однозначно определяет байт, слово или двойное слово, расположенные в сегменте данных.  
    Косвенная базовая (регистровая) адресация:  
    При такой адресации эффективный адрес операнда может находиться в любом из регистров общего назначения, кроме sp/esp и bp/ebp (это специфические регистры для работы с сегментом стека). Синтаксически в команде этот режим адресации выражается заключением имени регистра в квадратные скобки [ ]. К примеру, команда mov ax,[ecx] помещает в регистр ax содержимое слова по адресу из сегмента данных со смещением, хранящимся в регистре ecx. Так как содержимое регистра легко изменить в ходе работы программы, данный способ адресации позволяет динамически назначить адрес операнда для некоторой машинной команды. Это свойство очень полезно, например, для организации циклических вычислений и для работы с различными структурами данных типа таблиц или массивов.  
    Косвенная базовая (регистровая) адресация со смещением:  
    Этот вид адресации очень похож на косвенную базовую адресацию со смещением. Здесь также для формирования эффективного адреса используется один из регистров общего назначения. Но индексная адресация обладает одной интересной особенностью, которая очень удобна для работы с массивами. Она связана с возможностью так называемого масштабирования содержимого индексного регистра. К примеру, в команде mov ax,mas[si\*2] значение эффективного адреса второго операнда вычисляется выражением mas+(si)\*2. В связи с тем, что в ассемблере нет средств для организации индексации массивов, то программисту своими силами приходится ее организовывать.  
    Наличие возможности масштабирования существенно помогает в решении этой проблемы, но при условии, что размер элементов массива составляет 1, 2, 4 или 8 байт.  
    Косвенная базовая индексная адресация:  
    При этом виде адресации эффективный адрес формируется как сумма содержимого двух регистров общего назначения: базового и индексного. В качестве этих регистров могут применяться любые регистры общего назначения, при этом часто используется масштабирование содержимого индексного регистра
6. **Какую принципиальную роль играет оператор безусловного перехода JMP при организации ветвлений?** Безусловный переход в программе на Ассемблере производится по команде JMP. Полный формат команды следующий:  
    JMP [модификатор] адрес\_перехода.  
    Адрес перехода может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода.  
    В системе команд микропроцессора существуют несколько кодов машинных команд безусловного перехода. Их различия определяются дальностью перехода и способом задания целевого адреса. Дальность перехода определяется местоположением операнда адрес\_перехода. Этот адрес может находиться в текущем сегменте кода или в некотором другом сегменте. В первом случае переход называется внутрисегментным или близким, а во втором случае – межсегментным или дальним.
7. **Что означает корректное завершение программы?**

Программа, запрашивающая сервис DOS, должна подготовить всю необходимую информацию в регистрах и управляющих блоках, указать в регистре AH номер желаемой функции DOS и затем вызвать прерывание INT 21H. Оно служит главным входом большинства функций DOS.

1. **Реальный и защищённый режимы работы процессора. Вычисление физических адресов ячеек памяти.**

Для совместимости с первыми компьютерами в последующих версиях микропроцессоров была реализована возможность использовать их в двух режимах: реальном (real mode) — так назвали режим работы первых 16-разрядных микропроцессоров — и защищенном (protected mode),означающем, что параллельные вычисления могут быть защищены аппаратно-программными механизмами.  
 Для определения физического адреса команды содержимое регистра сегмента кода (Code Segment, CS) умножается на 16 за счет добавления справа (к младшим битам) четырех нулей, после чего к полученному значению прибавляется содержимое регистра указателя ко- манд (Instruction Pointer, IP). Получается 20-разрядное значение, которое и позволяет указать любой байт из 2020.  
 При переключении микропроцессора в защищенный режим он начинает совершенно другим образом, чем в реальном режиме, вычислять физические адреса команд и операндов. Прежде всего, содержимое сегментных регистров начинает интерпретироваться иначе: считается, что там содержится не адрес начала, а номер соответствующего сегмента. Для того чтобы подчеркнуть этот факт, сегментные регистры CS, SS, DS, ES, FS, GS начинают даже называть иначе — селекторами сегментов. При этом каждый селекторный регистр разбивается на три поля.  
 Поле индекса (Index) — старшие 13 битов (3-15) определяет собственно номер сегмента (его индекс в соответствующей таблице дескрипторов).  
 Поле индикатора таблицы сегментов (Table Index, TI) — бит с номером 2 определяет часть виртуального адресного пространства (общее или принадлежащее только данной задаче). Если TI = 0, то поле индекса указывает на элемент в глобальной таблице дескрипторов (GDT), то есть идет обращение к общей памяти. Если TI = 1, то идет обращение к локальной области памяти текущей задачи; это пространство описывается локальной таблицей дескрипторов (LDT).  
 Поле уровня привилегий идентифицирует запрашиваемый уровень привилегий (Requested Privilege Level, RPL).