**Федеральное государственное образовательное учреждение   
высшего образования**

**«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИМЕНИ И.С. ТУРГЕНЕВА»**

**Институт приборостроения, автоматизации и информационных технологий**

**Кафедра информационных систем и цифровых технологий**

Отчет к лабораторной работе № 1

«Линейное исполнение программ.   
Арифметические и поразрядные логические   
операции над целыми двоичными числами»

Выполнила:

Студентка группы 21ПГ

Банных Мария Алексеевна

Приняла:

доцент кафедры   
информационных систем   
и цифровых технологий  
Конюхова Оксана Владимировна  
  
доцент кафедры   
информационных систем   
и цифровых технологий  
Амелина Ольга Викторовна

Орёл 2022 год

Ответы на контрольные вопросы

1. *Понятие сегмента, характеристики сегмента, организация сегмента.*

Сегмент – это прямоугольная область памяти, характеризующаяся начальным адресом и длиной.

Ячейка, с которой начинается сегмент, называется начальным адресом.

Длина сегмента – это количество входящих в него ячеек памяти. Все ячейки, расположенные внутри сегмента, перенумеровываются, начиная с нуля, нумерация ведётся относительно начала сегмента. Адрес ячейки называется смещением или EA.

1. *На какие сегменты разбита память компьютера? В какие регистры записываются начальные адреса сегментов?*

Программа, размещённая в памяти, может иметь следующие сегменты: сегмент данных для хранения операндов, сегмент кода хранения операторов программы и сегмент стека – дополнительную память для временного размещения информации.

Начальные адреса сегментов помещаются микропроцессором в соответствующие сегментные регистры.   
Регистры общего назначения: РОН  
Регистры универсальные: AX (AH, AL); BX (BH, BL); CX (CH, CL); DX (DH, DL)

Могут использоваться для временного хранения любых данных, при этом можно работать с каждым регистром целиком, как двухбайтовым, а можно отдельно, с каждой его однобайтовой половиной.

1. *Какие регистры процессора используются при выполнении арифметических операций?*

РОН

1. *На какие флаги воздействуют арифметические команды?*

CF – флаг переноса. Устанавливается в 1, если при выполнении арифметических и некоторых операций сдвига возникает «перенос» из старшего разряда.

PF – флаг чётности. Проверяет младшие 8 битов результатов над данными. Чѐтное число единиц приводит к установке этого флага в 1, нечётное – в 0.

AF – флаг логического переноса в двоично-десятичной арифметике. Устанавливается в 1, если арифметическая операция приводит к переносу или займу четвѐртого справа бита однобайтового операнда. Используется при арифметических операциях над двоично-десятичными кодами и кодами ASCII.

ZF – флаг нуля. Устанавливается в 1, если результат операции равен 0, в противном случае ZF обнуляется.

SF – флаг знака. Устанавливается в 1, если результат арифметической операции является отрицательным, в 0, если результат положительный.

OF – флаг переполнения. Устанавливается в единицу при арифметическом переполнении, когда результат выходит за пределы разрядной сетки.

1. *Особенности выполнения команд сложения и вычитания. Требования к операндам этих команд.*

Сложение:

add ax,5     ;AX = AX + 5

add dx,cx    ;DX = DX + CX

add dx,cl    ;Ошибка: разный размер операндов

Операнды должны иметь одинаковый размер. Результат на месте первого операнда.

Вычитание:

sub ax,13     ;AX = AX - 13

sub ax,bx    ;AX = AX + BX

sub bx,cl    ;Ошибка: разный размер операндов

Операнды должны иметь одинаковый размер. Результат на месте первого операнда.

1. *Особенности выполнения операции умножения. Особенности выполнения операции деления. Распределение регистров.*

Операция умножения:

**MUL** – команда умножения чисел без знака. У этой команды только один операнд — второй множитель, который должен находиться в регистре или в памяти. Местоположение первого множителя и результата задаётся неявно и зависит от размера операнда:

|  |  |  |
| --- | --- | --- |
| **Размер операнда** | **Множитель** | **Результат** |
| 1 байт | AL | AX |
| 2 байта | AX | DX:AX |
| 4 байта | EAX | EDX:EAX |

mul bl    ;AX = AL \* BL

mul ax    ;DX:AX = AX \* AX

**IMUL** – команда умножения чисел со знаком. Эта команда имеет три формы, различающиеся количеством операндов:

1. С одним операндом — форма, аналогичная команде MUL. В качестве операнда указывается множитель. Местоположение другого множителя и результата определяется по таблице.

2. С двумя операндами — указываются два множителя. Результат записывается на место первого множителя. Старшая часть результата в этом случае игнорируется. Кстати, эта форма команды не работает с операндами размером 1 байт.

3. С тремя операндами — указывается положение результата, первого и второго множителя. Второй множитель должен быть непосредственным значением. Результат имеет такой же размер, как первый множитель, старшая часть результата игнорируется. Это форма тоже не работает с однобайтными множителями.

imul cl            ;AX = AL \* CL

imul bx,ax         ;BX = BX \* AX

imul cx,-5         ;CX = CX \* (-5)

imul dx,bx,134h    ;DX = BX \* 134h

Деление:

**DIV** – команда деления чисел без знака. У этой команды один операнд — делитель, который должен находиться в регистре или в памяти. Местоположение делимого, частного и остатка задаётся неявно и зависит от размера операнда:

|  |  |  |  |
| --- | --- | --- | --- |
| **Размер операнда** | **Делимое** | **Частное** | **Остаток** |
| 1 байт | AX | AL | AH |
| 2 байта | DX:AX | AX | DX |
| 4 байта | EDX:EAX | EAX | EDX |

При выполнении команды DIV может возникнуть *прерывание*:

* если делитель равен нулю;
* если частное не помещается в отведённую под него разрядную сетку (например, если при делении слова на байт частное больше 255)

div cl   ;AL = AX / CL, остаток в AH

div di   ;AX = DX:AX / DI, остаток в DX

**IDIV** – команда деления чисел со знаком. Единственным операндом является делитель. Местоположение делимого и частного то же, как и для DIV. Эта команда тоже генерирует прерывание при делении на ноль или слишком большом частном.

1. *Основные логические операции и принципы их выполнения.*

AND. Команда AND выполняет операцию логического И между соответствующими парами битов операндов команды и помещает результат на место операнда получателя данных: AND получатель, источник.

OR. Команда OR выполняет операцию логического ИЛИ между соответствующими парами битов операндов команды и помещает результат на место операнда получателя данных: OR получатель, источник.

XOR. Команда XOR выполняет операцию ИСКЛЮЧАЮЩЕГО ИЛИ между соответствующими парами битов операндов команды и помещает результат на место операнда получателя данных: XOR получатель, источник.

NOT. Команда NOT позволяет выполнить инверсию всех битов операнда, в результате чего получается обратный код числа. В команде допускаются следующие типы операндов: NOT reg или NOT mem.

Команда логического сравнения TEST выполняет операцию логического умножения над операндами; при этом сами операнды не изменяются. Результат операции формируется во временной памяти: бит результата равен 1, если соответствующие биты операндов равны 1, в противном случае бит результата равен 0.

1. *Правила формирования масок для установки и сброса битов.*

Размерность операндов должна быть одинакова.

1. *Каким образом выполняются логические команды над словами?*

Таким же, как и с байтами, только второй операнд в логической операции тоже должен быть словом.

Практическая часть:

1. (a + b)^2 mod (b – x)^2

raz segment ;создаём сегмент с переменными

a db 2

b db 3

x db 1

raz ends

kod segment ;создаём сегмент с коммандами

assume DS: raz, CS: kod

begin:

mov ax, raz

mov ds, ax

mov ah, a ;добавляем в регистры переменные

mov al, b

mov bh, x

add ah, al ;складываем в регистр ah

mov al, ah ;записываем в регистр al значение ah

mul al ;умножаем регистр сам на себя

mov cx, ax ;записываем в регистр cx значение ax

mov ah, b ;записываем значение переменных

mov al, x

sub ah, al ;вычитаем из регистра ah al

mov al, ah ;записываем в регистр al значение ah

mul al ;умножаем регистр сам на себя

mov dx, ax ;записываем в регистр dx значение ax

mov ax, cx ; записываем в регистр ax значение cx

div dl ;делим регистр al на dl

mov ah, 4ch ;завершаем программу

int 21h

kod ends

end begin

1. Сбросить биты 5, 6, 7, установить биты 0 и 1, инвертировать

kod segment ;создаём сегмент кода

assume CS: kod

begin:

mov ah, 237 ;записываем в регистр ah число 237

and ah, 00011111b ;сбрасываем 5, 6, 7 биты

or ah, 00000011b ;добавляем 0 и 1 биты

not ah ;инвертируем число

mov ah, 4ch ;завершаем программу

int 21h

kod ends

end begin