«Архитектура информационных систем»

контрольная работа

Стрельников Сергей Александрович 123802

1. Ответы на контрольные вопросы.

1. Формат команды «сложить», ее операнды.

Команда **ADD** осуществляет сложение первого и второго операнда, при этом исходное значение первого операнда (**dst** - приёмника) теряется, замещаясь результатом сложения. Второй операнд (**src** –источник) не изменяется.

ADD dst, src; dst: = (dst) + (src).

В качестве первого операнда можно указывать регистр (кроме сегментного) или ячейку памяти, в качестве второго – регистр (кроме сегментного), ячейку памяти или непосредственное значение, однако не допускается определять оба операнда одновременно как ячейки памяти. Операнды могут быть байтами или словами и являться числами со знаком или без знака. Команда воздействует на флаги **OF**, **SF**, **ZF**, **AF**, **PF** и **CF**.

При сложении беззнаковых чисел, когда размерность результата операции выходит за разрядную сетку операндов, могут возникнуть ошибки с определением точного результата. Для этих целей в процессоре предназначен **флаг переноса CF**, который необходимо проконтролировать при выполнении операции сложения.

При операциях с числами со знаком нужно учитывать возможный перенос в старший значащий разряд, так как при этом может измениться знак результата. Для этих целей может помочь анализ **флага переполнения OF**, так как он устанавливается в 1, если происходит перенос в старший значащий разряд (в 7-ой или 15-ый) для положительных чисел или из старшего значащего разряда для отрицательных чисел.

2. Формат команды «вычесть», ее операнды.

Команда **SUB** вычитает второй операнд из первого и помещает результат на место первого операнда, т.е.

SUB dst, src; ; dst: = (dst) - (src).

Операнды аналогичны команде целочисленного сложения. Команда воздействует на флаги **OF**, **SF**, **ZF**, **AF**, **PF** и **CF**.

3. Формат команды «умножить», ее операнды.

Для умножения чисел без знака предназначена команда **MUL**, которая имеет следующий формат:

MUL src; AX:= AL*src - при умножении байтов,
(DX:AX):= AX*src – при умножении слов.

Как видно, второй операнд должен находиться или в регистре-аккумуляторе **AL** (в случае умножения на байт), или в регистре-аккумуляторе **AX** (в случае умножения на слово). После выполнения операции с однобайтовыми числами, 16-и битовый результат записывается в регистр-аккумулятор **AX**; для двухбайтовых чисел произведение длиной в 32 бита формируется в паре регистров **DX:AX** (в **DX** – старшая часть, в **AX** – младшая). Предыдущее содержимое регистра **DX** затирается.

Если содержимое регистра AH после однобайтового умножения или содержимое регистра DX после двухбайтового умножения не равны 0, флаги CF и OF устанавливаются в 1. В противном случае оба флага сбрасываются в 0.

В качестве операнда-сомножителя команды **MUL** можно указывать регистр (кроме сегментного) или ячейку памяти; не допускается умножение на непосредственное значение.

Для умножения чисел со знаком предназначена команда

IMUL src.

Эта команда выполняется так же, как и команда MUL. Отличительной особенностью команды IMUL является только формирование знака. Если результат мал и умещается в одном регистре (то есть если CF=OF=0), то содержимое другого регистра (старшей части)

является расширением знака – все его биты равны старшему биту (знаковому разряду) младшей части результата. В противном случае (если CF=OF=1) знаком результата является знаковый бит старшей части результата, а знаковый бит младшей части является значащим битом двоичного кода результата.

4. Формат команды «делить», ее операнды.

Команды деления для знаковых и беззнаковых операндов **DIV** и **IDIV** выполняют целочисленное деление, формируя целое частное и целый остаток. Формат команды:

DIV src; AL:=quot ((AX)/(src)); частное и остаток
AH:=rem ((AX)/(src)); при делении на байт.
AX:=quot ((DX:AX)/(src)); частное и остаток
DX:=rem ((DX:AX)/(src)); при делении на слово.

При этом делимое находится в регистрах **AX** (в случае деления на байт) или **DX:AX** (в случае деления на слово). Размер делимого в два раза меньше размеров делителя и остатка.

После выполнения операции с однобайтовыми числами, частное записывается в регистр **AL**, остаток – в регистр **AH**; для двухбайтовых чисел – частное в **AX**, остаток в **DX.**

Если делитель равен 0, или если частное не помещается в назначенный регистр, возбуждается прерывание с вектором 0.

Команды не воздействуют на флаги процессора.

5. Каков диапазон беззнаковых чисел допустим в программах 16-ти разразрядного микропроцессора?

Диапазон беззнаковых слов (16 битов) равен = 0…65535.

6. Каков диапазон чисел со знаком допустим в программах 16-ти разразрядного микропроцессора?

Диапазон слов со знаком (16 битов) равен = -32768…+32767

7. Какую информацию содержат арифметические флаги операций?

Арифметические флаги это:

CF= индицирует переполнение при арифметических операциях с беззнаковыми числами

SF = устанавливается, если левый бит результата = 1 (то есть если число отрицательное)

ZF= устанавливается когда результат операции = 0

OF = устанавливается когда происходит переполнение разрядной сетки процессора при операциях с числами со знаком.

8. Какие флаги устанавливаются при выполнении команд «сложить» и «вычесть».

Команды «сложить» и «вычесть» воздействуют на флаги **OF**, **SF**, **ZF**, **AF**, **PF** и **CF**. Флаги устанавливаются в соответствии с пунктами 1,2 (см. выше).

9. Какие флаги устанавливаются при выполнении команд «умножить» и «делить».

Команды умножения воздействуют на флаги CF,OF. Команды деления не изменяют содержимого регистра флагов. Установка флагов для MUL, IMUL описана в п.3 (см. выше).

10.Как выполнить сложение (вычитание) двух операндов, находящихся в памяти?

Непосредственно выполнить сложение (вычитание) двух операндов, находящихся в памяти, невозможно. Надо один из операндов предварительно записать в какой-либо регистр и произвести операцию с этим регистром.

11. Как выполнить умножение двух операндов, находящихся в памяти?

Непосредственно выполнить умножение двух операндов, находящихся в памяти, невозможно. Надо один из операндов предварительно записать в какой-либо регистр и произвести операцию с этим регистром.

12. Как выполнить деление двух операндов, находящихся в памяти?

Непосредственно выполнить деление двух операндов, находящихся в памяти, невозможно. Надо один из операндов предварительно записать в какой-либо регистр и

произвести операцию с этим регистром.

13. С числами какой системы счисления может работать Ассемблер?

Ассемблер работает с двоичными числами. Частным случаем двоичной системы является 16-ричная запись. С десятичными числами CPU не работает, но есть возможность коррекции результата при операциях с двоично-десятичными числами (в упакованном и неупакованном формате).

14. Найдите ошибки в нижеприведенных командах:

MOV AL, E4h; - не хватает нуля; должно быть: MOV AL, 0E4h

ADD 64, BL; - нельзя прибавлять к immediate значению (нарушение формата команды ADD)

MUL 3Fh; - - нельзя умножать на immediate значение (нарушение формата команды MUL)

MOV DS, 3F3Fh. – сегментные регистры нельзя инициировать непосредственно.

2. Выполнение работы

2.1 Вариант 4 Вычислить арифметическое выражение

№ варианта	Функция	Данные		
		A	B	C
4	$X = \frac{\left(A^2 + D\right)}{C - B}$	**15**	**150h**	**5**

Значение переменной D=4 (последняя цифра зачетной книжки)

Подставив значения переменных в выражение вычислим результат:
Частное = 0, остаток 229d = E5h

```
;prog_4.asm Арифметические операции
Data SEGMENT            ; Открыть сегмент данных
A DB 15                 ;Инициализировать
B DW 150h               ;переменные A, B, C, D, X
C DW 5
D DW 4
X DW ?
Data ENDS               ;Закрыть сегмент данных
Ourstack SEGMENT Stack ;Открыть сегмент стека
DB 100h DUP (?)         ;Отвести под стек 256 байт
Ourstack ENDS           ;Закрыть сегмент стека
ASSUME CS:Code, DS:Data, SS:Ourstack ;Назначить сегментные ;регистры
Code SEGMENT            ;Открыть сегмент кодов
Start: mov AX, Data     ;Инициализировать       (1)
mov DS, AX              ;сегментный регистр DS  (2)
xor AX, AX              ;очистить регистр AX    (3)
;=====команды вычисления арифметического выражения==========
mov AL,A                ;AL:= (A)               (4)
mul A                   ;AX:=A*A                (5)
add AX, D               ;AX:=(AX) + (D)         (6)
cwd                     ;преобразовать (AX) в двойное слово (7)
mov BX, C               ;BX:= (C)               (8)
sub BX,B                ;BX:=(BX)-(B)           (9)
idiv BX                 ;делить (DX:AX) на (BX) (10)
                        ;частное в AX, остаток в DX
mov AX, 4C00h           ;Завершить программу (11)
int 21h                 ;с помощью DOS       (12)
Code ENDS               ;Закрыть сегмент кодов
END Start               ;Конец исходного модуля.
```

Таблица 1 – Результаты пошагового выполнения программы prog_4.exe

Вариант 4						
№ строки	Команда Ассемблера	Машинный код	Длина машинного кода, байт	Логический адрес в памяти	Состояние регистров и флагов	Выводы
1	mov AL,A	A00000	3	48FE:7	AX=000F BX=0 , CX=0 , DX=0 , SP=0100 , BP=0 , SI=0 , DI=0 , DS=48ED SS=48EE , CS=48FE ES=48DD CF=0 , ZF=1 , SF=0 , OF=0 , PF=1 , AF=0 .	При выполнении команды mov флаги не изменяются
2	mul A	F6260000	4	48FE:A	AX=00E1 BX=0 , CX=0 , DX=0 , SP=0100 , BP=0 , SI=0 , DI=0 , DS=48ED SS=48EE , CS=48FE , ES=48DD CF=0 , ZF=0 , SF=0 , OF=0 , PF=1 , AF=0 .	
3	add al,D	02060400	4	48FE:E	AX=00E3 BX=0 , CX=0 , DX=0 , SP=0100 , BP=0 , SI=0 , DI=0 , DS=48ED	SF=1 так как операнд (AL) имеет 1 в старшем разряде

					SS=48EE CS=48FE ES=48DD CF=0 , ZF=0 , SF=1 , OF=0 , PF=0 , AF=0 .	
4	mov bl,C	8A1E0300	4	48FE:12	AX=00E3 BX=0005 CX=0 , DX=0 , SP=0100 , BP=0 , SI=0 , DI= , DS=48ED SS=48EE CS=48FE ES=48DD CF=0 , ZF=0 , SF=1 , OF=0 , PF=0 , AF=0 .	При выполнении команды mov флаги не изменяются
5	xor bh,bh	32FF	2	48FE:16	AX=00E3 BX=0005 CX=0 , DX=0 , SP=0100 , BP=0 , SI=0 , DI= , DS=48ED SS=48EE CS=48FE ES=48DD CF=0 , ZF=1 , SF=0 , OF=0 , PF=1 , AF=0 .	ZF=1 так как операнд стал =0, SF=0 так как левый бит операнда не установлен
6	sub bx,B	2B1E0100	4	48FE:18	AX=00E3 BX=FEB5 CX=0 , DX=0 , SP=0100 , BP=0 , SI=0 , DI=	CF=1 так как произошел перенос из младшего байта (BL)в старший (BH), SF=1 так как левый бит результата (BX) установлен

					, DS=48ED SS=48EE CS=48FE ES=48DD CF=1 , ZF=0 , SF=1 , OF=0 , PF=0 , AF=0 .	
7	xor dx,dx	33D2	2	48FE:1C	AX=00E3 BX=FEB5 CX=0 , DX=0 , SP=0100 , BP=0 , SI=0 , DI= , DS=48ED SS=48EE CS=48FE ES=48DD CF=0 , ZF=1 , SF=0 , OF=0 , PF=0 , AF=0 .	ZF=1, так как результат = 0
8	idiv bx	F7FB	2	48FE:1E	AX=0000 BX=FEB5 CX=0 , DX=00E3 SP=0100 , BP=0 , SI=0 , DI= , DS=48ED SS=48EE CS=48FE ES=48DD CF=0 , ZF=1 , SF=0 , OF=0 , PF=0 , AF=0 .	ZF=1, так как результат = 0 Остаток выводится как модуль.
Значение переменной X: частное = (AX) = 0 остаток = (DX) = E3 (модуль)						

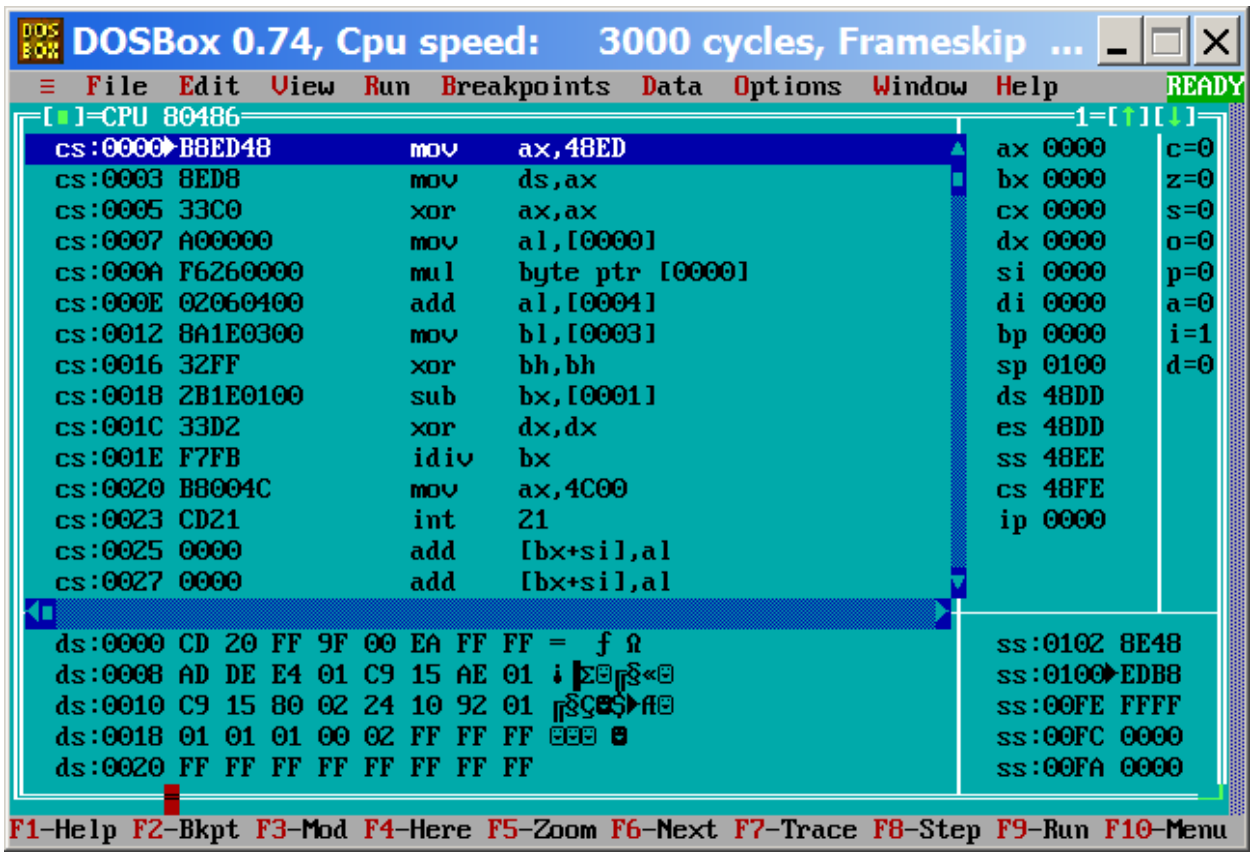

Рис. 1 Окно TD.exe после загрузки программы prog_4.exe

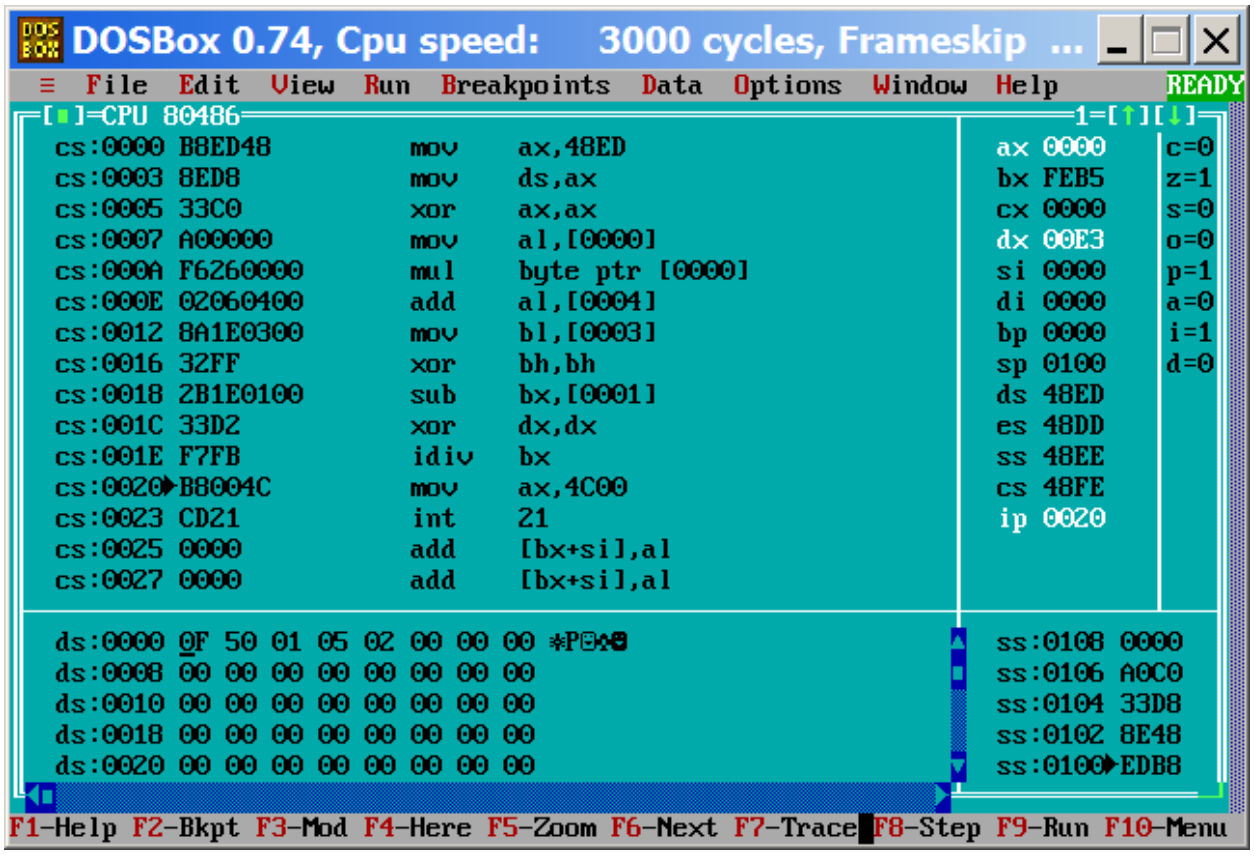

Рис. 2 Окно TD.exe после выполнения программы prog_4.exe

2.2 Определение начальных и конечных адресов сегментов

Определение начальных и конечных адресов сегментов кодов, данных, стека в программе prog_4.exe.

Значения CS,SS,DS после загрузки программы следующие:
CS=48FE, SS=48EE, DS=48ED (после инициализации)

Сегмент кодов.
Начальный адрес 48FE:0000
Конечный адрес 48FE:0024

Длина 25h

Сегмент данных
Начальный адрес 48ED:0000
Конечный адрес 48ED:0007

Длина 7

Сегмент стека
Начальный адрес 48EE:0010
Конечный адрес 48EE:010F
Длина 100h

Данные файла prog_4.map

Start Stop Length Name Class

00000H 00006H 00007H DATA
00010H 0010FH 00100H OURSTACK
00110H 00134H 00025H CODE

Address Publics by Name

Address Publics by Value

Program entry point at 0011:0000

Вывод: вычисленные длины сегментов соответствуют файлу prog_4.map

Рисунок (таблица) расположения программы prog_4.exe в памяти

Имя сегмента	Физич. адреса	Начало	Конец	Длина
Data	48ED0-48ED7	48ED:0000	48ED:0007	7
not used	48ED8-48EDF			9
Stack	48EE0-48FDF	48EE:0010	48EE:010F	100H
Code	48FE0-49024	48FE:0000	48FE:0024	25H

Примечание: область памяти с хранящимися данными соответствует сегменту Data