# Лабораторная работа №2. Арифметические команды

**Цель работы**

Изучение арифметических команд и получение навыка работы с ними.

**Постановка задачи**

Занести числа в память:

a= 5'678'901;

b= 6'789'012;

c= 7'890'123;

d= 8'901'234;

e= 9'012'345;

f=10'123'456.

Произвести операции над числами a,b,c,d,e,f по вариантам, заданным в таблице 3. Результат вывести в память и распечатать в *Memo*.

**Содержание отчета**

Отчет по лабораторной работе должен содержать номер, название, цель лабораторной работы, задание на лабораторную работу, текст программы на языке ассемблера, распечатки формы в активном состоянии с результатами выполнения лабораторной работы в виде дампа памяти и фрагмента программы, содержащей ассемблерную вставку, а также контрольный пример и выводы по работе.

**Варианты заданий**

Таблица 3.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Номер варианта** | **Выражение** | **Номер варианта** | **Выражение** | **Номер варианта** | **Выражение** |
| **1** |  | **17** |  | **33** |  |
| **2** |  | **18** |  | **34** |  |
| **3** |  | **19** |  | **35** |  |
| **4** |  | **20** |  | **36** |  |
| **5** |  | **21** |  | **37** |  |
| **6** |  | **22** |  | **38** |  |
| **7** |  | **23** |  | **39** |  |
| **8** |  | **24** |  | **40** |  |
| **9** |  | **25** |  | **41** |  |
| **10** |  | **26** |  | **42** |  |
| **11** |  | **27** |  | **43** |  |
| **12** |  | **28** |  | **44** |  |
| **13** |  | **29** |  | **45** |  |
| **14** |  | **30** |  | **46** |  |
| **15** |  | **31** |  | **47** |  |
| **16** |  | **32** |  | **48** |  |

**Краткие теоретические сведения**

**Регистр флагов**

При выполнении некоторых команд процессору необходима дополнительная информация, так же как и после выполнения некоторых команд необходимо сохранить дополнительную информацию о результатах выполнения команды. Чаще всего необходимость в дополнительной информации при выполнении команд возникает при выполнении арифметических команд. Поскольку процессор может оперировать данными определенных размеров (Byte, Word, LongWord), а при выполнении арифметических операций результат может не уместиться в используемый тип данных, то необходимо эту дополнительную информацию сохранять для дальнейшего учета. Местом хранения этой дополнительной информации является регистр флагов EFLAGS. Регистр флагов представляет собой 32-битный регистр специального назначения (РСН), где фиксируется информация о текущем состоянии процессора. Многие их его битов используются при выполнении команд и устанавливаются после выполнения команд. Расположение битов в регистре флагов представлено в таблице 4. Первая строка таблицы содержит номера битов регистра флагов, вторая – их аббревиатуры.

Таблица 4.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ID | VIP | VIF | AC | VM | RF | 0 | NT | IOPL | | OF | DF | IF | TF | SF | ZF | 0 | AF | 0 | PF | 1 | CF |

Флаг **CF** называется флагом переноса (от английского *Carry Flag*). Флаг переноса устанавливается (становиться равным 1) при переносе (заёме), возникающем при сложении (вычитании). Иначе сбрасывается (становиться равным 0). Может быть установлен и в других случаях в зависимости от выполняемой команды.

Флаг **ZF** – флаг нулевого результата (*Zero Flag*). Если результат последней операции равен нулю, то флаг устанавливается, иначе сбрасывается.

Флаг **AF** – флаг вспомогательного переноса (*Auxiliary Carry Flag*). Устанавливается при переносе (заёме) из бита 3 (в бит 3) при сложении (вычитании). Используется только для двоично-десятичной арифметики, которая оперирует исключительно младшими байтами.

Флаг **SF** – флаг знака (*Sign Flag*). Дублирует старший бит результата, который при использовании дополнительного кода соответствует знаку числа.

Флаг **OF** – флаг переполнения (*Overflow Flag*). Сигнализирует о потере старшего бита результата в связи с переполнением разрядной сетки при работе со знаковыми числами. Равен 1, когда:

- при сложении был перенос из старшего бита, но не было переноса в старший бит;

- при сложении был перенос в старший бит, но не было переноса из старшего бита;

- при вычитании был заем в старший бит, но не было заёма из старшего бита;

- при вычитании был заем из старшего бита, но не было заёма в старший бит.

Флаг **PF** – флаг четности (*Parity Flag*). Устанавливается в 1 при наличие четного числа единиц в двоичном представлении младшего байта результата (**ЧЕТНОСТЬ НЕ ПО МАТЕМАТИКЕ**!)

Флаг **DF** – флаг направления (*Direction Flag*). Используется в командах работы со строками (цепочками). При DF=1 регистр(ы) индекса, используемый(ые) в командах работы со строками, увеличивается(ются) на 1 при каждом следующем выполнении команды, при DF=0 - регистр(ы) индекса уменьшается(ются) на 1.

Остальные флаги являются достаточно специфичными и выходят за рамки данного пособия. Их описание можно найти, например, в [2-3].

К арифметическим командам относятся:

ADD (Add) – Сложить;

ADC (Add with Carry) – Сложить с учетом переноса;

XADD (eXchange and Add) – Обменять и сложить;

SUB (Subtract) – Вычесть;

SBB (Subtract with borrow) – Вычесть с учетом заёма;

INC (Increment) – Приращение;

DEC (Decrement) – Декремент (отрицательное приращение), уменьшение;

NEG (Negative) – Выдать число с противоположным (негативным) знаком;

MUL (Multiplication) – Умножение;

IMUL (Integer Multiplication) – Умножение с учета знака;

DIV (Division) – Деление;

IDIV (Integer Division) – Деление с учетом знака;

CWD (Convert Word to Double Word) – Преобразовать слово в двойное слово;

CDQ (Convert Double Word to Quad Word) – Преобразовать двойное слово в четверное слово;

CBW (Convert Byte to Word) – Преобразовать байт в слово;

CWDE (Convert Word to Double Word Extended) – Преобразовать слово в двойное слово, дополнительная функция.

**Команды ADD, ADC, XADD**

Формат команды: ADDП,И

ADCП,И

XADDП,И

где П – приемник информации. В качестве приемника может использоваться РОН или ячейка памяти;

И – источник информации. В качестве источника информации могут использоваться РОН, ячейка памяти или константа (непосредственные данные, указанные в самой команде ассемблера).

Приемник и источник должны быть одинаковой размерности! Нельзя в одной команде в качестве и приемника и источника использовать ячейки памяти!

Команда **ADD** выполняет следующую операцию: П=П+И;

Команда **ADC** выполняет следующую операцию:

П=П+И+значение флага CF;

Команда **XADD** выполняет следующие операции: 1) П↔И, 2) П=П+И;

Все три команды влияют на флаги OF,SF,ZF,AF,PF,CF, т.е. эти флаги принимают значения по результатам выполнения команды.

Связка команд ADD и ADC, как правило, используются для сложения чисел повышенной точности. Для примера, сложим два 64-битных числа. Поскольку в архитектуре IA32, к которой относится большинство микропроцессоров ПК, нет 64-х разрядных РОН, то хранение и сложение чисел большей разрядности, чем 32, приходиться выполнять по частям, задействовав по два 32-х разрядных РОН для каждого числа. Пусть одно число находиться в паре регистров EDX:EAX (EAX – младшее двойное слово, EDX – старшее двойное слово), а другое - в паре регистров EBX:ECX. При разбиении числа на части необходимо максимально использовать разряды РОН для младшей части. Рассмотрим, для примера, число 10'000'000'00010. В шестнадцатеричной системе счисления это число равно 2'540B'E40016. Как видно, это число занимает больше 32-х разрядов. При разбиении на части, младшая часть будет равна 540B'E40016, а старшая 0000'000216. Программа сложения 64-битных чисел будет выглядеть так:

ADD EAX,ECX

ADC EDX,EBX

Если при сложении младших двойных слов произошел перенос из старшего (15-ого) разряда, т.е. флаг CF стал равным 1, то он будет учтен в команде ADC при сложении старших частей. Стоит отметить, что сложение обязательно начинать с младших частей.

В большинстве встраиваемых систем используются 8-ми и 16-ти разрядные микропроцессоры и микроконтроллеры с соответствующей им разрядностью РОН. Необходимость сложения чисел по частям для таких систем возникает уже для чисел больше 25510=FF16, т.е. для чисел, занимающих более 8-ми двоичных разрядов. Часто для 8-ми разрядных систем возникает ситуация, что результирующие числа могут не умещаться и в два байта. При этом сложение чисел, занимающих несколько байтов в 8-ми разрядной системе, ведется последовательно для всех частей, продвигаясь от младшей части к старшей, причем для младших частей применяется команда ADD, а для всех остальных ADC. Между командами ADC нельзя применять команды, изменяющие значение флага CF.

**Команды SUB и SBB**

Формат команды: SUBП,И

SBBП,И

где П – приемник информации. В качестве приемника может использоваться РОН или ячейка памяти;

И – источник информации. В качестве источника информации могут использоваться РОН, ячейка памяти или константа (непосредственные данные, указанные в самой команде ассемблера).

Приемник и источник должны быть одинаковой размерности! Нельзя в одной команде в качестве и приемника и источника использовать ячейки памяти!

Команда **SUB** выполняет следующую операцию: П=П-И;

Команда **SBB** выполняет следующую операцию:

П=П-И-значение флага CF;

Команды SUB и SBB влияют на флаги OF,SF,ZF,AF,PF,CF, т.е. эти флаги принимают значения по результатам выполнения команды.

Связка команд SUB, SBB, как и ADD, ADC, как правило, используются для вычитания чисел повышенной точности, когда младшие части чисел вычитаются командой SUB, а остальные командой SBB.

**Команда INC, DEC**

Формат команды: INCП

DECП

где П – приемник информации. В качестве приемника может использоваться РОН или ячейка памяти.

Команда **INC** выполняет следующую операцию: П=П+1;

Команда **DEC** выполняет следующую операцию: П=П-1;

Команда INC увеличивает содержимое приемника на 1, а команда DEC – уменьшает на 1. Отличие от команд сложения и вычитания в том, что не затрагивается флаг CF (флаг переноса). Остальные влаги OF,SF,ZF,AF,PF устанавливаются в соответствии с результатом выполнения команды.

**Команда NEG**

Формат команды: NEGП

где П – приемник информации. В качестве приемника информации могут использоваться РОН или ячейка памяти.

Команда NEG выполняет следующую операцию: П=0-П;

Команда NEG изменяет значение знака содержимого приемника, что вытекает из математического смысла вычитания из нуля. Команда NEG влияет на флаги OF,SF,ZF,AF,PF,CF, т.е. эти флаги принимают значения по результатам выполнения команды вычитания из нуля приемника.

**Команда MUL**

Формат команды: MULИ

где И – источник информации. В качестве источника информации могут использоваться РОН или ячейка памяти.

Команда MUL выполняет беззнаковое умножение. При выполнении команды используются неявные операнды:

- если И 8-ми разрядный, то AX=AL\*И. Если в результате AH=0, то CF=OF=0, иначе CF=OF=1;

- если И 16-ти разрядный, то DX:AX=AX\*И. Если в результате DX=0, то CF=OF=0, иначе CF=OF=1;

- если И 32-ух разрядный, то EDX:EAX=EAX\*И. Если в результате EDX=0, то CF=OF=0, иначе CF=OF=1.

Значение флагов SF,ZF,AF,PF неопределенно, т.е. стоит считать их значения после выполнение команды случайными.

**Команда IMUL**

Команда IMUL выполняет операцию умножения с учетом знака.

1-ый формат команды (однооперандный): IMULИ

где И – источник информации. В качестве источника информации могут использоваться РОН или ячейка памяти.

В данном формате команды используются неявные операнды:

- если И 8-ми разрядный, то AX=AL\*И. Если в результате AH=0 или AH=FF16, то CF=OF=0, иначе CF=OF=1;

- если И 16-ти разрядный, то DX:AX=AX\*И. Если в результате DX=0 или DX=FFFF16, то CF=OF=0, иначе CF=OF=1;

- если И 32-ух разрядный, то EDX:EAX=EAX\*И. Если в результате EDX=0 или EDX=FFFF'FFFF16, то CF=OF=0, иначе CF=OF=1.

Значение флагов SF,ZF,AF,PF неопределенно, т.е. стоит считать их значения после выполнение команды случайными.

2-ой формат команды (двухоперандный): IMULП,И

где И – источник информации. В качестве источника информации могут использоваться РОН, константа или значение из памяти;

П – приемник информации. В качества приемника может выступать только РОН.

Возможные варианты команды IMUL в двухоперандном виде:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| П |  | П |  | И |
| 16-ти разрядный РОН | = | 16-ти разрядный РОН | \* | 16-ти разрядный РОН или ЯП |
| 32-ух разрядный РОН | = | 32-ух разрядный РОН | \* | 32-ух разрядный РОН или ЯП |
| 16-ти разрядный РОН | = | 16-ти разрядный РОН | \* | 8-ми разрядная константа1 |
| 32-ух разрядный РОН | = | 32-ух разрядный РОН | \* | 8-ми разрядная константа2 |
| 16-ти разрядный РОН | = | 16-ти разрядный РОН | \* | 16-ми разрядная константа |
| 32-ух разрядный РОН | = | 32-ух разрядный РОН | \* | 32-ми разрядная константа |

1 - 8-ми разрядная константа, перед умножение знакорасширяется до 16-ти разрядов.

2 - 8-ми разрядная константа, перед умножение знакорасширяется до 32-ух разрядов.

3-ий формат команды (трехоперандный): IMULП,И1,И2

где И1 – первый источник информации. В качестве И1 могут использоваться РОН или значение из памяти;

И2 – второй источник информации. В качестве И2 может использоваться только константа;

П – приемник информации. В качества приемника может выступать только РОН.

Возможные варианты команды IMUL в трехоперандном виде:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| П |  | И1 |  | И2 |
| 16-ти разрядный РОН | = | 16-ти разрядный РОН или ЯП | \* | 8-ми разрядная константа1 |
| 32-ух разрядный РОН | = | 32-ух разрядный РОН или ЯП | \* | 8-ми разрядная константа2 |
| 16-ти разрядный РОН | = | 16-ти разрядный РОН или ЯП | \* | 16-ми разрядная константа |
| 32-ух разрядный РОН | = | 32-ух разрядный РОН или ЯП | \* | 16-ми разрядная константа3 |
| 32-ух разрядный РОН | = | 32-ух разрядный РОН или ЯП | \* | 32-ми разрядная константа |

1 - 8-ми разрядная константа, перед умножение знакорасширяется до 16-ти разрядов.

2 - 8-ми разрядная константа, перед умножение знакорасширяется до 32-ух разрядов.

3 - 16-ти разрядная константа, перед умножение знакорасширяется до 32-ух разрядов.

Когда используется однооперандная форма команды IMUL, то результат умножения доступен, даже если установлен флаг переполнения, так как РОН результата в два раза больше по размеру, чем множимое и множитель. Этого достаточно, чтобы сохранить любой возможный результат.

Для форм с двумя и тремя операндами возможна потеря старших значащих битов результата. Если это происходит, то флаги CF и OF устанавливаются.

**Команда DIV, IDIV**

Формат команды: DIVИ

IDIVИ

где И – источник информации. В качестве источника информации могут использоваться РОН или ячейки памяти.

Команда DIV выполняет операцию целочисленного беззнакового деления. Команда IDIV выполняет операцию целочисленного деления с учетом знака. При выполнении и команды DIV, и команды IDIV используются неявные операнды:

- если И 8-ми разрядный, то AX делится на И. Целочисленное частное сохраняется в AL, целочисленный остаток в AH;

- если И 16-ти разрядный, то DX:AX делится на И. Целочисленное частное сохраняется в AX, целочисленный остаток в DX;

- если И 32-ух разрядный, EDX:EAX делится на И. Целочисленное частное сохраняется в EAX, целочисленный остаток в EDX.

Значение флагов OF,SF,ZF,AF,PF,CF неопределенно, т.е. стоит считать их значения после выполнение команды случайным.

Если результирующее частное слишком велико, чтобы поместиться по назначению, или если делитель равен нулю, то генерируется исключительная ситуация, которая отслеживается на уровне операционной системы, однако при этом программа может продолжить свое выполнение.

Следующие четыре команды очень похожи на команду MOVSX. Однако, команда MOVSX является универсальной и может использовать любые РОН и ячейки памяти в качестве своих операндов. Рассматриваемые же здесь команды не имеют входных операндов, операнды используются неявно. Это дает возможность экономить память программ и, более того, эти команда используют те же самые РОН, что и команды умножения и деления, а, значит, использование их в связке повышает эффективность программного кода.

**Команда CBW**

Формат команды: CBW

Команда CBW знаково расширяет байт, находящийся в регистре AL, до слова и сохраняет результат в регистре AX.

**Команда CWD**

Формат команды: CWD

Команда CWD знаково расширяет слово, находящиеся в регистре AX, до двойного слова и сохраняет результат в паре регистров DX:AX.

**Команда CDQ**

Формат команды: CDQ

Команда CDQ знаково расширяет двойное слово, находящееся в EAX, в учетверенное слово, и результат помещает в пару регистров EDX:EAX.

**Команда CWDE**

Формат команды: CWDE

Команда CWDE знаково расширяет слово, находящееся в регистре AX, в двойное слово, и результат помещает в регистр EAX.

**Примерные контрольные вопросы и задачи**

1. Чему будут равны регистры CH, CL, CX, BL, BH после выполнения следующего фрагмента программы:

mov cl,0

mov bl,0

mov bl,$32

mov cl,24

add bl,cl

1. Чему будут равны регистры AH, AL, DX, AL после выполнения следующего фрагмента программы:

mov eax,0

mov edx,0

mov ax,$32

mov dx,2

sub ax,dx

1. Чему будут равны регистры EAX, ECX после выполнения следующего фрагмента программы:

mov eax,0

mov edx,0

mov ax,$3d

mov cl,2

mul cl

1. Чему будут равны регистры EAX, ECX, DX, AX после выполнения следующего фрагмента программы:

mov eax,0

mov edx,0

mov ax,200

mov cl,2

div cl

1. Чему будут равны регистры EAX, ECX, DX, AX после выполнения следующего фрагмента программы:

mov eax,0

mov edx,0

mov ax,$200

mov cx,$2

div cx

1. Чему будет равен регистры EAX после выполнения следующего фрагмента программы:

mov ax,243

cwdE