Лабораторная работа №3

Арифметические команды центрального процессора

Цель работы: изучение арифметических команд центрального процессора для работы с целыми числами.

**Теоретические сведения**

Арифметико-логическое устройство центрального процессора содержит следующие основные команды для работы с целочисленной арифметикой:

ADD <операнд\_1>, <операнд\_2>. Команда складывает два числа, помещённых в регистры или память. Результат записывается на место первого операнда: **операнд\_1 = операнд\_1 + операнд\_2**.

ADC <операнд\_1>, <операнд\_2>. Сложение чисел с учётом знака переноса. Команда складывает значения двух операндов со значением флага переноса CF.

XADD <операнд\_1>, <операнд\_2>. Команда сначала обменивает содержимое двух операндов, а затем складывает их и пересылает результат на место первого операнда.

INC <операнд>. Команда выполняет инкремент содержимого регистра или ячейки памяти.

SUB <операнд\_1>, <операнд\_2>. Команда выполняет вычитание значения второго операнда из первого и записывает результат на место первого операнда.

SBB <операнд\_1>, <операнд\_2>. Команда вычитает из значения первого операнда значение второго операнда и флага CF: **операнд\_1 = операнд\_1 – (операнд\_2 + CF)**.

DEC <операнд>. Выполняет уменьшение на единицу значения регистра или ячейки памяти.

MUL <операнд>. Команда выполняет умножение содержимого регистра AL/AX/EAX на значение операнда **без учёта знака**. Местоположение результата зависит размерности операнда и представлено в следующей таблице:

|  |  |  |  |
| --- | --- | --- | --- |
| Размер операндов | Первый множитель | Результат | Пример |
| байт | AL | AX | MUL BL |
| слово | AX | DX:AX | MUL CX |
| двойное слово | EAX | EDX:EAX | MUL ESI |

IMUL. Команда выполняет умножение чисел **с учётом знака** и имеет три формы:

1. IMUL <операнд\_множитель>. Механизм работы данной команды похож на работу команды MUL с тем отличием, что произведение выполняется с учётом знака. Местоположение множителей и результата смотреть в вышеприведённой таблице.
2. IMUL <операнд\_1>, <операнд\_2>. Команда выполняет умножение значения первого операнда на значение второго. Результат записывается на место первого:

операнд\_1 = операнд\_1 \* операнд\_2.

1. IMUL <операнд\_1>, <операнд\_2>, <операнд\_3>. Команда выполняет умножение второго и третьего операндов: **операнд\_1 = операнд\_2 \* операнд\_3**. Результат записывает на место первого операнда. Третий операнд может иметь только непосредственную адресацию.

DIV <операнд>. Команда выполняет деление без учёта знаков. Делимое задаётся неявно. Местоположение делимого и результат зависит от размерности операнда-делителя и определяется следующей таблицей:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер операнда | Делимое | Частное | Остаток | Максимальное частное |
| байт | AX | AL | AH | 255 |
| слово | DX:AX | AX | DX | 65535 |
| двойное слово | EDX:EAX | EAX | EDX | 232-1 |

IDIV <операнд>. Команда выполняет деление с учётом знаков. Делимое задаётся неявно. Механизм работы данной команды похож на механизм работы команды DIV.

Команды изменения размерности и знака числа

Следующая группа команд расширяет число в два раза, сохраняя при этом его знак. Знак сохраняется за счёт копирования старшего (знакового) бита числа в старшую половину результирующего числа.

CBW. Команда расширяет байт до размерности слова, копируя старший бит регистра AL во все биты регистра AH.

CWD. Команда расширяет слово до размерности двойного слова, копируя старший бит регистра AX во все биты регистра DX.

CWDE. Команда расширяет слово до размерности двойного слова, копируя старший бит регистра AX во все биты старшей половины регистра EAX.

CDQ. Команда расширяет двойное слово до размерности учетверённого слова, копируя старший бит регистра EAX во все биты регистра EDX.

NEG <операнд>. Команда изменяет знак числа.

**Задания для выполнения к работе**

1. Написать программу для вычисления значения арифметического выражения согласно варианту задания. Все переменные, используемые в программе, требуется использовать как знаковые и расширять до размерности двойного слова. Результат должен быть записан в регистр EAX. Если результат содержит остаток от деления, оставить его в регистре EDX. Подобрать набор тестовых данных (не менее 3).
2. Написать программу для сложения или вычитания целых беззнаковых чисел большой размерности (размерность и операция зависят от варианта задания). Младшие байты при этом хранить по младшему адресу. Подобрать наборы тестовых данных (не менее 3). Для выполнения этого задания изучить теоретический материал главы «Вычитание и сложение операндов большой размерности», начиная со страницы 176 учебника Юрова «Assembler».

Пример выполнения первого задания:

|  |  |  |  |
| --- | --- | --- | --- |
| # |  | *g* – word  *r* – byte  *h* – word | сложение  7 байт |

Переменные *g*, *r*, *h* разместим в сегменте данных. Программа, вычисляющая значение этого выражения, имеет вид:

.386

.model flat, stdcall

option casemap: none

include d:\masm32\include\kernel32.inc

includelib d:\masm32\lib\kernel32.lib

.data

g dw -102

r db -1

h dw 15

.code

start:

MOV AX, g ; AX = g

CWDE ; Расширение в регистре AX слова до двойного в EAX

IMUL EAX ; EAX = EAX \* EAX = g2

MOV EBX, EAX ; EBX = EAX

MOV AL, r ; AL = r

CBW ; Расширение до слова

CWDE ; Расширение до двойного слова

MOV ECX, 5 ; ECX = 5

CDQ ; Расширение перед делением до двойного слова, т.к.

; следующая команда оперирует содержимым EDX.

IDIV ECX ; EAX = EDX:EAX / 5 = r / 5

ADD EAX, EBX ; EAX = EAX + EBX = r / 5 + g2

SUB EAX, 9\*9\*9\*9 ; EAX = EAX - 94 = r / 5 + g2 - 94

MOV EBX, EAX ; Копирование в EBX

MOV AX, h ; AX = h

CWDE ; Расширение h до двойного слова

XCHG EAX, EBX ; Обмен местами содержимого EAX и EBX

CDQ ; Расширение до двойного слова

IDIV EBX ; EAX = EDX:EAX / EBX = (r / 5 + g2 - 94) / h

push 0

call ExitProcess ; Выход из программы

end start

Тестовые данные

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *g* | *r* | *h* | Частное (EAX) | Остаток (EDX) |
| -1000 | 50 | 100 | 000026CEh=  9934 | 00000031h=  49 |
| 10 | -100 | 4 | FFFFF9AСh в дополни-тельном коде  соответствует  -1620 | FFFFFFFFh в дополни-тельном коде  соответствует  -1 |
| 300 | 60 | -1000 | 00000053h=  83 | 000001C3h=  451 |

*При выполнении второго задания числа требуется хранить в виде последовательности байт следующим образом:*

.386

.model flat, stdcall

option casemap: none

include d:\masm32\include\kernel32.inc

includelib d:\masm32\lib\kernel32.lib

.data

a db 2Ah, 03h, 12h, 0DE, 43h, 0E2h, 34h ; 7 байт

b db 15h, 0DDh, 34h, 4Bh, 57h, 7Fh, 0CDh ; 7 байт

r db 8 dup(?) ; Для результата резервируется на один байт больше

.code

start:

…

push 0

call ExitProcess ; Выход из программы

end start

В некоторых случаях целесообразно складывать сразу по 2 или 4 байта для уменьшения количества операций.

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Выражение | Размер входных параметров | Операция  Размерность  (2-е задание) |
| 1 |  | *a* – dword  *b* – byte  *d* – word | сложение  17 байт |
| 2 |  | *a* – word  *b* – word  *d* – byte | вычитание  14 байт |
| 3 |  | x – word  y – word  z – word | сложение  30 байт |
| 4 |  | *a* – byte  *b* – dword  *e* – byte | вычитание  17 байт |
| 5 |  | *t* – dword  *r* – word  *s* – byte | сложение  16 байт |
| 6 |  | *k* – word  *l* – dword  *m* – byte | вычитание  15 байт |
| 7 |  | *i*, *j*, *k* – word | сложение  21 байт |
| 8 |  | *x*, *y*, *z* – word | вычитание  15 байт |
| 9 |  | *t* – массив из 3 чисел типа byte  *x*, *a* – word | сложение  17 байт |
| 10 |  | *k* – dword  *m* – byte | вычитание  22 байта |
| 11 |  | *r* – dword  *s* – byte | сложение  20 байт |
| 12 |  | *i*, *j*, *k* – word | вычитание  19 байт |
| 13 |  | *a*, *b*, *d* – byte  *x* – word | сложение  14 байт |
| 14 |  | *a*, *b*, *d* – byte | вычитание  18 байт |
| 15 |  | *x*, *y*, *z* – byte | сложение  27 байт |
| 16 |  | *x*, *y*, *z* – byte | вычитание  25 байт |
| 17 |  | *p* – dword  *q* – byte  *s* – word | сложение  26 байт |
| 18 |  | *n* – word  *m* – byte  *r* – word | вычитание  25 байт |
| 19 |  | *v* – word  *t* – dword  *g* – word | сложение  25 байт |
| 20 |  | *f* – word  *g* – dword  *h* – byte | вычитание  16 байт |
| 21 |  | *i*, *j*, *k* – byte | Сложение  14 байт |
| 22 |  | *x*, *y*, *z* – word | вычитание  15 байт |
| 23 |  | *m* – byte  *l* – word  *n* – byte | сложение  19 байт |
| 24 |  | *a* – dword  *b* – word  *c* – word  *d* – byte | вычитание  26 байт |
| 25 |  | *c*, *e*, *k* – byte | сложение  28 байт |