Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебная дисциплина «Встраиваемые микропроцессорные системы»

Инструкция

по выполнению лабораторной работы
«Программирование на языке Ассемблер. Изучение команд арифметических операций»

Минск

2017

Лабораторная работа № 6

Тема работы: «Программирование на языке Ассемблер. Изучение команд арифметических операций»

1. **Цель работы**:

Формирование практических навыков по использованию команд арифметических операций.

2. Задание

Изучить команды арифметических операций. Написать на языке ассемблера и выполнить программу для вычисления алгебраического выражения.

3. Оснащение работы

Техническое задание, ПК, эмулятор DOSBox.

4. Основные теоретические сведения

В язык Ассемблера входят пять групп арифметических команд:

- команды преобразования типов,
- команды двоичной арифметики,
- команды десятичной арифметики,
- вспомогательные команды,
- прочие команды с арифметическим принципом действия.

Микропроцессор может работать с целыми числами со знаком и с целыми беззнаковыми числами. Целые беззнаковые числа не имеют знакового разряда, поэтому все его двоичные биты отводятся под мантиссу числа. В числах со знаком под знак отводится самый старший бит двоичного числа: 0 – положительное число; 1 – отрицательное. Поэтому диапазон значений двоичного числа зависит от его размера и трактовки старшего бита числа. Необходимо помнить, что числа со знаком представляются в дополнительном коде.

Сложение и вычитание. Команды ADD и SUB выполняют сложение и вычитание байтов или слов, содержащих двоичные данные. Вычитание выполняется по методу сложения с отрицательным числом, представленным в дополнительном коде. Во всем, кроме первого шага, операции сложения и вычитания идентичны. На рисунке 6.1 представлены примеры команд ADD и SUB, обрабатывающие байты или слова. В процедуре B10ADD используется

команда ADD для сложения байтов, а в процедуре C10SUB команда SUB вычитает слова. Примеры (рисунок 6.1) показывают все пять возможных ситуаций:

- сложение/вычитание регистр-регистр;
- сложение/вычитание память-регистр;
- сложение/вычитание регистр-память;
- сложение/вычитание регистр-непосредственное значение;
- сложение/вычитание память-непосредственное значение.

```
TITLE   EXADD   (COM) Сложение и вычитание
CODESG  SEGMENT PARA 'Code'
        ASSUME  CS:CODESG,DS:CODESG,SS:CODESG
        ORG     100H
BEGIN:  JMP     SHORT MAIN
; -----------------------------------------
BYTEA   DB      64H             ;Элементы данных
BYTEB   DB      40H
BYTEC   DB      16H
WORDA   DW      4000H
WORDB   DW      2000H
WORDC   DW      1000H
; -----------------------------------------
MAIN    PROC    NEAR            ;Основная процедура:
        CALL    B10ADD          ;Вызвать сложение ADD
        CALL    C10SUB          ;Вызвать вычитание SUB
        RET
MAIN    ENDP
;               Пример сложения байт:
;               --------------------
B10ADD  PROC
        MOV     AL,BYTEA
        MOV     BL,BYTEB
        ADD     AL,BL           ;Регистр и
регистр         ADD     AL,BYTEC          ;Память
и регистр
        ADD     BYTEA,BL        ;Регистр и память
        ADD     BL,10H          ;Непосредственное и регистр
        ADD     BYTEA,25H       ;Непосредственное и память
        RET
B10ADD  ENDP
;               Пример вычитания слов:
;               ---------------------
C10SUB  PROC
        MOV     AX,WORDA
        MOV     BX,WORDB
        SUB     AX,BX           ;Регистр из регистра
        SUB     AX,WORDC        ;Память из регистра
        SUB     WORDA,BX        ;Регистр из памяти
        SUB     BX,1000H        ;Непосредственное из регистра
        SUB     WORDA,256H      ;Непосредственное из памяти
        RET
C10SUB  ENDP
```

```
CODESG  ENDS
        END     BEGIN
```

Рисунок 6.1 - Примеры команд ADD и SUB

Поскольку прямой операции память-память не существует, эта операция выполняется через регистр. В следующем примере к содержимому слова WORDB прибавляется содержимое слова WORDA, описанных как DW:

```
MOV  AX,WORDA
ADD  AX,WORDB
MOV  WORDB,AX
```

Умножение и деление.

Операция умножения для беззнаковых данных выполняется командой MUL, для знаковых – IMUL (Integer MULtiplication – умножение целых чисел). Ответственность за выбор подходящей команды умножения лежит на самом программисте.

Умножение байта на байт: Множимое находится в регистре AL, множитель – в байте памяти или однобайтовом регистре. После умножения произведение находится в регистре AX. Операция игнорирует и стирает любые данные, которые находились в регистре AH.

Умножение слова на слово: Множимое находится в регистре AX, множитель – в слове памяти или регистре. После умножения произведение находится в двойном слове, для которого требуется два регистра: старшая (левая) часть произведения находится в регистре DX, а младшая (правая) часть – в регистре AX. Операция игнорирует и стирает любые данные, которые находились в регистре DX.

В единственном операнде команд MUL и IMUL указывается множитель. Рассмотрим следующую команду:

```
MUL  MULTR
```

Если поле MULTR определено как байт (DB), то операция предполагает умножение содержимого AL на значение байта из поля MULTR. Если поле MULTR определено как слово (DW), то операция предполагает умножение содержимого AX на значение слова из поля MULTR. Если множитель находится в регистре, то длина регистра определяет тип операции, как показано ниже:

```
MUL  CL  ;Байт-множитель: множимое в AL, произведение в AX
MUL  BX  ;Слово-множитель: множимое в AX, произведение в DX:AX
```

Беззнаковое умножение: команда MUL. Команда MUL (MULtiplication – умножение) умножает беззнаковые числа. На рисунке 6.2 в процедуре C10MUL дано три примера умножения: байт на байт, слово на слово и слово на байт.

```
TITLE   EXMULT  (COM) Пример команд умножения
CODESG  SEGMENT PARA 'Code'
        ASSUME  CS:CODESG,DS:CODESG,SS:CODESG
        ORG     100H
BEGIN:  JMP     SHORT MAIN
; -----------------------------------------
BYTE1   DB      80H
BYTE2   DB      40H
WORD1   DW      8000H
WORD2   DW      4000H
; -----------------------------------------
MAIN    PROC    NEAR                ;Основная процедура:
        CALL    C10MUL              ;Вызвать умножение MUL
        CALL    D10IMUL             ;Вызвать умножение IMUL
        RET
MAIN    ENDP
;               Пример умножения MUL:
;               --------------------
C10MUL  PROC
        MOV     AL,BYTE1            ;Байт * байт
        MUL     BYTE2               ; произведение в AX
        MOV     AX,WORD1            ;Слово * слово
        MUL     WORD2               ; произведение в DX:AX
        MOV     AL,BYTE1            ;Байт * слово
        SUB     AH,AH               ; расширенное множимое в AH
        MUL     WORD1               ; произведение в DX:AX
        RET
C10MUL  ENDP
;               Пример умножения IMUL:
;               ----------------------
D10IMUL PROC
        MOV     AL,BYTE1            ;Байт * байт
        IMUL    BYTE2               ; произведение в AX
        MOV     AX,WORD1            ;Слово * слово
        IMUL    WORD2               ; произведение в DX:AX
        MOV     AL,BYTE1            ;Байт * слово
        CBW                         ; расширенное множимое в AH
        IMUL    WORD1               ; произведение в DX:AX
        RET
D10IMUL ENDP

CODESG  ENDS
        END     BEGIN
```

Рисунок 6.2 - Беззнаковое и знаковое умножение

Первый пример команды MUL умножает шест. 80 (128) на шест. 47 (64). Произведение шест. 2000 (8192) получается в регистре AX.

Второй пример команды MUL генерирует шест. 10000000 в регистрах DX:AX.

Третий пример команды MUL выполняет умножение слова на байт и требует расширение байта BYTE1 до размеров слова. Так как предполагаются беззнаковые величины, в примере левый бит регистра AH равен нулю. (При использовании команды CBW значение левого бита регистра AL может быть 0 или 1). Произведение – шест. 00400000 – получается в регистрах DX:AX.

Знаковое умножение: команда IMUL. Команда IMUL (Integer MULtiplication – умножение целых чисел) умножает знаковые числа. На рисунке 6.2 в процедуре D10IMUL используются те же три примера умножения, как в процедуре C10MUL, но вместо команд MUL записаны команды IMUL.

Первый пример команды IMUL умножает шест. 80 (отрицательное число) на шест. 40 (положительное число). Произведение – шест. E000 – получается в регистре AX. Используя те же данные, команда MUL дает в результате шест. 2000, так что можно видеть разницу в использовании команд MUL и IMUL. Команда MUL рассматривает шест. 80 как +128, а команда IMUL – как -128. В результате умножения -128 на +64 получается -8192 или шест. E000.

Второй пример команды IMUL умножает шест. 8000 (отрицательное значение) на шест. 2000 (положительное значение). Произведение – шест. F0000000 – получается в регистрах DX:AX и представляет собой отрицательное значение.

Третий пример команды IMUL перед умножением выполняет расширение байта BYTE1 до размеров слова в регистре AX. Так как значения предполагаются знаковые, в примере используется команда CBW для перевода левого знакового бита в регистр AH: шест. 80 в регистре AL превращается в шест. FF80 в регистре AX. Поскольку множитель в слове WORD1 тоже имеет отрицательное значение, произведение должно получится положительное. В самом деле: шест. 00400000 в регистрах DX:AX – такой же результат, как в случае умножения командой MUL, которая предполагала положительные сомножители.

Итак, если множимое и множитель имеет одинаковый знаковый бит, команды MUL и IMUL генерируют одинаковый результат. Но если сомножители имеют разные знаковые биты, то команда MUL вырабатывает положительный результат умножения, команда IMUL – отрицательный.

Деление. Операция деления для беззнаковых данных выполняется командой DIV, а для знаковых – IDIV. Ответственность за подбор подходящей команды лежит на программисте. Существуют два основных вида операции деления.

Деление слова на байт: Делимое находится в регистре AX, делитель – в байте памяти или однобайтовом регистре. После деления остаток получается в регистре AH, частное – в AL.

Так как однобайтовое частное очень мало (максимально 255 (шест. FF) для беззнакового деления и +127 (шест. 7F) для знакового), эта команда используется ограниченно.

Деление двойного слова на слово: Делимое находится в регистровой паре DX:AX, делитель – в слове памяти или регистре. После деления остаток получается в регистре DX, частное – в регистре AX. Частное в одном слове допускает максимальное значение 32767 (шест. FFFF) для беззнакового деления и +16383 (шест. 7FFF) для знакового.

В единственном операнде команд DIV и IDIV указывается делитель. Рассмотрим следующую команду:

DIV DIVISOR

Если поле DIVISOR определено как байт (DB), то операция предполагает деление слова на байт. Если поле DIVISOR определено как слово (DW), то операция предполагает деление двойного слова на слово.

При делении, например, 13 на 3, получается результат 4 1/3. Частное есть 4, а остаток – 1. Значение содержит целую часть (4) и дробную часть (,333). Значение 1/3 и 333... суть дробные части, тогда как 1 есть остаток от деления.

Беззнаковое деление: команда DIV. Команда DIV делит беззнаковые числа. На рисунке 6.3 в процедуре D10DIV дано четыре примера деления: слово на байт, байт на байт, двойное слово на слово и слово на слово. Первый пример команды DIV делит шест. 2000 (8092) на шест. 80 (128). В результате остаток 00 получается в регистре AH, частное – шест. 40 (64) – в регистре AL.

Второй пример команды DIV выполняет прежде расширение байта BYTE1 до размеров слова. Так как здесь предполагается беззнаковая величина, то в примере левый бит регистра AH равен нулю. В результате деления остаток – шест. 12 – получается в регистре AH, частное – шест. 05 – в регистре AL.

Третий пример команды DIV генерирует остаток шест. 1000 в регистре DX и частное шест. 0080 в регистре AX.

В четвертом примере команды DIV сначала выполняется расширение слова WORD1 до двойного слова в регистре DX. После деления остаток шест. 0000 получится в регистре DX, а частное шест. 0002 – в регистре AX.

```
TITLE   EXDIV   (COM) Пример операций DIV и IDIV
CODESG  SEGMENT PARA 'Code'
        ORG     100H
BEGIN:  JMP     SHORT MAIN
; -----------------------------------------
BYTE1   DB      80H           ;Data items
BYTE2   DB      16H
WORD1   DW      2000H
WORD2   DW      0010H
WORD3   DW      1000H
; -----------------------------------------
MAIN    PROC    NEAR          ;Основная процедура
```

```
        CALL    D10DIV          ;Вызов подпрограммы DIV
        CALL    E10IDIV         ;Вызов подпрограммы IDIV
MAIN    ENDP
;               Примеры с командой DIV:
; ---------------------------------------
D10DIV  PROC
        MOV     AX,WORD1        ;Слово / байт
        DIV     BYTE1           ; остаток:частное в AH:AL
        MOV     AL,BYTE1        ;Байт / байт
        SUB     AH,AH           ; расширить делимое в AH
        DIV     BYTE3           ; остаток:частное в
AH:AL
        MOV     DX,WORD2        ;Двойное слово / слово
        MOV     AX,WORD3        ; делимое в DX:AX
        DIV     WORD1           ; остаток:частное в DX:AX
        MOV     AX,WORD1        ;Слово / слово
        SUB     DX,DX           ; расширить делимое в DX
        DIV     WORD3           ; остаток:частное в DX:AX
        RET
D10DIV  ENDP
;               Примеры с командой IDIV:
; ---------------------------------------
E10IDIV PROC
        MOV     AX,WORD1        ;Слово / байт
        IDIV    BYTE1           ; остаток:частное в AH:AL
        MOV     AL,BYTE1        ;Байт / байт
        CBW                     ; расширить делимое в AH
        IDIV    BYTE3           ; остаток:частное в
AH:AL
        MOV     DX,WORD2        ;Двойное слово / слово
        MOV     AX,WORD3        ; делимое в DX:AX
        IDIV    WORD1           ; остаток:частное в DX:AX
        MOV     AX,WORD1        ;Слово / слово
        CWD                     ; расширить делимое в DX
        IDIV    WORD3           ; остаток:частное в DX:AX
        RET
E10DIV  ENDP
CODESG  ENDS
        END     BEGIN
```

Рисунок 6.3 - Беззнаковое и знаковое деление

Знаковое деление: команда IDIV. Команда IDIV (Integer DIVide) выполняет деление знаковых чисел. На рисунке 6.3 в процедуре E10IDIV используются те же четыре примера деления, что и в процедуре D10DIV, но вместо команд DIV записаны команды IDIV. Первый пример команды IDIV делит шест. 2000 (положительное число) на шест. 80 (отрицательное число). Остаток от деления – шест. 00 получается в регистре AH, а частное – шест. C0

(-64) – в регистре AL. Команда DIV, используя те же числа, генерирует частное +64.

Только в примере 4 вырабатывается такой же результат, как для команды DIV. Таким образом, если делимое и делитель имеют одинаковый знаковый бит, то команды DIV и IDIV генерируют одинаковый результат. Но если делимое и делитель имеют разные знаковые биты, то команда DIV генерирует положительное частное, а команда IDIV – отрицательное частное. Можно обнаружить это, используя отладчик для трассировки этих примеров.

Используя команды DIV и особенно IDIV, очень просто вызвать переполнение. Переполнения приводят к непредсказуемым результатам. При делении предполагается, что частное значительно меньше, чем делимое. Деление на ноль всегда вызывает прерывание. Но деление на 1 генерирует частное, равное делимому, что тоже может вызвать прерывание. Рекомендуется использовать следующее правило: если делитель – байт, то его значение должно быть меньше, чем левый байт (AH) делителя: если делитель – слово, то его значение должно быть меньше, чем левое слово (DX) делителя.

Для команды IDIV нужно учитывать тот факт, что делимое либо делитель могут быть отрицательными; так как сравниваются абсолютные значения, нужно использовать команду NEG для временного перевода отрицательного значения в положительное.

Преобразование знака. Команда NEG обеспечивает преобразование знака двоичных чисел из положительного в отрицательное и наоборот. Практически команда NEG устанавливает противоположные значения битов и прибавляет 1. Примеры:

```
NEG  AX
NEG  BL
NEG  BINAMT    ;(байт или слово в памяти)
```

Остается небольшая проблема: с числами, представленными в двоичном формате, удобно выполнять арифметические операции, если сами числа определены в программе. Данные, вводимые в программу с файла, тоже могут иметь двоичный формат. Но данные, вводимые с клавиатуры, представлены в ASCII-формате. ASCII-коды удобны для отображения и печати, но требуют специальных преобразований в двоичный формат для арифметических вычислений.

5. Порядок выполнения работы

Используя текстовый редактор, создать и отредактировать исходный модуль программы Prog_6.asm, которая вычисляет значение X в соответствии с вариантом задания. Номер функции и значения переменных A, B и C взять из таблицы 6.1. Значение переменной D берётся равным номеру варианта. После выполнения операции деления, в дальнейших операциях учитывать только частное.

```
;Program_4 – Арифметические операции, вариант …
Data SEGMENT                    ;Открыть сегмент
                                 данных
        A  DB  1                ;Инициализировать
        B  DB 2                 ;переменные A, B, C,
                                D, X
        C  DB  3
        D  DB  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
                               ;Назначить сегментные регистры
ASSUME CS:Code, DS:Data, SS:Ourstack
Code SEGMENT                   ;Открыть сегмент кодов
Start:  mov  AX, Data           ;Инициализировать
        mov  DS, AX             ;сегментный
                                регистр DS
        xor  AX, AX             ;Очистить регистр
                                AX
```

Здесь должны быть команды вычисления арифметического выражения

```
        movAX, 4C00h   ;Завершить программу
         int 21h                ;с помощью DOS
Code  ENDS                      ;Закрыть сегмент кодов
END Start                       ;Конец исходного модуля.
```

Таблица 6.1 - Варианты заданий

№варианта	Функция	Данные		
		A	B	C
1	$X = \frac{2*A+B*D}{C-3}$	64h	14h	-4
2	$X = \frac{D*C}{2*A+B}$	16h	-50	1Bh
3	$X = \left(1 + \frac{A}{5}\right) * B - C * D$	150	111b	48h
4	$X = \frac{A^2+D}{C-B}$	15	150h	5

5	$X = (48 + 3 * A) - \frac{B}{C} * D$	5Ah	5 5h	1 1h
6	$X = \frac{B-25}{A+1} + (B + D)$	-5	3 1	
7	$X = (A + B) * (C - 4000) * (D +$	A1h	- 150	F B0h
8	$X = (A * B - C * D)^2$	Fh	1 4	1 0h
9	$X = \frac{A^2+B^2}{D-C}$	7	1 2	- 15
10	$X = \frac{(B-C)*A}{D-12}$	5	E 2h	2 25
11	$X = \frac{300-D+B*C}{A}$	8	2 6h	- 10
12	$X = \frac{65528-A*B}{(D+C)^2}$	BFh	1 4h	2
13	$X = \frac{A*(B+1)}{C} - D$	32	F h	8 0
14	$X = 3 * (A - B) + \frac{D}{C}$	99h	D 9h	1 55
15	$X = \frac{-1*(D+1)}{A+B*C}$	Ch	4	9

6. Форма отчета о работе

Лабораторная работа № ___
Номер учебной группы _____
Фамилия, инициалы учащегося _____
Дата выполнения работы _____
Тема работы: _____
Цель работы: _____
Оснащение работы: _____
Результат выполнения работы: _____

_____

_____

7. Контрольные вопросы и задания

1. Формат команды «сложить», ее операнды.
2. Формат команды «вычесть», ее операнды.
3. Формат команды «умножить», ее операнды.

4. Формат команды «делить», ее операнды.
5. Каков диапазон беззнаковых чисел допустим в программах 16-тиразрядного микропроцессора?
6. Каков диапазон чисел со знаком допустим в программах 16-ти разрядного микропроцессора?
7. Какую информацию содержат арифметические флаги операций?
8. Какие флаги устанавливаются при выполнении команд «сложить» и «вычесть».
9. Какие флаги устанавливаются при выполнении команд «умножить» и «делить».
10. Как выполнить сложение (вычитание) двух операндов, находящихся в памяти?
11. Как выполнить умножение двух операндов, находящихся в памяти?
12. Как выполнить деление двух операндов, находящихся в памяти?

8. Рекомендуемая литература

Финогенов, К. Г. Основы языка Ассемблера [Текст] / К. Г. Финогенов. – М.: Радио и связь, 2000.
Финогенов, К. Г. Использование языка Ассемблера [Текст]: учеб. пособие для вузов / К.Г. Финогенов. – М.: Горячая линия Телеком, 2004.
Юров, В. И. Assembler [Текст]: учеб. пособие для вузов / В. И. Юров. 2-е изд. – СПб.: Питер, 2007.