МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПУТЕЙ СООБЩЕНИЯ (МИИТ)

Кафедра «Управление и информатика в технических системах»

М.А. Васильева

Утверждено
редакционно –издательским
советом университета

МАШИННО-ОРИЕНТИРОВАННЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Методические указания к лабораторным работам
по курсу
«Машинно-ориентированные языки программирования»

для студентов специальности
«Управление и информатика в технических системах»

Москва – 2004

УДК 681.3 (075)

В-19

Васильева М.А. Машинно-ориентированные языки программирования: методические указания к лабораторным работам. - М.:МИИТ, 2004. –29с.

В методических указаниях представлен необходимый практический материал по изучению основ машинно—ориентированного программирования, сформулированы задания, даны указания по выполнению лабораторных работы.

Лабораторная работа № 1
Внутреннее представление целочисленных данных в IBM PC

Цель работы. Выполнить перевод заданных преподавателем чисел из десятичной в двоичную систему счисления. Дать их внутреннее (машинное) представление в соответствии с диапазоном в знаковых и беззнаковых форматах типов **ShortInt, Byte, Integer, Word, LongInt.** Машинное представление данных должно быть дано в двоичной и шестнадцатеричной системах счисления.

Порядок работы

1) вычислить числа для своего варианта;
2) перевести целые числа из 10-тичной в 2-ичную (или 16-ричную) систему счисления;
3) получить их внутреннее представление;
4) написать программу описания этих чисел на языке Ассемблера и получить листинг;
5) проверить правильность своих выкладок.

Целочисленные данные должны быть представлены во всех возможных для формы **Winl6** форматах с учетом их диапазона представления.
В отчете по лабораторной работе должен быть представлен подробный проток перевода всех заданных чисел из 10-тичной в 2-ичную и 16-ричную системы счисления **(теоретическое введение).**

Пример решения типового варианта

Преподаватель задает два базовых числа **X=± 2235** и **Y=± 40**. Студент должен прибавить и отнять от них № своего варианта.
Например, пусть № = **19**. Тогда получатся следующие восемь целых чисел для варианта №= 19, а именно:

1) 40 + 19 = 59;
2) 40 – 19 = 21;
3) - 40 + 19 = - 21;
4) - 40 – 19 = - 59;
5) 2235 + 19 = 2254;
6) 2235 – 19 = 2216;
7) -2235 + 19 = - 2216;
8) -2235 – 19 = - 2254

Решение:

Для перевода целого числа , представленного в десятичной системе счисления, в систему счисления с основанием **q** необходимо данное число делить на основание до получения целого остатка, меньшего **q**. Полученное частное снова необходимо делить на основание до получения целого остатка, меньшего **q**, и т. д. до тех пор, пока последнее частное будет меньше **q**. Число в системе счисления с основанием **q** представится в виде упорядоченной последовательности остатков деления в порядке, обратном их получению. Причем старшую цифру числа дает последний остаток, а младшую — первый.

Для того чтобы сменить знак числа, нужно инвертировать все его биты и прибавить к нему единицу — получим представление ОТРИЦАТЕЛЬНОГО числа в дополнительном коде.

Для целых типов со знаком, под знак отводится старший бит, причем для положительных чисел он равен 0, а для отрицательных — 1.

Протокол перевода чисел.

59	**2**				
58	**29**	**2**			
1	**28**	**14**	**2**		
	1	**14**	**7**	**2**	
		0	**6**	**3**	**2**
			1	**2**	**1**
				1	

59d → 0011.1011b (BYTE) →3Bh

-59d → 1) 0011.1011b – двоичный код числа |-59|

2) 1100.0100b – инверсия

3) + 1

1100.0101b – дополнительный код → C5h

21	2			
20	10	2		
1	10	5	2	
	0	4	2	2
		1	2	1
			0	

21d → 0001.0101b (BYTE) → 15h

-21 → 1) 0001.0101b – двоичный код числа |-21|

2) 1110.1010b– инверсия

3) + 1

1110.1011b – дополнительный код →EBh

2254	2										
2254	1127	2									
0	1126	563	2								
	1	562	281	2							
		1	280	140	2						
			1	140	70	2					
				0	70	35	2				
					0	34	17	2			
						1	16	8	2		
							1	8	4	2	
								0	4	2	2
									0	2	1
										0	

2254d → 0000.1000.1100.1110b (WORD) →08CEh

-2254d→ 1) 0000.1000.1100.1110b – двоичный код числа |-2254|

2) 1111.0111.0011.0001b– инверсия

3) + 1

1111.0111.0011.0010b – дополнительный код

→F732h

2216	2										
2216	1108	2									
0	1108	554	2								
	0	554	277	2							
		0	276	138	2						
			1	138	69	2					
				0	68	34	2				
					1	34	17	2			
						0	16	8	2		
							1	8	4	2	
								0	4	2	2
									0	2	1
										0	

2216d → 0000.1000.1010.1000b (WORD)→08A8h

-2216d→ 1) 0000.1000.1010.1000b – двоичный код числа |-2216|

2) 1111.0111.0101.0001b– инверсия

3) + 1

1111.0111.0101.1000b – дополнительный код

→F732h

В формате **LongInt** (4 байта):

2216 → 000008A8
2254 → 000008CE
-2254 → FFFFF732
-2216 → FFFFF758

Теперь напишем программу на языке Ассемблера с указанием всех данных. Назовем этот файл **Lab_1.asm**. Поскольку данные могут находиться либо в сегменте данных, либо в сегменте кода, определимся, например, на сегменте данных (для данного примера это НЕ принципиально). Модель для этого примера тоже безразлична.

Исходный текст программы Lab_1.asm

```
title lab1
.MODEL tiny
.DATA
;----------------byte------------
i1      db      59
        db      21
;----------------word------------
iw      dw      59
        dw      21
        dw      2254
        dw      2216
;-------------shortint-----------
is      db      -59
        db      -21
;-------------integer------------
ii      dw      -59
        dw      -21
        dw      -2254
        dw      -2216
;-------------longint------------
il      dd      2254
        dd      2216
iil     dd      -2254
        dd      -2216

        END
```

Чтобы посмотреть, правильно ли мы представили данные в машинных форматах, необходимо выйти из **Pascal**'я и войти в **Norton Commander** (желательно при отладке и компиляции всех своих файлов на Ассемблере сохранять их временно в директории **../BP/Bin/**). Далее набираем командную строку **tasm.exe lab_1.asm/L**. Данная команда создаст файл листинга – **lab_1.lst**. Чтобы посмотреть файл листинга необходимо нажать клавишу **F3**.

Лабораторная работа №2
Команды пересылки и обмена информацией

Цель работы. Написать и отладить программу обмена данными на алгоритмическом языке Паскаль, используя модуль, выполняющий пересылку информации, написанный на языке Ассемблер.

Порядок работы.

1) Вызвать среду программирования Borland Pascal. В редакторе одно окно (файл) назвать, например, **Lab_2.pas**, другое окно (файл), например, **Lab2.asm**. СЛЕДИТЕ, ЧТОБЫ ИМЕНА МОДУЛЕЙ БЫЛИ РАЗНЫЕ.
2) Написать и отладить программу ввода и вывода данных, заданных **8, 16** и **32** битами на алгоритмическом языке Паскаль, используя при вводе данных проверку на допустимый диапазон;
3) Написать модуль обмена информацией на языке Ассемблер. Откомпилировать файл **Lab2.asm**, вызвав через **Shift-F3** компилятор **TASM**. Если нет ошибок, должен получиться объектный файл **Lab2.obj**;
4) Подключить модуль, написанный на Ассемблере, в исполняемую программу;
5) Выполнить тестовые примеры и сделать выводы.

Пример решения типового варианта (БЕЗ ПРОВЕРКИ НА ДОПУСТИМЫЙ ДИАПАЗОН!!!)

2) Исходный текст программы на языке Паскаль.

```
Program Lab_2;
Uses crt;
Var
   y,k:LongInt;                  // 4 байта
   x,a:Integer; {x,a,c:Word}     // 2 байта (16 бит)
   b,z:Byte; {b,z:ShortInt}      // 1 байт (8  бит)
begin
      ClrScr;
      Writeln(' Вычислить:        y = k; y,k :
LongInt;');
      Write ('Введите значение k = ');
```

```
      Readln(k);
      y :=k;
      Writeln(' Вычислить:        x = a; a,x :
Integer;');
      Write ('Введите значение a = ');
      Readln(a);
      x := a;
      Writeln(' Вычислить:        z = b; b,z : Byte;');
      Write ('Введите значение b =');
      z := b;
      writeln ('ПАСКАЛЬ: x=',x,';   z=',z,';  y=',y );
      repeat until keypressed;

end.
```

3) Исходный текст модуля на языке Ассемблер.

```
        Title Lab2
        .Model Large
        .Data
        Extrn y:DWord,  k:DWord     ;Extrn - директива
описания внешних имен. Она указывает компилятору и
компоновщику, что данное имя имеет определенный тип,
его предполагается использовать в данном
ассемблеровском модуле, но память для него выделена в
другом месте
                                    ; y,k:LongInt
        Extrn a:Word,   x:Word      ; x,a:Integer
        Extrn b:Byte,   z:Byte      ; z,b: byte

        .code
        Public Lab2Asm  ;Public  - директива описания
общих имен - указывает компилятору и компоновщику, что
данное имя (его адрес) должно быть доступно для других
программ.Имена могут быть метками, переменными или
именами подпрограмм, как в нашем случае Lab2Asm - имя
процедуры
Lab2Asm proc far        ;начало процедуры Lab2Asm
        mov ax,a        ; ax  ←  a
        mov x,ax        ; x   ←  <ax>
```

```
        mov bl,b          ; bL  ←   b
        mov z,bl          ; z   ←   <bL>
        mov cx,Word ptr k   ;cx  ← младшая часть k
        mov dx,word ptr k+2 ;dx  ← старшая часть k   } <Cx:Dx> ⇐ k
        mov Word ptr y,cx     ; младшая часть y
        mov Word ptr y+2,dx   ; старшая часть y      } y ⇐ <Cx:Dx>
  ret                     ; КОМАНДА возврата в
                            точку вызова процедуры
Lab2Asm endp                    ;конец процедуры Lab2Asm
        end
```

4) В исходный текст программы на Паскале необходимо вставить несколько строк – соглашение по интерфейсу. Вставленные строки отмечены жирным шрифтом.

```
Program Lab_2;
{$L Lab2}// вызов ASM-модуля Lab2.obj
{$F+}    //Директива FAR-вызова процедур и функций
Uses crt;
Var
   y,k:LongInt;                  // 4 байта
   x,a:Integer; {x,a,c:Word}     // 2 байта (16 бит)
   b,z:Byte; {b,z:ShortInt}      // 1 байт (8  бит)

Procedure Lab2Asm(var x:Integer;var z:Byte;var
y:LongInt);external;
{Описание внешней процедуры – она реализована на Ассемблере}
begin
      ClrScr;
      Writeln(' Вычислить:        y = k; y,k :
LongInt;');
      Write ('Введите значение k = ');
      Readln(k);
      y :=k;
      Writeln(' Вычислить:        x = a; a,x :
Integer;');
      Write ('Введите значение a = ');
      Readln(a);
      x := a;
      Writeln(' Вычислить:        z = b; b,z : Byte;');
```

```
      Write ('Введите значение b =');
      z := b;
      writeln ('ПАСКАЛЬ: x=',x,';   z=',z,';  y=',y );
      {Делаем вычисления этих переменных на
                                        Ассемблере}

      y:=0;
      z:=0;
      x:=0;
      Lab2Asm(x,z,y);//Вызов процедуры на Ассемблере
      Writeln('Ассемблер x=',x,';   z=',z,';  y=',y);
      repeat until keypressed;
end.
```

Для отладки модуля, написанного на Ассемблере, можно использовать **ИНТЕГРИРОВАННЫЙ ОТЛАДЧИК Turbo Pascal – 7.0x:**

1) Занести в EXE – файл отладочную информацию – **Options/Compiler/Debugging:**

- **Debug Information**
- **Local Symbols**

2) Подключить отладчик – **Option/Debugger:**

- **Integrated**

3)Установить контрольные точки: **F4 – Go to Cursor.**

4) Подключить окно CPU: **Debug/Register**.

5) Запустить отладочный режим: **F7 – trace** (с трассировкой подпрограмм), **F8 – Step** (БЕЗ трассировки подпрограмм).

ЗАМЕЧАНИЕ.

Использование в языке Паскаль меток является "дурным тоном" и снижает "читаемость" программы. Использование процедур и функций, напротив, только приветствуется. Поэтому, в качестве ПРИМЕРА обеспечения корректности вычислений на Паскале приводится данная программа.

```
Program Correction;
Uses crt;
Const
     B='Это не числовой тип!';
var
   A1:Shortint;
```

```
   A2:Integer;
   A3:Byte;
   A4:Word;
   A:LongInt;
   code:Integer;
   k:string;
function War(sl:string):boolean;
begin
     Writeln(sl);
     War:=false;

end;
function Cor(var
A:LongInt;Shortmin,ShortMax:LongInt;k:string):boolean;
var
   Ok:boolean;
begin
     Ok:=true;
     Val(k,A,code);
     if code=0 then
        begin
             if (A<ShortMin) or (A>ShortMax) then
             Ok:=War('Число выходит за рамки
требуемого типа!');
        end
     else Ok:=War(B);
     Cor:=Ok;
end;
Procedure Input(var k:string;Inv:string);
begin
     Write('Input '+Inv);
     Readln(k);
end;
begin
   clrscr;
   Input(k,'ShortString A1=');
   if Cor(A,-128,127,k) then
      begin
           A1:=A;
           k:='';
            .........
```

```
      end;
   Input(k,'Integer A2=');
   if Cor(A,-32768,32767,k) then
      begin
           A2:=A;
           k:='';
            .........
      end;
 .........
//Аналогично для форматов данных Byte и Word
.........
   Input(k,'LongInt A5=');
   Val(k,A,code);
   if code<>0 then Writeln(B)
   else  .........
   repeat until keypressed;

      end.
```

Лабораторная работа №3
Команда сложения ADD

Цель работы. Написать и отладить программу сложения 8 и 16 битных данных на алгоритмическом языке Паскаль, используя модуль, выполняющий сложение, написанный на языке Ассемблер.

Порядок работы.

1) Написать и отладить программу сложения 8 и 16 битных данных на языке Паскаль;
2) Написать и отладить модуль сложения 8 и 16 битных данных на языке Ассемблер;
3) Подключить модуль к основной программе;
4) Выполнить тестовые примеры.

Пример решения типового варианта

Исходный текст модуля на языке Ассемблер

```
Title p_ADD
        .Model Large
        .Data
        Extrn A1:Byte,B1:Byte,C1:Byte
        Extrn A2:Word,B2:Word,C2:Word
        .Code
        Public ADD_Byte ;проц.сложения 8-битных данных
ADD_Byte proc far
         Mov A1,A1      ;AL ← A1
         Add A1,B1      ;AL ← <AL>+B1
         Mov C1,A1      ;C1 ← <AL>
         Ret
ADD_Byte endp
        Public ADD_Word ; проц.сложения 16-битных д.
ADD_Word proc far
         Mov Ax,A2      ;Ax ← A2
         Add Ax,B2      ;Ax ← <Ax>+B2
         Mov C2,Ax      ;C2← <Ax>
         Ret
ADD_Word endp
         end
```

Лабораторная работа №4
Команда сложения ADC

Цель работы. Написать и отладить программу сложения 32 битных данных на алгоритмическом языке Паскаль, используя модуль, выполняющий сложение, написанный на языке Ассемблер.

Порядок работы.

1) Написать и отладить программу сложения 32 битных данных на языке Паскаль;
2) Написать и отладить модуль сложения 32 битных данных на языке Ассемблер;
3) Подключить модуль к основной программе;
4) Выполнить тестовые примеры.

Пример решения типового варианта

Исходный текст модуля на языке Ассемблер

```
Title p_ADC
        .Model Large
        .Data
        Extrn A3:DWord,B3:DWord,C3:DWord
        .Code
        Public ADC_Dword ;Процедура сложения 32-битных
                          данных
ADC_DWord proc far
         Mov Ax,Word Ptr A3   ;AX ← мл.часть A3
         Mov Bx,Word Ptr A3+2 ;Bx ← ст.часть A3
         Mov Cx,Word Ptr B3   ;Cx ← мл.часть B3
         Mov Dx,Word Ptr B3+2 ;Dx ← ст.часть B3
         Add Ax,Cx                ;Ax ← <Ax>+<Cx>
         Adc Bx,Dx                ;Bx ←
                                     <Bx>+<Dx>+<CF>
         Mov Word Ptr C3,Ax   ;мл.часть C3 ← <Ax>
         Mov Word Ptr C3+2,Bx ;ст.часть C3 ← <Bx>
         Ret
ADC_DWord endp
          End
```

Лабораторная работа №5
Команда вычитания SUB

Цель работы. Написать и отладить программу вычитания 8 и 16 битных данных на алгоритмическом языке Паскаль, используя модуль, выполняющий вычитание, написанный на языке Ассемблер.

Порядок работы.

1) Написать и отладить программу вычитания 8 и 16 битных данных на языке Паскаль;
2) Написать и отладить модуль вычитания 8 и 16 битных данных на языке Ассемблер;
3) Подключить модуль к основной программе;
4) Выполнить тестовые примеры.

Пример решения типового варианта

Исходный текст модуля на языке Ассемблер

```
Title p_SUB
        .Model Large
        .Data
        Extrn A1:Byte,B1:Byte,C1:Byte
        Extrn A2:Word,B2:Word,C2:Word
        .Code
        Public SUB_Byte ;проц.вычитания 8-битных дан.
SUB_Byte proc far
         Mov A1,A1              ;AL ← A1
         SUB A1,B1              ;AL ← <AL>-B1
         Mov C1,A1              ;C1 ← <AL>
         Ret
SUB_Byte endp
        Public SUB_Word ; проц.вычитания 16-битных д.
Sub_Word proc far
         Mov Ax,A2              ;Ax ← A2
         SUB Ax,B2              ;Ax ← <Ax>-B2
         Mov C2,Ax              ;C2 ← <Ax>
         Ret
SUB_Word endp
         end
```

Лабораторная работа №6
Команда вычитания SBB

Цель работы. Написать и отладить программу вычитания 32 битных данных на алгоритмическом языке Паскаль, используя модуль, выполняющий вычитание, написанный на языке Ассемблер.

Порядок работы.

1) Написать и отладить программу вычитания 32 битных данных на языке Паскаль;
2) Написать и отладить модуль вычитания 32 битных данных на языке Ассемблер;
3) Подключить модуль к основной программе;
4) Выполнить тестовые примеры.

Пример решения типового варианта

Исходный текст модуля на языке Ассемблер

```
Title p_SBB
        .Model Large
        .Data
        Extrn A3:DWord,B3:DWord,C3:DWord
        .Code
        Public SBB_DWord
SBB_DWord proc far ;процедура вычитания 32-битных
                   данных
         Mov Ax,Word Ptr A3     ;AX ← мл.часть A3
         Mov Bx,Word Ptr A3+2   ;Bx ← ст.часть A3
         Mov Cx,Word Ptr B3     ;Cx ← мл.часть B3
         Mov Dx,Word Ptr B3+2   ;Dx ← ст.часть B3
         SUB Ax,Cx              ;Ax ← <Ax>-<Cx>
         SBB Bx,Dx              ;Bx ← <Bx>-<Dx>-<CF>
         Mov Word Ptr C3,Ax     ;мл.часть C3 ← <Ax>
         Mov Word Ptr C3+2,Bx   ;ст.часть C3 ← <Bx>
         Ret
SBB_DWord endp
          End
```

Лабораторная работа №7
Команда умножения MUL

Цель работы. Написать и отладить программу умножения 8 и 16 битных беззнаковых данных на алгоритмическом языке Паскаль, используя модуль, выполняющий умножение, написанный на языке Ассемблер.

Порядок работы.

1) Написать и отладить программу умножения 8 и 16 битных беззнаковых данных на языке Паскаль;
2) Написать и отладить модуль умножения 8 и 16 битных беззнаковых данных на языке Ассемблер;
3) Подключить модуль к основной программе;
4) Выполнить тестовые примеры.

Пример решения типового варианта

Исходный текст модуля на языке Ассемблер

```
Title p_MUL
        .Model Large
        .Data
        Extrn A1:Byte,B1:Byte
        Extrn A2:Word,B2:Word,C1:Word
        Extrn C2:DWord
        .Code
        Public MUL_Byte

MUL_Byte proc far      ;процедура беззнакового
                        умножения 8-битных данных
         Mov AL,A1     ;AL ← A1
         Mov BL,B1     ;BL ← B1
         Mul BL        ;AX ← <AL>*<BL>
         Mov C1,Ax     ;C1 ← <Ax>
         Ret
Mul_Byte endp
```

```
        Public Mul_Word
Mul_Word proc far          ; процедура беззнакового
                             умножения 16-битных данных
         Mov Ax,A2               ;Ax ← A2
         Mov Bx,B2               ;Bx ← B2
         Mul Bx                  ;<AX:DX> ← <Ax>*<Bx>
         Mov Word Ptr C2,Ax      ;мл. разряды C2 ← <Ax>
         Mov Word Ptr C2+2,Dx    ;ст. разряды C2 ← <Dx>
         Ret
MUL_Word endp
         End
```

Лабораторная работа №8
Команда умножения IMUL

Цель работы. Написать и отладить программу умножения 8 и 16 битных знаковых данных на алгоритмическом языке Паскаль, используя модуль, выполняющий умножение, написанный на языке Ассемблер.

Порядок работы.

1) Написать и отладить программу умножения 8 и 16 битных знаковых данных на языке Паскаль;
2) Написать и отладить модуль умножения 8 и 16 битных знаковых данных на языке Ассемблер;
3) Подключить модуль к основной программе;
4) Выполнить тестовые примеры.

Пример решения типового варианта

Исходный текст модуля на языке Ассемблер

```
Title p_IMUL
        .Model Large
        .Data
        Extrn A1:Byte,B1:Byte
        Extrn A2:Word,B2:Word,C1:Word
        Extrn C2:DWord
        .Code

        Public IMUL_Byte
IMUL_Byte proc far    ; процедура знакового умножения
                        8-битных данных
         Mov AL,A1       ;AL ← A1
         Mov BL,B1       ;BL ← B1
         IMul BL         ;AX ← <AL>*<BL>
         Mov C1,Ax       ;C1 ← <Ax>
         Ret
IMul_Byte endp
```

```
Public IMul_Word
IMul_Word proc far      ; процедура знакового
                          умножения 16-битных данных

          Mov Ax,A2                 ;Ax ← A2
          Mov Bx,B2                 ;Bx ← B2
          IMul Bx                   ;AX:DX ← <Ax>*<Bx>
          Mov Word Ptr C2,Ax        ;мл. разряды C2 ← <Ax>
          Mov Word Ptr C2+2,Dx      ;ст. разряды C2 ← <Dx>
          Ret
IMUL_Word endp
          End
```

Лабораторная работа №9
Команда деления DIV.
Делимое – в два раза больше по формату, чем делитель.

Цель работы. Написать и отладить программу деления 8 и 16 битных беззнаковых данных на алгоритмическом языке Паскаль, используя модуль, выполняющий деление, написанный на языке Ассемблер. Делимое – в два раза больше по формату, чем делитель.

Порядок работы.

1) Написать и отладить программу деления 8 и 16 битных беззнаковых данных на языке Паскаль;
2) Написать и отладить модуль деления 8 и 16 битных беззнаковых данных на языке Ассемблер;
3) Подключить модуль к основной программе;
4) Выполнить тестовые примеры.

Пример решения типового варианта

Исходный текст модуля на языке Ассемблер

```
Title p_DIV
        .Model Large
        .Data
        Extrn C1:Byte,B1:Byte
        Extrn C2:Word,B2:Word,A1:Word
        Extrn A2:DWord
        .Code
        Public DIV_Byte ;процедура деления 8-
                         битных беззнаковых данных
DIV_Byte proc far
         Mov AX,A1       ;Ax ← A1
         Mov BL,B1       ;BL ← B1
         DIV BL          ;AL ← <Ax>/<BL>
         Mov C1,AL       ;C1 ← <AL>
         Ret
DIV_Byte endp
```

```
        Public DIV_Word
DIV_Word proc far              ;процедура деления 16-
                               битных беззнаковых данных


         Mov Ax,Word Ptr A2   ;Ax ← мл.часть A2
         Mov Dx,Word Ptr A2+2;Dx ← ст.часть A2
         Mov Bx,B2             ;Bx ← B2
         DIV Bx                ;Ax ← <Ax:DX>/<Bx>
         Mov C2,Ax             ;C2 ← <Ax>
         Ret
DIV_Word endp
         end
```

Лабораторная работа №10
Команда деления IDIV.
Делимое – в два раза больше по формату, чем делитель.

Цель работы. Написать и отладить программу деления 8 и 16 битных знаковых данных на алгоритмическом языке Паскаль, используя модуль, выполняющий деление, написанный на языке Ассемблер. Делимое – в два раза больше по формату, чем делитель.

Порядок работы.

1) Написать и отладить программу деления 8 и 16 битных знаковых данных на языке Паскаль;
2) Написать и отладить модуль деления 8 и 16 битных знаковых данных на языке Ассемблер;
3) Подключить модуль к основной программе;
4) Выполнить тестовые примеры.

Пример решения типового варианта

Исходный текст модуля на языке Ассемблер

```
Title p_IDIV
        .Model Large
        .Data
        Extrn C1:Byte,B1:Byte
        Extrn C2:Word,B2:Word,A1:Word
        Extrn A2:DWord
        .Code
        Public IDIV_Byte          ;процедура деления 8-
                                битных знаковых данных
IDIV_Byte proc far
         Mov AX,A1                ;Ax ← A1
         Mov B1,B1                ;BL ← B1
         IDIV B1                  ;AL ← <Ax>/<BL>
         Mov C1,AL                ;C1 ← <AL>
         Ret
IDIV_Byte endp
```

```
        Public IDIV_Word
IDIV_Word proc far                  ;процедура деления 16-
                                    битных знаковых данных

          Mov Ax,Word Ptr A2        ;Ax ← мл.часть A2
          Mov Dx,Word Ptr A2+2      ;Dx ← ст.часть A2
          Mov Bx,B2                 ;Bx ← B2
          IDIV Bx                   ;Ax ← <Ax:DX>/<Bx>
          Mov C2,Ax                 ;C2 ← <Ax>
          Ret
IDIV_Word endp
    End
```

Лабораторная работа №11
Команда деления DIV

Цель работы. Написать и отладить программу деления 8 и 16 битных беззнаковых данных на алгоритмическом языке Паскаль, используя модуль, выполняющий деление, написанный на языке Ассемблер.

Порядок работы.

1) Написать и отладить программу деления 8 и 16 битных беззнаковых данных на языке Паскаль;
2) Написать и отладить модуль деления 8 и 16 битных беззнаковых данных на языке Ассемблер;
3) Подключить модуль к основной программе;
4) Выполнить тестовые примеры.

Пример решения типового варианта

Исходный текст модуля на языке Ассемблер

```
Title p_XOR
        .Model Large
        .Data
        Extrn A1:Byte,B1:Byte,C1:Byte
        Extrn A2:Word,B2:Word,C2:Word
        .Code

        Public XOR_Byte
XOR_Byte proc far          ;процедура деления 8-битных
                            данных

         Mov AL,A1         ;AL ← A1
         Xor Ah,Ah         ;BL ← B1
         Mov BL,B1         ;AL ← <Ax>/<BL>
         DIV BL            ;C1 ← <AL>
         Mov C1,AL
         Ret
XOR_Byte endp
```

```
        Public Xor_Word
Xor_Word proc far          ;процедура деления 16-битных
                            данных

         Mov Ax,A2         ;AX ← мл.часть A2
         Xor Dx,Dx         ;Dx ← ст.часть A2
         Mov Bx,B2         ;Bx ← B2
         DIV Bx            ;Ax ← <Ax:DX>/<Bx>
         Mov C2,Ax         ;C2 ← <Ax>
         Ret
Xor_Word endp

         end
```

Лабораторная работа №12
Команда деления IDIV.

Цель работы. Написать и отладить программу деления 8 и 16 битных знаковых данных на алгоритмическом языке Паскаль, используя модуль, выполняющий деление, написанный на языке Ассемблер.

Порядок работы.

1) 1.Написать и отладить программу деления 8 и 16 битных знаковых данных на языке Паскаль;
2) Написать и отладить модуль деления 8 и 16 битных знаковых данных на языке Ассемблер;
3) Подключить модуль к основной программе;
4) Выполнить тестовые примеры.
5)

Пример решения типового варианта

Исходный текст модуля на языке Ассемблер

```
Title p_ IDIV
        .Model Large
        .Data
        Extrn A1:Byte,B1:Byte,C1:Byte
        Extrn A2:Word,B2:Word,C2:Word
        .Code

        Public CBW_Byte ;процедура деления 8-битных
                          данных

CBW_Byte proc far
         Mov AL,A1      ;AX ← A1
         CBW            ;команда распространения знака
                                для формата Byte
         Mov BL,B1      ;BL ← B1
         IDIV BL        ;AL ← <Ax>/<BL>
         Mov C1,AL      ;C1 ← <AL>
         Ret
CBW_Byte endp
```

```
        Public CWD_Word
CWD_Word proc far       ;процедура деления 16-битных
                         данных

         Mov Ax,A2      ;AX ← A2
         CWD            ;команда распространения знака
                                для формата Word
         Mov Bx,B2      ;Bx ← B2
         IDIV Bx        ;Ax ← <Ax:DX>/<Bx>
         Mov C2,Ax      ;C2 ← <Ax>
         Ret
CWD_Word endp

         end
```

Учебно-методическое издание

Васильева Марина Алексеевна

Машинно-ориентированные языки программирования

Методические указания
к лабораторным работам

Подписано к печ. - 14.09.04. Формат - 60×84/16
Усл. печ.л. - 2,0, Тираж - 100.
Заказ - 557. Изд.№ 116-04. Цена - 14руб.00коп.

127994, Москва, ул. Образцова, 15. Типография МИИТа