2-1.3. Рекомендації до виконання роботи

Знання мови Паскаль може допомогти у вивченні мови Асемблера. Дійсно, знаючи оператори мови Паскаль, а також маючи результати трансляції Паскаль операторів на мову Асемблера, не важко зрозуміти, ЩО САМЕ виконують окремі команди невеликого фрагмента програми мовою Асемблера, який реалізує окремий Паскаль оператор.

Перша проблема, що необхідно вирішити, – це створити файл, в якому після кожного Паскаль оператора містились би машинні інструкції – результати трансляції Паскаль оператора на мову Асемблера. Більшість компіляторів мов високого рівня мають стандартний режим генерації такого файла, але він відсутній в інтегрованому середовищі ТурбоПаскаль. Існує багато варіантів створення вказаного файлу. Всі вони грунтуються на перетворенні програми в кодах команд процесора в програму мовою Асемблера. В даній лабораторній роботі для цього рекомендується використовувати відповідну команду налагоджувача AFD. Розглянемо процес створення такого файлу та його подальшого аналізу на прикладі нескладної програми мовою Паскаль.

Програма мовою Паскаль містить:

- невелику кількість операторів;
- дані типу integer, byte, word та масиви даних цих типів;
- найуживаніші оператори мови Паскаль (присвоєння, **if..then..else, for , while, repeat..until**);
- контрольне виведення даних на екран;

Приклад файлу test0000.pas

```
{1}   program test0000;
{2}   var
{3}     c,d: byte;
{4}     k: integer;
{5}     A: array [3..10] of integer;
{6}
{7}   begin
{8}   k:=6;
{9}   d:=0;
```

```
{10}  for c:=3 to 10 do
{11}  begin
{12}      if k<d then
{13}           d:=d+k
{14}      else d:=k-d;
{15}      A[c]:=d;
{16}      k:=k+1;
{17}  end;
{18}   for c:=3 to 10 do
{19}   write (A[c]:4);writeln;
{20}   end.
```

Настроювання інтегрованого середовища ТурбоПаскаль

Необхідно виконати наступне настроювання інтегрованого середовища ТурбоПаскаль:

- Меню **Compile**. Опцію **Destination** встановити у стан **Disk**.
- Меню **Options**. У підменю **Memory Size**, опцію **High heap limit** встановити в 0 (цим вивільняємо пам'ять, щоб уможливити завантаження програми у налагоджувач AFD).
- У підменю **Linker** опцію **Map file** встановити в стан **Detailed**, опцію **Link bufer** у стан **Disk** (цим забезпечується формування файлу карти пам'яті *.map*, який містить розподіл об'єктів програми в пам'яті).

Файл test0.map

(перелік сегментів)

Start	Stop	Length	Name	Class
00000H	000AEH	000AFH	test0000	CODE
000B0H	00950H	008A1H	System	CODE
00960H	00C11H	002B2H	DATA	DATA
00C20H	04C1FH	04000H	STACK	STACK
04C20H	04C20H	00000H	HEAP	HEAP

(структура сегмента даних)

Address	**Publics by Value**
0000:0000	@
0096:0002	OvrCodeList
0096:0004	OvrHeapSize
0096:0006	OvrDebugPtr

0096:000A OvrHeapOrg
0096:000C OvrHeapPtr
0096:000E OvrHeapEnd
0096:0010 OvrLoadList
0096:0012 OvrDosHandle
0096:0014 OvrEmsHandle
0096:0016 HeapOrg
0096:001A HeapPtr
0096:001E HeapEnd
0096:0022 FreeList
0096:0026 FreeZero
0096:002A HeapError
0096:002E ExitProc
0096:0032 ExitCode
0096:0034 ErrorAddr
0096:0038 PrefixSeg
0096:003A StackLimit
0096:003C InOutRes
0096:003E RandSeed
0096:0042 SelectorInc
0096:0044 Seg0040
0096:0046 SegA000
0096:0048 SegB000
0096:004A SegB800
0096:004C Test8086
0096:004D Test8087
0096:004E FileMode
0096:0052 c
0096:0053 d
0096:0054 k
0096:0056 A
0096:0066 Input
0096:0166 Output
0096:0266 SaveInt00
0096:026A SaveInt02
0096:026E SaveInt1B
0096:0272 SaveInt21
0096:0276 SaveInt23
0096:027A SaveInt24
0096:027E SaveInt34
0096:0282 SaveInt35
0096:0286 SaveInt36
0096:028A SaveInt37
0096:028E SaveInt38
0096:0292 SaveInt39
0096:0296 SaveInt3A
0096:029A SaveInt3B
0096:029E SaveInt3C
0096:02A2 SaveInt3D
0096:02A6 SaveInt3E
0096:02AA SaveInt3F
0096:02AE SaveInt75

(структура сегмента кодів)

Line numbers for test0000(TEST0000.PAS) segment test0000

7 0000:0000	8 0000:000F	9 0000:0015	10 0000:001A
12 0000:0025	13 0000:0030	14 0000:003E	15 0000:004D
16 0000:0061	17 0000:0068	18 0000:006F	19 0000:007A
20 0000:00A7			

Program entry point at 0000:0000

end of test0000.map

Файл карти пам'яті складається із трьох частин: *таблиці сегментів*, *структури сегмента даних* і *структури сегмента кодів*. З *таблиці сегментів* видно, що компілятор мови Паскаль формує додатковий сегмент кодів System, який значно об'ємніший за основний. Цей сегмент містить стандартні процедури компілятора, навіть якщо вони не всі потрібні для конкретної програми. Із *структури сегмента даних* також зрозуміло, що значна частина сегменту заповнена системними даними. Очевидно, що більшість із них в даній програмі також не використовується. У структурі сегменту даних жирним шрифтом відмічені дані, що визначені в програмі. Кожний ідентифікатор із структури є символічним зображенням адреси (зміщення) в сегменті, яка вказана ліворуч від ідентифікатора та праворуч від символу ":" Ліворуч від символу ":" позначається одна із можливих адрес сегменту.

У *структурі сегменту кодів* подані адреси (зміщення) у сегменті кодів послідовностей команд ПЕОМ, які згенеровані транслятором із рядків Паскаль програми з відповідними порядковими номерами. Наприклад, 11-ий рядок з ключовим словом *Begin*, який започатковує тіло циклу *for*, не має реалізації в кодах команд, а 17-ий рядок з ключовим *End*, який вказує на закінчення тіла циклу *for*, – має .

Робота з налагоджувачем AFD

Звичайно, попередньо програму testxxxx.exe необхідно завантажити у налагоджувач за допомогою команди **L** (**L**oad). Налагоджувач AFD дозволяє створювати файли, що містять результати перетворення програми в кодах

команд ЕОМ у програму мовою Асемблера (*процес зворотної трансляції*). Для цього використовується команда **PD**, що має 3 параметри:

- початкова адреса ділянки пам'яті;
- довжина ділянки;
- ім'я файлу результата.

Перший із них – початкова адреса – дорівнює молодшому (правому від символу "**:**") значенню *Program entry point*, показаному у файлі testxxxx.map (це компонента зміщення у сегменті логічної адреси точки входу в програму – Offset Program entry point). Для визначення довжини L ділянки пам'яті рекомендується використати наступну формулу:

L = Addr_end +8 - Offset_Program_entry_point

де Addr_end – зміщення у сегменті кодів останнього рядка Паскаль програми (*end*.), 8 – сукупна кількість байтів команд ПЕОМ, які реалізують цей рядок.

ЗАУВАЖЕННЯ. Другий параметр команди PD (довжина) також задається у 16-ковому форматі.

Таким чином, із файла test0000.map маємо:

L = 0A7h+8-0 = 0**Af**h,

а команда PD матиме вигляд: **PD 0, 0af, test0000.prn**

Приклад файлу test0000.prn

```
>> AFD-Pro print out                9-9-2003    21:53
9717:0000    9A00002297      CALL    9722:0000
9717:0005    55              PUSH    BP
9717:0006    89E5            MOV     BP,SP
9717:0008    31C0            XOR     AX,AX
9717:000A    9ACD022297      CALL    9722:02CD
9717:000F    C70654000600    MOV     [0054],0006
9717:0015    C606530000      MOV     [0053],00
9717:001A    C606520003      MOV     [0052],03
9717:001F    EB04            JMP     0025
9717:0021    FE065200        INC     B/[0052]
9717:0025    A05300          MOV     AL,[0053]
9717:0028    30E4            XOR     AH,AH
9717:002A    3B065400        CMP     AX,[0054]
9717:002E    7E0E            JNG     003E
9717:0030    A05300          MOV     AL,[0053]
9717:0033    30E4            XOR     AH,AH
```

```
9717:0035  03065400        ADD     AX,[0054]
9717:0039  A25300          MOV     [0053],AL
9717:003C  EB0F            JMP     004D
9717:003E  A05300          MOV     AL,[0053]
9717:0041  30E4            XOR     AH,AH
9717:0043  8BD0            MOV     DX,AX
9717:0045  A15400          MOV     AX,[0054]
9717:0048  2BC2            SUB     AX,DX
9717:004A  A25300          MOV     [0053],AL
9717:004D  A05300          MOV     AL,[0053]
9717:0050  30E4            XOR     AH,AH
9717:0052  8BD0            MOV     DX,AX
9717:0054  A05200          MOV     AL,[0052]
9717:0057  30E4            XOR     AH,AH
9717:0059  8BF8            MOV     DI,AX
9717:005B  D1E7            SHL     DI,1
9717:005D  89955000        MOV     [0050+DI],DX
9717:0061  A15400          MOV     AX,[0054]
9717:0064  40              INC     AX
9717:0065  A35400          MOV     [0054],AX
9717:0068  803E52000A      CMP     [0052],0A
9717:006D  75B2            JNZ     0021
9717:006F  C606520003      MOV     [0052],03
9717:0074  EB04            JMP     007A
9717:0076  FE065200        INC     B/[0052]
9717:007A  BF6601          MOV     DI,0166
9717:007D  1E              PUSH    DS
9717:007E  57              PUSH    DI
9717:007F  A05200          MOV     AL,[0052]
9717:0082  30E4            XOR     AH,AH
9717:0084  8BF8            MOV     DI,AX
9717:0086  D1E7            SHL     DI,1
9717:0088  8B855000        MOV     AX,[0050+DI]
9717:008C  99              CWD
9717:008D  52              PUSH    DX
9717:008E  50              PUSH    AX
9717:008F  6A00            PUSH    0000
9717:0091  9A91062297      CALL    9722:0691
9717:0096  9ADD052297      CALL    9722:05DD
9717:009B  9A91022297      CALL    9722:0291
9717:00A0  803E52000A      CMP     [0052],0A
9717:00A5  75CF            JNZ     0076
9717:00A7  C9              LEAVE
9717:00A8  31C0            XOR     AX,AX
9717:00AA  9A16012297      CALL    9722:0116
```

end of test0.prn

Оброблення файлу testxxxx.prn

Виконати ці пункти зручніше або при використанні роздруківок файлів testxxxx.pas, testxxxx.map і testxxxx.prn, або ж при розміщенні, наприклад, за допомогою додатку Notepad, вказаних 3-х відкритих файлів на екрані наступним чином:

файл testxxxx.prn	файл testxxxx.pas
файл testxxxx.map	

Далі виконується почергове перенесення рядків Паскаль програми testxxxx.pas у файл testxxxx.prn і запис їх перед командами з адресами рядків, вказаними у файлі testxxxx.map. Таким чином встановлюється відповідність між Паскаль операторами та їх реалізацією командами Асемблера.

Заміна числових адресних посилань на символічні виконується за допомогою інформації з файлу testxxxx.map. Наприклад, **0052** змінюється на логічне ім'я **c**, **0053** на **d** і т.д. В командах мовою Асемблера, у деяких випадках, необхідно вказувати довжину в байтах змінної, яка адресується. Відлагоджувач AFD у цих випадках генерує символи W/ або B/. Їх необхідно замінити відповідно на *word ptr* або *byte ptr*. В результаті проведених замін файл test0000.prn буде мати наступний вигляд:

```
>> AFD-Pro print out                9-9-2003      21:53
  begin
9717:0000    9A00002297             CALL          9722:0000
9717:0005    55                     PUSH          BP
9717:0006    89E5                   MOV           BP,SP
9717:0008    31C0                   XOR           AX,AX
9717:000A    9ACD022297             CALL          9722:02CD

    k:=6;
9717:000F    C70654000600           MOV           [k],0006
    d:=0;
9717:0015    C606530000             MOV           [d],00
    for c:=3 to 10 do
9717:001A    C606520003             MOV           [c],03
9717:001F    EB04                   JMP           0025
9717:0021    FE065200               INC           Byte ptr [c]
         begin
           if k<d then
9717:0025    A05300                 MOV           AL,[d]
```

```
9717:0028    30E4              XOR       AH,AH
9717:002A    3B065400          CMP       AX,[k]
9717:002E    7E0E              JNG       003E
        d:=d+k
9717:0030    A05300            MOV       AL,[d]
9717:0033    30E4              XOR       AH,AH
9717:0035    03065400          ADD       AX,[k]
9717:0039    A25300            MOV       [d],AL
9717:003C    EB0F              JMP       004D
        else d:=k-d;
9717:003E    A05300            MOV       AL,[d]
9717:0041    30E4              XOR       AH,AH
9717:0043    8BD0              MOV       DX,AX
9717:0045    A15400            MOV       AX,[k]
9717:0048    2BC2              SUB       AX,DX
9717:004A    A25300            MOV       [d],AL
        A[c]:=d;
9717:004D    A05300            MOV       AL,[d]
9717:0050    30E4              XOR       AH,AH
9717:0052    8BD0              MOV       DX,AX
9717:0054    A05200            MOV       AL,[c]
9717:0057    30E4              XOR       AH,AH
9717:0059    8BF8              MOV       DI,AX
9717:005B    D1E7              SHL       DI,1
9717:005D    89955000          MOV       [A+DI-6],DX
        k:=k+1;
9717:0061    A15400            MOV       AX,[k]
9717:0064    40                INC       AX
9717:0065    A35400            MOV       [k],AX
      end;
9717:0068    803E52000A        CMP       [c],0A
9717:006D    75B2              JNZ       0021
 for c:=3 to 10 do
9717:006     C606520003        MOV       [c],03
9717:0074    EB04              JMP       007A
9717:0076    FE065200          INC       byte ptr [c]
   write (A[c]:4);
   writeln;
9717:007A    BF6601            MOV       DI,0166
9717:007D    1E                PUSH      DS
9717:007E    57                PUSH      DI
9717:007F    A05200            MOV       AL,[c]
9717:0082    30E4              XOR       AH,AH
9717:0084    8BF8              MOV       DI,AX
9717:0086    D1E7              SHL       DI,1
9717:0088    8B855000          MOV       AX,[A+DI-6]
9717:008C    99                CWD
9717:008D    52                PUSH      DX
9717:008E    50                PUSH      AX
9717:008F    6A00              PUSH      0000
9717:0091    9A91062297        CALL      9722:0691
9717:0096    9ADD052297        CALL      9722:05DD
```

```
9717:009B   9A91022297      CALL     9722:0291
9717:00A0   803E52000A      CMP      [c],0A
9717:00A5   75CF            JNZ      0076
 end.
9717:00A7   C9              LEAVE
9717:00A8   31C0            XOR      AX,AX
9717:00AA   9A16012297      CALL     9722:0116
                  end of test0000.prn
```

Ознайомлення з реалізацією Паскаль операторів мовою Асемблера

Цей пункт найважливіший в даній лабораторній роботі. За допомогою довідника з системи команд мікропроцесора (наприклад, див. [2]) необхідно ознайомитись з командами, які реалізують відповідний оператор Паскаль програми, та зрозуміти (вивчити) методику реалізації цих операторів.

Асемблерна вставка

Асемблерна вставка у Паскаль програмі починається оператором **asm** і закінчується оператором **end;**. Між ними записуються машинні інструкції мови Асемблера і директиви визначення пам'яті. Всі мітки повинні починатися символом **@**. У директивах визначення пам'яті поле мітки може містити мітку, а не ім'я. В адресних частинах команд як адресні посилання можуть використовуватися імена змінних і процедур, котрі видимі (доступні) в даній точці програми. Необхідно використовувати покажчики довжини змінних (*word ptr* або *byte ptr*). Константи задаються за правилами як Паскаля, так і Асемблера. Коментарі задаються за правилами Паскаля.

Уважне вивчення команд ПЕОМ дозволяє в ряді випадків зробити більш ефективну реалізацію окремих операторів Паскаля порівняно з результатами роботи транслятора:

```
{1}  program test0000;
{2}  var
{3}    c,d: byte;
{4}    k: integer;
{5}    A: array [3..10] of integer;
{6}
{7}  begin
{8}  k:=6;
{9}  d:=0;
```

```
{10}   for c:=3 to 10 do
{11}   begin
       asm
{12     if k<d then}
       mov    al,d
       xor      ah,ah
       cmp    ax,k
       jng      @20
{13     d:=d+k}
       mov  al,byte ptr k
       add   byte ptr d,al
       jmp    @30
{14     else d:=k-d;}
@20:
       mov   al,d
       mov dl,byte ptr k
       sub   dl,al
       mov   byte ptr d,dl
{15     A[c]:=d;}
@30:
       mov   al,d
       xor     ah,ah
       mov    bl,c
       xor     bh,bh
       shl     bx,1
       mov    word ptr A[bx-6],ax
{16     k:=k+1;}
       inc    word ptr k
       end; {asm}
{17}   end;
{18}   for c:=3 to 10 do
{19}   write (A[c]:4);writeln;
{20}   end.
```

Навіть у цій дуже простій програмі можна досягти деякої економії пам'яті та підвищення швидкодії при реалізації рядків 13, 15 та 16.

2-1.4. Завдання на виконання роботи

Перше заняття

1) Скопіювати програму мовою Паскаль testxxxx.pas (табл.2-1.4), де xxxx – номер варіанту. Перевірити працездатність програми, відкомпілювавши та запустивши її у середовищі Турбо Паскаль.

2) Створити файли testxxxx.exe і testxxxx.map, здійснивши відповідне настроювання середовища Турбо Паскаль. Для виконання даного та

наступних пунктів необхідно використовувати надані вище Рекомендації до виконання роботи та приклади.

3) За допомогою команди PD налагоджувача AFD створити файл testxxxx.prn.

4) У файлі testxxxx.prn перед групою команд ПЕОМ, які сформовані транслятором на основі чергового рядка Паскаль програми testxxxx.pas, записати цей Паскаль-рядок. Інформація про початкові адреси рядків береться із структури сегмента кодів, що у файлі testxxxx.map.

5) За інформацією з файла testxxxx.map замінити у файлі testxxxx.prn числові адресні посилання на символічні.

Друге заняття

1) Ознайомитись з реалізацією операторів Паскаль програми командами мови Асемблера.

2) Вибрати декілька Паскаль операторов (наприклад, тіло циклу) і замінити їх у програмі testxxxx.pas асемблерною вставкою, намагаючись досягти, якщо можливо, економії часу виконання і/або пам'яті.

3) Переконатись у правильності функціонування модифікованої програми testxxxx.pas шляхом виведення на екран результатів та їх порівняння з результатами немодифікованої програми.

Таблиця 2-1.4

Варіанти завдання

1.	**2.**

Program test001;
Var
  i,j,k: integer;
  A:array[3..11] of integer;
Begin
  k:=1;
  i:=3;
  repeat
  j:=i+k*2;
  if j>11
  then A[i]:=j
  else
  begin
	Program test002;
	Var
	  i,j,k: integer;
	  A:array[2..11] of integer;
	Begin
	  k:=1;
	  i:=2;
	  while i<=11 do
	  begin
	  j:=k+i*2;
	  if j>19
	  then A[i]:=j
	  else

```
      k:=k+1;
      A[i]:=k;
    end;
  inc(i);
 until i>11;
 for i:=2 to 10 do write(A[i]:4);
 writeln;
end.
```

```
      begin
        k:=k+3;
        A[i]:=k;
      end;
    inc(i);
  end;
 for i:=2 to 11 do
 write(A[i]:4);writeln;
end.
```

3.

```
Program test003;
Var
 i,j,k: integer;
 A:array[3..11] of integer;
Begin
 k:=1;
 for i:=3 to 11 do
   begin
     j:=k+i*3;
     if j>18
       then A[i]:=j
     else
       begin
         k:=k+2;
         A[i]:=k;
       end;
   end;
 for i:=3 to 11 do
   write(A[i]:4);writeln;
end.
```

4.

```
Program test004;
Var i,d:integer;
 A:array [4..11] of integer;
Begin
 d:=1;
 i:=11;
 repeat
   d:= i mod 2;
    if d=0 then
      A[i]:=i
    else A[i]:=-i;
     dec(i);
 until i<2;
 for i:=2 to 7 do
   write(A[i]:4);writeln;
End.
```

5.

```
Program test005;
Var i,d:integer;
  A:array [5..11] of integer;
Begin
  d:=1;
  i:=5;
 while i<=11 do
   begin
     A[i]:=i or d;
     d:=d + A[i];
      if d>10 then A[i]:=127-i;
        inc(i);
   end;
 for i:=5 to 11 do
    write(A[i]:4);writeln;
 End.
```

6.

```
Program test006;
Var
 p,k:integer;
A:array[6..13] of integer;
Begin
   k:=1;
   p:=6;
 repeat
     if p< 10 then
        k:=k or p
     else k:=k and p;
        A[p]:=k;
        p:=p+1;
until p>=14;
    for p:=6 to 13 do
    write(A[p]:4);writeln;
 End.
```

7.

```
Program test007;
Var  a1,c:integer;
       b:byte;
       A:array[7..13] of integer;
Begin
 a1:=3;
 c:=0;
 for  b:=7 to 13 do
  begin
    c:=c+3;
     if c<9 then
         a1:=5*c+b+1
      else a1:=b shl 2;
         A[b]:=a1;
  End;
for b:=7 to 13 do
     write(A[b]:4);writeln;
End.
```

8.

```
Program test008;
 Var  a1,c:integer;
       b:byte;
      A:array[8..20] of integer;
Begin
  A1:=2;
  b:=20;
  while  b>=8 do
   begin
     if b<15 then
       c:=a1*2
     else c:=c-a1;
     A[b]:=c;
    A1:=a1+b;
   dec(b);
  end;
for b:=8 to 20 do
     write(A[b]:4);writeln;
End.
```

9.

```
Program test009;
 Var  a,c:integer;
       b:byte;
       A1:array[9..20] of integer;
Begin
  a:=12;
  b:=9;
  repeat
     if a>b then  c:=a*3-b
     else c:= a*2+b;
      A1[b]:=c;
      inc(b);
  until b=21;
 for b:=9 to 20 do
     write(A1[b]:4);writeln;
 End.
```

10.

```
Program test0010;
Var  l,k:integer;
       j:byte;
     A:array[0..10] of integer;
Begin
  l:=2;
  k:=0;
   for  j:=0 to 10 do
  begin
       A[j]:=k;
      if  (j<2) or (j=4) then   k:=l+k
      else k:=l-k;
   end;
    for l:=0 to 10 do
      write(A[l]:4);writeln;
 End.
```

11.

```
Program test0011;
Var  k:integer;
       j:byte;
      A:array[1..11] of integer;
Begin
  k:=3;  j:=1;
  while j<=11 do
   begin
     if  j<>7 then
```

12.

```
Program test0012;
Var  l,k:integer;
       j:byte;
      A:array[2..12] of integer;
Begin
  l:=2;   k:=64;
  j:=12;
  repeat
     if  (j mod 4)=0  then
```

```
      begin
        k:=k+j;
      end;
      A[j]:=k;
      j:=j+1;
  end;
  for j:=1 to 11 do
      write(A[j]:4);writeln;
End.
```

```
        k:=k div l
      else k:=k+l;
        A[j]:=k;
        dec(j);
  until j<2;
 for j:=2 to 12 do
      write(A[j]:4);writeln;
End.
```

13.

```
Program test0013;
Var p,j:integer;
       A: array [3..13] of integer;
Begin
   for p:=3 to 13 do
   begin
       A[p]:=4*p;
       If A[p]<9 then A[p]:=A[p]+1;
   End;
for p:=3 to 13 do
      write(A[p]:4);writeln;
End.
```

14.

```
Program  test0014;
Var i,d:byte;
   A:array [2..5] of integer;
Begin
   d:=1;
   for i:=2 to 5 do
     begin
       A[i]:=d*I+1;
       d:= A[i]-d+1;
          if d>3 then  d:=3;
     end;
 for i:=2 to 5 do
      write(A[i]:4);writeln;
End.
```

15.

```
Program test0015;
Var  j:integer;
       A:array [5..15] of byte;
Begin
      j:=7 ;
     A[6]:=1;    A[5]:=1;
     repeat
       A[j]:=A[j-1]+A[j-2];
          If A[j]>127
        Then A[j]:=0;
       inc(j);
    until j>15;
   for j:=5 to 15 do
      write(A[j]:4);writeln;
End.
```

16.

```
Program  test0016;
Var i,s:integer;
   A:array [6..16] of integer;
Begin
  s:=0;
  for i:=6 to 16 do
     begin
       A[i]:=2*i;
       s:=s+A[i];
         if s>10 then A[i]:=s;
     end;
for i:=6 to 16 do
      write(A[i]:4);writeln;
End.
```

17.

```
Program test0017;
Var i,s:integer;
   A:array [7..17]of integer;
Begin
  s:=0; i:=17
```

18.

```
Program  test0018;
Var i,s:integer;
   A:array [8..18] of integer;
Begin
  s:=0;
```

```
while i>=7 do
    begin
      A[i]:=2*i;
      s:=s+A[i];
        if s>15 then A[i]:=s;
      dec(i);
    end;
 for i:=7 to 17 do
     write(A[i]:4);writeln;
End.
```

```
i:=8;
  repeat
      A[i]:=2*i;
      s:=s+A[i];
         If s>80 then
      A[i]:=s;
      inc(i);
  until i>18;
 for i:=8 to 18 do
     write(A[i]:4);writeln;
End.
```

19.

```
Program  test0019;
Var i,m:byte;
   A:array [9..19] of byte;
Begin
   m:=1;
   for i:=9 to 19 do
   begin
       m:=(m xor i) and 1;
       if m=1 then
          A[i]:=2*i
      Else A[i]:=2*i+1;
    end;
 for i:=9 to 19 do
     write(A[i]:4);writeln;
End.
```

20.

```
Program test0020;
Var i,m:integer;
   A:array [10..20] of integer;
Begin
   m:=1;
   i:=10;
 while i<=20 do
    begin
      A[i]:=3*i;
      m:=m+A[i];
         if m>100 then A[i]:=2*i;
      inc(i);
    end;
 for i:=10 to 20 do
     write(A[i]:4);writeln;
End.
```

21.

```
Program test0021;
Var    i,m:integer;
   A:array [13..21] of integer;
Begin
   m:=0;
   i:=21;
   repeat
      m:=m+i;
        if m<50 then
          A[i]:=5*i
       Else A[i]:=m;
      dec(i)
    until i<13;
for i:=13 to 21 do
     write(A[i]:4);writeln;
End.
```

22.

```
Program  test0022;
Var i,r:integer;
   A:array [13..22] of integer;
Begin
  r:=0;
  for i:=13 to 22 do
  begin
      r:=r+i*2;
       if r<100 then
         A[i]:=r
      Else
         A[i]:=r and i;
  end;
 for i:=13 to 22 do
     write(A[i]:4);writeln;
End.
```

23.

```
Program test0023;
Var i,r:integer;
   A:array [13..23] of integer;
Begin
  r:=100;
  i:=23;
  while i>=13 do
  begin
       A[i]:=2*i;
       r:=r-A[i];
         if r<0 then
           A[i]:=-A[i]+1;
       dec(i);
  end;
 for i:=13 to 23 do
      write(A[i]:4);writeln;
End.
```

24.

```
Program  test0024;
Var i,r:integer;
   A:array [14..24] of integer;
Begin
  r:=0; i:=15;
  A[14]:=4;
  repeat
       A[i]:=A[i-1]+r;
       r:=r or i;
          if r>21 then
             A[i]:=r;
       inc(i);
  until i>24;
for i:=14 to 24 do
      write(A[i]:4);writeln;
End.
```

25.

```
Program test0025;
Var  j:integer;
       A:array [4..14] of integer;
Begin
     j:=5;
    A[4]:=0;
    while  j<=14 do
    begin
       if A[j-1]<5 then
         A[j]:=A[j-1]+j;
       inc(j);
     end;
for j:=4 to 14 do
      write(A[j]:4);writeln;
   end.
```

26.

```
Program test0026;
Var
 p,i:integer;
A:array[16..26] of byte;
Begin
   p:=16;
   i:=0;
  while p<26 do
  begin
      if p>=20 then
           i:=i or p
      else i:=i and p;
      p:=p+1;
      A[p]:=i;
  end;
    for i:=5 to 10 do
      write(A[i]:4);writeln;
End.
```

27.

```
Program test0027;
Var
p,i:integer;
A:array[2..7] of word;
Begin
  p:=6;
  for i:=7 downto 2 do
  begin
     if i<6 then
       p:=p+1
```

28.

```
Program test0028;
Var
i,j:integer;
A:array[2..8] of integer;
Begin
  i:=2;
  j:=3;
  repeat
      if j<5 then
         i:=i+2*j
```

Продовження табл. 2-1.4

else p:=p-1;
A[i]:=p;
end;
for i:=2 to 7 do
write(A[i]:4);writeln;
End.
	else i:= i+j;
	A[j]:=i;
	j:=j+1;
	until j>8;
	for i:=2 to 8 do
	write(A[i]:4);writeln;
	End.

29.
Program test0029;
Var
i,j:integer;
A:array[2..10] of integer;
Begin
i:=0;
j:=2;
while i<=20 do
begin
i:=i+2;
j:=j+1;
if i<17 then
A[j]:=i
else A[j]:=3;
end;
for i:=2 to 10 do
write(A[i]:4);writeln;
End.
	30.
	Program test0030;
	Var
	i,k:integer;
	A:array[1..9] of word;
	Begin
	k:=0;
	for i:=1 to 9 do
	begin
	k:=k+i;
	if k<9 then
	A[i]:=i
	else A[i]:=k;
	end;
	for l:=1 to 9 do
	write(A[i]:4);writeln;
	End.

2-1.5. Контрольні запитання

1. Що відображає мовою Асемблера ідентифікатор змінної мови Паскаль?

2. Як реалізуються мовою Асемблера оператори присвоєння?

3. Як реалізуються мовою Асемблера оператори **if...then** ?

4. Як реалізуються мовою Асемблера оператори **for**?

5. Як реалізуються мовою Асемблера оператори **while**?

6. Як реалізуються мовою Асемблера оператори **repeat..until**?

7. Які вимоги ставлять до асемблерних вставок у програмах мовою Паскаль?