МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Р. Рикалюк, Л. Галамага, Р. Селіверстов

ЛАБОРАТОРНИЙ (СИМУЛЯЦІЙНИЙ) ПРАКТИКУМ
з курсу
“АРХІТЕКТУРА ОБЧИСЛЮВАЛЬНИХ СИСТЕМ”
Частина 2

Львів ЛНУ 2017

Рикалюк Р. Є., Галамага Л. Б., Селіверстов Р. Г. Лабораторний (симуляційний) практикум з курсу “Архітектура обчислювальних систем”. Частина 2. — Видавн. центр Львів. ун-ту, 2017. — 20 с.

Розглянуто найуживаніші команди цілочислового мікропроцесора та співпроцесора, синтасис мови програмування Assembler, сформульовано завдання до проведення лабораторних робіт з курсу “Архітектура обчислювальних систем” (теми “Базові принципи організації мікропроцесора”, “Машинна мова”, “Програмна архітектура процесора”, “Переривання”, “Режими адресації пам'яті та пристроїв вводу-виводу”, “Специфіка виконання команд цілочисловим процесором та співпроцесором”).

Для студентів факультету прикладної математики та інформатики.

Практикум містить методичні рекомендації та завдання до виконання циклу лабораторних робіт, під час виконання якого студенти вивчають структуру персонального комп'ютера, команди цілочислового мікропроцесора та співпроцесора, можливості низькорівневого програмування.

ЗАГАЛЬНІ МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

Виконання лабораторних робіт спрямоване на практичне засвоєння студентами теоретичного матеріалу лекційного курсу "Архітектура обчислювальних систем".

Перш ніж допустити студента до виконання лабораторної роботи, викладач перевіряє його теоретичну підготовку з теми роботи, знання методики проведення лабораторної роботи та наявність оформленого звіту про попередню роботу.

Лабораторну роботу виконують згідно з планом, що наведений в інструкції. Для виконання окремих робіт студенти отримують індивідуальні завдання від викладача.

Звіти про виконання лабораторних робіт потрібно оформлювати у текстовому редакторі Microsoft Word або LibreOffice Writer та зберігати окремими файлами з іменем: <Прізвище студента>_ас<№ лабораторної роботи> у форматі .docx, .odt або .pdf.

У звіті необхідно подати:

1) назву та мету роботи;

2) розширений план виконання роботи;

3) скріншоти з текстом та результатами виконання програм (програма повинна містити лаконічні змістовні коментарі);

4) висновки.

За результатами виконаної лабораторної роботи, які наведені у звіті, відбувається захист лабораторної роботи.

ЛАБОРАТОРНА РОБОТА № 7

Тема: Вступ до мови програмування Assembler

Мета роботи. Використовуючи середовище Visual Studio, реалізувати програму виведення текстового повідомлення на мові Assembler.

Теоретичні відомості:

У програмах на мові Assembler кожна команда займає один рядок. Аргументами команди можуть бути регістри процесора, числа або адреси пам'яті.

В межах цієї лабораторної роботи використовуються 32-розрядні регістри загального призначення EAX, EBX, EDI (у процесорах x86-64 вони є молодшими частинами відповідних 64-розрядних регістрів RAX, RBX та RDI відповідно), команди MOV, PUSH, POP, CALL та оператор OFFSET.

mov <призначення> <джерело>

Регістру або пам'яті (<призначення>) присвоюється безпосереднє значення або значення з пам'яті чи регістра (<джерело>).

push <операнд>

У стек "заштовхується" операнд (пам'яті, на яку вказує вказівник стеку (регістр ESP), присвоюється значення операнда, після чого значення ESP зменшується на відповідний розмір).

pop <операнд>

Зі стеку "виштовхується" елемент (значення ESP збільшується на відповідний розмір) і записується у регістр чи пам'ять (<операнд>).

call <функція>

Викликається функція, аргументи якої беруться із стеку. Виклик функції не очищає стек, аргументи й надалі залишаються там.

offset <вираз>

Повертає відносну адресу — зміщення виразу (в байтах) відносно початку сегменту, у якому цей вираз визначений.

Для зручності виконання код на мові Assembler може бути вставлений у програму, написану на високорівневій мові. Наприклад, у мові C++ для цього використовується асемблерна вставка __asm{}.

Хід роботи:

1. У середовищі Visual Studio створити простий проект на мові C++.
2. Вибравши з контекстного меню проекту пункт *Build Customization ...*, підключити інструмент *Microsoft Macro Assembler*, активізувавши опцію *masm*.
3. Створити асемблерну вставку *_asm {}*. Використовуючи набір інструкцій для роботи з текстовими змінними, реалізувати програму для виведення слів «Hello World»:

```
#include <iostream>
using namespace std;

char FORMAT[] = "%s %s\n";
char HELLO[] = "Hello";
char WORLD[] = "world";

void main() {

     __asm { // початок асемблерного коду

          mov  eax, offset WORLD
          push eax
          mov  eax, offset HELLO
          push eax
          mov  eax, offset FORMAT
          push eax

          mov edi, printf
          call edi

          // Очищення стеку для коректного завершення процедури main
          pop  ebx
          pop  ebx
          pop  ebx

     } // кінець асемблерного коду

     system("pause");}
```

3. Створити та запустити проект за допомогою команди *Debug -> Start Debugging* та спостерігати за результатом роботи програми.

4. Замінити текст «Hello World» на повідомлення про автора програми (наприклад, «Design by ..., 2017») на мові Assembler, врахувавши вимоги до форматування змінних.

5. Перезапустити проект.

6. Прокоментувати код.

7. Оформити звіт.

ЛАБОРАТОРНА РОБОТА № 8

Тема: Обчислення простих виразів на мові Assembler

Мета роботи. Використовуючи середовище Visual Studio, реалізувати програму обчислення математичного виразу на мові Assembler.

Теоретичні відомості:

EFLAGS (RFLAGS) — 32(64)-розрядний регістр ознак. Кожен біт регістра означає певний стан або результат після виконання команди.

cdq

Перетворює подвійне слово на четвірне (розширює значення регістра EAX на EAX:EDX).

add <операнд_1> <операнд_2>

Додає операнди і зберігає результат у першому.

sub <операнд_1> <операнд_2>

Віднімає від першого операнда другий і зберігає результат у першому.

mul <операнд>

Виконує беззнакове множення операнда на регістр EAX (RAX) і зберігає результат в регістрах EDX:EAX (RDX:RAX).

div <операнд>

Виконує беззнакове ділення регістрів EDX:EAX (RDX:RAX) на операнд. Результат ділення зберігається в EAX (RAX), а остача в EDX (RDX).

imul (idiv) <операнд_1> <операнд_2>

Виконує знакове множення (ділення) операндів і записує результат у першому.

test <операнд_1> <операнд_2>

Встановлює біт регістра ознак ZF (zero flag) в 1, якщо результат логічного множення (порівняння) двох операндів дорівнює нулю. Біт ZF у подальшому використовується командами умовного переходу. Часто

використовується для порівняння регістра з 0 (у записі команди обидва операнди дорівнюють цьому регістру).

Мітка повертає адресу пам'яті, за якою вона знаходиться. Використовується для передачі управління (переходу до виконання іншої команди). Способи задання мітки:

```
mitka:                          label mitka
     ...                                ...
```

jmp <мітка>
Виконується безумовний перехід до вказаної мітки.

jz <мітка>
Виконується перехід до вказаної мітки, якщо біт ZF дорівнює 1.

jnz <мітка>
Виконується перехід до вказаної мітки, якщо біт ZF дорівнює 0.

Хід роботи:

1. У середовищі Visual Studio створити простий проект на мові C++.
2. Вибравши з контекстного меню проекту пункт *Build Customization ...*, підключити інструмент *Microsoft Macro Assembler*, активізувавши опцію *masm*.
3. З використанням асемблерної вставки реалізувати програму для з'ясування, чи є рік високосним:

```
#include <iostream>
using namespace std;

void main(){

     int year, leap;
     cin >> year;

     __asm{

          mov ebx, 0
          mov ecx, 400
          mov eax, year
          cdq
```

```
        div ecx
        test edx, edx
        jz is_leap

        mov eax, year
        cdq
        mov ecx, 4
        div ecx
        test edx, edx
        jz maybe_leap
        jmp result

    is_leap:
        mov ebx, 1
        jmp result

    maybe_leap:
        mov eax, year
        cdq
        mov ecx, 100
        div ecx
        test edx, edx
        jz result
        jmp is_leap

    result:
        mov leap, ebx

    }

    if (leap > 0) {cout << "Yes" << endl;}
    else {cout << "No" << endl;}
    system("pause");
}
```

4. Розібратись у структурі програми та прокоментувати код.

5. Реалізувати програму, обравши завдання згідно зі своїм варіантом (див. Додаток).

6. Оформити звіт.

ЛАБОРАТОРНА РОБОТА № 9

Тема: Команди передачі управління та організація циклів в Assembler

Мета роботи. Використовуючи середовище Visual Studio, на прикладі обчислення математичного виразу освоїти програмування з використанням команд переходу та циклів на мові Assembler.

Теоретичні відомості:

loop <мітка>

Команда для організації циклу. В якості лічильника використовується регістр CX. Команда виконує його декремент, а потім перевіряє значення. Якщо воно не дорівнює 0, то виконується перехід до мітки, інакше виконується наступна команда.

cmp <операнд_1> <операнд_2>

Порівнює операнди та відповідним чином змінює регістр ознак, який надалі використовується командою переходу.

Команда	Умова переходу
JG	операнд_1 > операнд_2
JGE	операнд_1 ≥ операнд_2
JL	операнд_1 < операнд_2
JLE	операнд_1 ≤ операнд_2
JE	операнд_1 = операнд_2
JNE	операнд_1 ≠ операнд_2

Хід роботи:

1. У середовищі Visual Studio створити простий проект на мові C++.

2. Вибравши з контекстного меню проекту пункт *Build Customization ...*, підключити інструмент *Microsoft Macro Assembler*, активізувавши опцію *masm*.

3. З використанням асемблерної вставки реалізувати програму для обчислення значення виразу *y* для усіх цілих значень *x* з проміжку [- 2;2]:

$$y=\begin{cases} 2ax+5, & x<1; \\ \dfrac{a-b}{d}, & x=1; \\ \dfrac{a^2-x}{c}, & x>1 \end{cases} \qquad a=-6;\ b=4;\ c=8;\ d=2;\ -2\le x\in Z\le 2.$$

<u>Примітка.</u> Під дробом мається на увазі цілочислове ділення.

```
#include <iostream>
using namespace std;

     int a, b, c, d, x;
     int values[5]; // масив для збереження y(x)

void main(){

     a = -6;   b = 4;    c = 8;    d = 2;

     __asm{

          mov eax, offset values
          push eax       // заштовхуємо у стек адресу початку масиву
          mov cx, 5      // кількість проходів циклу

     start:              // початок циклу
          mov ax, 3      // для обчислення x = 3 - cx
          sub ax, cx     // отримуємо поточне значення x
          cwde           // розширюємо ax до eax
          mov ebx, eax   // записуємо x в ebx
          mov eax, a     // стала a

          cmp ebx, 1     // порівнюємо x з одиницею
          jl less        // якщо x менше 1, переходимо до мітки less
          je equals      // якщо x дорівнює 1, переходимо до equals

                         // якщо x > 1:
          imul eax       // a^2
          sub eax, ebx   // a^2-x
          cdq            // розширюємо a^2-x до четвірного слова
          mov ebx, c     // стала c
          idiv ebx       // (a^2-x)/c
          jmp result     // переходимо до мітки result
```

```
    less:
        mov edx, 2
        imul edx        // a*2
        imul ebx        // a*2*x
        add eax, 5      // a*2*x+5
        jmp result      // переходимо до мітки result

    equals:
        mov ebx, b      // стала b
        sub eax, ebx    // a-b
        cdq             // розширюємо a-b до четвірного слова
        mov ebx, d      // стала d
        idiv ebx        // (a-b)/d
        jmp result      // переходимо до мітки result

    result:
        pop ebx         // виштовхуємо зі стеку адресу масиву
        mov [ebx], eax  // записуємо результат за поточною адресою
        add ebx, 4      // зсуваємо вказівник масиву
        push ebx        // заштовхуємо в стек нову адресу
        loop start      // кінець циклу

    }
    for (int i = 0; i < 5; i++) {
        cout << values[i] << endl;}
    system("pause");
}
```

4. Реалізувати програму обчислення математичного виразу, обравши завдання згідно зі своїм варіантом (див. Додаток).

5. Оформити звіт.

ЛАБОРАТОРНА РОБОТА № 10

Тема: Логічні операції

Мета роботи. Використовуючи середовище Visual Studio, освоїти методи програмування з використанням логічних операцій на мові Assembler.

Теоретичні відомості:

or <операнд_1>, <операнд_2>

Операція бітового логічного АБО. Виконує побітове логічне додавання операндів і зберігає результат в першому.

and <операнд_1>, <операнд_2>

Операція бітового логічного І. Виконує побітове логічне множення операндів і зберігає результат в першому.

not <операнд>

Операція бітового логічного заперечення.

xor <операнд_1>, <операнд_2>

Операція бітового виключного АБО. Результат зберігається в першому операнді. Часто використовується для обнулення регістрів (в якості операндів треба вказати один і той же регістр).

shl (shr) <операнд>, <число_бітів>

Виконує зсув операнда вліво (вправо) на вказане число бітів. У біти, що вивільнилися справа (зліва), записуються нулі.

rol (ror) <операнд>, <число_бітів>

Виконує циклічний зсув операнда вліво (вправо) на вказане число бітів.

inc (dec) <операнд>

Виконує збільшення (зменшення) значення операнда на одиницю

Хід роботи:

1. У середовищі Visual Studio створити простий проект на мові C++.

2. Підключити інструмент *Microsoft Macro Assembler*.

3. З використанням асемблерної вставки реалізувати програму для аналізу стану бітів двійкового масиву (підрахунку в масиві з восьми чотирибітових слів кількості слів з парним числом одиниць в слові):

```
#include <iostream>
using namespace std;

void main(){

     __int32 arr=848843586;    // (0011 0010 1001 1000
                               // 0101 0011 0100 0010)
     __int32 res=0;

     __asm{
          mov cx, 8       // кількість проходів циклу
          mov eax, 1      // маска

     start_outer: // початок зовнішнього циклу (по словах)
          mov bx, 4       // кількість проходів внутрішнього циклу
          xor di, di      // обнулення лічильника одиниць

     start_inner:         // початок внутрішного циклу (по бітах)
          mov edx, arr
          and edx, eax    // накладання маски
          jz go_next      // якщо результат – 0
          inc di          // якщо результат – 1, збільшуємо лічильник

     go_next:
          shl eax, 1      // побітовий зсув вліво на 1 біт
          dec bx          // зменшуємо лічильник проходів на 1
          test bx, bx     // перевіряємо умову завершення циклу
          jnz start_inner// повторюємо цикл, якщо результат не 0
          and di, 1       // перевірка на парність
          jnz end_        // якщо непарне
          inc res         // якщо парне, збільшуємо лічильник

     end_:
          loop start_outer     // закінчення зовнішнього циклу
     }
          cout << res << endl;
     system("pause");
}
```

4. Реалізувати програму аналізу двійкового масиву, обравши завдання згідно зі своїм варіантом (див. Додаток).

5. Оформити звіт.

ЛАБОРАТОРНА РОБОТА № 11

Тема: Масиви з індексуванням, стек, робота з матрицями

Мета роботи. Використовуючи середовище Visual Studio, написати програму опрацювання масивів даних на мові Assembler.

Хід роботи:

1. У середовищі Visual Studio створити простий проект на мові C++.
2. Підключити інструмент *Microsoft Macro Assembler*.
3. Реалізувати з використанням асемблерної вставки програму для обчислення добутку матриць *AB*, де *A* — задана матриця розмірності *n x n*, а елементи матриці *B* задаються формулою $b_{ij} = i + j - 1$:

```
#include <iostream>
using namespace std;

void main()
{
     // ФОРМУВАННЯ МАТРИЦІ
     int n;
     cout << “Введіть n : ”;
     cin >> n;
     int** a = new int*[n];          // матриця A
     int** b = new int*[n];          // матриця B
     int** res = new int*[n];        // результуюча матриця
     cout << “Введіть елементи матриці A : ”;
     for (int i = 0; i < n; i++)
     {
          a[i] = new int[n];
          b[i] = new int[n];
          res[i] = new int[n];
          for (int j = 0; j < n; j++)
          {
               cin >> a[i][j];
               b[i][j] = i + j – 1;
               res[i][j] = 0;
          }
     }

     // ВІЗУАЛІЗАЦІЯ СФОРМОВАНИХ МАТРИЦЬ
     cout << “ A: ” << endl;
     for (i = 0; i < n; i++)
     {
          for (j = 0; j < n; j++)
```

```
     {
          cout << a[i][j] << " ";
     }
     cout << endl;
}
cout << endl;
cout << " B: " << endl;
for (i = 0; i < n; i++)
{
     for (j = 0; j < n; j++)
     {
          cout << b[i][j] << " ";
     }
     cout << endl;
}
cout << endl;

// АСЕМБЛЕРНА ВСТАВКА (алгоритм множення матриць)
// Для зручності цикли записано на C++
for (i = 0; i < n; i++)
{
     for (j = 0; j < n; j++)
     {
          __asm
          {
               mov edx, res   // edx = res
               mov esi, i
               imul esi, 4
               add edx, esi   // edx = &res[i]
               push [edx]
               pop edx        // edx = res[i]
               mov esi, j
               imul esi, 4
               add edx, esi   // edx = &res[i][j]
               push edx
               xor esi, esi
               mov ecx, n
          start:
               mov eax, a
               mov ebx, b
          a_to_eax:
               mov edx, i
               mov eax, [eax + 4 * edx] // eax = a[i]
               mov edx, esi
               mov eax, [eax + 4 * edx] // eax = a[i][esi]
          b_to_ebx:
               mov edx, esi
               mov ebx, [ebx + 4 * edx] // ebx = b[esi]
               mov edx, j
```

```
                    mov ebx, [ebx + 4 * edx] // ebx = b[esi][j]
               end:
                    pop edx
                    imul eax, ebx  // eax = a[i][esi]*b[esi][j]
                    add [edx], eax
                    push edx
                    inc esi
                    loop start
                    pop edx
               }
          }
     }

// ВИВЕДЕННЯ РЕЗУЛЬТАТУ
cout << “ A*B: ” << endl;
     for (i = 0; i < n; i++)
     {
          for (j = 0; j < n; j++)
          {
               cout << res[i][j] << “ ”;
          }
          cout << endl;
     }
     cout << res << endl;
     system("pause");
}
```

4. Реалізувати програму, обравши завдання згідно зі своїм варіантом (див. Додаток).

5. Оформити звіт.

ЛАБОРАТОРНА РОБОТА № 12

Тема: Програмування співпроцесора з використанням команд обчислення трансцендентних функцій для даних дійсного типу

Мета роботи. Ознайомитися з принципами роботи математичного співпроцесора і використати його можливості для обчислення трансцендентних функцій та реалізації розгалужень.

Теоретичні відомості:

Команди порівняння

У центральному процесорі (цілочисловому) команди умовних переходів виконуються відповідно до значень окремих бітів регістра ознак процесора. У арифметичному співпроцесорі існують спеціальні команди порівнянь, за наслідками виконання яких, встановлюються біти кодів умов в регістрі стану. Якщо операнди не вказані явно після команди, то за замовчуванням перший операнд міститься у вершині стеку ST(0), а другий у регістрі ST(1).

FCOM

Команда порівняння операндів. Віднімає вміст операнда від значення у вершині стеку співпроцесора ST(0). Результат віднімання нікуди не записується і покажчик вершини стеку ST не змінюється. Значення бітів кодів умови після виконання команди FCOM x:

C3 = 0, C0 = 0	ST(0) > x
C3 = 0, C0 = 1	ST(0) < x
C3 = 1, C0 = 0	ST(0) = x
C3 = 1, C0 = 1	ST(0) і x непорівнювані

FICOM

Команда цілочисельного порівняння. Аналогічна команді FCOM.

FCOMP/FICOMP

Команди порівняння/цілочисельного порівняння із вилученням операнда зі стеку ST(0) після виконання порівняння.

FCOMPP

Команда порівняння із вилученням обох операндів зі стеку (ST(0) і ST(1)) після виконання порівняння.

FTST

Команда порівняння з нулем. Значення бітів після її виконання:

C3 = 0, C0 = 0	ST(0) > 0
C3 = 0, C0 = 1	ST(0) < 0
C3 = 1, C0 = 0	ST(0) = 0
C3 = 1, C0 = 1	ST(0) і 0 непорівнювані

FXAM

Команда аналізу операнда на тип числа (скінчене, денормалізоване, нуль, нескінченність та ін.) Аналізує вміст ST(0). Після її виконання встановлюються коди умов, згідно з якими можна визначити знак числа (C1 = 0 — додатне; C1 = 1 — від'ємне), його скінченність (C0 = 0) або нескінченність (C0 = 1) і т. д.

Дії над дійсними числами

FABS

Обчислює модуль ST(0).

FCHS

Змінює знак ST(0) на протилежний.

FADD

Додає до ST(0) операнд.

FADD

Додає до ST(0) операнд.

FSUB

Віднімає від ST(0) операнд.

FMUL (FDIV)

Множить (ділить) ST(0) на операнд.

Трансцендентні команди

FPTAN

Обчислює частковий тангенс ST(0), розміщуючи в стеку такі два числа *x* та *y*, що *y/x = tg(ST(0)).* Для сучасних співпроцесорів *x* = 1, а *y* = *tg(ST(0)).* Після виконання команди число *y* розташовується в ST(0), а *x* заноситься у ST(1). Для старих співпроцесорів (Intel 8087, 80287) аргумент команди FPTAN повинен бути нормалізованим і знаходитися в межах 0 ≤ ST(0) ≤ п/4. Для нових співпроцесорів аргумент команди повинен бути поданий в радіанах в межах $\pm 2^{63}$.

FPATAN

Обчислює частковий арктангенс: z = arctg(ST(1)/ST(0)) = arctg(x/y). Перед виконанням команди числа *x* і *y* розміщуються в ST(1) і ST(0) відповідно. Аргументи команди повинні знаходиться в межах: *0 ≤ x < y < +∞* . Результат записується в ST(1), а операнд в ST(0) вилучається зі стеку. Таким чином результат після виконання команди розміщується в ST(0).

FYL2X

Обчислює вираз y*$\log_2$(x), операнди *x* і *y* розміщуються відповідно в ST(0) і ST(1). Операнди вилучаються зі стеку, а результат записується в стек. *x* повинен бути додатним числом. Користуючись результатом виконання цієї команди, можна обчислити:

$\log_2$(x) = FYL2(x);

ln(x) = ln(2)*$\log_2$(x) = FYL2X(ln(2), x) = FYL2X(FLDLN2, x);

lg(x) = lg(2)*$\log_2$ (x) = FYL2X(lg(2), x) = FYL2X(FLDLG2, x).

FYL2XP1

Обчислює вираз y*$\log_2$(x+1), де *x* відповідає ST(0), а *y* — ST(1). Результат записується в ST(0), обидва операнди вилучаються зі стеку і втрачаються. На операнд *x* накладається обмеження: *0 < x < 1 – 1/sqrt(2).*

F2XM1

Обчислює вираз 2x–1, де *x* — ST(0). Результат записується в ST(0), параметр повинен знаходиться в межах *0 ≤ x ≤ 0,5.*

FCOS

Обчислює cos(x). Параметр *x* знаходиться в ST(0), туди ж записується результат виконання команди.

FSIN

Аналогічна команді FCOS, але обчислює значення синуса ST(0).

FSINCOS

Обчислює одночасно значення синуса і косинуса параметра ST(0). Значення синуса записується в ST(1), косинуса — в ST(0).

FSQRT

Обчислює квадратний корінь з ST(0).

Команди керування

Команди керування призначені для роботи з нечисловими регістрами співпроцесора. Мнемоніки цих команд можуть починатися з FN або F. Перший варіант відповідає командам "Без очікування". Для таких команд процесор не перевіряє, чи зайнятий співпроцесор виконанням команди. Особливі випадки також ігноруються. Варіанти команд "З очікуванням" діють так само, як і звичайні команди співпроцесора.

FNSTCW (FSTCW)

Записати управляюче слово (записує вміст регістра керування в оперативну пам'ять).

FLDCW

Завантажити управляюче слово (завантажує регістр керування з оперативної пам'яті. Зазвичай використовується для зміни режиму роботи співпроцесора).

FNSTSW (FSTSW)

Записати слово стану (записує вміст регістра стану в оперативну пам'ять).

FNSTSW AX (FSTSW AX)

Записати слово стану в AX (записує вміст регістра стану в регістр AX центрального процесора, де можливий аналіз вмісту за допомогою команд умовних переходів).

FNCLEX (FCLEX)

Скинути особливі випадки (скидає ознаки особливих випадків в регістрі стану співпроцесора, також скидаються біти ES і B).

FNINIT (FINIT)

Ініціалізувати співпроцесор (регістри стану, керування і тегів):

регістр керування — проектна нескінченність, округлення до найближчого, розширена точність, всі особливі випадки замасковані;

регістр стану — B = 0 (біт зайнятості скинутий), код умови не визначений, ST = ES = 0, ознаки особливих випадків встановлені в нуль;

регістр тегів — усі поля регістра тегів містять значення 11 (порожній регістр).

FNSTENV (FSTENV)

Записати оточення (записує в пам'ять вміст всіх регістрів, окрім числових, у визначеному форматі). Команда корисна при обробці особливих випадків.

FLDENV

Завантажити оточення (завантажує регістри, збережені командою FNSTENV).

FNSAVE (FSAVE)

Записати повний стан (діє аналогічно команді FNSTENV, але додатково зберігає вміст числових регістрів).

FRSTOR

Відновити повний стан (діє аналогічно команді FLDENV, але додатково відновлює вміст числових регістрів).

FINCSTP/FDECSTP

Збільшити/зменшити покажчик стека SP на 1

FFREE

Звільнити регістр (визначає числовий регістр ST, вказаний як операнд, як порожній, записуючи у відповідне поле регістра тегів 11).

FNOP

Порожня команда, немає операції (не робить жодних дій).

FSETPM

Встановлює захищений режим роботи (переводить співпроцесор в захищений режим роботи).

Хід роботи:

1. У середовищі Visual Studio створити простий проект на мові C++.

2. Реалізувати програму, яка обчислює трансцендентний вираз, обравши завдання згідно зі своїм варіантом (див. Додаток). Вхідні дані повинні вводитися з клавіатури під час виконання програми в десятковому форматі зі знаком. Програма повинна складатися з двох модулів:

Головний модуль — створюється мовою C++ і має забезпечити ввід необхідних даних, виклик асемблерної процедури для обчислення виразу та вивід результату обчислень.

Модуль безпосередніх обчислень — здійснює необхідні арифметичні дії з використанням математичного співпроцесора.

3. Оформити звіт.

Приклад. Лістинг програми на мові C++ з використанням асемблерної вставки, яка обчислює значення виразу –sin(x+d)cos(x+d)/4 для трьох значень змінної x:

```
#include <iostream>
#include <cmath>
using namespace std;

const int n = 3;                // кількість значень змінної x
const float neg4 = -4;
float d;
float x[n];                     // значення змінної
float res[n];                   // значення виразу

void calculateASM()
{
     __asm
     {                          // заповнення регістрів
          lea esi, x            // адреса початку масиву x
          lea edi, res          // адреса початку масиву res
          mov ecx, n            // розмір масиву
          finit                 // ініціалізація співпроцесора

          iteration :           // початок циклу

               fld [esi]        // значення x в стек співпроцесора
               fadd d           // x + d
               fsincos          // sin(x+d) і cos(x+d) в стек
               fmul             // sin(x+d)*cos(x+d)
```

```
                fld neg4        // -4 в стек
                fdiv            // sin(x+d)cos(x+d)/(-4)
                fstp [edi]      // запис значення виразу в масив
                // перехід до наступних елементів масивів
                add esi, 4
                add edi, 4

                loop iteration // кінець циклу
        }
}

void enteringData()
{
        cout << " Введіть d : ";
        cin >> d;
        cout << " Введіть x[i] :" << endl;
        for (int i = 0; i < n; ++i)
        {
                cout << " x[" << i << "] = ";
                cin >> x[i];
        }
}

void printRes(char* sym)
{
        cout << " Обчислено в " << sym << endl;
        for (int i = 0; i < n; ++i)
        {
                cout << " result: [" << i + 1 << "] = " << res[i] << endl;
        }
}

void calculateCpp()
{
        for (int i = 0; i < n; i++)
        {
                res[i] = -0.125 * sin(2*(d + x[i]));
        }
}

void main()
{
        enteringData();
        calculateASM();
        printRes("ASM");
        calculateCpp();
        printRes("C++");
        system("pause");
}
```

ДОДАТОК

Варіанти завдань до лабораторної роботи № 8

1. Вивести третю з кінця цифру в записі додатного цілого числа.
2. Обчислити суму цифр трицифрового числа.
3. Йде k-та секунда доби. Визначити, скільки повних годин і хвилин пройшло до цього моменту.
4. Визначити кут (в градусах) між положенням годинної стрілки на початку доби і її положенням в певний момент часу hh:mm:ss.
5. Визначити повну кількість годин і хвилин, які пройшли від початку доби, якщо годинникова стрілка повернулась на f (ціле число) градусів.
6. Вивести порядковий номер дня тижня, на який припадає k-ий день не високосного року, у якому 1 січня — понеділок.
7. Обчислити цілу частину середнього арифметичного трьох заданих додатних цілих чисел.
8. Обчислити довжину кола, площу круга і об'єм кулі заданого радіуса (число пі прийняти рівним 3).
9. Обчислити периметр і площу прямокутного трикутника за заданими двома катетами.
10. Знайти добуток цифр заданого чотиризначного числа.
11. Вивести число, утворене записом цифр заданого трицифрового числа у зворотньому порядку.
12. Визначити дискримінант квадратного рівняння за заданими його коефіцієнтами.
13. Визначити, чи дорівнює сума двох перших цифр заданого чотирицифрового числа сумі двох його останніх цифр.
14. Визначити, чи дорівнює квадрат заданого тризначного числа кубу суми його цифр.
15. Вивести цифри заданого трицифрового числа у порядку зростання.
16. З'ясувати, чи серед цифр заданого трицифрового числа є однакові.
17. Яка сума буде на рахунку через k років, якщо покласти S грн під n річних простих відсотків?

18. З'ясувати, чи є трикутник із заданими сторонами прямокутним.
19. З'ясувати, чи є трикутник із заданими сторонами рівнобедреним.
20. Обчислити добуток цифр трицифрового числа.
21. Скільки одиниць товару вартістю k грн можна придбати на суму n грн? Скільки грошей залишиться?
22. Визначити, чи є введене число парним, чи непарним.
23. Визначити, чи є число n дільником числа k.
24. Обчислити площу поверхні прямокутного паралелепіпеда за заданими сторонами.
25. Скільки літрів води поміщається у кубі, сторона якого задана у дм.

Варіанти завдань до лабораторної роботи № 9

1) $y=y_1+y_2;\ \ y_1=\begin{cases} a+x, & x>a \\ 2a-x, & x\le a \end{cases};\ \ y_2=\begin{cases} ax, & x>10 \\ x, & x\le 10 \end{cases}.$

2) $y=y_1-y_2;\ \ y_1=\begin{cases} x-2, & x\ge 2 \\ 8, & x<2 \end{cases};\ \ y_2=\begin{cases} 4, & x=0 \\ a-x, & x\ne 0 \end{cases}.$

3) $y=y_1\cdot y_2;\ \ y_1=\begin{cases} x-a, & x>a \\ 5, & x\le a \end{cases};\ \ y_2=\begin{cases} a, & a>x \\ ax, & a\le x \end{cases}.$

4) $y=y_1+y_2;\ \ y_1=\begin{cases} 2-x, & x<2 \\ a+3, & x\ge 2 \end{cases};\ \ y_2=\begin{cases} a-1, & x<a \\ ax-1, & x\ge a \end{cases}.$

5) $y=y_1-y_2;\ \ y_1=\begin{cases} |x|, & x<0 \\ x-a, & x\ge 0 \end{cases};\ \ y_2=\begin{cases} a+x, & x \bmod 3=1 \\ 7, & x \bmod 3\ne 1 \end{cases}.$

6) $y=y_1+y_2;\ \ y_1=\begin{cases} x \bmod 4, & x>a \\ a, & x\le a \end{cases};\ \ y_2=\begin{cases} ax, & x/a>3 \\ x, & x/a\le 3 \end{cases}.$

7) $y=y_1+y_2;\ \ y_1=\begin{cases} 4-x, & |x|<3 \\ a+x, & |x|\ge 3 \end{cases};\ \ y_2=\begin{cases} 2, & x-парне \\ a+2, & x-непарне \end{cases}.$

8) $y=y_1+y_2;\ \ y_1=\begin{cases} 4x, & x\le 4 \\ x-a, & x>4 \end{cases};\ \ y_2=\begin{cases} 7, & x-непарне \\ x/2+a, & x-парне \end{cases}.$

9) $y=y_1\cdot y_2;\ \ y_1=\begin{cases} ax, & x \bmod 3=2 \\ 9, & x \bmod 3\neq 2 \end{cases};\ \ y_2=\begin{cases} a-x, & a>x \\ a+2, & a\leq x \end{cases}.$

10) $y=y_1-y_2;\ \ y_1=\begin{cases} a+|x|, & x>a \\ a-7, & x\leq a \end{cases};\ \ y_2=\begin{cases} 3a, & a>3 \\ 11, & a\leq 3 \end{cases}.$

11) $y=y_1 \bmod y_2;\ \ y_1=\begin{cases} 10+x, & x>1 \\ |x|+a, & x\leq 1 \end{cases};\ \ y_2=\begin{cases} 2, & x>4 \\ x, & x\leq 4 \end{cases}.$

12) $y=y_1/y_2;\ \ y_1=\begin{cases} 15+x, & x>7 \\ |a|+9, & x\leq 7 \end{cases};\ \ y_2=\begin{cases} 3, & x>2 \\ |x|-5, & x\leq 2 \end{cases}.$

13) $y=y_1\cdot y_2;\ \ y_1=\begin{cases} 3+x, & x=a \\ a-x, & x\neq a \end{cases};\ \ y_2=\begin{cases} |a|, & a<x \\ |a|-x, & a\geq x \end{cases}.$

14) $y=y_1-y_2;\ \ y_1=\begin{cases} 2x+a, & x>2 \\ 2x+1, & x\leq 2 \end{cases};\ \ y_2=\begin{cases} |x|+1, & x>0 \\ a-1, & x\leq 0 \end{cases}.$

15) $y=y_1 \bmod y_2;\ \ y_1=\begin{cases} 8+|x|, & x<1 \\ 2|a|, & x\geq 1 \end{cases};\ \ y_2=\begin{cases} 3, & x=a \\ a+1, & x\neq a \end{cases}.$

16) $y=y_1+y_2;\ \ y_1=\begin{cases} 4+x, & x\leq 3 \\ ax, & x>3 \end{cases};\ \ y_2=\begin{cases} |a|-2, & x>a \\ x, & x\leq a \end{cases}.$

17) $y=y_1-y_2;\ \ y_1=\begin{cases} a+|x|, & x<0 \\ x-a, & x\geq 0 \end{cases};\ \ y_2=\begin{cases} 7, & x<3 \\ a, & x\geq 3 \end{cases}.$

18) $y=y_1 \bmod y_2;\ \ y_1=\begin{cases} 7+x, & x<3 \\ |a|+x, & x\geq 3 \end{cases};\ \ y_2=\begin{cases} 1, & x>5 \\ a+x, & x\leq 5 \end{cases}.$

19) $y=|y_1|+y_2;\ \ y_1=\begin{cases} -5, & x>4 \\ x-a, & x\leq 4 \end{cases};\ \ y_2=\begin{cases} |a|, & x>a \\ 9, & x\leq a \end{cases}.$

20) $y=y_1\cdot y_2;\ \ y_1=\begin{cases} 2x, & x<5 \\ |a|+x, & x\geq 5 \end{cases};\ \ y_2=\begin{cases} 3, & a=x \\ a-x, & a\neq x \end{cases}.$

21) $y=y_1+|y_2|;\ \ y_1=\begin{cases} 3, & x \bmod 3=1 \\ x-a, & x \bmod 3\neq 1 \end{cases};\ \ y_2=\begin{cases} a/x, & x\neq 0 \\ 4, & x=0 \end{cases}.$

22) $y=y_1-y_2;\ y_1=\begin{cases}|x|+|a|, & x\le 3\\ xa, & x>3\end{cases};\ y_2=\begin{cases}3, & a=x\\ a-x, & a\ne x\end{cases}.$

23) $y=y_1+y_2;\ y_1=\begin{cases}2x, & |x|>4\\ 4+a, & |x|\le 4\end{cases};\ y_2=\begin{cases}9, & x=0\\ a/x, & x\ne 0\end{cases}.$

24) $y=y_1\cdot y_2;\ y_1=\begin{cases}x, & x \bmod 4\ne 2\\ a+x, & x \bmod 4=2\end{cases};\ y_2=\begin{cases}a-x, & x<a\\ ax, & x\ge a\end{cases}.$

25) $y=y_1/y_2;\ y_1=\begin{cases}12, & x<12\\ x+1, & x\ge 12\end{cases};\ y_2=\begin{cases}2, & x>2\\ a+x, & x\le 2\end{cases}.$

Варіанти завдань до лабораторної роботи № 10

1. Задано масив з 9 байт. У скількох його байтах скинуті 6-й і 4-й біти.
2. Задано масив з 8 байт. Розглядаючи його, як масив з 64 біт, порахувати кількість одиниць.
3. Задано масив з 8 байт. Розглядаючи його як масив логічних значень $x_0x_1x_2x_3x_4x_5x_6x_7$ (true — ненульові біти в байті, false — всі біти нульові), обчислити логічну формулу:
   $f=(x_7\wedge x_6\wedge x_1)\vee(x_6\wedge x_4\wedge x_2\wedge x_1\wedge x_0)\vee(x_7\wedge x_6\wedge x_3\wedge x_1).$
4. Задано масив з 10 байт. У скількох його байтах рівно три одиниці?
5. Розглядаючи байт як набір логічних значень $x_7x_6x_5x_4x_3x_2x_1x_0$ (true —1, false — 0), обчислити логічну формулу:
   $f=(x_7\wedge x_6\wedge x_3)\vee(x_6\wedge x_4\wedge x_2\wedge x_1)\vee(x_7\wedge x_6\wedge x_2\wedge x_0).$
6. Задано масив з 8 байт. Розглядаючи його, як масив з 64 біт, порахувати довжину найдовшої послідовності одиниць.
7. Задано масив з 10 байт. Порахувати кількість одиниць у всіх розрядах, кратних трьом: 3, 6, 9, ..., 75, 78.
8. Задано масив з 5 байт. Розглядаючи його як масив з 8 п'ятирозрядних слів, знайти всі "виключні або" всіх 8 слів виразу "10101".
9. Задано масив з 6 байт. Розглядаючи його, як масив з 48 біт, порахувати в ньому кількість нулів.
10. Задано масив з 8 байт. Розглядаючи його, як масив з 64 біт, порахувати кількість пар одиниць в оточенні нулів. Кінець послідовності розглядати як нуль.

11. Задано масив з 7 байт. Розглядаючи його, як масив з 8-ми семибітних слів, порахувати кількість слів з непарним числом нулів.
12. Задано масив з 9 байт. Розглядаючи його як масив з 72 біт, порахувати число переходів між нулями і одиницями.
13. Задано масив з 3 байт. Розглядаючи його, як масив з 24 біт, порахувати кількість одиночних одиниць в оточенні нулів. Кінець послідовності розглядати як нуль.
14. Задано масив з 6 байт. Порахувати кількість байт число одиниць в яких не перевищує 3.
15. Задано масив з 11 байт. Порахувати кількість байт, в яких немає одиниць, що стоять поруч.
16. Задано масив з 4 байт. Розглядаючи його, як масив з 32 біт порахувати довжину найдовшої послідовності нулів.
17. Задано масив з 6 байт. Порахувати кількість одиниць у всіх розрядах, кратних п'яти: 5, 10, ..., 45.
18. Задано масив з 3 байт. Розглядаючи його як масив з 8 трирозрядних слів, знайти "виключне або" всіх 8 слів висловлення "101".
19. Задано масив з 7 байт. Розглядаючи його, як масив з 56 біт, порахувати в ньому кількість нулів, що стоять після одиниці. Кінець послідовності розглядати як нуль.
20. Задано масив з 8 байт. Розглядаючи його, як масив з 64 біт, порахувати кількість пар одиниць в оточенні нулів. Кінець послідовності розглядати як нуль.
21. Задано масив з 6 байт. Порахувати кількість байт, число одиниць в яких не перевищує 3.
22. Задано масив з 6 байт. Розглядаючи його, як масив з 48 біт, порахувати число двох одиниць, що стоять між нулями. Кінець і початок послідовності розглядати як нулі.
23. Задано масив з 3 байт. Розглядаючи його, як масив з 24 біт, порахувати кількість одиночних одиниць в оточенні нулів. Кінець послідовності розглядати як нуль.
24. Задано масив з 5 байт. Розглядаючи його, як масив з восьми п’ятибітних слів, порахувати кількість слів з парним числом одиниць в слові.
25. Задано масив з 11 байт. Порахувати кількість байт, в яких немає одиниць, що стоять поруч.

Варіанти завдань до лабораторної роботи № 11

1. Симетричні квадратні матриці A та B задані послідовністями з $n(n+1)/2$ числами: спочатку іде n елементів 1-го рядка, потім, починаючи з другого елемента, $n-1$ елементів 2-го рядка і т. д., з останнього рядка береться лише n-й елемент. Отримати в аналогічному вигляді матрицю AB.
2. Симетрична квадратна матриця задана послідовністю $n(n+1)/2$ чисел (див. умову задачі 1). Крім цього, заданий вектор $\boldsymbol{b}$ з n елементів. Знайти вектор $A\boldsymbol{b}$.
3. Задані дві праві трикутні матриці A та B порядку n, так як описано в задачі 1. Отримати в аналогічному вигляді матрицю $A\,(E + B^2)$.
4. Задані дві праві трикутні матриці A та B порядку n, так як описано в задачі 1. Отримати в аналогічному вигляді матрицю $B\,(E - A^2)$.
5. Задані дві праві трикутні матриці A та B порядку n, так як описано в задачі 1. Отримати в аналогічному вигляді матрицю AB.
6. Елемент матриці називається сідловою точкою, якщо він є найбільшим в своєму рядку і найменшим у своєму стовпці або, навпаки, є найменшим в своєму рядку і найбільшим в своєму стовпці. Для заданої цілої матриці вивести індекси її сідлових точок.
7. Задані квадратна матриця порядку m та натуральне число n. Порахувати сліди матриць A, A^2, ..., A^n. Під час обчислення матриці A^n для скорочення обчислень скористатись розкладом n за степенями 2 і тоді обчислювати $(A^2)^k$, де $n = 2k$. Якщо ж $n = 2k + 1$, то $A^n = (A^2)^k A$.
8. Задана квадратна матриця A порядку m. Отримати матриці $(A + A^T)/2$ та $(A - A^T)/2$.
9. Задана матриця A розміру $m \times n$. Отримати матрицю AA^T.
10. Задані квадратна матриця A порядку m, натуральне число n, дійсні числа $p_n, p_{n-1}, \ldots, p_0$. Отримати матрицю $p_nA^n + p_{n-1}A^{n-1} + \ldots + p_1A^1 + p_0E$.
11. Задані квадратна матриця A порядку m, натуральне число n. Отримати матрицю $E + A^1 + A^2 + \ldots + A^n$.
12. Нехай задана матриця A порядку m та деяке натуральне число n. Знайти A^n (скористатись розкладом n за степенями 2 і тоді обчислювати $(A^2)^k$, де $n = 2k$. Якщо ж $n = 2k + 1$, то $A^n = (A^2)^k A$).
13. Задані квадратні матриці A та B порядку n. Отримати матрицю $A(B - E) + C$, де E — одинична матриця порядку n, а елементи матриці C обчислюються за формулою: $c_{ij} = 1 / (i + j)$; $i, j = 1, 2, \ldots, n$.

14. Задані квадратні матриці A, B та C порядку n. Отримати матрицю $(A + B)C$.
15. Задана квадратна матриця A порядку n і вектори $\boldsymbol{x}$ та $\boldsymbol{y}$ розміром n. Отримати вектор $A(x^3 + y^2)$.
16. Задана квадратна матриця A порядку n. Отримати матрицю AB, де елементи матриці B обчислюються за формулою: $b_{ij} = 1 / (i + j - 1)$.
17. Задана квадратна матриця A порядку n. Отримати матрицю AB, де елементи матриці B: $b_{ij} = 1 / (i + j - 1)$ та $b_{ij} = 1 / (i + j + 1)$ при $i \le j$ та $i > j$ відповідно.
18. Задана квадратна матриця A порядку n. Отримати матрицю AB, де елементи матриці B: $b_{ij} = 1 / (i + j - 1)$, $b_{ij} = 0$ та $b_{ij} = -1 / (i + j - 1)$ при $i < j$, $i = j$ та $i > j$ відповідно.
19. Задана цілочислова квадратна матриця, всі елементи якої відмінні між собою. Знайти скалярний добуток рядка, в якому знаходиться найбільший елемент, та стовпця, в якому знаходиться найменший елемент.
20. Задана цілочислова квадратна матриця, всі елементи якої відмінні між собою. Знайти скалярний добуток рядка, в якому знаходиться найменший елемент, та стовпця, в якому знаходиться найбільший елемент.
21. Визначити, чи задана цілочислова квадратна матриця є ортонормованою, тобто в якої скалярний добуток кожної пари різних рядків дорівнює 0, а скалярний добуток кожного рядка на себе дорівнює 1.
22. Визначити, чи задана цілочислова квадратна матриця є магічним квадратом, тобто такою, в якої суми елементів у всіх рядках та стовпцях однакові.
23. За заданою матрицею A та вектором $\boldsymbol{b}$ розв'язати методом Гауса матричне рівняння $A\boldsymbol{x} = \boldsymbol{b}$.
24. Задана матриця A розміру m x n. Отримати матрицю A^T.
25. Елемент матриці називається сідловою точкою, якщо він є найбільшим в своєму рядку і найменшим у своєму стовпці або, навпаки, є найменшим в своєму рядку і найбільшим в своєму стовпці. Для заданої цілочислової матриці вивести індекси її сідлових точок.

Варіанти завдань до лабораторної роботи № 12

Примітка. Для усіх варіантів $i = 1, 2, 3, 4, 5$.

Варіант	Завдання	Варіант	Завдання
1	$y_i = \dfrac{2c - d + \sqrt{21 a_i}}{\dfrac{a_i}{4} - 1}.$	2	$y_i = \dfrac{32d - 2c}{tg\left(\dfrac{a_i}{4} - 1\right)},\ c > d.$
3	$y_i = \dfrac{33 - 2c - \sin\dfrac{a_i}{d}}{\dfrac{a_i}{4} - b},\ c \le d.$	4	$y_i = \dfrac{a_i + \dfrac{c}{d} - \sqrt{16d}}{4ab + 1},\ c > d.$
5	$y_i = \dfrac{2c - \ln\dfrac{a_i + d}{c}}{\dfrac{c}{3} - 1},\ c > d.$	6	$y_i = \dfrac{a - 4c - 1}{\dfrac{c}{18} + \mathrm{tg}\left(a_i d\right)},\ c > d.$
7	$y_i = \dfrac{4c - \ln\dfrac{d}{4}}{a_i^2 - 1},\ c > d.$	8	$y_i = \dfrac{\lg\left(19 - a_i\right) \cdot \dfrac{c}{4}}{1 + \dfrac{c}{a_i} + d},\ c > d.$
9	$y_i = \dfrac{5c - \dfrac{d}{21}}{\ln\left(1 - \dfrac{a_i}{4}\right)},\ c > d.$	10	$y_i = \dfrac{2b - 20c}{\dfrac{1}{c}\mathrm{arctg}\left(d + a_i\right) + 1},\ c > d.$
11	$y_i = \dfrac{6c - d\sqrt{\dfrac{12}{d}}}{c + a_i - 1},\ c > d.$	12	$y_i = \dfrac{a_i \cdot \dfrac{c}{4} - 1}{\sqrt{21 - d} - c a_i},\ c > d.$
13	$y_i = \dfrac{\mathrm{arctg}\left(c - \dfrac{d}{2}\right)}{7a_i - 1},\ c > d.$	14	$y_i = \dfrac{\ln\left(a_i d + 2\right)c}{22 - \dfrac{d}{c} + 1},\ c > d.$
15	$y_i = \dfrac{c\,\mathrm{tg}\left(d + 16\right)}{\dfrac{a_i}{2} - 4d - 1},\ c > d.$	16	$y_i = \dfrac{2c + \mathrm{tg}\left(a_i - 23\right)}{\dfrac{c}{a_i} + d + 1},\ c > d.$

17	$y_i=\frac{3c+18\lg d}{d-a_i-1},\ c>d.$	18	$y_i=\frac{24\lg(d-1)-c}{2a_i+\frac{d}{c}},\ c>d.$
19	$y_i=\frac{10c-\frac{d}{2}+1}{a_i^2-\ln(d-5)},\ c>d.$	20	$y_i=\frac{25\cdot\frac{\ln d}{c}+1}{2c+a_i},\ c>d.$
21	$y_i=\frac{\operatorname{arctg}\frac{12}{c}+11}{a_i^2-1},\ c>d.$	22	$y_i=\frac{\operatorname{arctg}(a_i-c)d+26}{\frac{4c}{a_i}+1},c>d.$
23	$y_i=\frac{\sqrt{\frac{24}{a_i}+d-4a_i}}{1+a_ic},\ c>d.$	24	$y_i=\frac{\sqrt{27a_i+\frac{d}{c}}}{a_i-\frac{c}{4}+1},\ c>d.$
25	$y_i=\frac{-\frac{13}{a_i}+c-\operatorname{tg}c}{1+c\cdot\frac{d}{2}},\ c>d.$		

Література

1. Аблязов Р. З. Программирование на ассемблере на платформе x86-64. — М. : ДМК Пресс, 2011. — 304 с.

2. Bartlett J. Programming from the Ground Up. — http://www.freebookcentre.net/ComputerScience-Books-Download/Programming-from-the-Ground-Up-(J.-Bartlett).html

3. Мархивка В. С., Олексів М. В., Акимишин О. І., Мороз І. В. Програмування співпроцесора з використанням команд обчислення трансцендентних функцій та реалізація розгалужень при порівнянні даних дійсного типу: Методичні вказівки до лабораторної роботи № 4 з курсу “Системне програмування” для студентів базового напряму 6.050102 “Комп’ютерна інженерія”. — Львів: Національний університет “Львівська політехніка”, 2011. — 11 с.