## Лабораторна робота №1

**Назва**

Порівняння програми, написаної на мові високого рівня з її компільованим виглядом

**Мета**

Ознайомитися з основам дизасемблінгу та з представленням основних базових арифметичних дій, структур умови та переходу, з представленням локальних та глобальних змінних, циклів, викликів функцій, визначення масиву у мові Assembler в архітектурі ІА-32.

**Теоретичні відомості**

Дивитися пункти 1(Архітектурні особливості процесорів ІА-32), 3 (Команди асемблера MASM 32)

Асемблер, як мова програмування був винайдений як засіб уникнення написання двійкового кодування команд процесора та вказання абсолютних адрес в пам’яті комп’ютера. Однак, це не тільки мова символьного кодування, а також і мова збірки(саме так перекладається англійське слово аssembler), яка дозволяє спеціальній програмі, яка власне і асемблером. перетворити рядки на мові Assembler у виконуваний код для заданого процесора. Загальновідомо, що процесори бувають різної архітектури, операційні системи по різному працюють з прикладними програмами, тому і різновидів асемблерів існує велика кількість. Проте, не зважаючи на це, мають в цілому, а не в деталях певні спільні риси. Це не стосується систем команд, чи способів адресації. В першу чергу це стосується структури самої програми. Проте і там можуть мати місце певні відмінності. Тому важливо спочатку подивитись на це з не якоїсь конкретної реалізації, з певної спільної платформи. Така, на щастя, давно існує і успішно використовується до цього часу – це мова С. Її створив видатний американський програміст Денніс Рітчі ще у 1970 році саме як інтелектуальний асемблер.

Це єдина у світі мова, яка сертифікована за стандартом ISO, тобто код на цій мові має бути однаковим, незалежно від платформи, на якій написана. Стандартизація настільки універсальна, що навіть вихідний модуль має одне і те ж розширення .с, на відміну від усіх інших рівнів компіляції. Значно пізніше на основи мови С була розроблена об’єктно-орієнтована мова С++, яка проте зберегла всі можливості С.

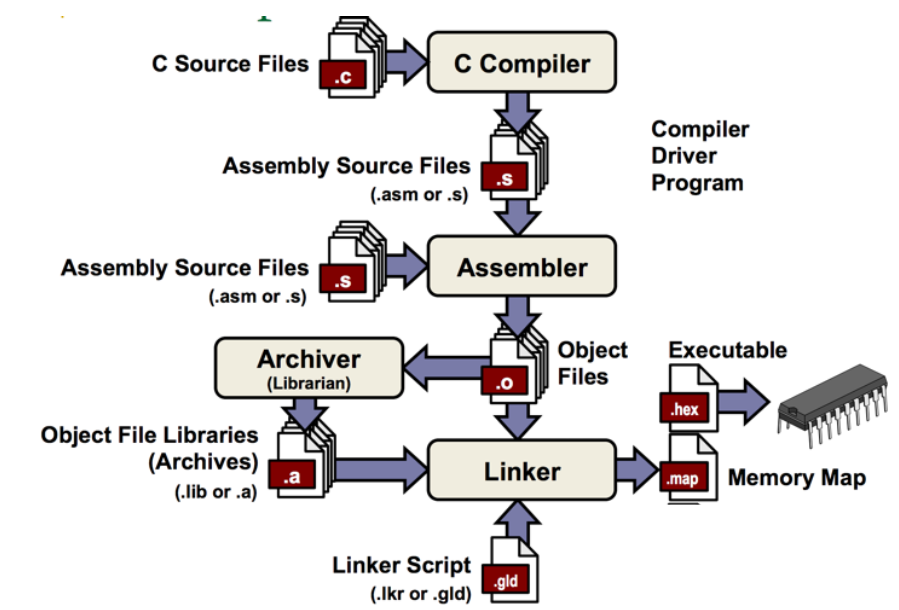


Рис. Асемблер і компілятор С.

На Рис. схематично показано тісний зв’язок між асемблером та компілятором С. Перший етап перетворення коду, написаного на мові високого рівня і є власне компіляцією. І тільки з цього асемблерного коду відбувається перетворення у машинний код. Тому знайомство з мовою С є корисним для наступного переходу на використання одного з варіантів асемблера. Треба спуститися на один поверх абстракції мови нижче. Слід зазначити, що сучасні компілятори С++ налаштовані так, що вони використовують всі конструкції мови С як свої. Проте ряд відмінностей між конструкціями чистого С все ж є. Наприклад, це стосується реалізації вводу/виводу: для ввводу/виводу у чистому С використовуються спеціальні функції, а у - С++ перевантажена операція зсуву.

Основними частинами типової структури програми на С++ є такі:

* директиви препроцесорної обробки;
* опис зовнішніх змінних (вихідних даних і результатів) та функцій;
* функції програми;
* головна функція — програми **main()**, що має вигляд:

main()

{

опис змінних;

виконувані оператори;

}

## Приклад синтаксиса програми на C++

#include <iostream>  
using namespace std;  
  
int main() {  
 cout << "Hello World!"; // виведення рядка операцією зсуву, під’єднаної завдяки препроцесору

return 0;  
}

* #include <iostream> — це бібліотека файлів заголовків, яка дозволяє працювати з об’єктами введення та виведення, такими як cout (використовується в рядку 5). Файли заголовків додають функціональність програмам C++.
* використання простору імен std означає, що ми можливо використовувати імена для об’єктів і змінних зі стандартної бібліотеки.
* іnt main(). Ім‘я, після якого йдуть круглі дужки, називається функцією. Будь-який код у фігурних дужках {} буде виконано.
* cout - це об'єкт, який використовується разом з операцією зсуву (<<) для виведення/друку тексту. У прикладі він виведе "Hello World".
* кожен оператор C++ закінчується крапкою з комою ;
* повернення 0 завершує функцію main.

У загальному випадку програма складається з декількох функцій, які не перетинаються (тобто «вкладення» однієї функції в іншу неприпустиме). Перед функціями і між ними можуть бути присутні оголошення об’єктів даних і оператори препроцесорної обробки. Функції користувача, які викликаються у головній функції main(), слід обов’язково описати до їх використання.

Суттєві особливості програми на C такі:

**Вказівники:** дозволяє посилатися на місце пам’яті за іменем, призначеним йому в програмі.

**Розподіл пам'яті:** під час визначення пам'ять призначається імені змінної, що дозволяє динамічно розподіляти пам'ять. Це означає, що сама програма може попросити операційну систему звільнити пам'ять для використання під час виконання.

**Рекурсія:** коли функція викликає сама себе, це називається рекурсією.

**Маніпуляція бітами:** відноситься до маніпулювання даними в їх найнижчій формі. Він також відомий як біти. У пам’яті комп’ютера інформація зберігається у двійковому форматі (0 і 1).

Тобто, маючи риси мови високого рівня, мова С має всі можливості асемблера, особливо при роботі з пам’яттю. Нижче наведено розділи програми на C:

1. Розділ документації
2. Розділ препроцесора
3. Розділ визначення
4. Глобальна декларація
5. Основна функція
6. Визначені користувачем функції

## Змінні C++

Змінні є контейнерами для зберігання значень даних. У C++ існують різні типи змінних (визначених різними ключовими словами), наприклад:

• int - зберігає цілі числа (цілі числа) без десяткових знаків, наприклад 123 або -123

• double - зберігає числа з плаваючою комою з десятковими дробами, наприклад 19,99 або -19,99

• char - зберігає окремі символи, наприклад 'a' або 'B'. Значення символів оточені одинарними лапками

• string - зберігає текст, наприклад "Hello World". Рядкові значення взяті в подвійні лапки

• bool - зберігає значення з двома станами: true або false тип variableName = значення;

int myNum = 5;  
double myFloatNum = 5.99;  
char myLetter = 'D';  
string myText = "Hello";  
bool myBoolean = true;

Основні типи даних

Тип даних визначає розмір і тип інформації, яку зберігатиме змінна:

|  |  |  |
| --- | --- | --- |
| Тип | Розмір | Визначення |
| boolean | 1 байт | Зберігає значення true або false |
| char | 1 байт | Зберігає один символ/букву/цифру або значення ASCII |
| int | 2 або 4 байта | Зберігає цілі числа без десяткових знаків |
| float | 4 байта | Зберігає дробові числа, що містять один або більше десяткових знаків. Досить для зберігання 7 десяткових цифр |
| double | 8 байтів | Зберігає дробові числа, що містять один або більше десяткових знаків. Достатньо для зберігання 15 десяткових цифр |

## C++ Strings

Окремо розглянемо представлення текстових даних, тобто рядків. Рядки в строгому розумінні не є типом даних, а специфічним масивом символів. Проте рядки використовуються для зберігання тексту. Рядкова змінна містить набір символів у подвійних лапках:

string greeting = "Hello";

## Арифметичні операції

|  |  |  |  |
| --- | --- | --- | --- |
| Операція | Назва | Визначення | Приклад |
| + | Addition | Додає одне значення до іншого | x + y |

|  |  |  |  |
| --- | --- | --- | --- |
| - | Subtraction | Віднімає одне значення від іншого | x - y |
| \* | Multiplication | Перемножує два значення | x \* y |
| / | Division | Розділяє одне значення на інше | x / y |
| % | Modulus | Повертає залишок від ділення | x % y |
| ++ | Increment | Збільшує значення змінної на 1 | ++x |
| -- | Decrement | Зменшує значення змінної на 1 | --x |

**Оператори С /C++**

## Конструкція else if

Оператор else if потрібен, щоб указати нову умову, якщо перша хибна.

### Синтаксис

if (condition1) {  
} else if (condition2) {  
} else {  
}

## C++ Switch

Оператор switch портібен, щоб вибрати один із багатьох блоків коду для виконання.

### Синтаксис

switch(expression) {  
case x:  
 break;  
case y:  
 break;  
default:

}

Як працює:

• Вираз switch обчислюється один раз

• Значення виразу порівнюється зі значеннями кожного випадку

• Якщо є збіг, виконується відповідний блок коду

• Ключові слова break і default є необов'язковими

## C++ For Цикл

### Синтаксис

for (*statement 1*; *statement 2*; *statement 3*) {  
}

### Приклад

for (int i = 0; i < 5; i++) {  
 cout << i << "\n";  
}

Оператор *statement 1* виконується (один раз) перед виконанням блоку коду. *statement 2* визначає умову для виконання блоку коду. *statement 3* виконується (щоразу) після виконання блоку коду. У прикладі вище будуть надруковані числа від 0 до 4

## C++ While Цикл

Цикл while перебирає блок коду, доки виконується задана умова:

### Синтакс

while (condition) {  
  
}

У наведеному нижче прикладі код у циклі виконуватиметься знову і знову, доки значення змінної (i) менше 5:

### Приклад

int i = 0;  
while (i < 5) {  
 cout << i << "\n";  
 i++;  
}

## C++ Масив

Масиви використовуються для зберігання кількох значень в одній змінній замість оголошення окремих змінних для кожного значення. Щоб оголосити масив, визначте тип змінної, вкажіть ім’я масиву, а потім квадратні дужки та вкажіть кількість елементів, які він має зберігати:

string array[4]; - оголосили змінну, яка містить масив із чотирьох рядків.

Для визначення певних значень, скористаємся ініціалізацією, тобто розмістимо значення у списку, розділеному комами, у фігурних дужках:

string array[4] = {"Beatles", "Aerosmith", "ACDC", "Motley Crue"};

Аналогічно ініціалізуємо масив із трьох цілих чисел:

int num[3] = {1, 2, 3};

Отримуєте доступ до елемента масиву, посилаючись на номер індексу в квадратних дужках []. Цей оператор отримує доступ до значення першого елемента в масиві про автомобі:

*string array[4] = {"Beatles", "Aerosmith", "ACDC", "Motley Crue"};  
for (int i = 0; i < 4; i++) {*

*cout << array[i] << "\n";  
}*

**Приклади**

Почнемо з простої програми в стилі чистого С

**Приклад 1:** знайти суму двох чисел, наданих користувачем

*/\* Сума двох чисел \*/ Розділ документації*

*#include<stdio.h> //Розділ препроцесора*

*іnt s; //Глобальне оголошення*

*int main() //Основна функція*

*int a, b, сума;//Локальні визначення*

*printf( "Введіть два числа, які потрібно додати " );// Функція виводу бібліотечної //функції, яка була під’єднана завдяки препроцесору*

*scanf( "%d %d" , &a, &b); // Функція вводу бібліотечної функції, яка була під’єднана завдяки препроцесору*

*s= a + b;// обчислення суми*

*printf( "%d + %d = %d" , a, b, s); // Функція виводу*

*return 0;// повертає ціле значення в сумі*

*}*

**Результат на консолі**



### Приклад програми на С++, в якій використані тільки базові арифметичні операції та порівняння рядкових змінних

Зверніть увагу на розділ препроцесора, де замість *<stdio.h> прописано<iostream>*

*#include<iostream>*

*#include<string>*

*using namespace std;*

*int main()*

*{*

*cout<<"Task1"<<endl;*

*inta,b,c,d;*

*a=2;*

*b=3;*

*cout<<a<"+"<<b<<"="<<a+b<<endl;*

*cout<<a<<"-"<<b<<"="<<a-b<<endl;*

*cout<<a<<"\*"<<b<<"="<<a\*b<<endl;*

*cout<<a<<"/"<<b<<"="<<a/b<<endl;*

*cout<< "\nEnter a number:";*

*cin>>c;*

*cout<<"Enter another number:";*

*cin>>d;*

*cout<<c<<"+"<<d<<"="<<c+d<<endl;*

*cout<<c<<"-"<<d<<"="<<c-d<<endl;*

*cout<<c<<"\*"<<d<<"="<<c\*d<<endl;*

*cout<<c<<"/"<<d<<"="<<c/d<<endl;*

*cout<<"\nTask2"<<endl;*

*int num1=5;*

*int num2=1;*

*if (num1>num2){*

*cout<<"First number \n";*

*}*

*else{*

*cout<<"Second number\n";*

*}*

*string str1="first";*

*string str2="second";*

*cout<<"\nFirst string is :"<<str1<<endl;*

*cout<<"\nFirst string is :"<<str1<<endl;*

*if(str1==str2){*

*cout<<"The two strings are equal.\n";*

*} else {*

*cout << "The two strings are different.\n"; }*

*cout<<"\nTask3"<<endl;*

*int task3=30;*

*switch(task3) {*

*case 17:*

*cout <<"too little \n";*

*break;*

*case 30:*

*cout <<"Got it !\n";*

*break;*

*case54:*

*cout<<"Too much \n";*

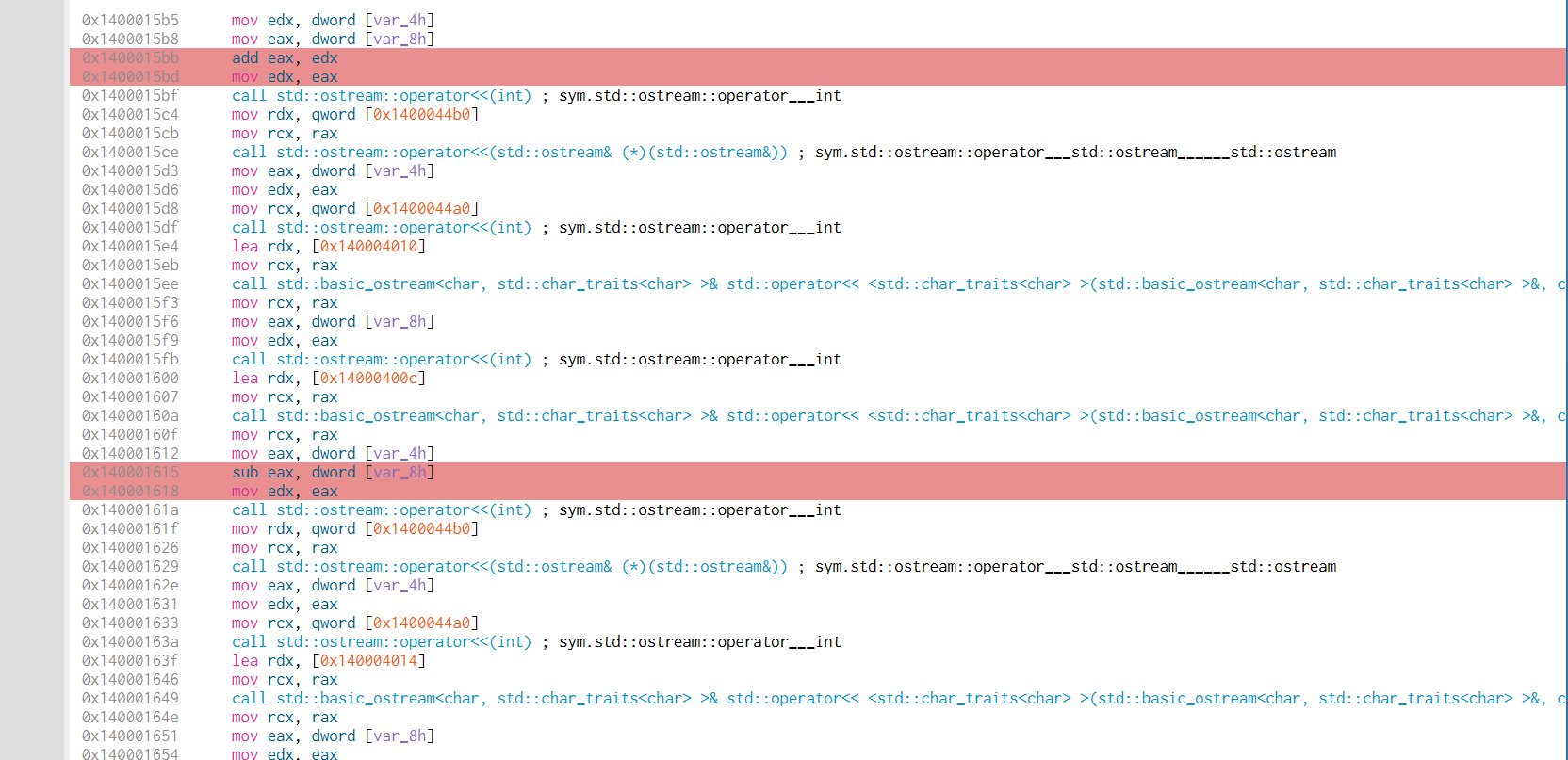
*break;*

}

}

Після компіляції програми — завантажуємо її у програму декомпіляції за вашим вибором (у данному прикладі — Cutter). На Рис. показано асемблерний код декомпільованої програми. Червоним наведено опкоди ассемблера, які відповідають двом базовим арифметичним операція (додаванню та відніманню).

***Приклад скріншоту дизасембльованого коду в програмі Cutter***



опкод add для додавання 2х чисел

опкод sub для віднімання 2х чисел

**Приклад програми на С++, в якій зображено створення масиву, використання циклів, глобальні та локальні змінні**

*#include <iostream>*

*using namespace std;*

*// Глобальні змінні*

*int num = 3;*

*void kingcrimson() {*

*cout<<"It erases time"<<endl;*

}

*class Node{*

*public:*

*int data;*

*Node\*next;*

*};*

*void printList(Node\* n) {*

*while (n!=NULL){*

*cout<<n->data<<"";*

*n=n->next;*

*}*

*}*

*int main () {*

*// Локальні змінні*

*int num= 2;*

*cout<<"\*\*\*Task1\*\*\*\n";*

*cout<<"local:"<<num<<endl;*

*cout<<"global:"<<::num<<endl;*

*cout<<"\n\*\*\*Task2\*\*\*\n";*

*for(inti=0;i<5; i++){*

*for(inti=0;i<5; i++){*

*}*

*cout<<""<<endl;*

*int a=10;*

*while (a<13){*

*cout <<"a:"<<a<<endl;*

*a++;*

*}*

*cout<<"\n\*\*\*Task3\*\*\*\n";*

*kingcrimson();*

*cout<<"\n\*\*\*Task4\*\*\*\n";*

*int arr[5]={1,2,3,4,5};*

*cout << arr[2];*

*cout<<"\n\n\*\*\*Task5\*\*\*\n";*

*Node\*head= NULL;*

*Node\*second=NULL;*

*Node\*third=NULL;*

*head=new Node();*

*second =new Node();*

*third=new Node();*

*head→data = 1;*

*head→next = second;*

*second→data = 2;*

*second→next = third;*

*third→data = 3;*

*third→next =NULL;*

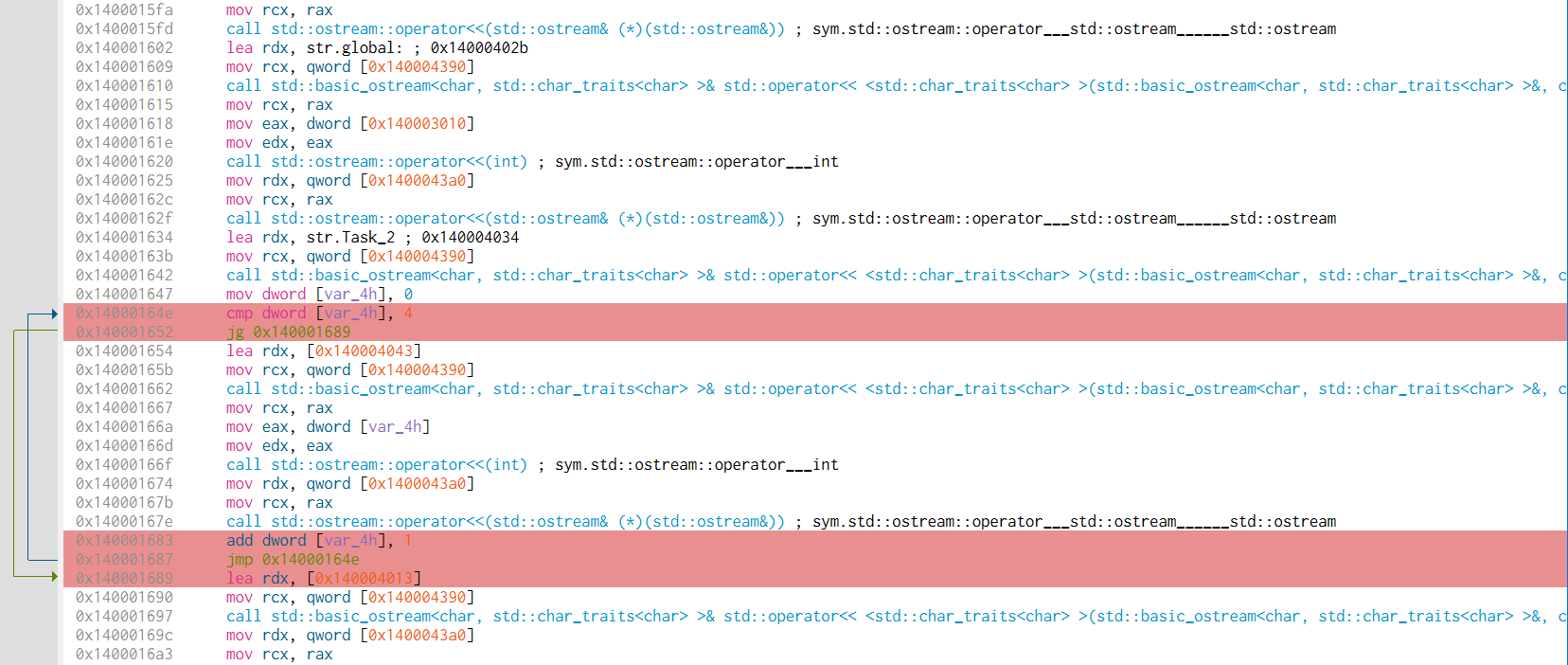
*printList(head);*

*return 0;*

}

Після компіляції програми — завантажуємо її у програму декомпіляції за вашим вибором (у данному прикладі — Cutter). На Рис. червоним кольором виділені опкоди ассемблера, які відповідають порівнянню двох значень та безумовному переходу.

***Приклад скріншоту дизасембльованого коду в програмі Cutter***



Опкод порівняння

Приклад переходу (jmp)

Аналогічно потрібно зробити з усіма пунктами, які вказані у завданні

**Завдання**

1.Розробити C/C++ програму згідно варіанту, яка повинна мати пункти, визначені відповідно до варіанту (наведено нижче). Отримати асемблерний код. Проаналізувати асемблерний код, де вказати ті ділянки, які відповідають пунктам індивідуального завдання

Код асемблерного виду для кожного пункту має містити

1.1Стан регістрів для кожного пункту

1.2. Дерево програми (якщо можливо)

2. Підготувати звіт для захисту

### Варіанти індивідуальних завдань

Варіант 1

1. Базові арифметичні дії (додавання, множення) з хардкоженими значеннями

2. If стейтмент для чисел

3. Switch на 2 пунктів

4. Локальними змінними

5. Циклом for

6. Викликом функції

7. Массивом з такими елементами: 1,2,3,4,5,6,7

Варіант 2

1. Базові арифметичні дії (віднімання, ділення) над елементами масиву

2. if стейтмент рядок

3. Switch на 4 пунктів

4. Глобальними змінними

5. Циклом while

6. Викликом функції, яку ви створили

7. Массивом з такими елементами: 12,13,14,151,6

Варіант 3

1. Базові арифметичні дії (додавання, ділення), значення елементів визначає користувач

2. if стейтмент для чисел

3. Switch на 5 пунктів

4. Локальними змінними

5. Циклом do while

6. Викликом функції

7. Массивом з такими елементами: 5,7,8,9,11,2

Варіант 4

1. Базові арифметичні дії (віднімання, множення), значення елементів визначає користувач

2. if стейтмент рядок

3. Switch на 3 пунктів

4. Глобальними та локальними змінними

5. Циклом for

6. Викликом функції, яку ви створили

7. Массивом з такими елементами: 34,56,32,12

Варіант 5

1. Базові арифметичні дії (додавання, інкремент) з хардкоженими значеннями

2. If стейтмент для чисел

3. Switch на 4 пунктів

4. Глобальними змінними

5. Циклом for

6. Викликом функції

7. Массивом з такими елементами: 5,3,78,12,34

Варіант 6

1. Базові арифметичні дії (віднімання, декремент) , значення елементів визначає користувач

2. if стейтмент рядок

3. Switch на 5 пунктів

4. Локальними змінними

5. Циклом while

6. Викликом функції, яку ви створили

7. Массивом з такими елементами: 100, 101, 103, 104, 105

Варіант 7

1. Базові арифметичні дії (додавання, ділення) над елементами масиву

2. if стейтмент для чисел

3. Switch на декілька пунктів

4. Локальними змінними

5. Циклом do while

6. Викликом функції

7. Массивом з такими елементами: 34, 78, 23, 45

Варіант 8

1. Базові арифметичні дії (віднімання, множення), значення елементів визначає користувач

2. if стейтмент рядок

3. Switch на декілька пунктів

4. Глобальними та локальними змінними

5. Циклом for

6. Викликом функції, яку ви створили

7. Массивом з такими елементами: 1, 34, 56, 7, 12

Варіант 9

1. Базові арифметичні дії (додавання, множення) з хардкоженими значеннями

2. If стейтмент для чисел

3. Switch на декілька пунктів

4. Локальними змінними

5. Циклом for

6. Викликом функції

7. Массивом з такими елементами: 1, 11, 111, 1111

Варіант 10

1. Базові арифметичні дії (віднімання, ділення) над елементами масиву

2. if стейтмент рядок

3. Switch на декілька пунктів

4. Глобальними змінними

5. Циклом while

6. Викликом функції, яку ви створили

7. Массивом з такими елементами: 23, 344, 56, 12

Варіант 11

1. Базові арифметичні дії (додавання, ділення), значення елементів визначає користувач

2. if стейтмент для чисел

3. Switch на декілька пунктів

4. Локальними змінними

5. Циклом do while

6. Викликом функції

7. Массивом з такими елементами: 14, 45, 34, 56

Варіант 12

1. Базові арифметичні дії (віднімання, множення), значення елементів визначає користувач

2. if стейтмент рядок

3. Switch на декілька пунктів

4. Глобальними та локальними змінними

5. Циклом for

6. Викликом функції, яку ви створили

7. Массивом з такими елементами: 23, 89, 56, 0

Варіант 13

1. Базові арифметичні дії (віднімання, ділення) над елементами масиву

2. if стейтмент рядок

3. Switch на декілька пунктів

4. Глобальними та локальними змінними

5. Циклом for

6. Викликом функції, яку ви створили

7. Массивом з такими елементами: 25, 56, 34, 1

Варіант 14

1. Базові арифметичні дії (додавання, ділення), значення елементів визначає користувач

2. if стейтмент рядок

3. Switch на 4 пунктів

4. Глобальними змінними

5. Циклом while

6. Викликом функції, яку ви створили

7. Массивом з такими елементами: 1, 2, 2, 3, 5

Варіант 15

1. Базові арифметичні дії (додавання, ділення), значення елементів визначає користувач

2. if стейтмент для чисел

3. Switch на декілька пунктів

4. Локальними змінними

5. Циклом do while

6. Викликом функції

7. Массивом з такими елементами: 67, 68, 69, 70

Варіант 16

1. Базові арифметичні дії (додавання, віднімання) над елементами масиву

2. if стейтмент рядок

3. Switch на декілька пунктів

4. Глобальними та локальними змінними

5. Циклом for

6. Викликом функції, яку ви створили

7. Массивом з такими елементами: 45, 56, 67, 78

Варіант 17

1. Базові арифметичні дії (додавання, множення) з хардкоженими значеннями

2. If стейтмент для чисел

3. Switch на 2 пунктів

4. Локальними змінними

5. Циклом while

6. Викликом функції

7. Массивом з такими елементами: 23, 34, 45, 56

Варіант 18

1. Базові арифметичні дії (віднімання, множення), значення елементів визначає користувач

2. if стейтмент рядок

3. Switch на декілька пунктів

4. Глобальними та локальними змінними

5. Циклом do while

6. Викликом функції, яку ви створили

7. Массивом з такими елементами: 78, 2, 5, 79

Варіант 19

1. Базові арифметичні дії (віднімання, ділення) над елементами масиву

2. if стейтмент рядок

3. Switch на декілька пунктів

4. Глобальними та локальними змінними

5. Циклом for

6. Викликом функції, яку ви створили

7. Массивом з такими елементами: 9, 8, 7, 6, 5

Варіант 20

1. Базові арифметичні дії (віднімання, додавання) з хардкоженими значеннями

2. if стейтмент рядок

3. Switch на декілька пунктів

4. Глобальними та локальними змінними

5. Циклом while

6. Викликом функції, яку ви створили

7. Массивом з такими елементами: 81, 82, 83, 84, 85

Варіант 21

1. Базові арифметичні дії (віднімання, додавання) з хардкоженими значеннями

2. if стейтмент рядок

3. Switch на декілька пунктів

4. Глобальними та локальними змінними

5. Циклом while

6. Викликом функції, яку ви створили

7. Массивом з такими елементами: 90, 82, 43, 84, 85,93

### Контрольні запитання

1. Які команди ассемблера відповідають за додавання та віднімання

2. Які команди ассемблера відповідають за множення та ділення

3. Як зображаються захардкожені значення змінних у мові ассемблер

4. Як виглядає if структура у мові ассемблер

5. Як працює умовний перехід та відповідні оператори

6. Як зображається структура switch у мові ассемблер

7. В чому різниця між глобальними та локальними зміннами

8. Як різниця між глобальними та локальними змінними відображена у мові ассемблер

9. В чому різниця між циклами