**Индивидуальное задание №2**

**1. Задание**

Необходимо реализовать на языке ассемблера алгоритм, предложенный в варианте. Программа оформляется в виде ассемблерной вставки языка C. Программный код не должен содержать комментариев. Обучающийся должен быть способен пояснить каждую строку программы по просьбе преподавателя. Индивидуальное задание засчитывается, если программный код успешно компилируется, программа выполняется без ошибок и обучающийся отвечает на все вопросы преподавателя по программному коду.

**2. Справочные данные**

В языке C массив определяется как последовательность элементов одинакового типа, в квадратных скобках после названия массива указывается его размер, например, int A[3] – определяет массив с названием A из 3 целочисленных элементов. Многомерные массивы определяются аналогичным образом, для каждой следующей степени размерности необходимо указать в квадратных скобках количество элементов. Например, int B[3][3] – определяет двумерный массив с названием B из 3×3 целочисленных элементов.

Массивы в памяти создаются последовательно, без пропусков ячеек памяти, поэтому можно вычислить адрес любого элемента, зная адрес начала массива и размер одного элемента. В случае одномерного массива адрес вычисляется по следующей формуле: базовый\_адрес + смещение \* размер\_элемента. Для двумерных массивов изменения в формуле незначительны, поскольку элементы массива хранятся в памяти построчно. Формула примет вид: базовый\_адрес + (длина\_строки \* номер\_строки + номер\_столбца) \* размер\_элемента.

lea – инструкция получения базового адреса. Содержит два операнда. В результате выполнения инструкции первый операнд принимает значение адреса второго, второй при этом не меняется. Например, lea ebx, A. В результате ebx = адрес переменной A, второй операнд не изменится.

loop – оператор цикла. Выполняет последовательно операции: 1) уменьшение регистра-счетчика на 1; 2) сравнение регистра-счетчика с 0; 3) если регистр-счетчик не равен 0, то происходит возврат на метку, иначе переход на следующую строку (выход из цикла). В качестве операнда loop требуется указать адрес\_перехода, простейший способ – использовать метку. Метка – символическое имя, ничего не выполняет, служит для использования в командах перехода. Пример организации цикла:

mov eax, 0 // eax = 0

mov ecx, 3 // ecx = 3, это счетчик цикла, цикл выполнится 3 раза

metka:

add eax, 4 // увеличиваем eax на 4

loop metka // вычитаем 1 из ecx, пока ecx не обнулится возврат на metka

// в результате eax = 12

cmp – команда сравнения. Содержит два операнда. Выполняет вычитание из первого операнда второго, но не сохраняет результат, а только изменят состояние арифметических флагов zf, pf, sf. По состоянию этих флагов возможна реализация условных переходов.

neg – изменить знак операнда. Один операнд. Выполняет операнд = 0 – операнд.

Условные переходы. Все условные переходы начинаются с j, затем идет код условия. В качестве операнда указывается адрес\_перехода (обычно - метка). Переходы возможны по соотношению операндов (источником условия является команда сравнения), также возможны переходы по состоянию флагов.

Для чисел со знаком:

|  |  |
| --- | --- |
| **Команда** | **Условие перехода** |
| jg/jnle (больше) | > |
| jl/jnge (меньше) | < |
| jge/jnl (больше или равно) | ≥ |
| jle/jng (меньше или равно) | ≤ |

Для чисел без знака:

|  |  |
| --- | --- |
| **Команда** | **Условие перехода** |
| ja/jnbe (выше) | > |
| jb/jnae (ниже) | < |
| jae/jnb (выше или равно) | ≥ |
| jbe/jna (ниже или равно) | ≤ |

Для всех чисел:

|  |  |
| --- | --- |
| **Команда** | **Условие перехода** |
| je (равно) | = |
| jne (не равно) | ≠ |
| jc (перенос) | cf = 1 |
| js (знак) | sf = 1 |
| jz (ноль) | zf = 1 |
| jp (четное число единичных бит) | pf = 1 |
| jo (переполнение) | of = 1 |
| jnc (нет переноса) | cf = 0 |
| jns (нет знака) | sf = 0 |
| jnz (не ноль) | zf = 0 |
| jnp (нечетное число единичных бит) | pf = 0 |
| jno (нет переполнения) | of = 0 |

**3. Пример выполнения задания**

Рассмотрим один из возможных вариантов задания:

**2.0.** Написать программу, вычисляющую значение выражения: Summ(|Ai|) \* X + Summ(Bij) \* Y где Ai – одномерный массив из 3 целочисленных элементов, Bij – двумерный массив из целочисленных элементов размером 3×3, X, Y – целочисленные переменные. Программа должна запрашивать значение всех указанных переменных и всех элементов массивов с клавиатуры.

Разработаем алгоритм, решающий данную задачу:

1) Вывод на экран сообщения пользователю: «Данная программа вычисляет значение выражения Summ(|Ai|) \* X + Summ(Bij) \* Y».

2) Считывание переменных и элементов массивов с клавиатуры.

3) Вычисление значения выражения на языке С для проверки результата.

4) Открыть ассемблерную вставку.

5) Обнулить регистр-аккумулятор и регистр общего назначения (не использовать регистр-счетчик и регистр-базу).

6) Присвоить регистру-счетчику значение 3.

7) Получить в базовый регистр адрес массива A.

8) Поставить метку, обозначив точку входа в цикл.

9) Поместить значение, расположенное по адресу в базовом регистре, в регистр общего назначения.

10) Сравнить регистр общего назначения с нулем.

11) Перейти на метку (будет определена позже), в случае если значение в регистре общего назначения ≥ 0.

12) Вызвать команду смены знака, передав в качестве операнда регистр общего назначения.

13) Поставить метку, переход на которую предусмотрен в пункте 11.

14) Вызвать команду сложения, передав в качестве операндов регистр-аккумулятор и регистр общего назначения.

15) Вызвать команду сложения, передав в качестве операндов регистр-базу и размер элемента массива.

16) Применить оператор цикла с указанием метки из пункта 8.

17) Вызвать инструкцию умножения, передав в качестве операнда переменную X.

18) Поместить результат умножения в переменную X.

19) Обнулить регистр-аккумулятор.

20) Присвоить регистру-счетчику значение 9.

21) Получить адрес массива B в базовый регистр.

22) Поставить метку, обозначив точку входа в цикл.

23) С помощью команды сложения прибавить значение, расположенное по адресу в базовом регистре, в регистр-аккумулятор.

24) Вызвать команду сложения, передав в качестве операндов регистр-базу и размер элемента массива.

25) Применить оператор цикла с указанием метки из пункта 22.

26) Вызвать инструкцию умножения, передав в качестве операнда переменную Y.

27) Вызвать команду сложения, передав в качестве операндов переменную X и регистр-аккумулятор.

28) Закрыть ассемблерную вставку.

29) Вывести на экран результат вычислений на языке C и результат вычислений на языке ассемблера.

30) Задержка и ожидание нажатия произвольной клавиши для завершения работы программы.

Реализация алгоритма по шагам будет следующей:

1) Вывод на экран сообщения пользователю: «Данная программа вычисляет значение выражения Summ(|Ai|) \* X + Summ(Bij) \* Y».

printf("Данная программа вычисляет значение выражения: Summ(|Ai|) \* X + Summ(Bij) \* Y\n");

2) Считывание переменных и элементов массивов с клавиатуры.

for (int i = 0; i < 3; i++)

{

printf("Введите значение элемента A[%i]:\n", i+1);

scanf\_s("%i", &A[i]);

}

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

printf("Введите значение элемента B[%i][%i]:\n", i + 1, j + 1);

scanf\_s("%i", &B[i][j]);

z = z + B[i][j];//в переменную z копим сумму элементов массива B

}

}

printf("Введите число X:\n");

scanf\_s("%i", &x);

printf("Введите число Y:\n");

scanf\_s("%i", &y);

3) Вычисление значения выражения на языке С для проверки результата.

z = ((abs(A[0])+abs(A[1])+abs(A[2]))\*x) + z \* y;

4) Открыть ассемблерную вставку.

\_asm{

5) Обнулить регистр-аккумулятор и регистр общего назначения (не использовать регистр-счетчик и регистр-базу).

mov eax,0

mov esi,0

6) Присвоить регистру-счетчику значение 3.

mov ecx,3

7) Получить в базовый регистр адрес массива A.

lea ebx, A

8) Поставить метку, обозначив точку входа в цикл.

cycl1:

9) Поместить значение, расположенное по адресу в базовом регистре, в регистр общего назначения.

mov esi, [ebx]

10) Сравнить регистр общего назначения с нулем.

cmp esi,0

11) Перейти на метку (будет определена позже), в случае если значение в регистре общего назначения ≥ 0.

jge m1

12) Вызвать команду смены знака, передав в качестве операнда регистр общего назначения.

neg esi

13) Поставить метку, переход на которую предусмотрен в пункте 11.

m1:

14) Вызвать команду сложения, передав в качестве операндов регистр-аккумулятор и регистр общего назначения.

add eax,esi

15) Вызвать команду сложения, передав в качестве операндов регистр-базу и размер элемента массива.

add ebx,4

16) Применить оператор цикла с указанием метки из пункта 8.

loop cycl1

17) Вызвать инструкцию умножения, передав в качестве операнда переменную X.

imul x

18) Поместить результат умножения в переменную X.

mov x, eax

19) Обнулить регистр-аккумулятор.

mov eax, 0

20) Присвоить регистру-счетчику значение 9.

mov ecx, 9

21) Получить адрес массива B в базовый регистр.

lea ebx, B

22) Поставить метку, обозначив точку входа в цикл.

cycle2:

23) С помощью команды сложения прибавить значение, расположенное по адресу в базовом регистре, в регистр-аккумулятор.

add eax, [ebx]

24) Вызвать команду сложения, передав в качестве операндов регистр-базу и размер элемента массива.

add ebx, 4

25) Применить оператор цикла с указанием метки из пункта 22.

loop cycle2

26) Вызвать инструкцию умножения, передав в качестве операнда переменную Y.

imul y

27) Вызвать команду сложения, передав в качестве операндов переменную X и регистр-аккумулятор.

add x, eax

28) Закрыть ассемблерную вставку.

}

29) Вывести на экран результат вычислений на языке C и результат вычислений на языке ассемблера.

printf("Result on Asm: %i\n", x);

printf("Result on C++: %i\n", z);

30) Задержка и ожидание нажатия произвольной клавиши для завершения работы программы.

system("pause");

В программе используются, но не объявлены переменные x, y, z и массивы A и B. Можно объявить эти переменные и массивы в самом начале программы:

int A[3], B[3][3], x, y, z = 0;

Кроме этого необходимо подключить библиотеку stdio.h для правильной работы функций ввода вывода и библиотеку cstdlib для выполнения функции system():

#include <stdio.h>

#include <cstdlib>

Также необходимо подключить библиотеку locale.h и выполнить функцию setlocale(LC\_ALL, "Russian")

для вывода текста на русском языке.

Окончательный код программы имеет вид:

#include <stdio.h>

#include <locale.h>

#include <cstdlib>

int main()

{

setlocale(LC\_ALL, "Russian");

int A[3], B[3][3], x, y, z = 0;

printf("Данная программа вычисляет значение выражения: Summ(|Ai|) \* X + Summ(Bij) \* Y\n");

for (int i = 0; i < 3; i++)

{

printf("Введите значение элемента A[%i]:\n", i+1);

scanf\_s("%i", &A[i]);

}

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

printf("Введите значение элемента B[%i][%i]:\n", i + 1, j + 1);

scanf\_s("%i", &B[i][j]);

z = z + B[i][j];

}

}

printf("Введите число X:\n");

scanf\_s("%i", &x);

printf("Введите число Y:\n");

scanf\_s("%i", &y);

z = ((abs(A[0])+abs(A[1])+abs(A[2]))\*x) + z \* y;

\_asm

{

mov eax,0

mov esi,0

mov ecx,3

lea ebx, A

cycl1:

mov esi, [ebx]

cmp esi,0

jge m1

neg esi

m1:

add eax,esi

add ebx,4

loop cycl1

imul x

mov x, eax

mov eax, 0

lea ebx, B

mov ecx, 9

cycle2:

add eax, [ebx]

add ebx, 4

loop cycle2

imul y

add x, eax

}

printf("Result on Asm: %i\n", x);

printf("Result on C++: %i\n", z);

system("pause");

return 0;

}

**3. Варианты заданий**

Реализовать вычисление заданного выражения как ассемблерную вставку в программе на языке C. Все вычисления, кроме проверки результата, должны быть выполнены на языке ассемблера. Все переменные (X,Y) являются целочисленными. A, B, C – массивы. Элементы массивов являются целыми числами. Если у массива указан один индекс – массив одномерный размерностью 3 элемента. Если у массива 2 индекса – массив двумерный размерностью 3×3 элемента. Символы | | - означают модуль числа. Summ – означает, что необходимо посчитать сумму элементов массива. Сумма произведения массивов вычисляется как сумма элементов произведения матриц (например, Summ (|Ai| \* |Bi|) = |A1| \* |B1| + |A2| \* |B2| + |A3| \* |B3|).

*Примечание. Модуль суммы и сумма модулей – не одно и то же!*

**2-1.** Summ(Ai \* Xi) + Summ (Bi \* Y)  
**2-2.** Summ (Ai \* X) + Summ (Bi \* X2)  
**2-3.** Summ (Ai \* X + Summ (Bi\*XY) + Summ (Ci)\*Y  
**2-4.** Summ (Ai \* Bi) \* X + Summ (Ci) \* Y  
**2-5.** Summ (|Ai| \* |Bi|) \* X - Summ (|Ci|) \* Y  
**2-6.** Summ (|Ai|) \* X + Summ (Bi) \* Y  
**2-7.** Summ (|Ai|) \* X + Summ (|Bi|) \* X \* Y + Summ (Ci)  
**2-8.** (|Summ (Ai) + Summ (Bi)|) \* X \* Y  
**2-9.** |Summ (Ai + Bi)| \* X + X2

**2-10.** |Summ (Ai) \* X + Summ (Bi) \* Y|

**2-11.** Summ (Aij \* X) + Summ (Bi \* Y)  
**2-12.** Summ (Ai \* X) + Summ (Bij \* X2)  
**2-13.** Summ (Aij \* X) + Summ (Bi\*XY) + Summ (Ci)\*Y  
**2-14.** Summ (Ai \* Bij) \* X + Summ (Ci) \* Y  
**2-15.** Summ (|Ai| \* |Bij|) \* X - Summ (|Ci|) \* Y  
**2-16.** Summ (|Aij|) \* X + Summ (Bi) \* Y  
**2-17.** Summ (|Ai|) \* X + Summ (|Bi|) \* X \* Y + Summ (Cij)  
**2-18.** (|Summ (Ai) + Summ (Bij)|) \* X \* Y  
**2-19.** |Summ (Ai + Bij)| \* X + X2  
**2-20.** |Summ (Ai) \* X + Summ (Bij) \* Y|