**Лабораторная работа №2**

**ОРГАНИЗАЦИЯ УСЛОВНЫХ ПЕРЕХОДОВ**

**Студент гр. ИКПИ-22**

**Нестеренко Н.С.**

**Постановка задачи**

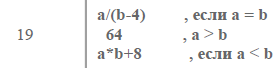
**ЦЕЛЬ РАБОТЫ**:

Вычислить заданное условное целочисленное выражение для данных в форматах INTEGER (int), WORD (unsigned int), используя команды сравнения, условного и безусловного переходов. Результат X тоже целочисленный и его диапазон (формат) зависит от специфики решаемого условного выражения. Исходные данные должны вводиться корректно(с проверкой на область допустимых значений). Результат также должен быть проверен на область допустимых значений. Обмен данными между Си и ASM - модулем должен осуществляться через глобальные переменные, определенные в модуле Си.

**ПОРЯДОК РАБОТЫ**:

* внимательно изучить свой вариант условного арифметического выражения;
* написать на языке Си программу ввода исходных данных, вычисления условного арифметического выражения и вывода полученного результата;
* написать модуль вычисления условного арифметического выражения на языке ASM;
* встроить вызов этого модуля в программу на языке Си;
* произвести тестовые проверки, отметить нормальные и аномальные результаты, сделать анализ результатов.

**ВАРИАНТ ЗАДАНИЯ:**



**Разработка алгоритма**

1. Входные данные:

(int16\_asm)

signed short (word): a\_i, b\_i

(uint16\_asm)

unsigned short (word): a\_ui, b\_ui

Ввод с клавиатуры в окне терминала (реализация на си)

1. Выходные данные:

(int16\_asm)

signed int (double word): res\_i

(uint16\_asm)

signed int (double word): res\_ui

Вывод в окно терминала результатов вычислений на си и ассемблере,

их сравнение

**Таблица идентификаторов**

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **Обозначение в задаче** | **Размер** | **Назначение** |
| **1** | **a\_i** | signed short (word)  16 бит | **Входные данные** |
| **2** | **b\_i** |
| **3** | **a\_ui** | unsigned short (word)  16 бит |
| **4** | **b\_ui** |
| **5** | **res\_i** | signed int (double word)  32 бит | **Выходные данные** |
| **6** | **res\_ui** |

**Контрольный расчет**

Результаты вычислений приведены ниже в таблице вычислений.

|  |  |  |  |
| --- | --- | --- | --- |
| **Назначение набора данных** | **Набор данных**  **(a\_i, b\_i)** | **Результаты вычислений**  **(res\_i)** | |
| **си** | **asm** |
| **Контрольный набор** | -34 56 | -1896 | -1896 |
| -32000 0 | 8 | 8 |
| 444 444 | 1 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Назначение набора данных** | **Набор данных**  **(a\_ui, b\_ui)** | **Результаты вычислений**  **(res\_ui)** | |
| **си** | **asm** |
| **Контрольный набор** | 34 56 | 1912 | 1912 |
| 123 456 | 56096 | 56096 |
| 1 65000 | 65008 | 65008 |

**Программа на языке Си**

#include <stdlib.h>

#include <stdio.h>

#include <stdint.h>

*int16\_t* a\_i, b\_i;

*int* res\_i;

*uint16\_t* a\_ui, b\_ui;

*int* res\_ui;

extern *void* int16\_asm(*void*);

extern *void* uint16\_asm(*void*);

*void* test(*char*);

//вариант 19

//(a == b) : a/(b-4)

//(a > b) : 64

//(a < b) : a\*b+8

*int* main(*void*) {

    printf("choose variation:\n1 - int16\n2 - uint16\n");

*char* c;

    c = getchar();

    if(c == '1') {

        printf("input a, b (-32768 - 32767): ");

        scanf("%hd%hd", &a\_i, &b\_i);

        if(b\_i-4 != 0) {

            if (a\_i == b\_i) res\_i = a\_i/(b\_i-4);

            else if (a\_i > b\_i) res\_i = 64;

            else if(a\_i < b\_i) res\_i = a\_i\*b\_i+8;

            printf("с: %d\n", res\_i);

            res\_i = 0;

            int16\_asm();

            printf("asm: %d\n", res\_i);

        }

        else printf("exeption: division by zero\n");

    }

    else if(c == '2') {

        printf("input a, b (0 - 65535): ");

        scanf("%hu%hu", &a\_ui, &b\_ui);

        if(b\_ui-4 != 0) {

            if (a\_ui == b\_ui) res\_ui = a\_ui/(b\_ui-4);

            else if (a\_ui > b\_ui) res\_ui = 64;

            else if(a\_ui < b\_ui) res\_ui = a\_ui\*b\_ui+8;

            printf("с: %d\n", res\_ui);

            res\_ui = 0;

            uint16\_asm();

            printf("asm: %d\n", res\_ui);

        }

        else printf("exeption: division by zero\n");

    }

    test(c);

    return 0;

};

*void* test(*char* *c*) {

*signed* *int* res\_c;

    if (c == '1') {

        for(a\_i = -2000; a\_i < 2000; ++a\_i) {

            for(b\_i = -2000; b\_i < 2000; ++b\_i) {

                if(b\_i-4 != 0) {

                    if (a\_i == b\_i) res\_c = a\_i/(b\_i-4);

                    else if (a\_i > b\_i) res\_c = 64;

                    else if(a\_i < b\_i) res\_c = a\_i\*b\_i+8;

                    res\_i = 0;

                    int16\_asm();

                    if(res\_i != res\_c) {

                        printf("exeption: (%d, %d)\n", a\_i, b\_i);

                        printf("с: %d\n", res\_c);

                        printf("asm: %d\n", res\_i);

                    }

                }

            }

        }

    }

    else if (c == '2') {

        for(a\_ui = -2000; a\_ui < 2000; ++a\_ui) {

            for(b\_ui = -2000; b\_ui < 2000; ++b\_ui) {

                if(b\_ui-4 != 0) {

                    if (a\_ui == b\_ui) res\_c = a\_ui/(b\_ui-4);

                    else if (a\_ui > b\_ui) res\_c = 64;

                    else if(a\_ui < b\_ui) res\_c = a\_ui\*b\_ui+8;

                    res\_ui = 0;

                    uint16\_asm();

                    if(res\_ui != res\_c) {

                        printf("exeption: (%d, %d)\n", a\_ui, b\_ui);

                        printf("с: %d\n", res\_c);

                        printf("asm: %d\n", res\_ui);

                    }

                }

            }

        }

    }

    printf("test complete!\n");

}

**Программа на языке asm (nasm)**

section .data

extern a\_i

extern b\_i

extern res\_i

extern a\_ui

extern b\_ui

extern res\_ui

section .text

global int16\_asm

global uint16\_asm

; вариант 19

; (a == b) : a/(b-4)

; (a > b) : 64

; (a < b) : a\*b+8

int16\_asm:

xor eax, eax

xor ecx, ecx

xor edx, edx

mov ax, [a\_i]

mov cx, [b\_i]

cmp ax, cx

je @int\_equal

jg @int\_great

jl @int\_less

@int\_equal:

xor eax, eax

xor ecx, ecx

mov ax, [b\_i]

cwde

sbb eax, 4

mov ecx, eax

xor eax, eax

mov ax, [a\_i]

cwde

cdq

idiv ecx

mov [res\_i], eax

jmp @int\_exit

@int\_great:

xor eax, eax

mov eax, 64

mov [res\_i], eax

jmp @int\_exit

@int\_less:

cwde

xor ecx, ecx

mov ecx, eax

mov eax, [b\_i]

cwde

imul ecx

adc eax, 8

mov [res\_i], eax

jmp @int\_exit

@int\_exit:

ret

uint16\_asm:

xor eax, eax

xor ecx, ecx

xor edx, edx

mov ax, [a\_ui]

mov cx, [b\_ui]

cmp ax, cx

je @uint\_equal

ja @uint\_great

jb @uint\_less

@uint\_equal:

xor eax, eax

xor ecx, ecx

mov ax, [b\_ui]

cwd

sbb eax, 4

mov ecx, eax

xor eax, eax

mov ax, [a\_ui]

cwd

cdq

idiv ecx

mov [res\_ui], eax

jmp @uint\_exit

@uint\_great:

xor eax, eax

mov eax, 64

mov [res\_ui], eax

jmp @uint\_exit

@uint\_less:

cwd

xor ecx, ecx

mov ecx, eax

mov eax, [b\_ui]

cwd

imul ecx

adc eax, 8

mov [res\_ui], eax

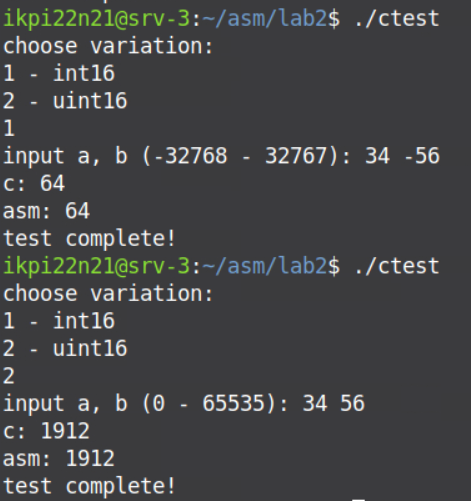
jmp @uint\_exit

@uint\_exit:

ret

**Результаты работы программы**

Далее предоставлены результаты выполнения программы с выбранным для тестирования контрольным набором исходных данных.



**Выводы**

Тот факт, что результаты расчетов на ассемблере, совпадают с результатами расчетов на си, свидетельствуют о том, что программа составлена правильно.