**Лабораторная работа №1**

**ВЫЧИСЛЕНИЕ ЦЕЛОЧИСЛЕННЫХ**

**АРИФМЕТИЧЕСКИХ ВЫРАЖЕНИЙ**

**Студент гр. ИКПИ-22**

**Нестеренко Н.С.**

**Постановка задачи**

Вычислить заданное целочисленное выражение для исходных данных в знаковых и беззнаковых форматах длиной 8 и 16 бит: signed char, unsigned char и signed int , unsigned int , используя арифметические операции ADD, ADC, INC, SUB, SBB , DEC, NEG, MUL, IMUL, DIV, IDIV , CBW, CWD. Исходные значения переменных вводятся пользователем с клавиатуры. Они должны быть максимально приближены к максимально-возможным для тех типов данных, с которыми решается задача. При вводе данных рекомендуется вывести диапазон возможных значений. Размер и тип числителя, знаменателя и результата зависит от заданного выражения.

Выражение:

**Разработка алгоритма**

1. Входные данные:

(uint8\_asm)

unsigned char (byte): a\_uc, b\_uc, d\_uc

(int8\_asm)

signed char (byte): a\_sc, b\_sc, d\_sc

(uint16\_asm)

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

(int16\_asm)

signed short (word): a\_si, b\_si, d\_si

Ввод с клавиатуры в окне терминала (реализация на си)

1. Выходные данные:

(uint8\_asm)

signed short (word): dividend\_uc, res\_uc

unsigned short (word): divider\_uc

(int8\_asm)

signed short (word): dividend\_sc, divider\_sc , res\_sc

(uint16\_asm)

signed int (double word): dividend\_ui, res\_ui

unsigned int (double word): divider\_ui

(int16\_asm)

signed int (double word): dividend\_si, divider\_si , res\_si

Вывод в окно терминала результатов вычислений на си и ассемблере,

их сравнение

**Таблица идентификаторов**

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **Обозначение в задаче** | **Размер** | **Назначение** |
| **1** | a\_uc | Unsigned char (byte)  8 бит | Входные данные |
| **2** | b\_uc |
| **3** | d\_uc |
| **4** | a\_sc | Signed char (byte)  8 бит |
| **5** | b\_sc |
| **6** | d\_sc |
| **7** | a\_ui | Unsigned short (word)  16 бит |
| **8** | b\_ui |
| **9** | d\_ui |
| **10** | a\_si | Signed short (word)  16 бит |
| **11** | b\_si |
| **12** | d\_si |
| **13** | dividend\_uc | Signed short (word)  16 бит | Выходные данные |
| **14** | res\_uc |
| **15** | divider\_uc | Unsigned short (word)  16 бит |
| **16** | dividend\_sc | Signed short (word)  16 бит |
| **17** | res\_sc |
| **18** | divider\_sc |
| **19** | dividend\_ui | Signed int (double word)  32 бит |
| **20** | res\_ui |
| **21** | divider\_ui | Unsigned int (double word)  32 бит |
| **22** | dividend\_si | Signed int (double word)  32 бит |
| **23** | res\_si |
| **24** | divider\_si |

**Контрольный расчет**

Результаты вычислений приведены ниже в таблице вычислений.

|  |  |  |  |
| --- | --- | --- | --- |
| **Назначение набора данных** | **Набор данных**  **(a\_uc, b\_uc, d\_uc)** | **Результаты вычислений**  **(res\_uc, dividend\_uc, divider\_uc)** | |
| **си** | **asm** |
| **Контрольный набор** | 4 5 6 | -1 -6 6 | -1 -6 6 |
| 1 1 5 | -31 -31 1 | -31 -31 1 |
| 0 255 0 | -255 -255 1 | -255 -255 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Назначение набора данных** | **Набор данных**  **(a\_sc, b\_sc, d\_sc)** | **Результаты вычислений**  **(res\_sc, dividend\_sc, divider\_sc)** | |
| **си** | **asm** |
| **Контрольный набор** | -5 -6 -70 | -7 -59 8 | -7 -59 8 |
| 13 -20 125 | -2 146 -64 | -2 146 -64 |
| -17 71 73 | 0 2 -300 | 0 2 -300 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Назначение набора данных** | **Набор данных**  **(a\_ui, b\_ui, d\_ui)** | **Результаты вычислений**  **(res\_ui, dividend\_ui, divider\_ui)** | |
| **си** | **asm** |
| **Контрольный набор** | 4 5 3000 | 498 2988 6 | 498 2988 6 |
| 123 456 789 | 0 333 14023 | 0 333 14023 |
| 1 65000 2 | -3 -64998 16251 | -3 -64998 16251 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Назначение набора данных** | **Набор данных**  **(a\_si, b\_si, d\_si)** | **Результаты вычислений**  **(res\_si, dividend\_si, divider\_si)** | |
| **си** | **asm** |
| **Контрольный набор** | -1 65000 -2 | 3 534 135 | 3 534 135 |
| -123 456 12345 | 0 11889 -14021 | 0 11889 -14021 |
| -56 11211 -3 | 0 -11214 -156953 | 0 -11214 -156953 |

**Программа на языке Си**

#include <stdlib.h>

#include <stdio.h>

#include <stdint.h>

uint8\_t a\_uc, b\_uc, d\_uc;

uint16\_t divider\_uc;

int16\_t dividend\_uc, res\_uc;

int8\_t a\_sc, b\_sc, d\_sc;

int16\_t divider\_sc, dividend\_sc, res\_sc;

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

*unsigned* *int* divider\_ui;

*signed* *int* dividend\_ui, res\_ui;

int16\_t a\_si, b\_si, d\_si;

*signed* *int* divider\_si, dividend\_si, res\_si;

extern *void* uint8\_asm(*void*);

extern *void* int8\_asm(*void*);

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

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

*void* test(*char*);

//var 19: (-35/b+d-b)/(1+a\*b/4)

*int* main(*void*) {

    printf("choose variation:\n1 - uint8\n2 - int8\n3 - uint16\n4 - int16\n");

*char* c;

    c = getchar();

    if(c == '1') {

        printf("input a, b, d (0 - 255): ");

        scanf("%hhu%hhu%hhu", &a\_uc, &b\_uc, &d\_uc);

        if(1+a\_uc\*b\_uc/4 != 0 && b\_uc!=0) {

            uint8\_asm();

            printf("с: %hd / %hu = %hd\n",

                (-35/b\_uc+d\_uc-b\_uc), (1+a\_uc\*b\_uc/4), (-35/b\_uc+d\_uc-b\_uc)/(1+a\_uc\*b\_uc/4));

            printf("asm: %hd / %hu = %hd\n",

                dividend\_uc, divider\_uc, res\_uc);

        }

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

    }

    if(c == '2') {

        printf("input a, b, d (-128 - 127): ");

        scanf("%hhd%hhd%hhd", &a\_sc, &b\_sc, &d\_sc);

        if(1+a\_sc\*b\_sc/4 != 0 && b\_sc!=0) {

            int8\_asm();

            printf("с: %hd / %hd = %hd\n",

                (-35/b\_sc+d\_sc-b\_sc), (1+a\_sc\*b\_sc/4), (-35/b\_sc+d\_sc-b\_sc)/(1+a\_sc\*b\_sc/4));

            printf("asm: %hd / %hd = %hd\n",

                dividend\_sc, divider\_sc, res\_sc);

        }

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

    }

    if(c == '3') {

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

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

        if(1+a\_ui\*b\_ui/4 != 0 && b\_ui!=0) {

            printf("с: %d / %d = %d\n",

                (-35/b\_ui+d\_ui-b\_ui), (1+a\_ui\*b\_ui/4), (-35/b\_ui+d\_ui-b\_ui)/(1+a\_ui\*b\_ui/4));

            uint16\_asm();

            printf("asm: %d / %d = %d\n",

                dividend\_ui, divider\_ui, res\_ui);

        }

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

    }

    if(c == '4') {

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

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

        if(1+a\_si\*b\_si/4 != 0 && b\_si!=0) {

            printf("с: %d / %d = %d\n",

                (-35/b\_si+d\_si-b\_si), (1+a\_si\*b\_si/4), (-35/b\_si+d\_si-b\_si)/(1+a\_si\*b\_si/4));

            int16\_asm();

            printf("asm: %d / %d = %d\n",

                dividend\_si, divider\_si, res\_si);

        }

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

    }

    test(c);

    return 0;

};

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

    if (*c* == '1') {

        for(a\_uc = 0; a\_uc < 255; ++a\_uc) {

            for(b\_uc = 0; b\_uc < 255; ++b\_uc) {

                for(d\_uc = 0; d\_uc < 255; ++d\_uc) {

                    if(1+a\_uc\*b\_uc/4 != 0 && b\_uc!=0) {

                        uint8\_asm();

                        if(res\_uc != (-35/b\_uc+d\_uc-b\_uc)/(1+a\_uc\*b\_uc/4)) {

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

                            printf("с: %hd\n", (-35/b\_uc+d\_uc-b\_uc)/(1+a\_uc\*b\_uc/4));

                            printf("asm: %hd\n", res\_uc);

                        }

                    }

                }

            }

        }

    }

    else if (*c* == '2') {

        for(a\_sc = -127; a\_sc < 127; ++a\_sc) {

            for(b\_sc = -127; b\_sc < 127; ++b\_sc) {

                for(d\_sc = -127; d\_sc < 127; ++d\_sc) {

                    if(1+a\_sc\*b\_sc/4 != 0 && b\_sc!=0) {

                        int8\_asm();

                        if(res\_sc != (-35/b\_sc+d\_sc-b\_sc)/(1+a\_sc\*b\_sc/4)) {

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

                            printf("с: %hd\n", (-35/b\_sc+d\_sc-b\_sc)/(1+a\_sc\*b\_sc/4));

                            printf("asm: %hd\n", res\_sc);

                        }

                    }

                }

            }

        }

    }

    else if (*c* == '3') {

        for(a\_ui = 65000; a\_ui < 65535; ++a\_ui) {

            for(b\_ui = 65000; b\_ui < 65535; ++b\_ui) {

                for(d\_ui = 65000; d\_ui < 65535; ++d\_ui) {

                    if(1+a\_ui\*b\_ui/4 != 0 && b\_ui!=0) {

                        uint16\_asm();

                        if(res\_ui != (-35/b\_ui+d\_ui-b\_ui)/(1+a\_ui\*b\_ui/4)) {

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

                            printf("с: %d\n", (-35/b\_ui+d\_ui-b\_ui)/(1+a\_ui\*b\_ui/4));

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

                        }

                    }

                }

            }

        }

    }

    else if (*c* == '4') {

        for(a\_si = -32768; a\_si < -32200; ++a\_si) {

            for(b\_si = -32768; b\_si < -32200; ++b\_si) {

                for(d\_si = -32768; d\_si < -32200; ++d\_si) {

                    if(1+a\_si\*b\_si/4 != 0 && b\_si!=0) {

                        int16\_asm();

                        if(res\_si != (-35/b\_si+d\_si-b\_si)/(1+a\_si\*b\_si/4)) {

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

                            printf("с: %d\n", (-35/b\_si+d\_si-b\_si)/(1+a\_si\*b\_si/4));

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

                        }

                    }

                }

            }

        }

    }

    printf("test complete!\n");

}

**Программа на языке asm (nasm)**

section .data

extern a\_sc

extern b\_sc

extern d\_sc

extern dividend\_sc

extern divider\_sc

extern res\_sc

extern a\_uc

extern b\_uc

extern d\_uc

extern dividend\_uc

extern divider\_uc

extern res\_uc

extern a\_si

extern b\_si

extern d\_si

extern dividend\_si

extern divider\_si

extern res\_si

extern a\_ui

extern b\_ui

extern d\_ui

extern dividend\_ui

extern divider\_ui

extern res\_ui

section .text

global int8\_asm

global uint8\_asm

global int16\_asm

global uint16\_asm

;var 19: (-35/b+d-b)/(1+a\*b/4)

int8\_asm:

xor ax, ax

xor cx, cx

;---dividend---

;-35/b+d-b

mov ax, [b\_sc]

cbw

mov cx, ax

mov ax, -35

cwd

idiv cx

mov cx, ax

xor ax, ax

mov ax, [d\_sc]

cbw

adc ax, cx

mov cx, ax

xor ax, ax

mov ax, [b\_sc]

cbw

sbb cx, ax

mov [dividend\_sc], cx

;---divider---

;1+a\*b/4

xor ax, ax

mov ax, [a\_sc]

xor ah, ah

mov cx, [b\_sc]

xor ch, ch

imul cl

mov cx, 4

xor ch, ch

cwd

idiv cx

mov cx, 1

xor ch, ch

adc ax, cx

cwd

mov [divider\_sc], ax

;---dividend/divider---

mov ax, [dividend\_sc]

mov cx, [divider\_sc]

cwd

idiv cx

mov [res\_sc], ax

ret

uint8\_asm:

xor ax, ax

xor cx, cx

;---dividend---

;-35/b+d-b

mov ax, -35

mov cx, [b\_uc]

xor ch, ch

cwd

idiv cx

mov cx, [d\_uc]

xor ch, ch

add ax, cx

mov cx, [b\_uc]

xor ch, ch

sub ax, cx

mov [dividend\_uc], ax

;---divider---

;1+a\*b/4

mov ax, [a\_uc]

xor ah, ah

mov cx, [b\_uc]

xor ch, ch

mul cx

mov cx, 4

div cx

mov cx, 1

add ax, cx

mov [divider\_uc], ax

;---dividend/divider---

mov ax, [dividend\_uc]

mov cx, [divider\_uc]

xor dx, dx

cwd

idiv cx

mov [res\_uc], ax

ret

int16\_asm:

xor eax, eax

xor ecx, ecx

;---dividend---

;-35/b+d-b

mov ax, [b\_si]

cwde

mov ecx, eax

xor eax, eax

mov eax, -35

cdq

idiv ecx

xor ecx, ecx

mov ecx, eax

xor eax, eax

mov ax, [d\_si]

cwde

adc eax, ecx

xor ecx, ecx

mov ecx, eax

xor eax, eax

mov ax, [b\_si]

cwde

sbb ecx, eax

mov [dividend\_si], ecx

;---divider---

;1+a\*b/4

xor eax, eax

xor ecx, ecx

mov ax, [a\_si]

cwde

mov ecx, eax

xor eax, eax

mov ax, [b\_si]

cwde

imul ecx

mov ecx, 4

cdq

idiv ecx

inc eax

xor ecx, ecx

mov [divider\_si], eax

;---dividend/divider---

xor eax, eax

xor ecx, ecx

xor edx, edx

mov eax, [dividend\_si]

mov ecx, [divider\_si]

cdq

idiv ecx

mov [res\_si], eax

ret

uint16\_asm:

xor eax, eax

xor ecx, ecx

;---dividend---

;-35/b+d-b

mov ax, [b\_ui]

cwd

mov ecx, eax

xor eax, eax

mov eax, -35

cdq

idiv ecx

xor ecx, ecx

mov ecx, eax

xor eax, eax

mov ax, [d\_ui]

cwd

adc eax, ecx

xor ecx, ecx

mov ecx, eax

xor eax, eax

mov ax, [b\_ui]

cwd

sbb ecx, eax

mov [dividend\_ui], ecx

;---divider---

;1+a\*b/4

xor eax, eax

xor ecx, ecx

mov ax, [a\_ui]

cwd

mov cx, [b\_ui]

mul ecx

mov ecx, 4

div ecx

inc eax

xor ecx, ecx

cwd

mov ecx, eax

mov [divider\_ui], eax

;---dividend/divider---

xor eax, eax

xor ecx, ecx

xor edx, edx

mov eax, [dividend\_ui]

mov ecx, [divider\_ui]

cdq

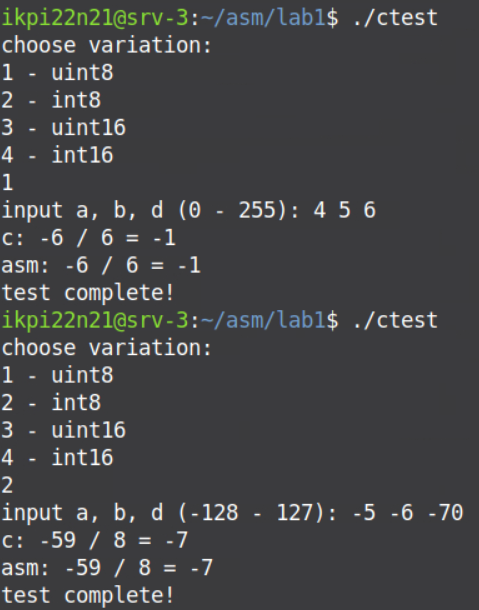
idiv ecx

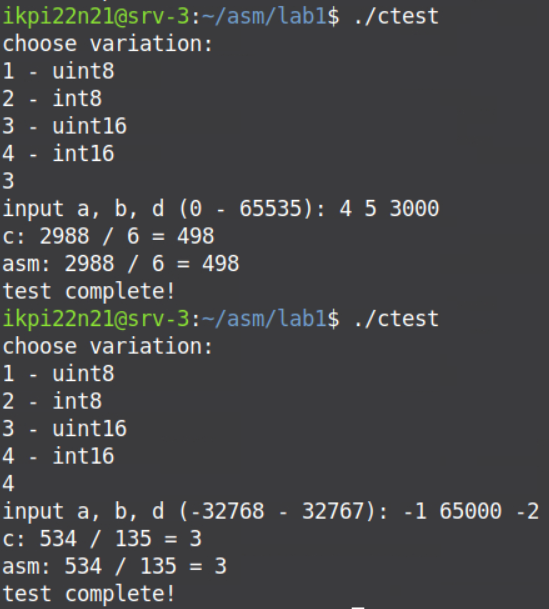
mov [res\_ui], eax

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.