Федеральное государственное бюджетное образовательное учреждение высшего образования «Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича»

Кафедра «Программная инженерия и вычислительная техника»

«Машино-зависимые языки программирования»

Отчет

по лабораторной работе №1

«Вычисление целочисленных арифметических выражений»

Выполнил

студент группы ИКПИ-33

А.Р.Коломиец

Проверил

Ст. преподаватель

И.Л. Савельев

2024 г.

1. Задание

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.

Вариант №7

$$(b*c-8/a)/(31+b-1)$$

2. Текст программы

2.1. Модуль main.c

```
#include <stdio.h>

// Переменные для signed char
signed char a_char, b_char, c_char;
int numerator_char, denominator_char, result_char;

// Variables for unsigned short (16-bit)
unsigned short a_uint, b_uint, c_uint;
int numerator_uint, denominator_uint, result_uint;
// Прототипы функций
void compute_signed_char();
void compute_unsigned_int();

int main() {
    int choice;

    // Ввод выбора типа переменных
    printf("Выберите тип переменных:\n");
    printf("0 - signed char (8 бит)\n");
    printf("1 - unsigned int (16 бит)\n");
    printf("Ваш выбор: ");
```

```
        scanf("%d", &choice);

        while (choice != 1 && choice != 0) {
            printf("Неверный ввод! Повторите снова\n");
            scanf("%d", &choice);
        }

        if (choice == 0) {
            // Работа с signed char (8 бит)
            printf("Введите значения для знаковых 8-битных переменных
(от -128 до 127):\n");
            printf("Введите a: ");
            scanf("%hhd", &a_char);
            printf("Введите b: ");
            scanf("%hhd", &b_char);
            printf("Введите c: ");
            scanf("%hhd", &c_char);

            while (a_char == 0) {
                printf("Ошибка: a не может быть равно 0. Введите a
снова: ");
                scanf("%hhd", &a_char);
            }

            // Вычисления на C
            numerator_char = (b_char * c_char - 8 / a_char);
            denominator_char = (31 + b_char - 1);

            if (denominator_char == 0) {
                printf("Ошибка: Знаменатель равен 0. Невозможно
выполнить деление.\n");
                return 1;
            }

            result_char = numerator_char / denominator_char;

            printf("\nРезультаты для signed char (C):\n");
            printf("Числитель: %d, Знаменатель: %d, Результат: %d\n",
numerator_char, denominator_char, result_char);

            // Вызов ассемблерной функции
            compute_signed_char();

            // Вывод результатов ассемблерной функции
            printf("\nРезультаты для signed char (ASM):\n");
```

```
            printf("Числитель: %d, Знаменатель: %d, Результат: %d\n",
numerator_char, denominator_char, result_char);

        } else if (choice == 1) {
            // Работа с unsigned int (16 бит)
            printf("Введите значения для беззнаковых 16-битных
переменных (от 0 до 65535):\n");
            printf("Введите a: ");
            scanf("%hu", &a_uint);
            printf("Введите b: ");
            scanf("%hu", &b_uint);
            printf("Введите c: ");
            scanf("%hu", &c_uint);

            while (a_uint == 0) {
                printf("Ошибка: a не может быть равно 0. Введите a
снова: ");
                scanf("%hu", &a_uint);
            }

            // Вычисления на C
            numerator_uint = (b_uint * c_uint - 8 / a_uint);
            denominator_uint = (31 + b_uint - 1);

            if (denominator_uint == 0) {
                printf("Ошибка: Знаменатель равен 0. Невозможно
выполнить деление.\n");
                return 1;
            }

            result_uint = numerator_uint / denominator_uint;

            printf("\nРезультаты для unsigned int (C):\n");
            printf("Числитель: %d, Знаменатель: %u, Результат: %d\n",
numerator_uint, denominator_uint, result_uint);

            // Вызов ассемблерной функции
            compute_unsigned_int();

            // Вывод результатов ассемблерной функции
            printf("\nРезультаты для unsigned int (ASM):\n");
            printf("Числитель: %d, Знаменатель: %u, Результат: %d\n",
numerator_uint, denominator_uint, result_uint);
        }

        return 0;
```

```
}
```

2.2. Модуль asm_signed.asm

```
; Use default RIP-relative addressing
default rel

section .data
    extern a_char          ; Extern declarations for variables
    extern b_char
    extern c_char
    extern numerator_char
    extern denominator_char
    extern result_char

section .text
    global compute_signed_char

compute_signed_char:
    ; Load b_char into ebx and c_char into ecx with sign
extension
    movsx ebx, byte [b_char]
    movsx ecx, byte [c_char]

    ; Multiply b_char * c_char
    imul ebx, ecx           ; ebx = b_char * c_char

    ; Prepare to compute 8 / a_char
    mov eax, 8              ; eax = 8 (dividend)
    cdq                     ; edx:eax = sign-extended dividend
    movsx ecx, byte [a_char]; ecx = a_char (divisor)

    ; Check for division by zero
    cmp ecx, 0
    je division_by_zero_error

    ; Perform signed division
    idiv ecx                ; eax = 8 / a_char

    ; Subtract the result of division from the multiplication
result
    sub ebx, eax            ; ebx = (b_char * c_char) - (8 /
a_char)

    ; Store the numerator
    mov [numerator_char], ebx
```

```
        ; Compute the denominator: 31 + b_char - 1
        movsx eax, byte [b_char]
        add eax, 30             ; eax = 31 + b_char - 1

        ; Store the denominator
        mov [denominator_char], eax

        ; Check for division by zero
        cmp eax, 0
        je division_by_zero_error

        ; Prepare to divide numerator by denominator
        mov eax, [numerator_char]
        cdq                     ; edx:eax = sign-extended dividend

        ; Perform signed division
        idiv dword [denominator_char]; eax = numerator_char /
denominator_char

        ; Store the result
        mov [result_char], eax

        ret

    division_by_zero_error:
        ; Handle division by zero error
        mov eax, 0
        mov [result_char], eax
        ret
```

2.3. Модуль asm_unsigned.asm

```
    section .data
        extern a_uint
        extern b_uint
        extern c_uint
        extern numerator_uint
        extern denominator_uint
        extern result_uint

    section .text
        global compute_unsigned_int

    compute_unsigned_int:
        ; Загружаем b_uint в eax с нулевым расширением до 32 бит
        movzx eax, word [b_uint]     ; eax = b_uint
        ; Загружаем c_uint в ecx с нулевым расширением до 32 бит
```

```
        movzx ecx, word [c_uint]     ; ecx = c_uint

        ; Беззнаковое умножение b_uint * c_uint
        mul ecx                      ; edx:eax = eax * ecx

        ; Сохраняем результат умножения в ebx
        mov ebx, eax                 ; ebx = результат умножения

        ; Загружаем a_uint в ecx с нулевым расширением до 32 бит
        movzx ecx, word [a_uint]     ; ecx = a_uint

        ; Подготовка к делению 8 / a_uint
        mov eax, 8                   ; eax = 8
        xor edx, edx                 ; edx = 0 (очищаем старшие 32
бита)

        ; Проверяем деление на ноль
        cmp ecx, 0
        je division_by_zero_error

        ; Выполняем **беззнаковое деление**: 8 / a_uint
        div ecx                      ; eax = 8 / a_uint

        ; Вычитаем результат деления из результата умножения
        sub ebx, eax                 ; ebx = (b_uint * c_uint) - (8 /
a_uint)

        ; Сохраняем числитель
        mov [numerator_uint], ebx

        ; Вычисляем знаменатель: 31 + b_uint - 1
        movzx eax, word [b_uint]
        add eax, 30                  ; eax = 31 + b_uint - 1

        ; Сохраняем знаменатель
        mov [denominator_uint], eax

        ; Проверяем деление на ноль
        cmp eax, 0
        je division_by_zero_error

        ; Загружаем числитель в eax для деления
        mov eax, [numerator_uint]    ; eax = числитель

        ; ПОДГОТОВКА К ЗНАКОВОМУ ДЕЛЕНИЮ
        ; Расширяем знак числителя из eax в edx
```

```
        cdq                             ; edx:eax = знаково расширенный
eax

        ; Выполняем ЗНАКОВОЕ деление: числитель / знаменатель
        idiv dword [denominator_uint] ; eax = числитель / знаменатель

        ; Сохраняем результат
        mov [result_uint], eax

        ret

    division_by_zero_error:
        ; Обработка деления на ноль
        mov eax, 0
        mov [result_uint], eax
        ret
```

2.4 Модуль Makefile

```
    all:
         gcc -m64 -c -g -o main.o main.c  # Добавляем флаг -m64 для
компиляции main.c
         nasm -f elf64 asm_signed.asm -o asm_signed.o
         nasm -f elf64 asm_unsigned.asm -o asm_unsigned.o
         gcc -m64 -no-pie -o program main.o asm_signed.o
asm_unsigned.o  # Флаг -m64 и -no-pie для линковки

    clean:
         rm -f *.o program
```

3. Сборка проекта

```
make
```

4. Выполнение программы

4.1. Запуск программы

```
./program
```

4.2. Входные данные

Выберите тип переменных:

0 - signed char

1 - unsigned int

Ваш выбор: 1

Введите значения для беззнаковых 16-битных переменных (от 1 до 65535):

Введите a: 12

Введите b: 23

Введите c: 34

4.3. Ожидаемый результат выполнения

Корректно подсчитанный результат выполнения уравнения на языках программирования C и ASM. Для данного набора переменных - Числитель: 782, Знаменатель: 53, Результат: 14

4.4. Результат выполнения

```
Выберите тип переменных:
0 - signed char (8 бит)
1 - unsigned int (16 бит)
Ваш выбор: 1
Введите значения для беззнаковых 16-битных переменных (от 0 до 65535):
Введите a: 12
Введите b: 23
Введите c: 34

Результаты для unsigned int (C):
Числитель: 782, Знаменатель: 53, Результат: 14

Результаты для unsigned int (ASM):
Числитель: 782, Знаменатель: 53, Результат: 14 _
```

5. Вывод

Результат выполнения программы соответствует ожидаемому результату. Работа выполнена в полном объеме.