

Правительство Российской Федерации
Федеральное государственное автономное образовательное учреждение
Высшего профессионального образования
Национальный исследовательский институт
«Высшая школа экономики»
Московский институт электроники и математики
Прикладная математика

Отчет
по лабораторной работе №2
по курсу «Язык ассемблера»

Вариант № 50

Ф.И.О. студента	Номер группы	Дата	Баллы
Спиридонов Даниил Андреевич	СКБ222	10.03.24	

Москва, 2024 г.

Задача

Написать программу, которая вычисляет значение выражения с помощью стандартных арифметических операций языка C и с помощью ассемблерной вставки. Проверить на двух тестовых наборах.

Формула:

$$v = 5 + \frac{-(z+2) \cdot x + 3}{y-1}$$

где y, z – байты, x, v – слова

Тесты:

- 1) $x = 2h, y = 4h, z = -8h, v = Ah$
- 2) $x = 7BEh, y = -7Eh, z = 7Eh, v = 7D2h$

Примечание: При вводе некорректных параметров (в этой задаче это значение $y = 1$) будет возвращаться $v = 5$.

Текст программы

```
int16_t asm_calculate(int16_t x, int8_t y, int8_t z) {
    int16_t v;

    if(y == 1) {
        return 5;
    }

    __asm {
        // Очищаем регистры
        xor ax, ax;
        xor bx, bx;

        // Вычисление числителя
        mov al, z;    // Так как z – байт, загружаем его в малый регистр
        cbw;          // Расширяем регистр до 16 бит, так как дальше будем работать со словами
        add ax, 2;     // Прибавляем к регистру 2
        neg ax;        // Инvertируем число (то есть ax := -ax)

        mov bx, x;    // Помещаем слово x в регистр bx
        imul bx;       // Умножаем регистр ax на bx. Получаем на выходе двойное слово

        add ax, 3;     // Прибавляем к ax 3
        sbb dx, 0;     // Вычитаем флаг переноса из dx, так как пара ax^dx участвует в делении
        mov bx, ax;    // Помещаем значения из ax в bx

        // Вычисление знаменателя
        xor ax, ax;    // Очищаем регистр ax
        mov al, y;     // Помещаем байт y в al
        cbw;          // Расширяем al до двойного слова
        sub ax, 1;     // Вычитаем из ax 1

        // Деление
        xchg ax, bx;   // Меняем местами ax и bx
        cwd;           // Расширяем до двойного слова
        idiv bx;       // Делим ax:dx на bx
        add ax, 5      // прибавляем к ax 5

        mov v, ax      // Помещаем в v ax
    };
    return v;
}

int16_t c_calculate(int16_t x, int8_t y, int8_t z) {
    int16_t num, div;

    if(y == 1) {
        return 5;
    }

    num = 3 - (z + 2) * x;
    div = y - 1;
    return 5 + num / div;
}
```

```

void run_test(int16_t x, int8_t y, int8_t z, int16_t v) {
    int8_t v_c, v_asm;

    printf("Ввод:\n\tx := %hd\n\ty := %hd\n\tz := %hd\n", x, y, z);

    printf("Правильный результат:\n\tv:= %hd\n", v);

    v_c = c_calculate(x, y, z);
    printf("Вычисления в Си:\n\tДесятичная: %hd\n\tШестнадцатеричная: 0x%04x\n\n", v_c, v_c);

    v_asm = asm_calculate(x, y, z);
    printf("Вычисления в Ассемблере:\n\tДесятичная: %hd\n\tШестнадцатеричная: 0x%04x\n\n", v_asm, v_asm);
    if(v_asm != v) {
        printf("Тест не пройден!\n\n");
    }
    else {
        printf("Тест пройден!\n\n");
    }
}

int main() {
    int8_t y, z;
    int16_t x, v;
    char ans;

    setlocale(LC_ALL, "Russian");

    printf("Тест #1\n");
    run_test(2, 4, -8, 0xA);

    printf("Тест #2\n");
    run_test(0x7BE, -0x7E, 0x7E, 0x7D2);

    printf("Продолжить ручное тестирование? (y/n):\t");
    scanf(" %c", &ans);

    while(ans != 'n') {
        printf("\nВведите x, y, z, v через пробел\n");
        scanf("%hd %hhd %hhd %hd", &x, &y, &z, &v);
        run_test(x, y, z, v);

        printf("Продолжить тестирование? (y/n):\t ");
        scanf(" %c", &ans);
    }

    return 0;
}

```

Функция с ассемблерной вставкой

```
int16_t asm_calculate(int16_t x, int8_t y, int8_t z) {
    int16_t v;

    if(y == 1) {
        return 5;
    }

    __asm {
        // Очищаем регистры
        xor ax, ax;
        xor bx, bx;

        // Вычисление числителя
        mov al, z;    // Так как z – байт, загружаем его в малый регистр
        cbw;          // Расширяем регистр до 16 бит, так как дальше будем работать со словами
        add ax, 2;     // Прибавляем к регистру 2
        neg ax;        // Инвертируем число (то есть ax := -ax)

        mov bx, x;    // Помещаем слово x в регистр bx
        imul bx;       // Умножаем регистр ax на bx. Получаем на выходе двойное слово

        add ax, 3;     // Прибавляем к ax 3
        sbb dx, 0;     // Вычитаем флаг переноса из dx, так как пара ax^dx участвует в делении
        mov bx, ax;    // Помещаем значения из ax в bx

        // Вычисление знаменателя
        xor ax, ax;    // Очищаем регистр ax
        mov al, y;     // Помещаем байт y в al
        cbw;           // Расширяем al до двойного слова
        sub ax, 1;     // Вычитаем из ax 1

        // Деление
        xchg ax, bx;   // Меняем местами ax и bx
        cwd;           // Расширяем до двойного слова
        idiv bx;       // Делим ax:dx на bx
        add ax, 5       // прибавляем к ax 5

        mov v, ax      // Помещаем в v ax
    };
    return v;
}
```

Тесты

Тест #1 (Из задания)

```
Ввод:
    x := 2
    y := 4
    z := -8
Правильный результат:
    v:= 10
Вычисления в Си:
    Десятичная: 10
    Шестнадцатеричная: 0x000a

Вычисления в Ассемблере:
    Десятичная: 10
    Шестнадцатеричная: 0x000a

Тест пройден!
```

Тест #2 (Из задания)

```
Ввод:
    x := 1982
    y := -126
    z := 126
Правильный результат:
    v:= 2002
Вычисления в Си:
    Десятичная: -61
    Шестнадцатеричная: 0xffffffffc3

Вычисления в Ассемблере:
    Десятичная: -61
    Шестнадцатеричная: 0xffffffffc3

Тест не пройден!
```

Тест #3 (Проверка граничных условий)

```
Введите x, y, z, v через пробел
2 1 2 5
Ввод:
    x := 2
    y := 1
    z := 2
Правильный результат:
    v:= 5
Вычисления в Си:
    Десятичная: 5
    Шестнадцатеричная: 0x0005

Вычисления в Ассемблере:
    Десятичная: 5
    Шестнадцатеричная: 0x0005

Тест пройден!
```

Проверка для теста #2

Дополнительный код: FFFFFFFC3

↓ Переводим в двоичную

1111 1111 1111 1111 1111 1111 1100 0011

↓ Вычитаем единицу

1111 1111 1111 1111 1111 1111 1100 0010

↓ Инвертируем все биты, кроме бита знака (1 для минуса)

0011 1101

↓ Переводим в десятичную СС

-61 (прямой код)