

ACS. Homework 3

Александр Васюков | БПИ235 (239)

Алгоритм вычисления целой части и остатка деления числа X на число Y в RISC-V:

- Записываем в регистр $t0$ делимое X , в регистр $t1$ делитель Y .
- Если $Y = 0$, выводится ошибка "Division by zero."
- Находим знак целой части через $XOR\ X\ Y$ и записываем результат в регистр $t2$. Если X и Y имеют одинаковый знак, то результат будет отрицательным, иначе положительным.
- Если $XOR < 0$, устанавливаем в регистр $t2$ значение 1 , иначе 0 . Это будет знаком целой части.
- Если $X < 0$, устанавливаем в регистр $t5$ значение 1 , иначе 0 . Это будет знаком остатка, он равен знаку делимого.
- Если $X < 0$ и $Y < 0$, делаем их положительными.
- В регистре $t3$ устанавливаем значение 0 , здесь будем хранить значение целой части.
- В регистр $t4$ перемещаем значение делимого, здесь будет храниться остаток.
- Пока остаток больше делителя: вычитаем из остатка делитель и прибавляем 1 к целой части.
- Устанавливаем отрицательный знак для целой части и остатка, если необходимо.
- Выводим результат.

Код на ассемлере:

Также можно найти на Гитхабе по ссылке: <https://github.com/vasyukov1/HSE-FCS-SE-2-year/blob/main/ACS/Homework/3/solution.asm>

```
.data
    dividend:      .asciz "Input dividend: "
    divisor:       .asciz "Input divider: "
    error_input:   .asciz "It isn't a number. Try again."
    error_zero:    .asciz "Division by zero."
    ceil:          .asciz "Ceil: "
    remainder:     .asciz "Remainder: "
    new_line:      .asciz "\n"

.text

main:
    # Input dividend
    li a7 4
    la a0 dividend
    ecall
    li a7 5
    ecall
    mv t0 a0

    # Input divivsor
    li a7 4
    la a0 divisor
    ecall
    li a7 5
    ecall
    mv t1 a0

    # Check by zero
    beqz t1 division_by_zero

    # Xor for dividend and divisor. If they have different signs, t2 < 0, else t2 >= 0
    xor t2 t1 t0

    # t2 = (t2 < 0) ? 1 : 0. If signs of dividend and divisor are different t2 = 1, else t2 = 0.
    slti t2 t2 0
    # t5 = (t0 < 0) ? 1 : 0. If the sign of dividend is negative, the sign of the remainder will be
    # negative too.
    slti t5 t0 0

    # if t0 < 0 goto negative_dividend
    bltz t0 negative_dividend
    j continue_dividend

    # Change sign of dividend to positive
negative_dividend:
    neg t0 t0
```

```

continue_dividend:
    # if t1 < 0 goto negative_divisor
    bltz t1 negative_divisor
    j continue_divisor

# Change sign of divisor to positive
negative_divisor:
    neg t1 t1

# Set the ceiling part and the remainder
continue_divisor:
    # ceil
    li t3 0
    # remainder
    mv t4 t0

# Subtraction cycle
division:
    blt t4 t1 check_ceil_sign
    sub t4 t4 t1
    addi t3 t3 1
    j division

# Change the sign if the ceiling part is negative
check_ceil_sign:
    beqz t2 check_remainder_sign
    neg t3 t3

check_remainder_sign:
    beqz t5 unput_result
    neg t4 t4

# Input result
unput_result:
    # Input ceiling part
    li a7 4
    la a0 ceil
    ecall
    mv a0 t3
    li a7 1
    ecall

    # New line for aesthetics
    li a7 4
    la a0 new_line
    ecall

    # Input remainder
    li a7 4
    la a0 remainder
    ecall
    mv a0 t4
    li a7 1
    ecall

    j exit

# Input error if divisor equals zero
division_by_zero:
    li a7 4
    la a0 error_zero
    ecall

# Stop
exit:
    li a7 10
    ecall

```

Тесты, покрывающие все случаи деления:

1. Положительное число на положительное
2. Положительное число на отрицательное
3. Отрицательное число на положительное
4. Отрицательное число на отрицательное

5. Число на ноль

Код на C++ для проверки:

```
#include <iostream>
using std::cin;
using std::cout;

void solution(const int&, const int&, const int&);

int main() {

    const int pos_dividend = 15;
    const int neg_dividend = -15;
    const int pos_divisor = 2;
    const int neg_divisor = -2;
    const int zero_divisor = 0;

    solution(pos_dividend, pos_divisor, 1);
    solution(pos_dividend, neg_divisor, 2);
    solution(neg_dividend, pos_divisor, 3);
    solution(neg_dividend, neg_divisor, 4);
    solution(pos_dividend, zero_divisor, 5);
}

void solution(const int& dividend, const int& divisor, const int& counter) {
    cout << "-----\nTEST " << counter << '\n';
    cout << "Dividend: " << dividend << '\n';
    cout << "Divisor: " << divisor << '\n';
    if (divisor == 0) {
        cout << "Division by zero.";
        return;
    }
    const int ceil = dividend / divisor;
    const int remainder = dividend % divisor;
    cout << "Ceil: " << ceil << '\n';
    cout << "Remainder: " << remainder << '\n';
}
```

Код с автоматическим тестированием: https://github.com/vasyukov1/HSE-FCS-SE-2-year/blob/main/ACS/Homework/3/solution_with_autotest.asm

```
.data
    dividend:    .asciz "Input dividend: "
    divisor:     .asciz "Input divider: "
    error_input: .asciz "It isn't a number. Try again."
    error_zero:  .asciz "Division by zero."
    ceil:        .asciz "Ceil: "
    remainder:   .asciz "Remainder: "
    new_line:    .asciz "\n"
    new_test:    .asciz "-----\nTEST "

    pos_dividend: .word 15
    neg_dividend: .word -15
    pos_divisor:  .word 2
    neg_divisor:  .word -2
    zero_divisor: .word 0

    counter:     .word 1

.text
main:
    # Number of tests
    la t6 counter
    lw t6 0(t6)

test1:
    # Test 1. 15 / 2
    la t0 pos_dividend
    lw t0 0(t0)
    la t1 pos_divisor
    lw t1 0(t1)
    j func

test2:
    # Test 2. 15 / -2
```

```

        la t0 pos_dividend
        lw t0 0(t0)
        la t1 neg_divisor
        lw t1 0(t1)
        j func
test3:
        # Test 3. -15 / 2
        la t0 neg_dividend
        lw t0 0(t0)
        la t1 pos_divisor
        lw t1 0(t1)
        j func
test4:
        # Test 4. -15 / -2
        la t0 neg_dividend
        lw t0 0(t0)
        la t1 neg_divisor
        lw t1 0(t1)
        j func
test5:
        # Test 5. 15 / 0
        la t0 pos_dividend
        lw t0 0(t0)
        la t1 zero_divisor
        lw t1 0(t1)
        j func

func:
        # Output the number of test
        li a7 4
        la a0 new_test
        ecall
        mv a0 t6
        li a7 1
        ecall
        # New line for aesthetics
        li a7 4
        la a0 new_line
        ecall

        # Input dividend
        li a7 4
        la a0 dividend
        ecall
        li a7 1
        mv a0 t0
        ecall
        # New line for aesthetics
        li a7 4
        la a0 new_line
        ecall

        # Input divivsor
        li a7 4
        la a0 divisor
        ecall
        li a7 1
        mv a0 t1
        ecall
        # New line for aesthetics
        li a7 4
        la a0 new_line
        ecall

        # Check by zero
        beqz t1 division_by_zero

        # Xor for dividend and divisor. If they have different signs then t2 < 0, else t2 >= 0
        xor t2 t1 t0

        # t2 = (t2 < 0) ? 1 : 0. If signs of dividend and divisor are different t2 = 1, else t2 = 0.
        slti t2 t2 0
        # t5 = (t0 < 0) ? 1 : 0. If the sign of dividend is negative, the sign of the remainder will be
negative too.
        slti t5 t0 0

        # if t0 < 0 goto negative_dividend

```

```

        bltz t0 negative_dividend
        j continue_dividend

# Change sign of dividend to positive
negative_dividend:
        neg t0 t0

continue_dividend:
        # if t1 < 0 goto negative_divisor
        bltz t1 negative_divisor
        j continue_divisor

# Change sign of divisor to positive
negative_divisor:
        neg t1 t1

# Set the ceiling part and the remainder
continue_divisor:
        # the ceiling part
        li t3 0
        # remainder
        mv t4 t0

# Substraction cycle
division:
        blt t4 t1 check_ceil_sign
        sub t4 t4 t1
        addi t3 t3 1
        j division

# Change the sign of the ceiling part if t2 < 0
check_ceil_sign:
        beqz t2 check_remainder_sign
        neg t3 t3

# Change the remainder's sign if t5 < 0
check_remainder_sign:
        beqz t5 unput_result
        neg t4 t4

# Input result
unput_result:
        # Input ceiling part
        li a7 4
        la a0 ceil
        ecall
        mv a0 t3
        li a7 1
        ecall

        # New line for aesthetics
        li a7 4
        la a0 new_line
        ecall

        # Input remainder
        li a7 4
        la a0 remainder
        ecall
        mv a0 t4
        li a7 1
        ecall

        # New line for aesthetics
        li a7 4
        la a0 new_line
        ecall

        j check_count

# Input error if divisor equals zero
division_by_zero:
        li a7 4
        la a0 error_zero
        ecall

# Check the number of test

```

```
check_count:
    addi t6 t6 1
    li t0 1
    beq t6 t0 test1
    li t0 2
    beq t6 t0 test2
    li t0 3
    beq t6 t0 test3
    li t0 4
    beq t6 t0 test4
    li t0 5
    beq t6 t0 test5

# Stop
exit:
    li a7 10
    ecall
```

Результат работы на ассемблере:

```
-----
TEST 1
Input dividend: 15
Input divider: 2
Ceil: 7
Remainder: 1
-----
TEST 2
Input dividend: 15
Input divider: -2
Ceil: -7
Remainder: 1
-----
TEST 3
Input dividend: -15
Input divider: 2
Ceil: -7
Remainder: -1
-----
TEST 4
Input dividend: -15
Input divider: -2
Ceil: 7
Remainder: -1
-----
TEST 5
Input dividend: 15
Input divider: 0
Division by zero.
-- program is finished running (0) --
```

Результат работы на C++:

```
/Users/alexvasyukov/Documents/GitHub/HSE-FCS-SE-2-year/ACS/Homework/3/main
-----
TEST 1
Dividend: 15
Divisor: 2
Ceil: 7
Remainder: 1
-----
TEST 2
Dividend: 15
Divisor: -2
Ceil: -7
Remainder: 1
-----
TEST 3
Dividend: -15
Divisor: 2
Ceil: -7
Remainder: -1
-----
TEST 4
Dividend: -15
Divisor: -2
Ceil: 7
Remainder: -1
-----
TEST 5
Dividend: 15
Divisor: 0
Division by zero.
Process finished with exit code 0
```