Отчёт по домашней работе

# 1. Введение

Целью работы является разработка программы на языке ассемблера RARS, осуществляющей целочисленное деление двух 32-битных чисел со знаком с использованием операций вычитания, ветвления и циклов. Программа должна корректно обрабатывать случаи деления на ноль, учитывать знаки делимого и делителя, а также вычислять остаток в соответствии с правилами языков программирования C/C++.

# 2. Листинг программы

# =================== RISC-V RV32 Деление через сдвиги ===================

# Главное меню с выбором действий

# Поддержка "выход" везде

# Случайные тесты

.data

# --- Строки для логирования ---

prompt\_log: .string "Хотите подробный лог? Введите точно: Хочу! (иначе нажмите Enter): \0"

seed\_prompt: .string "Для рандомизации тестов введите любые символы и нажмите Enter: \0"

newline: .string "\n\0"

# --- Строки для меню ---

menu\_title: .string "\n=== Главное меню ===\n\0"

menu\_option1: .string "1. Заготовленные тесты\n\0"

menu\_option2: .string "2. Случайные тесты\n\0"

menu\_option3: .string "3. Ручной ввод\n\0"

menu\_option0: .string "0. Выход\n\0"

menu\_prompt: .string "Выберите пункт меню (0-3): \0"

# --- Строки для ввода ---

prompt\_dividend: .string "Введите делимое (или 'выход' для возврата в меню): \0"

prompt\_divisor: .string "Введите делитель (или 'выход' для возврата в меню): \0"

# --- Строки ошибок ---

errdiv0: .string "Ошибка: деление на ноль! Повторите ввод делителя.\n\0"

errinput: .string "Ошибка: введено не целое число или переполнение! Повторите ввод.\n\0"

# --- Строки результатов ---

result\_prefix: .string "Делимое = \0"

result\_mid1: .string ", делитель = \0"

result\_mid2: .string " → частное = \0"

result\_suffix: .string ", остаток = \0"

# --- Отладочные строки ---

debug\_prefix: .string "DEBUG: \0"

debug\_sep: .string " | \0"

# --- Слово "выход" для проверки ---

exit\_word: .string "выход\0"

# --- Заготовленные тесты ---

num\_tests: .word 5

test\_dividends: .word 10, -10, 10, -10, 0

test\_divisors: .word 3, 3, -3, -3, 3

# --- Буфер ввода ---

buffer: .space 64

# --- Сид для генератора случайных чисел ---

rand\_seed: .word 123456789

.text

.globl main

# ================= Основная программа =================

main:

# --- Запрос на подробный лог (один раз в начале) ---

li x17, 4

la x10, prompt\_log

ecall

li x17, 8

la x10, buffer

li x11, 64

ecall

jal x1, clear\_newline

la x5, buffer

li x22, 0 # флаг логирования = 0 (выкл)

# Проверка "Хочу!" в UTF-8

lbu x6, 0(x5)

li x7, 208

bne x6, x7, skip\_log\_check

lbu x6, 1(x5)

li x7, 165

bne x6, x7, skip\_log\_check

lbu x6, 2(x5)

li x7, 208

bne x6, x7, skip\_log\_check

lbu x6, 3(x5)

li x7, 190

bne x6, x7, skip\_log\_check

lbu x6, 4(x5)

li x7, 209

bne x6, x7, skip\_log\_check

lbu x6, 5(x5)

li x7, 135

bne x6, x7, skip\_log\_check

lbu x6, 6(x5)

li x7, 209

bne x6, x7, skip\_log\_check

lbu x6, 7(x5)

li x7, 131

bne x6, x7, skip\_log\_check

lbu x6, 8(x5)

li x7, 33

bne x6, x7, skip\_log\_check

li x22, 1 # логирование включено

skip\_log\_check:

# --- Инициализация случайного seed на основе длины ввода ---

li x17, 4

la x10, seed\_prompt

ecall

li x17, 8

la x10, buffer

li x11, 64

ecall

jal x1, clear\_newline

# Подсчёт длины введённой строки

la x5, buffer

li x6, 0

count\_length\_loop:

lbu x7, 0(x5)

beq x7, x0, length\_done

addi x6, x6, 1

addi x5, x5, 1

j count\_length\_loop

length\_done:

# Генерация seed: (длина + 12345) \* 1664525 XOR 1013904223

li x7, 12345

add x6, x6, x7

li x7, 1664525

mul x6, x6, x7

li x7, 1013904223

xor x6, x6, x7

la x5, rand\_seed

sw x6, 0(x5)

# --- Главный цикл меню ---

main\_menu\_loop:

# Вывод меню

li x17, 4

la x10, menu\_title

ecall

li x17, 4

la x10, menu\_option1

ecall

li x17, 4

la x10, menu\_option2

ecall

li x17, 4

la x10, menu\_option3

ecall

li x17, 4

la x10, menu\_option0

ecall

li x17, 4

la x10, menu\_prompt

ecall

# Чтение строки

li x17, 8

la x10, buffer

li x11, 64

ecall

jal x1, clear\_newline

# Проверка на "выход"

jal x1, check\_for\_exit

bne x10, x0, exit\_program

# Получаем первый непробельный символ

la x10, buffer

jal x1, get\_first\_char

mv x5, x10

# Обработка выбора

li x6, '0'

beq x5, x6, exit\_program

li x6, '1'

beq x5, x6, run\_preset\_tests

li x6, '2'

beq x5, x6, run\_random\_tests

li x6, '3'

beq x5, x6, manual\_input\_mode

j main\_menu\_loop # неверный ввод — повторить

# ================= Очистка новой строки =================

clear\_newline:

la x5, buffer

clear\_loop:

lbu x6, 0(x5)

beq x6, x0, clear\_done

li x7, 10 # '\n'

beq x6, x7, clear\_replace

addi x5, x5, 1

j clear\_loop

clear\_replace:

sb x0, 0(x5)

clear\_done:

jr x1

# ================= Получить первый непробельный символ =================

get\_first\_char:

lbu x5, 0(x10)

beq x5, x0, get\_char\_done

li x6, 32 # пробел

beq x5, x6, skip\_char

li x6, 9 # таб

beq x5, x6, skip\_char

mv x10, x5

jr x1

skip\_char:

addi x10, x10, 1

j get\_first\_char

get\_char\_done:

li x10, 0

jr x1

# ================= Запуск заготовленных тестов =================

run\_preset\_tests:

la x5, num\_tests

lw x20, 0(x5)

li x21, 0

preset\_test\_loop:

beq x21, x20, return\_to\_menu

slli x7, x21, 2

la x6, test\_dividends

add x6, x6, x7

lw x8, 0(x6)

la x6, test\_divisors

add x6, x6, x7

lw x9, 0(x6)

mv s2, x8 # s2 = исходное делимое

mv s3, x9 # s3 = исходный делитель

beq x22, x0, skip\_preset\_debug

li x17, 4

la x10, debug\_prefix

ecall

li x17, 1

mv x10, x21

ecall

li x17, 4

la x10, debug\_sep

ecall

li x17, 1

mv x10, s2

ecall

li x17, 4

la x10, debug\_sep

ecall

li x17, 1

mv x10, s3

ecall

li x17, 4

la x10, newline

ecall

skip\_preset\_debug:

mv x10, s2

mv x11, s3

jal x1, fast\_divide

mv x12, x10

mv x13, x11

li x17, 4

la x10, result\_prefix

ecall

li x17, 1

mv x10, s2

ecall

li x17, 4

la x10, result\_mid1

ecall

li x17, 1

mv x10, s3

ecall

li x17, 4

la x10, result\_mid2

ecall

li x17, 1

mv x10, x12

ecall

li x17, 4

la x10, result\_suffix

ecall

li x17, 1

mv x10, x13

ecall

li x17, 4

la x10, newline

ecall

addi x21, x21, 1

j preset\_test\_loop

# ================= Запуск случайных тестов =================

run\_random\_tests:

li x20, 5

li x21, 0

random\_test\_loop:

beq x21, x20, return\_to\_menu

jal x1, rand

mv s2, x10

jal x1, rand

mv s3, x10

beq s3, x0, random\_test\_loop

mv x10, s2

mv x11, s3

jal x1, fast\_divide

mv x12, x10

mv x13, x11

li x17, 4

la x10, result\_prefix

ecall

li x17, 1

mv x10, s2

ecall

li x17, 4

la x10, result\_mid1

ecall

li x17, 1

mv x10, s3

ecall

li x17, 4

la x10, result\_mid2

ecall

li x17, 1

mv x10, x12

ecall

li x17, 4

la x10, result\_suffix

ecall

li x17, 1

mv x10, x13

ecall

li x17, 4

la x10, newline

ecall

addi x21, x21, 1

j random\_test\_loop

# ================= Режим ручного ввода =================

manual\_input\_mode:

input\_dividend:

li x17, 4

la x10, prompt\_dividend

ecall

li x17, 8

la x10, buffer

li x11, 64

ecall

jal x1, clear\_newline

jal x1, check\_for\_exit

bne x10, x0, return\_to\_menu

la x10, buffer

jal x1, parse\_int\_safe

bne x11, x0, input\_dividend

mv x18, x10

input\_divisor:

li x17, 4

la x10, prompt\_divisor

ecall

li x17, 8

la x10, buffer

li x11, 64

ecall

jal x1, clear\_newline

jal x1, check\_for\_exit

bne x10, x0, return\_to\_menu

la x10, buffer

jal x1, parse\_int\_safe

bne x11, x0, input\_divisor

mv x19, x10

beq x19, x0, div\_zero\_error

mv x10, x18

mv x11, x19

jal x1, fast\_divide

mv x12, x10

mv x13, x11

li x17, 4

la x10, result\_prefix

ecall

li x17, 1

mv x10, x18

ecall

li x17, 4

la x10, result\_mid1

ecall

li x17, 1

mv x10, x19

ecall

li x17, 4

la x10, result\_mid2

ecall

li x17, 1

mv x10, x12

ecall

li x17, 4

la x10, result\_suffix

ecall

li x17, 1

mv x10, x13

ecall

li x17, 4

la x10, newline

ecall

j manual\_input\_mode

div\_zero\_error:

li x17, 4

la x10, errdiv0

ecall

j input\_divisor

return\_to\_menu:

j main\_menu\_loop

exit\_program:

li x17, 10

ecall

# ================= Проверка на "выход" =================

check\_for\_exit:

la x5, exit\_word

la x6, buffer

li x7, 0

compare\_exit\_loop:

lbu x8, 0(x6)

lbu x9, 0(x5)

beq x8, x0, check\_exit\_end

beq x9, x0, not\_exit\_word

bne x8, x9, not\_exit\_word

addi x6, x6, 1

addi x5, x5, 1

j compare\_exit\_loop

check\_exit\_end:

lbu x9, 0(x5)

bne x9, x0, not\_exit\_word

li x10, 1

jr x1

not\_exit\_word:

li x10, 0

jr x1

# ================= Безопасный парсинг целого числа =================

parse\_int\_safe:

li x11, 0

li x5, 0

li x6, 0

mv x28, x10

skip\_spaces:

lbu x7, 0(x10)

li x8, 32

beq x7, x8, next\_space

li x8, 9

beq x7, x8, next\_space

j check\_sign

next\_space:

addi x10, x10, 1

j skip\_spaces

check\_sign:

lbu x7, 0(x10)

beq x7, x0, parse\_error

li x8, 45

beq x7, x8, minus\_sign

li x8, 43

beq x7, x8, plus\_sign

j digits\_loop

minus\_sign:

li x6, 1

addi x10, x10, 1

j digits\_loop

plus\_sign:

addi x10, x10, 1

j digits\_loop

digits\_loop:

lbu x7, 0(x10)

beq x7, x0, done\_parsing

li x8, 48

li x9, 57

blt x7, x8, parse\_error

bgt x7, x9, parse\_error

sub x7, x7, x8

slli x12, x5, 3

slli x13, x5, 1

add x12, x12, x13

add x5, x12, x7

addi x10, x10, 1

j digits\_loop

done\_parsing:

beq x6, x0, success

neg x5, x5

success:

mv x10, x5

li x11, 0

jr x1

parse\_error:

li x11, 1

li x10, 0

jr x1

# ================= Быстрое деление (C99-совместимое) =================

fast\_divide:

mv x5, x10

mv x6, x11

li x7, 0

bltz x5, neg\_a

mv x8, x5

j check\_b

neg\_a:

neg x8, x5

xori x7, x7, 1

check\_b:

bltz x6, neg\_b

mv x9, x6

j divide

neg\_b:

neg x9, x6

xori x7, x7, 1

divide:

li x12, 0

li x13, 0

li x14, 32

loop:

beq x14, x0, sign

slli x13, x13, 1

srli x15, x8, 31

or x13, x13, x15

slli x8, x8, 1

slli x12, x12, 1

blt x13, x9, no\_sub

sub x13, x13, x9

ori x12, x12, 1

no\_sub:

addi x14, x14, -1

j loop

sign:

beq x7, x0, remainder

neg x12, x12

remainder:

mul x15, x6, x12

sub x13, x5, x15

mv x10, x12

mv x11, x13

jr x1

# ================= Генератор случайных чисел (LCG) =================

rand:

la x5, rand\_seed

lw x6, 0(x5)

li x7, 1664525

mul x6, x6, x7

li x7, 1013904223

add x6, x6, x7

sw x6, 0(x5)

mv x10, x6

jr x1

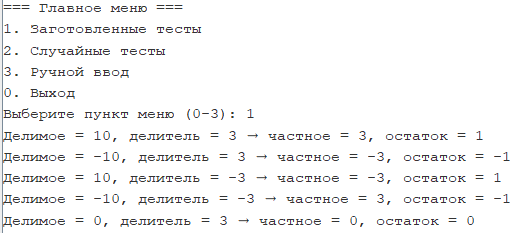
# 3. Таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| Делимое | Делитель | Ожидаемый результат | Фактический результат |
| 10 | 3 | Частное=3, Остаток=1 | Частное=3, Остаток=1 |
| -10 | 3 | Частное=-3, Остаток=-1 | Частное=-3, Остаток=-1 |
| 10 | -3 | Частное=-3, Остаток=1 | Частное=-3, Остаток=1 |
| -10 | -3 | Частное=3, Остаток=-1 | Частное=3, Остаток=-1 |
| 0 | 3 | Частное=0, Остаток=0 | Частное=0, Остаток=0 |
| 5 | 1 | Частное=5, Остаток=0 | Частное=5, Остаток=0 |
| 5 | -1 | Частное=-5, Остаток=0 | Частное=-5, Остаток=0 |
| 7 | 7 | Частное=1, Остаток=0 | Частное=1, Остаток=0 |
| 7 | -7 | Частное=-1, Остаток=0 | Частное=-1, Остаток=0 |
| 10 | 0 | Ошибка: деление на ноль! | Ошибка: деление на ноль! |

# 4. Скриншоты работы программы

Ниже приведены примеры скриншотов консоли симулятора RARS, демонстрирующие работу программы на различных входных данных.

Скриншот 1:



Скриншот 2:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

# 5. Заключение

В ходе работы была разработана программа на ассемблере RARS, реализующая деление двух 32-разрядных целых чисел со знаком через вычитания и циклы. Программа корректно обрабатывает ошибку деления на ноль, учитывает знаки делимого и делителя, а также формирует остаток в соответствии с правилами C/C++.