**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №5**

**по дисциплине «Организация ЭВМ и систем»**

Тема: **Знакомство с рабочей средой эмулятора Ripes для работы с процессором RISC-V. Базовый ISA, система команд, состав регистров. Разработка и выполнение простой программы на ассемблере RISC-V.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3342 |  | Мохамед М.Х. |
| Преподаватель |  | Морозов С.М. |

Санкт-Петербург

2024

**Цель работы.**

Целями работы являются:

1. Освоение работы с эмулятором Ripes: установка, настройка, трансляция ассемблерной программы, выполнение программы в автоматическом и отладочном режимах.

2. Изучение архитектуры RISC-V, базового набора инструкций и разработка простых программ на ассемблере.

а) Ознакомиться с основными компонентами компьютера RISC-V;

б) Освоение взаимосвязи между аппаратными и программными средствами компьютера на основе архитектуры системы команд (ISA- Instruction Set Architecture).

в) Изучение языка ассемблера RISC-V и кодирования операторов ассемблера в машинном коде.

г) Освоение этапов компиляции и выполнения ассемблерной программы в автоматическом и отладочном режимах.

**Задание.**

1. Разработайте процедуру на ассемблере, которая для целочисленных 32-битных входных переменных x, y, z и констант a, b, c вычисляет выражение R = f (x, y, z, a, b, c)

В выражениях используются следующие константы:

|  |  |
| --- | --- |
| Константа | Значение |
| a | [Сумма цифр студ. билета] |
| b | [Количество букв в фамилии] |
| c | [Количество букв в полном имени] |

Вариант 24

R = ((x ^ (-b)) & (z ^ c)) | (y + a)

Константы:  
a = 22 (№ студ. билета 334228)

b = 7 (Мохамед)

c = 6 (Махмуд)

Формула для вычисления:  
R = ((x ^ (-b)) & (z ^ c)) | (y + a)

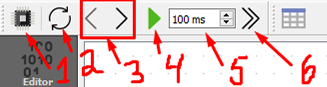
2. Напишите программу, которая для двух наборов исходных данных x, y, z выполняет вычисление заданного выражения с помощью разработанной процедуры, сохраняет в регистрах и выводит на экран результаты вычислений.

Начальные значения {x1, y1, z1} расположить в регистрах a2, a3, a4; значения {x2, y2, z2} расположить в регистрах a5, a6, a7; значения констант a, b, c расположить в регистрах s0, s1, s2. Результаты вычисления {r1, r2} записать в регистры а0, а1.

Моделируемые вычисления (формулы, входные данные, промежуточные значения, результаты) должны выводиться в консоль.

**Основные теоретические положения.**

Основные возможности эмулятора Ripes:

****

* Меню настройки процессора (цифра 1)
* Сброс состояния процессора, памяти, регистров и т.д. (цифра 2)
* Пошаговое выполнение (один такт вперёд или один такт назад) (цифра 3)
* Запуск в демонстрационном режиме, где каждый следующий такт выполняется через заданное количество времени (цифра 4)
* Установка временного интервала между тактами в демонстрационном режиме (цифра 5)
* “Моментальное” выполнение программы без задержек между тактами (цифра 6)

Системные вызовы:

1. PrintInt (a7=1): выводит число в регистре a0 в консоль в десятичной системе счисления

2. PrintString (a7=4): выводит строку, заканчивающуюся символом конца строки, адрес строки считывается из регистра a0

3. PrintChar (a7=11): выводит младший байт регистра a0 в консоль как

символ

4. Write (a7=64): выводит в файловый дескриптор, номер которого записан в a0, строку с адреса, записанного в a1, длиною a2. В конце в a0 записывается количество записанных байт

5. Exit (a7=93): завершает программу с кодом возврата в регистре a0

Программа на Ассемблере состоит из директив (рассматриваются на этапе трансляции), инструкций (выполняются при запуске программы) данных. Все они хранятся в соответствующих разделах в соответствии с их назначением. Разделы определяются с помощью директив. Основными разделами являются:

1. Раздел TEXT Раздел, доступный только для чтения, содержит фактические инструкции программы.

Синтаксис определения: .text

Описание. Этот раздел также известен как сегмент кода или просто текстовый сегмент программы. Он содержит исполняемые инструкции, которые не могут быть изменены во время выполнения. Любая попытка сохранить что-то в раздел text выдаст ошибку “Сегментация”, и программа немедленно завершится. Сегмент кода может в дополнение к инструкциям также содержать константы.

2. Раздел DATA Раздел, доступный для чтения и записи, содержит данные для переменных программы.

Синтаксис: .data Переменные

Описание. Раздел (сегмент) .data содержит инициализированные статические переменные, которые являются глобальными или статическими локальными переменными.

Инструкции и псевдо-инструкции:

Многие команды программ на ассемблере RISC-V не используют три

аргумента, так как являются псевдоинструкциями. Это означает, что они являются сокращениями для других инструкций.

Пример.

Псевдоинструкция li x5, 0x123456 (загрузка большого значения в регистр) преобразуется ассемблером в 2 инструкции:

lui x5, 0x123 (загрузить константу в старшие биты 31-12 регистра)

addi x5, x5, 0x456 (сложение того же регистра с младшими 12 битами).

**Выполнение работы.**

1. Процедура calc\_expression

Разработана процедура, вычисляющая выражение для целочисленных 32 битных входных переменных согласно варианту. Внутри вычисляется выражение для двух наборов данных {x1, y1, z1} и {x2, y2, z2}. Использованы инструкции xor (исключающее «или»), and (логическое «и»), add (сложение), а также псевдоинструкции neg (обратное число) и ret для возвращения из процедуры.

2. Основная программа

Состоит из двух разделов. В разделе data хранятся строки, которые впоследствии будут выводиться в консоль.

В разделе text содержится код программы. С помощью директивы equ определяются константы a, b, c, далее они загружаются в регистры инструкцией addi. Псевдо-инструкцией li в регистры загружаются значения для двух наборов

{x1, y1, z1} и {x2, y2, z2}, для больших чисел данная псевдо-инструкция преобразуется в две: lui (загружает константу в старшие биты регистра) и addi (складывает тот же регистр с младшими 12-ю битами). Далее с помощью псевдо-инструкции la в регистры загружаются адреса строк для печати, они выводятся с помощью системного вызова printString. Символы и числа для печати загружаются в регистр а0 после чего выводятся системными вызовами PrintChar и PrintInt соответственно. Процедура calc\_expression вызывается псевдо-инструкцией call, которая преобразуется в две инструкции: auipc и jalr. После вывода результатов, полученных в процедуре, программа завершается системным вызовом Exit.

Результаты отладки в пошаговом режиме под управлением отладчика с фиксацией содержимого используемых регистров и ячеек памяти до и после выполнения команд представлены в таблице 1. Тестирование программы на 3-х наборах для {x1, y1, z1} и 3-х наборах для {x2, y2, z2} представлены в таблице

2. Исходный код в приложении А.

Таблица 1 – результаты отладки программы в пошаговом режиме

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Адрес инстр. | (Псевдо-) инстр. | Инструк ция(и) | 16-ричный код инстр. | Содержимое регистров и ячеек памяти | |
| до вып. инстр. | после вып. инстр. |
| 0 |  | addi x8 x0 22 | 01600413 | x8 s0 0x00000000 | x8 s0 0x00000016 |
| 4 |  | addi x9 x0 7 | 00700493 | x9 s1 0x00000000 | x9 s1 0x00000007 |
| 8 |  | addi x18 x0 9 | 00600913 | x18 s2 0x00000000 | x18 s2 0x00000009 |
| c | li a2, -17 | addi x12 x0 -17 | ffef00613 | x12 a2 0x00000000 | x12 a2 0xffffffef |
| 10 | li a3, 5695 | lui x13 0x1 | 000016b7 | x13 a3 0x00000000 | x13 a3 0x00001000 |
| 14 | addi x13 x13 1599 | 63f68693 | x13 a3 0x00001000 | x13 a3 0x0000163f |
| 18 | li a4, 9 | addi x14 x0 9 | 00900713 | x14 a4 0x00000000 | x14 a4 0x00000009 |
| 1c | li a5, -132 | addi x15 x0 -132 | f7c00793 | x15 a5 0x00000000 | x15 a5 0xffffff7c |
| 20 | li a6, -11 | addi x16 x0 -11 | ff500813 | x16 a6 0x00000000 | x16 a6 0xfffffff5 |
| 24 | li a7, 310 | addi x17 x0 310 | 13600893 | x17 a7 0x00000000 | x17 a7 0x00000136 |
| 28 | mv s4, a7 | addi x20 x17 0 | 00088a13 | x20 s4 0x00000000 x17 a7 0x00000136 | x20 s4 0x00000136 x17 a7 0x00000136 |
| 2c | la a0,  formula | auipc x10 0x10000 | 10000517 | x10 a0 0x00000000 | x10 a0 0x1000002c |
| 30 | addi x10 x10 -44 | fd450513 | x10 a0 0x1000002c | x10 a0 0x10000000 |
| 34 |  | addi x17 x0 4 | 00400893 | x17 a7 0x00000136 | x17 a7 0x00000004 |
| 38 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| 3c | mv a0, a2 | addi x10 x12 0 | 00060513 | x10 a0 0x10000000  x12 a2 0xffffffef | x10 a0 0xffffffef x12 a2 0xffffffef |
| 40 |  | addi x17 x0 1 | 00100893 | x17 a7 0x00000004 | x17 a7 0x00000001 |
| 44 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| 48 |  | addi x10 x0 32 | 02000513 | x10 a0 0xffffffef | x10 a0 0x00000020 |
| 4с |  | addi x17 x0 11 | addi x10 x0 32 | x17 a7 0x00000001 | x17 a7 0x0000000b |
| 50 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| 54 | mv a0, a3 | addi x10 x13 0 | 00068513 | x10 a0 0x00000020  x12 a2 0xffffffef | x10 a0 0x0000163f  x12 a2 0xffffffef |
| 58 |  | addi x17 x0 1 | 00100893 | x17 a7 0x0000000b | x17 a7 0x00000001 |
| 5c |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| 60 | li a0, 32 | addi x10 x0 32 | 02000513 | x10 a0 0x0000163f | x10 a0 0x00000020 |
| 64 |  | addi x17 x0 11 | 00b00893 | x17 a7 0x00000001 | x17 a7 0x0000000b |
| 68 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| 6c | mv a0, a4 | addi x10 x14 0 | 00070513 | x10 a0 0x00000020 x14 a4 0x00000009 | x10 a0 0x00000009 x14 a4 0x00000009 |
| 70 |  | addi x17 x0 1 | 00100893 | x17 a7 0x0000000b | x17 a7 0x00000001 |
| 74 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| 78 | li a0, 10 | addi x10 x0 10 | 00a00513 | x10 a0 0x00000009 | x10 a0 0x0000000a |
| 7c |  | addi x17 x0 11 | 00b00893 | x17 a7 0x00000001 | x17 a7 0x0000000b |
| 80 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| 84 | la a0, data | auipc x10 0x10000 | 10000517 | x10 a0 0x0000000a | x10 a0 0x10000084 |
| 88 |  | addi x10 x10 -60 | fc450513 | x10 a0 0x10000084 | x10 a0 0x10000048 |
| 8c |  | addi x17 x0 4 | 00400893 | x17 a7 0x0000000b | x17 a7 0x00000004 |
| 90 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| 94 | mv a0, a5 | addi x10 x15 0 | 00078513 | x10 a0 0x10000048  x15 a5 0xffffff7c | x10 a0 0xffffff7c  x15 a5 0xffffff7c |
| 98 |  | addi x17 x0 1 | 00100893 | x17 a7 0x00000004 | x17 a7 0x00000001 |
| 9c |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| a0 | li a0, 32 | addi x10 x0 32 | 02000513 | x10 a0 0xffffff7c | x10 a0 0x00000020 |
| a4 |  | addi x17 x0 11 | 00b00893 | x17 a7 0x00000001 | x17 a7 0x0000000b |
| a8 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| ac | mv a0, a6 | addi x10 x16 0 | 00080513 | x10 a0 0x00000020  x16 a7 0xfffffff5 | x10 a0 0xfffffff5  x16 a7 0xfffffff5 |
| b0 |  | addi x17 x0 1 | 00100893 | x17 a7 0x0000000b | x17 a7 0x00000001 |
| b4 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| b8 |  | addi x10 x0 32 | 02000513 | x10 a0 0xfffffff5 | x10 a0 0x00000020 |
| bc |  | addi x17 x0 11 | 00b00893 | x17 a7 0x00000001 | x17 a7 0x0000000b |
| c0 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| c4 | mv a7, s4 | addi x17 x20 0 | 000a0893 | x17 a7 0x0000000b  x20 s4 0x00000136 | x17 a7 0x00000136  x20 s4 0x00000136 |
| c8 |  | addi x10 x17 0 | 00088513 | x10 a0 0x00000020  x17 a7 0x00000136 | x10 a0 0x00000136  x17 a7 0x00000136 |
| cc |  | addi x17 x0 1 | 00100893 | x17 a7 0x00000136 | x17 a7 0x00000001 |
| d0 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| d4 | li a0, 10 | addi x10 x0 10 | 00a00513 | x10 a0 0x00000136 | x10 a0 0x0000000a |
| d8 |  | addi x17 x0 11 | 00b00893 | x17 a7 0x00000001 | x17 a7 0x0000000b |
| dc |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| e0 | mv a7, s4 | addi x17 x20 0 | 000a0893 | x17 a7 0x0000000b  x20 s4 0x00000136 | x17 a7 0x00000136  x20 s4 0x00000136 |
| e4 | call calc\_expr ession | auipc x1 0x0  <\_start> | 00000097 | x1 ra 0x00000000 | x1 ra 0x000000e4 |
| e8 | jalr x1 x1 72 | 048080e7 | x1 ra 0x000000e4 | x1 ra 0x000000ec |
| 12c | neg s1, s1 | sub x9 x0 x9 | 409004b3 | x9 s1 0x00000007 | x9 s1 0xfffffff9 |
| 130 |  | xor x12 x12 x9 | 00964633 | x12 a2 0xffffffef  x9 s1 0xfffffff9 | x12 a2 0x00000016  x9 s1 0xfffffff9 |
| 134 |  | xor x14 x14 x18 | 01274733 | x18 s2 0x00000006  x14 a4 0x00000009 | x18 s2 0x00000006  x14 a4 0x0000000f |
| 138 |  | add x13 x13 x8 | 008686b3 | x8 s0 x00000016  x13 a3 0x0000163f | x8 s0 x00000016  x13 a3 0x00001655 |
| 13c |  | and x12 x12 x14 | 00e67633 | x12 a2 0x00000016  x14 a4 0x0000000f | x12 a2 0x00000006  x14 a4 0x0000000f |
| 140 |  | or x11 x13 x12 | 00c6e5b3 | x11 a1 0x00000000  x12 a2 0x00000006  x13 a3 0x00001655 | x11 a1 0x00001657  x12 a2 0x00000006  x13 a3 0x00001655 |
| 144 |  | xor x15 x15 x9 | 0097c7b3 | x15 a5 0xffffff7c  x9 s1 0xfffffff9 | x15 a5 0x00000085  x9 s1 0xfffffff9 |
| 148 |  | xor x17 x17 x18 | 0128c8b3 | x17 a7 0x00000136 x18 s2 0x00000006 | x17 a7 0x00000130 x18 s2 0x00000006 |
| 14c |  | add x16 x16 x8 | 00880833 | x8 s0 0x00000016  x16 a6 0xfffffff5 | x8 s0 0x00000016  x16 a6 0x0000000b |
| 150 |  | and x15 x15 x17 | 0117f7b3 | x15 a5 0x00000085  x17 a7 0x00000130 | x15 a5 0x00000000  x17 a7 0x00000130 |
| 154 |  | or x12 x16 x15 | 0107e633 | x12 a2 0x00000006  x15 a5 0x00000000  x16 a6 0x0000000b | x12 a2 0x0000000b  x15 a5 0x00000000  x16 a6 0x0000000b |
| 158 | ret | jalr x0 x1 0 | 00008067 | x1 ra 0x000000ec | x1 ra 0x000000ec |
| ec | la a0, results | auipc x10 0x10000 | 10000517 | x10 a0 0x0000000a | x10 a0 0x100000ec |
| f0 |  | addi x10 x10 -148 | f6c50513 | x10 a0 0x100000ec | x10 a0 0x10000058 |
| f4 |  | addi x17 x0 4 | 00400893 | x17 a7 0x00000002 | x17 a7 0x00000004 |
| f8 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| fc | mv a0, a1 | addi x10 x11 0 | 00058513 | x10 a0 0x10000058  x11 a1 0x00001616 | x10 a0 0x00001616  x11 a1 0x00001616 |
| 100 |  | addi x17 x0 1 | 00100893 | x17 a7 0x00000004 | x17 a7 0x00000001 |
| 104 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| 108 | li a0, 10 | addi x10 x0 10 | 00a00513 | x10 a0 0x00001616 | x10 a0 0x0000000a |
| 10c |  | addi x17 x0 11 | 00b00893 | x17 a7 0x00000001 | x17 a7 0x0000000b |
| 110 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| 114 | mv a0, a2 | addi x10 x12 0 | 00060513 | x10 a0 0x0000000a  x12 a2 0xffffff57 | x10 a0 0xffffff57  x12 a2 0xffffff57 |
| 118 |  | addi x17 x0 1 | 00100893 | x17 a7 0x0000000b | x17 a7 0x00000001 |
| 11c |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |
| 120 |  | addi x10 x0 0 | 00100513 | x10 a0 0xffffff57 | x10 a0 0x00000000 |
| 124 |  | addi x17 x0 93 | 05d00893 | x17 a7 0x00000001 | x17 a7 0x0000005d |
| 128 |  | ecall | 00000073 | x2 sp 7ffffff0 | x2 sp 7ffffff0 |

Таблица 2 – результаты тестирования

|  |  |
| --- | --- |
| Входные данные | Результаты |
| x1 = 10 | Formula: ((x ^ (-b)) & (z ^ c)) | (y + a) |
| y1 = 101 | Input data: |
| z1 = 11 | {x1, y1, z1} = 10 101 11 |
| x2 = 576 | {x2, y2, z2} = 576 89 3398 |
| y2 = 89 | Results: |
| z2 = 3398 | 123 |
|  | 3439 |
| x1 = 3210 | Formula: ((x ^ (-b)) & (z ^ c)) | (y + a) |
| y1 = 5747 | Input data: |
| z1 = -84 | {x1, y1, z1} = 3210 5747 -84 |
| x2 = -90423 | {x2, y2, z2} = -90423 85 308 |
| y2 = 85 | Results: |
| z2 = 308 | -2133 |
|  | 379 |
| x1 = 12 | Formula: ((x ^ (-b)) & (z ^ c)) | (y + a) |
| y1 = 13 | Input data: |
| z1 = -14 | {x1, y1, z1} = 12 13 -14 |
| x2 = 24 | {x2, y2, z2} = 24 28 30 |
| y2 = 28 | Results: |
| z2 = 30 | -9 |
|  | 50 |

**Выводы.**

В ходе выполнения лабораторной работы установлена, настроена и изучена среда эмулятора Ripes. Получены знания об архитектуре RISC-V, базовом наборе инструкций, псевдо-инструкциях и регистрах. Разработана простая программа на ассемблере.

**ПРИЛОЖЕНИЕ А.**

**ИСХОДНЫЙ КОД ПРОГРАММЫ.**

Название файла lb5.s:

.text # раздел с инструкциями программы

.global \_start

\_start:

.equ a, 22 # задаются константы

.equ b, 7

.equ c, 6

addi s0, x0, a # константы a, b, c размещаются

addi s1, x0, b # в регистры s0, s1

addi s2, x0, c # и s2 соответственно

li a2, 10 # размещение переменных

li a3, 101 # x1, y1, z1 в регистры

li a4, 11 # a2, a3, a4 соответственно

li a5, 576 # размещение переменных

li a6, 89 # x2, y2, z2 в регистры

li a7, 3398 # a5, a6, a7 соответственно

mv s4, a7 # сохранение значения из а7

la a0, formula # вывод строки по адресу в а0

addi a7, x0, 4 # с помощью системного вызова PrintString

ecall # (a7=4)

mv a0, a2 # перенос числа из а2 в а0

addi a7, x0, 1 # вывод его в 10-чной сс системным вызовом

ecall # PrintInt (a7=1)

li a0, 32 # в а0 ascii код пробела

addi a7, x0, 11 # вывод символа системным вызовом

ecall # PrintChar (a7=11)

mv a0, a3 # вывод числа из а3

addi a7, x0, 1

ecall

li a0, 32

addi a7, x0, 11

ecall

mv a0, a4 # вывод числа из а4

addi a7, x0, 1

ecall

li a0, 10 # вывод символа переноса строки

addi a7, x0, 11

ecall

la a0, data # вывод строки

addi a7, x0, 4

ecall

mv a0, a5 # вывод числа из а5

addi a7, x0, 1

ecall

li a0, 32

addi a7, x0, 11

ecall

mv a0, a6 # вывод числа из а6

addi a7, x0, 1

ecall

li a0, 32

addi a7, x0, 11

ecall

mv a7, s4 # восстановление значения в а7

mv a0, a7 # вывод числа из а7

addi a7, x0, 1

ecall

li a0, 10

addi a7, x0, 11

ecall

mv a7, s4

call calc\_expression # вызов процедуры calc\_expression

la a0, results # вывод строки

addi a7, x0, 4

ecall

mv a0, a1 # вывод числа из а1

addi a7, x0, 1

ecall

li a0, 10

addi a7, x0, 11

ecall

mv a0, a2 # вывод числа из а2

addi a7, x0, 1

ecall

addi a0, x0, 0 # завершение программы системным вызовом

addi a7, x0, 93 # Exit (a7=93) с кодом возврата в регистре а0

ecall

calc\_expression:

neg s1, s1 # теперь в s1 находится (-b)

#вычисление выражения для (x1, y1, z1)

xor a2, a2, s1 # теперь в a2 находится (x1 ^ (-b))

xor a4, a4, s2 # теперь в a4 находится (z1 ^ c)

add a3, a3, s0 # теперь в a3 находится (y1 + a)

and a2, a2, a4 # теперь в a2 находится (x1 ^ (-b)) & (z1 ^ c)

or a1, a3, a2 # теперь в a1 находится ((x1 ^ (-b)) & (z1 ^ c)) | (y1 + a)

#вычисление выражения для (x2, y2, z2)

xor a5, a5, s1 # теперь в a5 находится (x2 ^ (-b))

xor a7, a7, s2 # теперь в a7 находится (z2 ^ c)

add a6, a6, s0 # теперь в a6 находится (y2 + a)

and a5, a5, a7 # теперь в a5 находится (x2 ^ (-b)) & (z2 ^ c)

or a2, a5, a6 # теперь в a2 находится ((x2 ^ (-b)) & (z2 ^ c)) | (y2 + a)

ret # возврат из процедуры

.data # раздел с данными для переменных программы

formula: .asciz "Formula: ((x ^ (-b)) & (z ^ c)) | (y + a)\nInput data:\n{x1, y1, z1} = "

data: .asciz "{x2, y2, z2} = "

results: .asciz "Results:\n"