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. 2384 _____ Кузьминых Е.М.

Преподаватель _____ Морозов С.М.

Санкт-Петербург

2023

Цель работы

1. Изучение архитектуры RISC-V, базового набора инструкций и разработка простых программ на ассемблере. В ходе выполнения лабораторной работы написать программу, вычисляющая выражение для 3 переменных и 3 констант.

Задание

Вариант №14

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

$$((x \;\&;\; (-c)) \mid (z - a)) - (y + b)$$

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 $\{x_1,\ y_1,\ z_1\}$ расположить в регистрах *a2*, *a3*, *a4*; значения $\{x_2, y_2, z_2\}$ расположить в регистрах *a5*, *a6*, *a7*; значения констант a, b, c расположить в регистрах *s0*, *s1*, *s2*. Результаты вычисления $\{r_1, r_2\}$ записать в регистры *a1*, *a2*.

   В исходном коде обязательно должны быть употреблены следующие псевдоинструкции: call (ровно 1 раз), ret (ровно 1 раз), mv (как минимум 1 раз), li (как минимум 2 раза: 1 раз – преобразующаяся в две инструкции; 1 раз – преобразующаяся в одну инструкцию).

   Моделируемые вычисления (формула, входные данные, результаты) должны выводиться в консоль.

Основные теоретические положения

В архитектуре RISC-V имеется обязательное для реализации базовое подмножество в количестве 47 команд и несколько стандартных опциональных расширений. В базовый набор входят: минимальный набор команд арифметических/битовых операций на регистрах, команд для выполнения операций с памятью (load/store), команд условной и безусловной передачи управления/ветвления, а также небольшое число служебных инструкций (см. таблицу далее). Команды базового набора имеют длину 32 бита с выравниванием на границу 32-битного слова.

Операции условных переходов (ветвления) не используют каких-либо общих флагов, как результатов ранее выполненных операций сравнения, а непосредственно сравнивают свои регистровые операнды. Базис операций сравнения минимален, а для поддержки комплементарных операций операнды просто меняются местами.

В табл. 1 указан список регистров процессора RISC-V, их назначение и наименование.

Таблица 1. Регистры процессора RISC-V

Регистр(ы)	Наименование	Описание назначения
x0	*zero*	Константа 0
x1	*ra*	Адрес возврата
x2	*sp*	Указатель стека
x3	*gp*	Глобальный указатель
x4	*tp*	Указатель потока
x5-x7	*t0-t2*	Временные регистры
x8	*s0/fp*	Сохраняемый регистр/указатель фрейма
x9	*s1*	Сохраняемый регистр
x10-x11	*a0-a1*	Аргументы функции/возвращаемые значения
x12-x17	*a2-a7*	Аргументы функции
x18-x27	*s2-s11*	Сохраняемые регистры
x28-x31	*t3-t6*	Временные регистры

Вся арифметико-логическая обработка данных может производиться только над регистрами, при этом в основном формате регистр-регистр используются два регистра-источника операндов Rsrc1 и Rsrc2 (далее *rs1* и *rs2*)

и регистр результата Rdest (далее *rd*). Для использования ячеек основной памяти применяются инструкции загрузки (Load) данных из ячеек памяти в РОНы и выгрузки (Store) данных из РОН в ячейки памяти.

Архитектура использует только little-endian модель — первый байт операнда в памяти соответствует наименее значащим битам значений регистрового операнда (аналогично архитектуре x86).

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

1. Раздел TEXT (доступен для чтения, также известен как сегмент кода);
2. Раздел DATA (доступен для чтения и записи, содержит инициализированные статические переменные);
3. Раздел BSS (базовый служебный набор, доступен для чтения и записи, содержит неинициализированные данные).

Директивы для определения и экспорта символов:

1. GLOBAL (определяет символ глобальным);
2. LOCAL (ограничивает видимость символов);
3. EQU (задаёт значение символа в выражении).

Директивы ассемблера для задания данных:

1. HALF (инициализирует 16-разрядные выровненные целые числа);
2. WORD (задаёт естественно выровненные 32-битные слова);
3. DWORD (задаёт естественно выровненные 64-битные слова);
4. BYTE (служит для задания не выровненных 8-битных слов).

Директивы могут задавать несколько значений, разделённых запятыми. Указанные операнды могут быть десятичными, шестнадцатеричными, двоичными или символьными константами, но не метками.

Директивы задания строк:

1. ASCIZ (подобно директиве ASCII задаёт строку в пределах двойных кавычек, за каждой строкой следует нулевой байт);
2. STRING (задаёт строку в пределах двойных кавычек, за каждой строкой следует нулевой байт).

Многие команды программ на ассемблере RISC-V не используют три аргумента, так как являются псевдоинструкциями. Это означает, что они являются сокращениями для других инструкций.

Выполнение работы

1. Разработана процедура, которая вычисляет следующее выражение

$$R = ((\mathrm{x}\ \&\ (\text{-}\mathrm{c}))\ |\ (\mathrm{z}\ \text{-}\ \mathrm{a}))\ \text{-}\ (\mathrm{y} + \mathrm{b})$$

2. Определение констант: Программа начинается с определения констант a, b, c, x_1, y_1, z_1, x_2, y_2 и z_2 с помощью директивы equ. Эти константы затем используются в вашем коде для выполнения вычислений.
3. Определение строк: Далее программа определяет ряд строк, которые будут использоваться для вывода на экран. Эти строки включают в себя формулу функции, которая будет вычислена, а также имена и значения переменных.
4. Вывод информации на экран: В начале программы программа выводит на экран формулу функции и значения констант. Затем программа выводит значения переменных x_1, y_1, z_1, x_2, y_2 и z_2. Это делается с помощью процедуры print, которая принимает адрес строки и значение, которое нужно вывести, и выводит их на экран.
5. Вычисление функции: После вывода всех значений переменных программа вызывает процедуру calc, которая выполняет вычисления по формуле R = ((x & (-c)) | (z - a)) - (y + b). Результаты вычислений сохраняются в регистрах a1 и a2.
6. Вывод результатов: После вычисления функции программа выводит на экран результаты вычислений. Это делается с помощью процедуры print,

которая выводит строку "Answer: {r1, r2} = " и затем значения из регистров a1 и a2.

7. Выход из программы: В конце программы программа выходит из программы с помощью системного вызова ecall с аргументом 10.
8. Процедура print работает следующим образом: значение регистра a7 определяет тип системного вызова (вывод числа, строки). Сначала выводится строка, помещается аргумент 4, затем значение регистра a1, где храниться число, копируется в a0 (псевдоинструкция mv). и выводится в консоль, помещается в регистр a7 аргумент 1. Далее печатается Enter через помещение в a7 1.

Процедура calc делает вычисления для двух наборов данных и сохраняет результаты в регистрах a1 и a2.

Разработанный программный код см. в приложении А.

Таблица 2 – Протокол работы программы в отладочном режиме.

Адрес инстр.	Псевдоинстр.	Инструкция(и)	16-ричный код инстр.	Содержимое регистров и ячеек памяти	
				до вып. инстр.	после вып.
0	j start	jal x0 44 <start>	02c0006f	x0 = 0x00000000	x0=0x00000000
2c	la a0, my_func	auipc x10 0x10000	10000517	x10 =0x00000000	x10=0x1000002c
30		addi x10 x10 -44	fd450513	x10=0x1000002c	x10=0x10000000
34	li a7, 4	addi x17 x0 4	00400893	x17=0x00000000	x17=0x00000004
38	ecall	ecall	00000073		
3c	la a0, const_values	auipc x10 0x10000	10000517	x10=0x10000000	x10=0x1000003c
40		addi x10 x10 -20	fec50513	x10=0x1000003c	x10=0x10000026
44	li a7, 4	addi x17 x0 4	00400893	x17 = 0x00000004	x17 =0x00000004

48	ecall	ecall	00000073		
4c	la a0, str_x1	auipc x10 0x10000	10000517	x10=0x10000026	x10=0x1000004c
50		addi x10 x10 5	00550513	x10=0x1000004c	x10=0x1000004f
54	la a1, x_1	auipc x11 0x0	00000597	x11=0x00000000	x11=0x00000054
58		addi x11 x11 916	39458593	x11=0x00000054	x11=0x00000001
5c	jal print	jal x1 -88 <print>	fa9ff0ef	x1=0x00000000	x1=0x00000060
4	li a7, 4	addi x17 x0 4	00400893	x17=0x00000004	x17=0x00000004
8	ecall	ecall	00000073		
c	mv a0, a1	addi x10 x11 0	00058513	x10=0x1000004f x11=0x00000001	x10=0x00000001 x11=0x00000001
10	li a7 1	addi x17 x0 1	00100893	x17=0x00000004	x17=0x00000001
14	ecall	ecall	00000073		
18	li a7, 4	addi x17 x0 4	00400893	x17=0x00000001	x17=0x00000004
1c	la a0, line_end	auipc x10 0x10000	10000517	x10=0x00000001	x10=0x1000001c
20		addi x10 x10 99	06350513	x10=0x1000001c	x10=0x10000073
24	ecall	ecall	00000073		
28	ret	jalr x0 x1 0	00008067	x0=0x00000000 x1=0x00000060	x0=0x00000000 x1=0x00000060
c4	li a2, x_1	addi x12 x0 10	00a00613	x12=0x00000000	x12=0x00000001
c8	li a3, y_1	addi x13 x0 73	04900693	x13=0x00000000	x13=0x00000006
cc	li a4, z_1	addi x14 x0 -6	ffa00713	x14=0x00000000	x14=0xfffffffd
d0	li a5, x_2	addi x15 x0 -14	ff200793	x15=0x00000000	x15=0x00000008
d4	li a6, y_2	addi x16 x0 55	03700813	x16=0x00000000	x16=0x00000007
d8	li a7, z_2	addi x17 x0 64	04000893	x17=0x00000004	x17=0x00000005
dc	call calc	auipc x1 0x0	00000097	x1=0x000000c4	x1=0x000000dc
e0		jalr x1 x1 88	058080e7	x1=0x000000dc	x1=0x000000e4
134	addi s0, zero, a	addi x8 x0 22	01600413	x8=0x00000000	x8=0x00000017
138	addi s1, zero, b	addi x9 x0 7	00700493	x9=0x00000000	x9=0x00000009
13c	addi s2, zero, c	addi x18 x0 5	00500913	x18=0x00000000	x18=0x00000004
140	neg s2, s2	sub x18 x0	41200933	X18=0x00000004	X18=0xfffffffc

144	and s3, a2, s2	and x19 x12 x18	012679b3	x19=0x00000000 x12=0x00000001 x18=0xfffffffc	x19=0x00000000
148	sub s4, a4, s0	sub x20 x14 x8	40870a33	x20=0x00000000 x14=0xfffffffd x8=0x00000017	x20=0xffffffe6
14c	or s5, s3, s	or x21 x19 x20	0149eab3	x21=0x00000000 x19=0xffffffe6 x20=0x00000000	x21=0x00000008
150	add s6, a3, s1	add x22 x13 x9	00968b33	x22=0x00000000 x13=0x00000006 x9=0x00000009	x22=0x00000000
154	sub a1, s5, s6	sub x11 x21 x22	416a85b3	x11=0x00000005 x21=0xffffffe6 x22=0xffffffe6	x11=0x00000005
16c	jr ra	jalr x0 x1 0	00008067	x0=0x00000000	x0=0x00000000
e4	la a0, result	auipc x10 0x10000	10000517	x10=0x10000073	x10=0x100000e4
e8		addi x10 x10 -99	f9d50513	x10=0x100000e4	x10=0x10000075
ec	li a7, 4	addi x17 x0 4	00400893	x17=0x00000005	x17=0x00000004
f0	ecall	ecall	00000073		
f4	li a7 1	addi x17 x0 1	00100893	x17=0x00000004	x17=0x00000001
f8	mv a0, a2	addi x10 x12 0	00060513	x10=0x10000075	x10=0xffffffde
fc	ecall	ecall	00000073		
100	li a7, 4	addi x17 x0 4	00400893	x17=0x00000001	x17=0x00000004
104	la a0, separator	auipc x10 0x10000	10000517	x10=0xffffffd7	x10=0x10000104
108		addi x10 x10 -111	f9150513	x10=0x10000104	x10=0x1000008b
10c	ecall	ecall	00000073		
110	li a7 1	addi x17 x0 1	00100893	x17=0x00000004	x17=0x00000001
114	mv a0, a2	addi x10 x12 0	00060513	x10=0x10000089 x12=0xffffffde	x10=0xffffffde x12=0xffffffde
118	ecall	ecall	00000073		
11c	li a7, 4	addi x17 x0 4	00400893	x17=0x00000001	x17=0x00000004
120	la a0, line_end	auipc x10 0x10000	10000517	x10=0xffffffde	x10=0x10000120
124		addi x10 x10 -161	f5f50513	x10=0x10000120	x10=0x10000073

128	ecall	ecall	00000073		
12c	li a7, 10	addi x17 x0 10	00a00893	x17=0x00000004	x17=0x0000000a
130	ecall	ecall	00000073		

Тестирование

Результаты тестирования представлены в табл. 3.

Таблица 3 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1,6,-3	-41	Верная работа при наличии отрицательных чисел
2.	8,7,5	-34	Верное выполнение при положительных числах
3.	-5,-5,-5	-8	Верное выполнение при всех отрицательных одинаковых числах
4.	10000,-1000,0	984	Верное выполнение при больших числах, положительном, отрицательном и нуле.

Выводы

Была изучена архитектура RISC-V, основные наборы инструкций для работы с ним. В ходе выполнения лабораторной работы была написана программа, высчитывающая значение функции, выводящая значение на экран.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb5.s

```
.equ a 23 # 2+3+8+4+1+5
.equ b 9 # Кузьминых
.equ c 4 # Егор
.equ x_1 1
.equ y_1 6
.equ z_1 -3
.equ x_2 8
.equ y_2 7
.equ z_2 5

.data
  my_func: .string "R = ((x & (-c)) | (z - a)) - (y + b)\n"
  const_values: .string "consts: a = 23, b = 9, c = 4\n"

  newline: .string "\n"
  str_x1: .string "x1 = "
  str_x2: .string "x2 = "
  str_y1: .string "y1 = "
  str_y2: .string "y2 = "
  str_z1: .string "z1 = "
  str_z2: .string "z2 = "
  line_end: .string "\n"
  result: .string "Result: r1, r2 = "
  separator: .string ", "

.text
    j start

    print:

      li a7, 4
```

```
ecall

mv a0, a1
li a7 1
ecall

li a7, 4
la a0, line_end
ecall
ret

start:


la a0, my_func
li a7, 4
ecall

la a0, const_values
li a7, 4
ecall

#печать цифр
la a0, str_x1
la a1, x_1
jal print

la a0, str_y1
la a1, y_1
jal print

la a0, str_z1
la a1, z_1
jal print

la a0, str_x2
```

```
la a1, x_2
jal print

la a0, str_y2
la a1, y_2
jal print

la a0, str_z2
la a1, z_2
jal print


li a2, x_1
li a3, y_1
li a4, z_1
li a5, x_2
li a6, y_2
li a7, z_2

call calc


la a0, result
li a7, 4
ecall

li a7, 1
mv a0, a1
ecall

li a7, 4
la a0, separator
ecall

li a7, 1
mv a0, a2
```

```
    ecall

    li a7, 4
    la a0, line_end
    ecall


    li a7, 10
    ecall

calc: #вычисление значений

 addi s0, zero, a
 addi s1, zero, b
 addi s2, zero, c
 neg s2, s2

 and s3, a2, s2
 sub s4, a4, s0
 or s5, s3, s4
 add s6, a3, s1
 sub a1, s5, s6

 and s3, a5, s2
 sub s4, a7, s0
 or s5, s3, s4
 add s6, a6, s1
 sub a2, s5, s6

 ret
```