Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчёт по лабораторной работе № 2

Дисциплина: Низкоуровневое программирование

Тема: RISC-V

Выполнил студент гр. 3530901/10001 _____Солодовник И.Н.
(подпись)

Принял старший преподаватель _____ Коренев Д.А.
(подпись)

“11” декабря 2022 г.

Санкт-Петербург
2022

Оглавление

Описание задачи: 2

Алгоритм выполнения: 2

Реализация программы 2

Программа считывания коротких чисел: 2

Внешний цикл: 3

Внутренний цикл: 4

Результат работы программы: 4

Реализация подпрограммы 4

Main и setup: 4

Подпрограмма считывания коротких чисел: 5

Отличия программы от подпрограммы: 6

Вывод 6

Описание задачи:

Разработать программу реализующую загрузку последовательности «коротких» чисел. Числа кодируются обычным для EDSAC образом (в виде псевдоинструкций).

(Вариант 10)

Алгоритм выполнения:

Программа считывает первый символ, сравнивает и записывает его в 4 старших разряда результата и запоминает знак. Затем записываются цифры, сдвигаются влево на 1 разряд, а в младший разряд записывается символ S или L. После этого программа проверяет знак и переходит к следующему значению.

Реализация программы

Программа считывания коротких чисел:

```
.text
__start:
.globl __start # начало программы
  la a2, edsac_symbols # auipc + addi указатель на символ для сравнения
  la a3, cmd # указатель на сравниваемый символ
  la a5, result # указатель на результат
# считывание первого символа инструкции
preloop:
  add t0, a2, zero # копируем указатель, чтобы сократить обращения в память
  li t3, 0 # итератор
  lb t2, 0(a3) # считывание символа

first_char:
  lb t1, 0(t0) # считывание символа в порядке для edsac
  beq t1, t2, sign # сравниение символа из edsac_symbols и cmd
  addi t0, t0, 1 # смещение адресса на следующий символ(слово)
  addi t3, t3, 1 # i++
  jal first_char # цикл, пока не символы не совпадут
# запоминание знака
sign:
  li a6, 0 # запоминаем знак
  li t0, 16 # константа для нахождения знака
  bltu t3, t0, preres # если i < 16 -> сдвигаем символ на 12 бит
  addi t3, t3, -16 # избавляемся от старшего разряда
  li a6, 1 # запоминаем знак
# сдвиг
preres:
  slli t3, t3, 12 # сдвиг на 12 бит
  addi a7, t3, 0 # записываем в результат
  li t1, 0 # обнуляем t1
# обработка цифр
num:
  addi a3, a3, 1 # смещение адресса вводимой инструкции
  lb t0, 0(a3) # считывание символа(слова)
  addi t0, t0, -48 # переводим цифры из ascii кода
  li t3, 10
  bgeu t0, t3, cmd_end # если t0 > 10 переходим к проверке конца инструкции
  mul t1, t1, t3 # умножение промежуточного результата на 10
  add t1, t1, t0 # добавление цифры к промежуточному результату
  jal num # цикл, пока цифры не кончатся

cmd_end:
  slli t1, t1, 1 # смещение промежуточного результата на 1
  add a7, a7, t1 # запись в результат
```

```
45   addi t0, t0, -35 # проверка на S или L
46   beqz t0, negative # переход к обработке знака
47   addi a7, a7, 1 # L
48
49 negative:
50   beqz a6, fin_res # переход к окончательной записи результата
51   li t6, 65536
52   sub a7, a7, t6
53 # запись результата
54 fin_res:
55
56   add a1, a7, zero
57   li a0, 1
58   ecall
59   li a1, ' '
60   li a0, 11
61   ecall # вывод с консоль
62
63   sw a7, 0(a5) # запись в память
64   addi a5, a5, 4 # смещение адресса результата
65   addi a3, a3, 1 # смещение адресса вводимой инструкции
66   lb t0, 0(a3)
67   beqz t0, stop
68   addi a3, a3, 1 # пропуск пробела
69   j preloop # переход к следующей псевдоинструкции
70
71 stop:
72   li a0, 10
73   ecall # конец программы
74
75
76 .rodata # неизменяемая секция данных
77 edsac_symbols:
78   .string "PQWERTYUIOJ#SZK*.F@D!HNM&LXGABCV"
79 cmd:
80   .string "P0S .0S W43L M333L P0L *2047L"
81 .bss # выходной буфер
82 result:
83   .zero 512
```

Внешний цикл:

Внешний цикл отвечает за сборку инструкции почестям и за определение знака.

1. Сначала устанавливаются указатели и считывается первый символ.
2. Символ сравнивается с другими символами из строки, которая показывает в каком порядке символы кодируются в edsac. На каждой итерации цикла увеличивается специально выделенный итератор.
3. Как только символы совпадают, от итератора отнимается старший разряд и он сдвигается влево на 12 бит. При этом запоминается знак конечного результата.
4. Внутренний цикл.
5. Запись результата внутреннего цикла со сдвигом влево на 1 бит.
6. Затем программа проверяет какой символ стоит в конце псевдоинструкции и либо добавляет 1, либо переходит к следующему пункту.

7. Запись результата в память с выводом на консоль для проверки.
8. Переход к следующей псевдоинструкции или завершение программы.

Внутренний цикл:

Внутренний цикл отвечает со определение цифр в центре псевдоинструкции:

1. Смещается указатель и загружается необходимый байт.
2. Цифры переводятся из ASCII кода.
3. Если же символ меньше 10, то умножаем промежуточный результат на 10 и добавляем цифру.
4. Если символ больше 10, то проверяется конец псевдоинструкции.

Результат работы программы:

При корректных значениях P0S .0S W43L M333L P0L *2047L программа выдаёт следующий результат:

```
Console
0 -65536 8279 -36197 1 65535
```

```
Order = P 0 S  Integer 110S = 0
Order = . 0 S  Integer 109S = -65536
Order = W 43 L  Integer 108S = 8279
Order = M 333 L  Integer 107S = -36197
Order = P 0 L  Integer 106S = 1
Order = * 1023 L  Integer 106S = 65535
```

Реализация подпрограммы

Main и setup:

```
.text
main:
.globl main
  addi sp, sp, -16
  sw ra, 12(sp)

  la a3, cmd # указатель на сравниваемый символ
  la a5, result # указатель на результат
  call inst_reader

  la a3, cmd # указатель на сравниваемый символ
  la a5, result_2 # указатель на результат
  call inst_reader

  lw ra, 12(sp)
  addi sp, sp, 16

  ret # jalr zero, ra, 0

.rodata
cmd:
  .string "P0S .0S W43L M333L P0L *2047L"
.bss # выходной буфер
result:
  .zero 512
result_2:
  .zero 512
```

```
.text
__start:
.globl __start
  call main
finish:
  li a0, 10
  ecall
```

Подпрограмма считывания коротких чисел:

```
.text
inst_reader:
.globl inst_reader # начало программы
  la a2, edsac_symbols # auipc + addi указатель на символ для сравнения

# считывание первого символа инструкции
preloop:
  add t0, a2, zero # копируем указатель, чтобы сократить обращения в память
  li t3, 0 # итератор
  lb t2, 0(a3) # считывание символа

first_char:
  lb t1, 0(t0) # считывание символа в порядке для edsac
  beq t1, t2, sign # сравниение символа из edsac_symbols и cmd
  addi t0, t0, 1 # смещение адресса на следующий символ(слово)
  addi t3, t3, 1 # i++
  j first_char # цикл, пока не символы не совпадут
# запоминание знака
sign:
  li a6, 0 # запоминаем знак
  li t0, 16 # константа для нахождения знака
  bltu t3, t0, preres # если i < 16 -> сдвигаем символ на 12 бит
  addi t3, t3, -16 # избавляемся от старшего разряда
  li a6, 1 # запоминаем знак
# сдвиг
preres:
  slli t3, t3, 12 # сдвиг на 12 бит
  addi a7, t3, 0 # записываем в результат
  li t1, 0 # обнуляем t1
# обработка цифр
num:
  addi a3, a3, 1 # смещение адресса вводимой инструкции
  lb t0, 0(a3) # считывание символа(слова)
  addi t0, t0, -48 # переводим цифры из ascii кода
  li t3, 10
  bgeu t0, t3, cmd_end # если t0 > 10 переходим к проверке конца инструкции
  mul t1, t1, t3 # умножение промежуточного результата на 10
  add t1, t1, t0 # добавление цифры к промежуточному результату
  j num # цикл, пока цифры не кончатся

cmd_end:
  slli t1, t1, 1 # смещение промежуточного результата на 1
  add a7, a7, t1 # запись в результат
  addi t0, t0, -35 # проверка на S или L
```

```
45   beqz t0, negative # переход к обработке знака
46   addi a7, a7, 1 # L
47
48 negative:
49   beqz a6, fin_res # переход к окончательной записи результата
50   li t6, 65536
51   sub a7, a7, t6
52 # запись результата
53 fin_res:
54
55   add a1, a7, zero
56   li a0, 1
57   ecall
58   li a1, ' '
59   li a0, 11
60   ecall # вывод с консоль
61
62   sw a7, 0(a5) # запись в память
63   addi a5, a5, 4 # смещение адресса результата
64   addi a3, a3, 1 # смещение адресса вводимой инструкции
65   lb t0, 0(a3)
66   beqz t0, stop
67   addi a3, a3, 1 # пропуск пробела
68   j preloop # переход к следующей псевдоинструкции
69
70 stop:
71   ret # конец подпрограммы
72
73
74 .rodata # неизменяемая секция данных
75 edsac_symbols:
76   .string "PQWERTYUIOJ#SZK*.F@D!HNM&LXGABCV"
```

Отличия программы от подпрограммы:

Алгоритм выполнения задачи практически не изменился. First был выделен в подпрограмму inst_reader, которая вызывается из подпрограммы main. Main же вызывается из Setup’a.
Основные отличия:

- Несколько файлов необходимых для работы.
- Появление псевдоинструкции ret.
- .rodata cmd: и .bss вынесены в main, также как и установка указателей на память.
- Необходимость добавления регистра ra в стек, для устранения зацикливания подпрограммы.

Вывод

В ходе выполнения лабораторной работы была разработана программа на языке ассемблера RISC-V, реализующая загрузку коротких чисел, кодирующихся привычным для EDSAC образом.