**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

**отчет**

**по лабораторной работе №5**

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

**Тема: Взаимодействие с внешними компонентами.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3388 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Снигирёв А.А |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Молодцев Д.А |

Санкт-Петербург

2024

**Цель работы**

Освоить программирование на ассемблере RISC-V взаимодействия с внешними устройствами (ввода и вывода).

**Задание:**

Для выполнения работы требуется реализовать ряд комбинационных функций на ассемблере. Каждое вычисленное значение функции определяет состояние светодиода: 0 – не горит, 1 – горит. Значение переменных определяется переключателями: 0 – выкл., 1 – вкл.

Функции определяются по таблице истинности:

|  |  |
| --- | --- |
| 19 | 4,2,5 |

В задании требуется реализовать 3 функции для зажигания светодиодов. Порядок функций соответствует порядку светодиодов. Первый загорается красным, второй – зелёным, третий – синим. Также есть четвёртый светодиод, цвет которого – смесь всех цветов трёх других светодиодов.

**Основные теоретические положения:**

Эмулятор Ripes имеет возможность эмуляции внешних устройств по интерфейсу GPIO. Обеспечивается работа со следующими электронными компонентами:

* Светодиодные матрицы (LED Matrix)
* Клавиатура со стрелками (D-Pad)
* Переключатели (Switches)

Клавиатура и переключатели используются как устройства ввода, а светодиоды – как устройство вывода.

Для созданных компонентов определяются адреса в памяти процессора, по которым осуществляется взаимодействие с компонентами. Адреса указаны в виде констант для языка С, который можно скопировать для программ.

Так, для того, чтобы зажечь светодиод нужно определить его адрес и записать по нему значение цвета в формате RGB.

Чтобы определить состояние свитча, нужно определить его адрес и считать по нему значение.

**Выполнение работы**

1. Исходное задание было приведено ранее.

2. В регистры a7 и a6 сохраняются константы, соответствующие адресам матрицы диодов и переключателя.

3. Метка loop*.*  Начало итерации цикла. Обнуляет регистры s0,s1,s2,s3, которые будут хранить цвета соответствующих диодов.

Далее в a5 сохраняется адрес переключателя. После этого сохраняются в временные регистры состояния каждого отдельного переключателя, затем с помощью сдвигов они приводятся к булеву типу.

В регистры a0-a3 сохраняются обратные значения t0-t3, чтобы сократить число строк при расчете функций.

4. Diod\_1*.* Вызывает функцию Y4 и если ее результат не равен 0, загружает в регистр s0 значение 4080, соответствующее красному цвету.

Иначе сразу совершает переход на Diod\_2.

5. Diod\_2*.* Вызывает функцию Y2 и если ее результат не равен 0, загружает в регистр s1 значение 65280, соответствующее зеленому цвету.

Иначе сразу совершает переход на Diod\_3.

6. Diod\_3*.* Вызывает функцию Y5 и если ее результат не равен 0, загружает в регистр s2 значение 255, соответствующее синему цвету.

Иначе сразу совершает переход на Diod\_4.

7. Diod\_4. Считает среднее арифметическое трех компонент и загружает в регистр s3.

8. Install. Командами sw загружает в память по базе a7 и различным смещениям значения регистров s0-s3. После этого можно наблюдать свечение диодов различными цветами. Переходит на следующую итерацию.

9. Процедуры function\_2, function\_4, function\_5 расчитывают логическое выражение в соответствии с заданием и сохраняет результаты в регистры s4-s6.

Исходный код программы см. в **ПРИЛОЖЕНИИ А .** Тестирование см. в **ПРИЛОЖЕНИИ Б**.

**Вывод**

В результате выполнения лабораторной работы было реализовано простое взаимодействие с внешними устройствами ввода и вывода на языке ассемблера Risc-V.

**ПРИЛОЖЕНИЕ А**

**Код программы**

li a7 LED\_MATRIX\_0\_BASE

li a6 SWITCHES\_0\_BASE

# s0 - color[0]

# s1 - color[1]

# s2 - color[2]

# s3 - color[3]

# t0 - switch[0]

# t1 - switch[1]

# t2 - switch[2]

# t3 - switch[3]

# a5 - слово состояния ключей

loop: # бесконечный цикл

lw a5, 0[a6] # чтение состояния переключателя

mv s0, x0

mv s1, x0

mv s2, x0

mv s3, x0 # обнуление цветов

andi t0,a5,1

andi t1,a5,2

andi t2,a5,4

andi t3,a5,8 # перенос переключателей в отдельные регистры

srli t1,t1,1 # приведение к типу bool

srli t2,t2,2 #

srli t3,t3,3 #

not a0, t0 # вычисления !xi

not a1, t1

not a2, t2

not a3, t3

Diod\_1:

call function\_4

beqz s5, Diod\_2

lui s0, 4080 # Red

Diod\_2:

call function\_2

beqz s4, Diod\_3

li s1, 65280 # Green

Diod\_3:

call function\_5

beqz s6, Diod\_4

li s2, 255 # Blue

Diod\_4:

or s3, s3, s0

or s3, s3, s1

or s3, s3, s2

Install:

sw s0, 0(a7)

sw s1, 4(a7)

sw s2, 8(a7)

sw s3, 12(a7)

j loop

function\_2:

# (!x1 & !x2 & x3) | (x1 & !x2 & !x3 & !x4) | (x1 & x2 & x4) | (!x1 & !x2 & !x3) | (x2 & x3 & !x4)

mv s4, a0

and s4, s4, a1

and s4, s4, t2 # s4 = (!x1 & !x2 & x3)

mv s5, t0

and s5, s5, a1

and s5, s5, a2

and s5, s5, a3 # s5 = (x1 & !x2 & !x3 & !x4)

mv s6, t0

and s6, s6, t1

and s6, s6, t3 # s6 = (x1 & x2 & x4)

mv s7, a0

and s7, s7, a1

and s7, s7, a2 # s7 = (!x1 & !x2 & !x3)

mv s8, t1

and s8, s8, t2

and s8, s8, a3 # s8 = (x2 & x3 & !x4)

or s4, s4, s5

or s4, s4, s6

or s4, s4, s7

or s4, s4, s8

ret

function\_4:

# (!x1 & x2 & !x4) | (!x1 & x2 & x3) | (x2 & x3 & !x4) | (!x1 & x3 & !x4) | (x1 & !x2 & !x3 & x4)

mv s4, a0 # s4 = !x1

and s4, s4, t1 # s4 = !x1 & x2

and s4, s4, a3 # s4 = !x1 & x2 & !x4

mv s5, a0 # s5 = !x1

and s5, s5, t1 # s5 = !x1 & x2

and s5, s5, t2 # s5 = !x1 & x2 & x3

mv s6, t1 # s6 = x2

and s6, s6, t2 # s6 = x2 & x3

and s6, s6, a3 # s6 = x2 & x3 & !x4

mv s7, a0 # s7 = !x1

and s7, s7, t2 # s7 = !x1 & x3

and s7, s7, a3 # s7 = !x1 & x3 & !x4

mv s8, t0 # s8 = x1

and s8, s8, a1 # s8 = x1 & !x2

and s8, s8, a2 # s8 = x1 & !x2 & !x3

and s8, s8, t3 # s8 = x1 & !x2 & !x3 & x4

or s5, s4, s5 # s5 = (!x1 & x2 & !x4) | (!x1 & x2 & x3)

or s5, s5, s6 # s5 = (!x1 & x2 & !x4) | (!x1 & x2 & x3) | (x2 & x3 & !x4)

or s5, s5, s7 # s5 = (!x1 & x2 & !x4) | (!x1 & x2 & x3) | (x2 & x3 & !x4) | (!x1 & x3 & !x4)

or s5, s5, s8 # s5 = (!x1 & x2 & !x4) | (!x1 & x2 & x3) | (x2 & x3 & !x4) | (!x1 & x3 & !x4) | (x1 & !x2 & !x3 & x4)

ret

function\_5:

# x1 & x2 | x3 & x4 | !x1 & x4 | x1 & !x3 & !x4 | !x1 & !x2 & x3

mv s4, t0 # s4 = x1

and s4, s4, t1 # s4 = x1 & x2

mv s5, t2 # s5 = x3

and s5, s5, t3 # s5 = x3 & x4

mv s6, a0 # s6 = !x1

and s6, s6, t3 # s6 = !x1 & x4

mv s7, t0 # s7 = x1

and s7, s7, a2 # s7 = x1 & !x3

and s7, s7, a3 # s7 = x1 & !x3 & !x4

mv s8, a0 # s8 = !x1

and s8, s8, a1 # s8 = !x1 & !x2

and s8, s8, t2 # s8 = !x1 & !x2 & x3

or s6, s4, s6 # s6 = (x1 & x2) | (x3 & x4)

or s6, s6, s5 # s6 = (x1 & x2) | (x3 & x4) | (!x1 & x4)

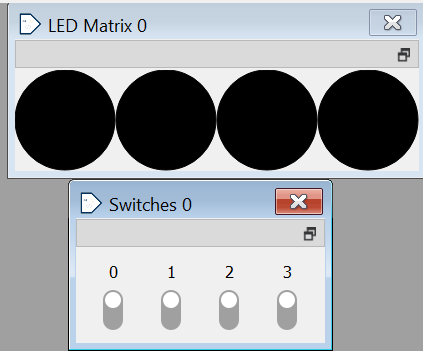
or s6, s6, s7 # s6 = (x1 & x2) | (x3 & x4) | (!x1 & x4) | (x1 & !x3 & !x4)

or s6, s6, s8 # s6 = (x1 & x2) | (x3 & x4) | (!x1 & x4) | (x1 & !x3 & !x4) | (!x1 & !x2 & x3)

ret

**ПРИЛОЖЕНИЕ Б**

**ТЕСТИРОВАНИЕ**

****

1.

Рисунок 1 — Результат при 0000

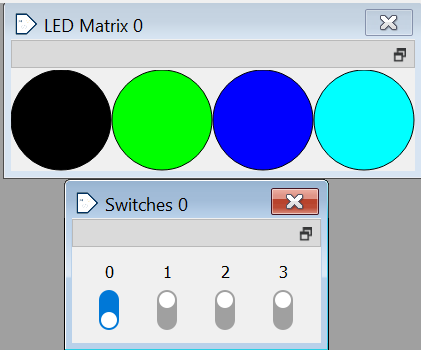
2.

Рисунок 2 — Результат при 1000

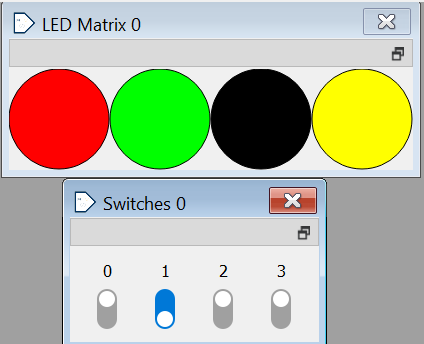
3.

Рисунок 3 — Результат при 0100

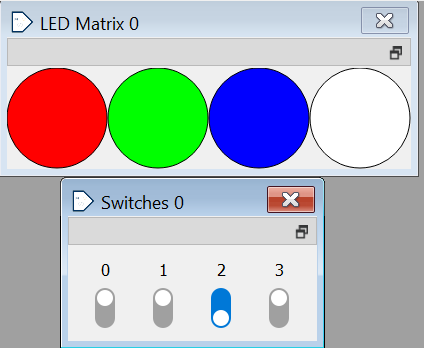
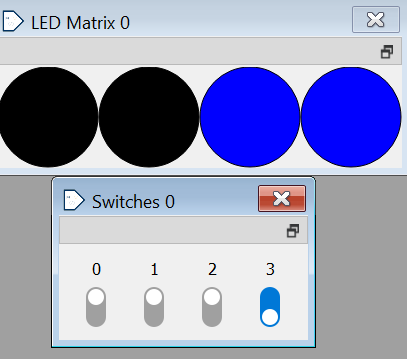
4.

Рисунок 4 — Результат при 0010



5.

Рисунок 5 — Результат при 0001