**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

# по лабораторной работе №7

**по дисциплине «Организация ЭВМ и систем» Тема: Взаимодействие с внешними компонентами**

|  |  |  |
| --- | --- | --- |
| Студент гр. 2384 |  | Кузьминых Е.М. |
| Преподаватель |  | Морозов С.М. |

Санкт-Петербург 2023

# Цель работы

Изучить способы взаимодействия с внешними элементами, написать программу для взаимодействия с устройствами по интерфейсу GPIO.

# 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, см. табл. 1. В задании требуется реализовать 3 функции для зажигания светодиодов. Порядок функций соответствует порядку светодиодов. Первый (самый левый) загорается красным, второй – зелёным, третий – синим. Также есть четвёртый светодиод, цвет которого – смесь всех цветов трёх других светодиодов. Так, если результат вычислений всех функций равен единице, то четвёртый светодиод горит белым цветом.

**Вариант № 14.**

Таблица 1 – Функции и параметры

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Переключатели | | | | Светодиоды | | |
| x1 | x2 | x3 | x4 | Y6(R) | Y3(G) | Y9(B) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 |

# Основные теоретические положения

Одной из важнейших функций, которые решаются микроконтроллером, является взаимодействие с внешними элементами. Для этого в микросхемах реализуются механизмы аппаратной передачи данных.

Самым простым механизмом является интерфейс GPIO (General-Purpose Input/Output), в котором основные операции - установка на контакте микросхемы определенного логического уровня и считывание сигнала с контакта.

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).

# Выполнение работы

В начале программы загружаются адреса различных компонентов в регистры. LED\_MATRIX\_0\_BASE, LED\_MATRIX\_0\_WIDTH, LED\_MATRIX\_0\_HEIGHT - это адреса, где хранятся данные о вашей LED-матрице. SWITCHES\_0\_BASE, SWITCHES\_0\_SIZE - это адреса и размер переключателей, которые вы используете для управления светодиодами.

После начинается цикл, в котором обновляются состояния светодиодов. Считывается текущее состояние переключателей из памяти, после обнуляются регистры s0, s1, s2 и s3 и вызываются три функции, которые отвечают за управление красным, зеленым и синим светодиодами соответственно. После смешиваются цвета, которые установлены в функциях redLED, greenLED и blueLED, в один цвет, который затем записываются в память. Затем происходит переход к следующей итерации цикла с помощью j loop. Смешивание цветов в цикле происходит следующим образом:  
beqz s4, skipColor0: Эта команда проверяет, равно ли значение в регистре s4 нулю. Если это так, то управление переходит к метке skipColor0. В противном случае, код продолжает выполнение следующей команды.

lui s0, 4080: Загружает в регистр s0 непосредственное значение 4080.

skipColor0: Это метка, к которой управление переходит, если значение в регистре s4 не равно нулю.

beqz s5, skipColor1: Проверяет, равно ли значение в регистре s5 нулю. Если это так, то управление переходит к метке skipColor1. В противном случае, код продолжает выполнение следующей команды.

li s1, 65280: Эта команда загружает в регистр s1 непосредственное значение 65280.

skipColor1: метка, к которой переходит управление, если s5 не равно 0

beqz s6, skipColor2: Эта команда проверяет, равно ли значение в регистре s6 нулю. Если это так, то управление переходит к метке skipColor2. В противном случае, код продолжает выполнение следующей команды.

li s2, 255: Эта команда загружает в регистр s2 непосредственное значение 255.

skipColor2: метка, к которой переходит управление, если s6 не равно 0

or s3, s3, s0: Эта команда выполняет логическую операцию ИЛИ между значениями в регистрах s3 и s0, и сохраняет результат в регистре s3.

or s3, s3, s1: Эта команда выполняет логическую операцию ИЛИ между значениями в регистрах s3 и s1, и сохраняет результат в регистре s3.

or s3, s3, s2: Эта команда выполняет логическую операцию ИЛИ между значениями в регистрах s3 и s2, и сохраняет результат в регистре s3.

redLED работает следующим образом:

li s4,: Эта команда устанавливает значение регистра s4 в 0, что означает, что красный светодиод изначально выключен.

bnez t0, redskip1: Проверяет, равно ли значение в регистре t0 нулю. Если это так, то управление переходит к метке redskip1. В противном случае, код продолжает выполнение следующей команды.

bnez t1, redskip1: Проверяет, равно ли значение в регистре t1 нулю. Если это так, то управление переходит к метке redskip1. В противном случае, код продолжает выполнение следующей команды.

beqz t2, redskip1: аналогично bnez t1, redskip1

li s4, 1: Эта команда устанавливает значение регистра s4 в 1, что означает, что красный светодиод включен.

j redendif: Эта команда переходит к метке redendif, завершая выполнение функции.

redskip1: Это метка, к которой управление переходит, если значение в регистре t0 или t1 не равно нулю, или значение в регистре t2 равно нулю.

bnez t0, redskip2, beqz t1, redskip2, bnez t2, redskip2: Эти команды проверяют значения в регистрах t0, t1 и t2, и в зависимости от их значений управление переходит к различным меткам.

li s4, 1: Эта команда устанавливает значение регистра s4 в 1, что означает, что красный светодиод включен.

j redendif: Эта команда переходит к метке redendif, завершая выполнение функции.

redskip2:, redskip3:, redskip4:, redskip5: Это метки, к которым управление переходит в зависимости от значений в регистрах t0, t1, t2 и t3.

redendif:: Это метка, к которой управление переходит, когда функция завершается.

blueLED и greenLED работают аналогичным образом.

Разработанный программный код см. в приложении А.

# Тестирование

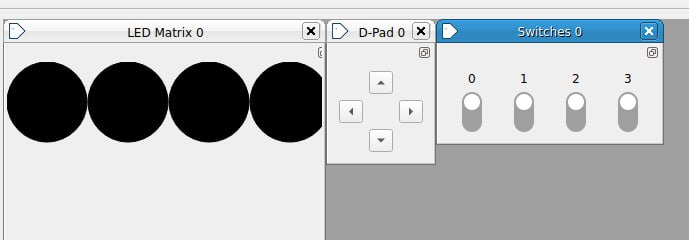


Рис.1 — x1 = 0, x2 = 0, x3 = 0, x4 = 0

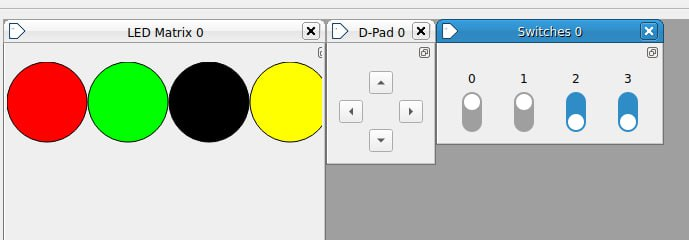


Рис.2 — x1 = 0, x2 = 0, x3 = 1, x4 = 1

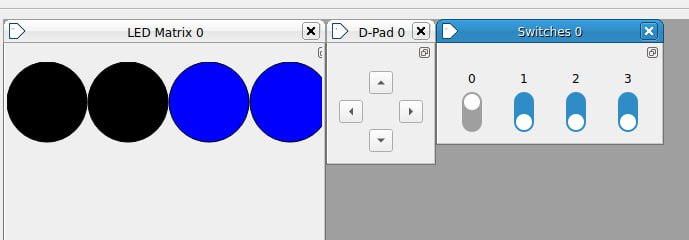


Рис.3 — x1 = 0, x2 = 1, x3 = 1, x4 = 1

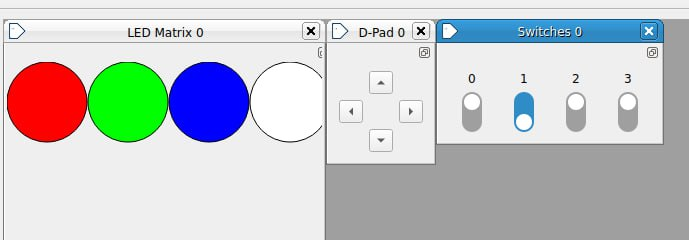


Рис.4 — x1 = 0, x2 = 1, x3 = 0, x4 = 0

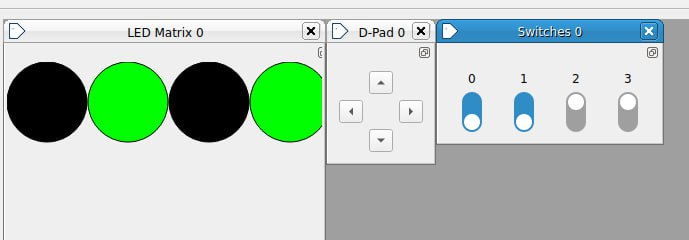


Рис.5 — x1 = 1, x2 = 1, x3 = 0, x4 = 0

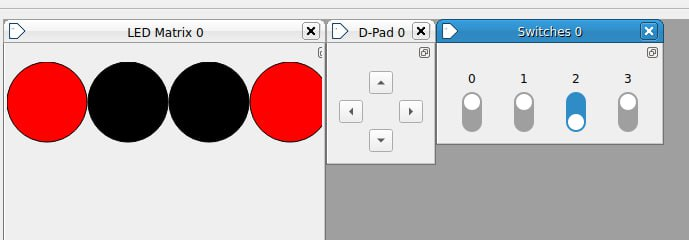


Рис.6 – x1 = 0, x2 = 0, x3 = 1, x4 = 0

# 

Рис.6 – x1 = 1, x2 = 1, x3 = 1, x4 = 0

# Выводы

Были изучены механизмы работы с внешними компонентами в архитектуре RISC-V. Написана программа для взаимодействия с устройствами по интерфейсу GPIO.

**ПРИЛОЖЕНИЕ А ИСХОДНЫЙ КОД ПРОГРАММЫ**

Название файла: lb7.s

.text

li a0 LED\_MATRIX\_0\_BASE

li a1 LED\_MATRIX\_0\_WIDTH

li a2 LED\_MATRIX\_0\_HEIGHT

li a3 SWITCHES\_0\_BASE

li a4 SWITCHES\_0\_SIZE

# 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 - key states

loop:

#считал состояние переключателя

lw a5, 0(a3)

mv s0, zero

mv s1, zero

mv s2, zero

mv s3, zero

#заполнил x1-x4

andi t0, a5, 1

andi t1, a5, 2

andi t2, a5, 4

andi t3, a5, 8

# зажигание фонаря (светодиода)

call redLED

call greenLED

call blueLED

#смешал цвета

beqz s4, skipColor0

lui s0, 4080

skipColor0:

beqz s5, skipColor1

li s1, 65280

skipColor1:

beqz s6, skipColor2

li s2, 255

skipColor2:

# beqz t3, skipColor3

or s3, s3, s0

or s3, s3, s1

or s3, s3, s2

skipColor3:

sw s0, 0(a0) # записал цвета в память

sw s1, 4(a0)

sw s2, 8(a0)

sw s3, 12(a0)

j loop

redLED:

li s4, 0 # выключил

bnez t0, redskip1

bnez t1, redskip1

beqz t2, redskip1 #0010 или 0011

li s4, 1

j redendif

redskip1:

bnez t0, redskip2

beqz t1, redskip2

bnez t2, redskip2 # 0100 или 0101

li s4, 1

j redendif

redskip2:

bnez t0, redskip3

beqz t1, redskip3

beqz t2, redskip3

bnez t3, redskip3 #0110

li s4, 1

j redendif

redskip3:

beqz t0, redskip4

bnez t1, redskip4

bnez t2, redskip4

bnez t3, redskip4 #1000

li s4, 1

redskip4:

beqz t0, redskip5

beqz t1, redskip5

bnez t2, redskip5

beqz t3, redskip5 #1101

li s4, 1

j redendif

redskip5:

beqz t0, redendif

beqz t1, redendif

beqz t2, redendif #1110 или 1111

li s4, 1

redendif:

ret

greenLED:

li s5, 0

beqz t0, greenskip1

beqz t1, greenskip1

bnez t2, greenskip1 #1100 или 1101

li s5, 1

j greenendif

greenskip1:

bnez t0, greenskip2

bnez t1, greenskip2

beqz t3, greenskip2 #0001 или 0011

li s5, 1

j greenendif

greenskip2:

bnez t0, greenskip3

beqz t1, greenskip3

bnez t2, greenskip3 # 0100

bnez t3, greenskip3

li s5, 1

j greenendif

greenskip3:

beqz t0, greenskip4

bnez t1, greenskip4

bnez t2, greenskip4

beqz t3, greenskip4 #1001

li s5, 1

j greenendif

greenskip4:

beqz t0, greenskip5

bnez t1, greenskip5

beqz t2, greenskip5

beqz t3, greenskip5 # 1011

li s5, 1

greenskip5:

beqz t0, greenskip6

beqz t1, greenskip6

bnez t2, greenskip6 #1100 или 1101

li s5, 1

greenskip6:

beqz t0, greenendif

beqz t1, greenendif

beqz t2, greenendif

bnez t3, greenendif # 1110

li s5, 1

greenendif:

ret

blueLED:

li s6, 0

bnez t0, blueskip1

beqz t1, blueskip1

bnez t2, blueskip1 # 0100 или 0101

li s6, 1

j blueendif

blueskip1:

bnez t0, blueskip2

beqz t1, blueskip2

beqz t2, blueskip2

beqz t3, blueskip2 #0111

li s6, 1

j blueendif

blueskip2:

beqz t0, blueendif

bnez t1, blueendif

beqz t3, blueendif #1001 или 1011

li s6, 1

j blueendif

blueendif:

ret