|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/ Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 1 |

**Название:** Инсталляция СУБД PostgreSQL 9.6 на виртуальной машине, создание базы данных и работа с ней

**Дисциплина:** Технология параллельных систем баз данных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-12М |  |  | С.В. Астахов | |
|  | (Группа) |  | (Подпись, дата) | | (И.О. Фамилия) |
|  |  |  |  | |  |
| Преподаватель |  |  |  | |  |
|  |  |  | (Подпись, дата) | | (И.О. Фамилия) |

Москва, 2023

**Введение**

**Цель работы:** формирование следующей компетенции: студент должен получить навыки инсталляции СУБД PostgreSQL в среде ОС Ubuntu. Он должен протестировать её работу, разработать небольшую базу данных и выполнить запросы к ней.

**Ход выполнения**

**Задание 1**

Скомпилировать исходный код представленной программы, привести дизассемблерный листинг и псевдокод программы.

Исходный код программы:

.section .text

.globl \_start;

len = 8 #Размер массива

enroll = 1 #Количество обрабатываемых элементов за одну итерацию

elem\_sz = 4 #Размер одного элемента массива

\_start:

addi x20, x0, len/enroll

la x1, \_x

lp:

lw x2, 0(x1)

add x31, x31, x2 #!

addi x1, x1, elem\_sz\*enroll

addi x20, x20, -1

bne x20, x0, lp

addi x31, x31, 1

lp2: j lp2

.section .data

\_x: .4byte 0x1

.4byte 0x2

.4byte 0x3

.4byte 0x4

.4byte 0x5

.4byte 0x6

.4byte 0x7

.4byte 0x8

Дизассемблерный листинг:

Disassembly of section .text:

80000000 <\_start>:

80000000: 00800a13 addi x20,x0,8

80000004: 00000097 auipc x1,0x0

80000008: 02408093 addi x1,x1,36 # 80000028 <\_x>

8000000c <lp>:

8000000c: 0000a103 lw x2,0(x1)

80000010: 002f8fb3 add x31,x31,x2

80000014: 00408093 addi x1,x1,4

80000018: fffa0a13 addi x20,x20,-1

8000001c: fe0a18e3 bne x20,x0,8000000c <lp>

80000020: 001f8f93 addi x31,x31,1

80000024 <lp2>:

80000024: 0000006f jal x0,80000024 <lp2>

Disassembly of section .data:

80000028 <\_x>:

80000028: 0001 c.addi x0,0

8000002a: 0000 unimp

8000002c: 0002 0x2

8000002e: 0000 unimp

80000030: 00000003 lb x0,0(x0) # 0 <enroll-0x1>

80000034: 0004 c.addi4spn x9,x2,0

80000036: 0000 unimp

80000038: 0005 c.addi x0,1

8000003a: 0000 unimp

8000003c: 0006 0x6

8000003e: 0000 unimp

80000040: 00000007 0x7

80000044: 0008 c.addi4spn x10,x2,0

Псевдокод программы:

#define len 8

#define enroll 1

#define elem\_sz 4

int \_x[]={1,2,3,4,5,6,7,8};

void \_start() {

int x20 = len/enroll;

int \*x1 = \_x;

do {

int x2 = x1[0];

x31 += x2;

x1 += enroll;

x20--;

} while(x20 != 0);

x31++;

while(1){}

}

Примечание: в результате работы данного кода регистр х31 будет содержать число 37 (0х25).

**Задание 2**

Получить снимок экрана, содержащий временную диаграмму выполнения стадий выборки и диспетчеризации команды с указанным адресом (0x8000000c, 1-я итерация).

Требуемый снимок экрана представлен на рисунке 1.

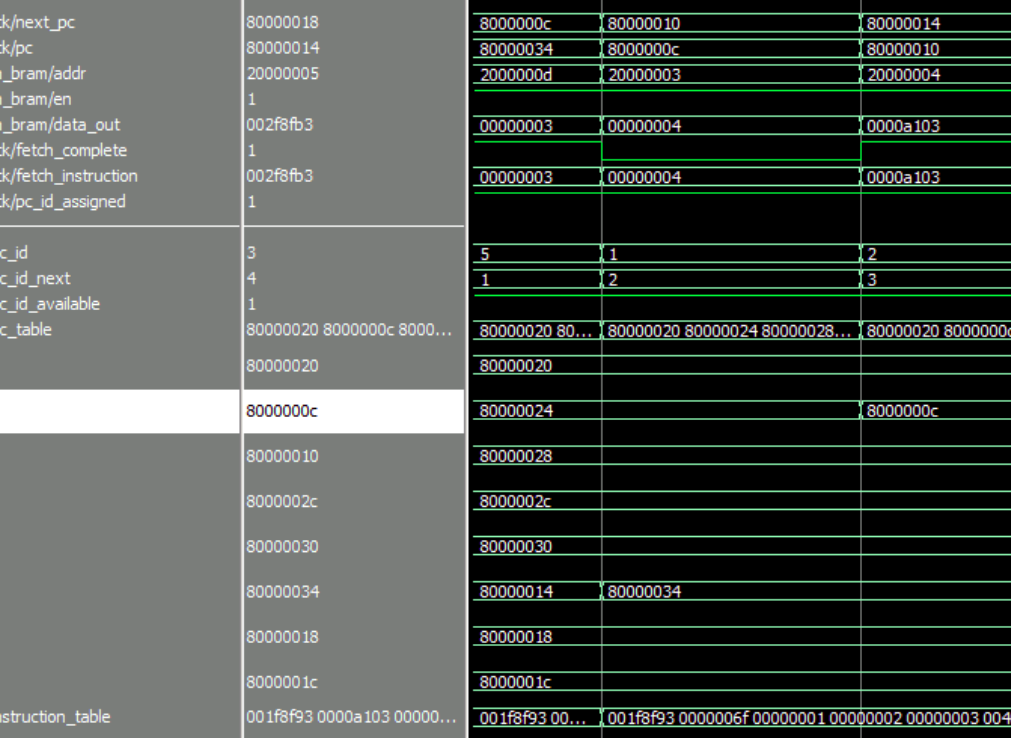


Рисунок 1 – стадия выборки и диспетчеризации

**Задание 3**

Получить снимок экрана, содержащий временную диаграмму выполнения стадии декодирования и планирования на выполнение команды с указанным адресом (0x80000018, 1-я итерация).

Требуемый снимок экрана представлен на рисунке 2.

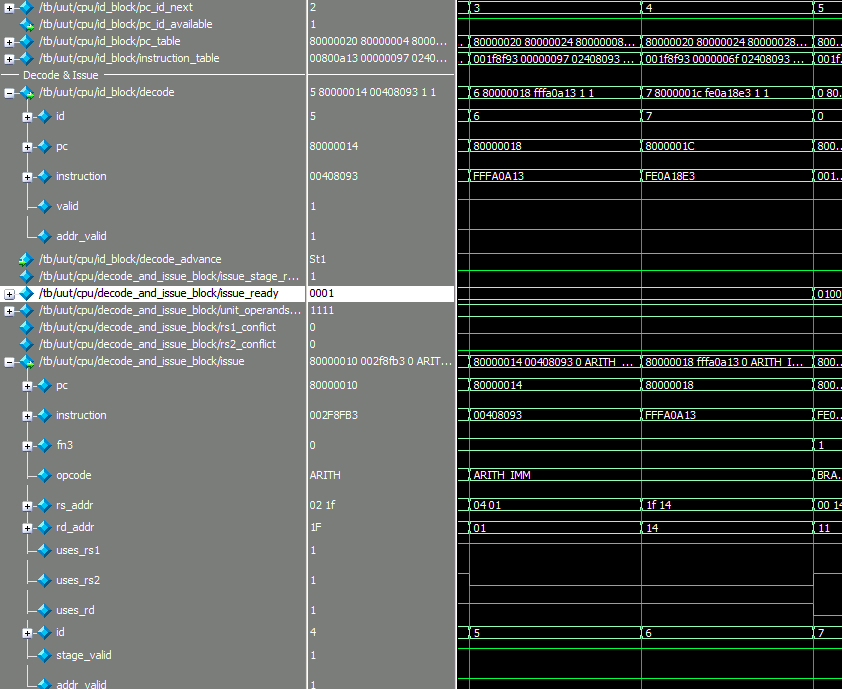


Рисунок 2 – декодирование команды

**Задание 4**

Получить снимок экрана, содержащий временную диаграмму выполнения стадии выполнения команды с указанным адресом (0x80000000).

Требуемый снимок экрана представлен на рисунке 3.

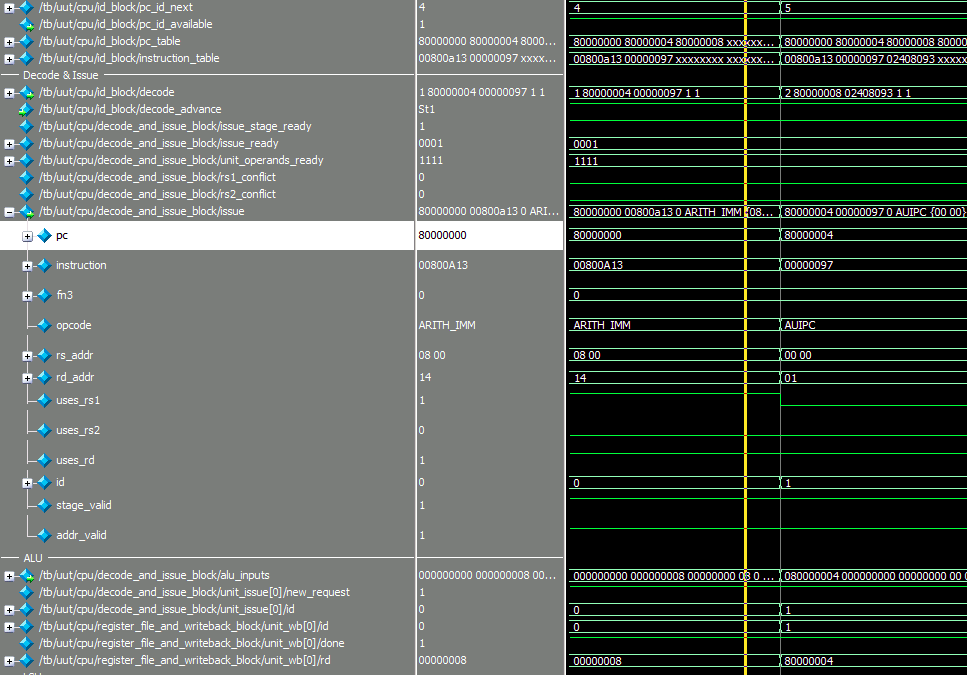


Рисунок 3 – стадия выполнения

**Задание 5.1**

Получить снимок экрана, содержащий временные диаграммы сигналов, соответствующих всем стадиям выполнения команды, обозначенной в тексте программы символом #!.

В тексте программы отмечена команда add x31, x31, x2 #! По адресу 0x80000010.

Этапы ее обработки показаны на рисунках 4-6.

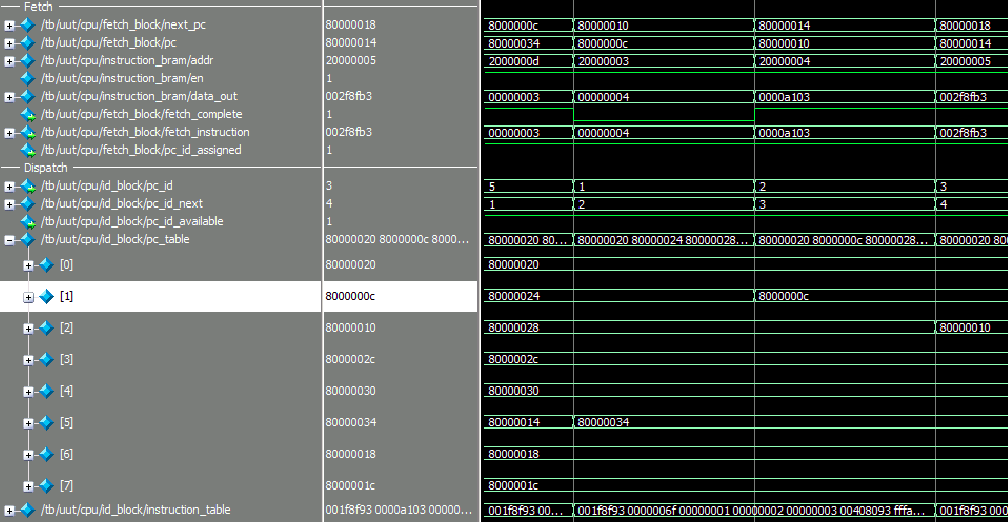


Рисунок 4 – стадия выборки

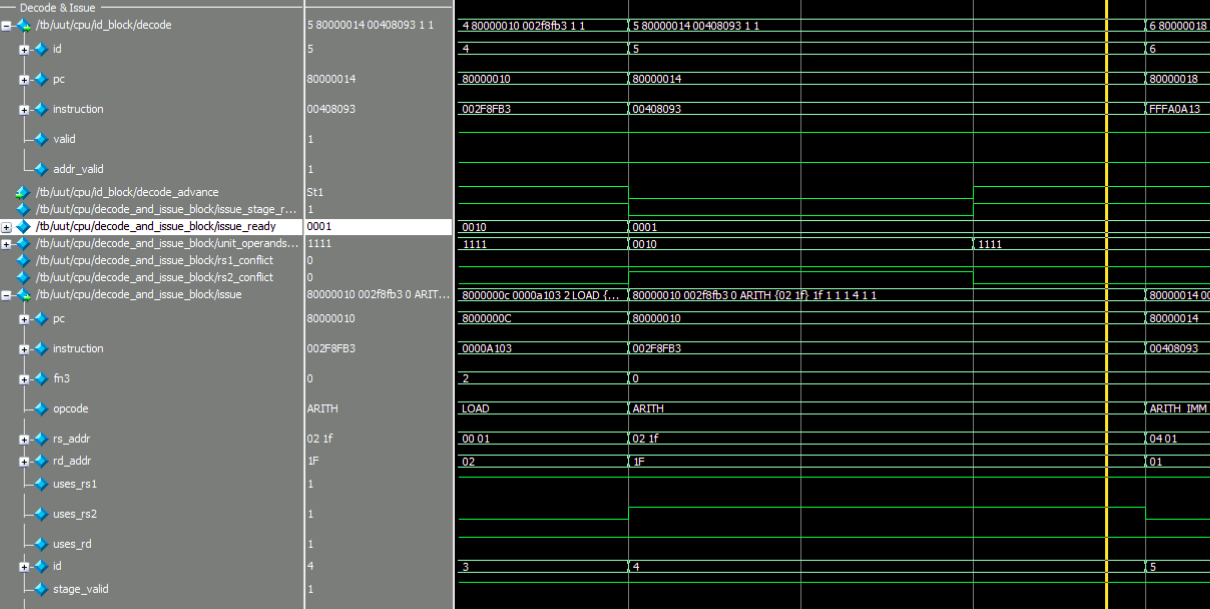


Рисунок 5 – стадия декодирования

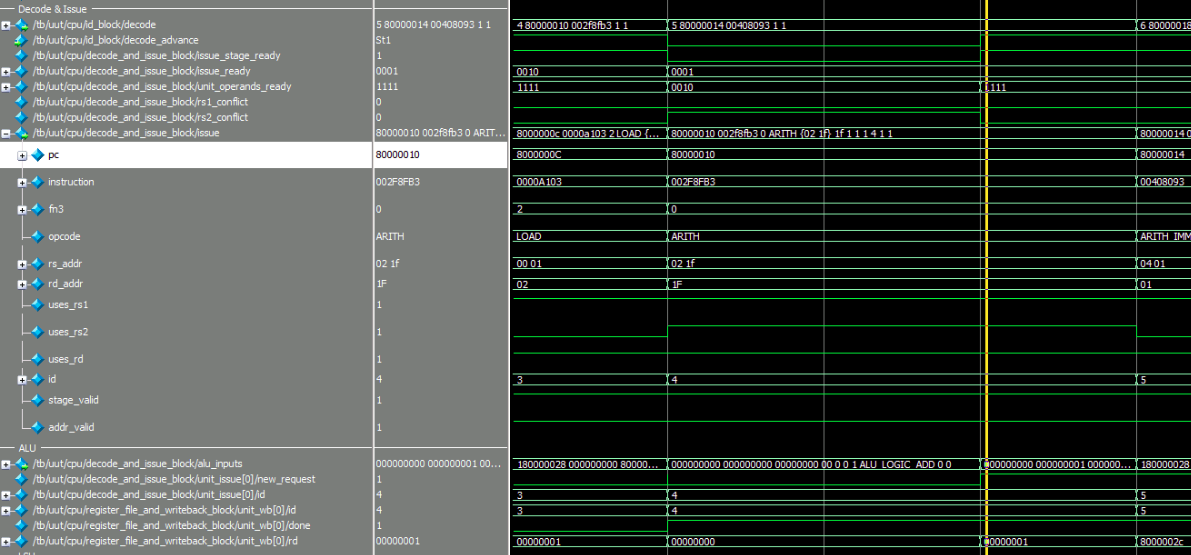


Рисунок 6 – стадия выполнения

**Задание 5.2**

Анализируя диаграмму заполнить трассу выполнения программы. Сделать вывод об эффективности выполнения программы и о путях оптимизации.

Трасса программы представлена на рисунке 7.

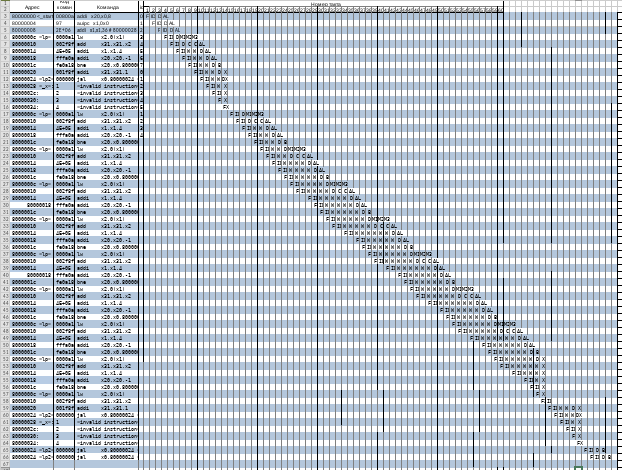


Рисунок 7 – трасса исходной программы

Программа имеет низкую эффективность, ее можно оптимизировать за счет устранения конфликтов на конвейере.

**Задание 5.3**

Провести оптимизацию программы путем перестановки команд для устранения конфликтов. Перекомпилировать программу и перезапустить симуляцию. Заполнить трассу выполнения оптимизированной программы. Сравнить трассы выполнения неоптимизированной и оптимизированной версии, сделать выводы.

Будем загружать данные в несколько регистров за цикл, а затем в том же порядке складывать их, чтобы устранить конфликты из-за продолжительной работы с памятью.

Исходный код измененной программы:

.section .text

.globl \_start;

len = 8 #Размер массива

enroll = 4 #Количество обрабатываемых элементов за одну итерацию

elem\_sz = 4 #Размер одного элемента массива

\_start:

addi x20, x0, len/enroll

la x1, \_x

lp:

lw x2, 0(x1)

lw x3, 4(x1)

lw x4, 8(x1)

lw x5, 12(x1)

add x31, x31, x2 #!

add x31, x31, x3 #!

add x31, x31, x4 #!

add x31, x31, x5 #!

addi x1, x1, elem\_sz\*enroll

addi x20, x20, -1

bne x20, x0, lp

addi x31, x31, 1

lp2: j lp2

.section .data

\_x: .4byte 0x1

.4byte 0x2

.4byte 0x3

.4byte 0x4

.4byte 0x5

.4byte 0x6

.4byte 0x7

.4byte 0x8

Дизассемблерный листинг оптимизированной программы:

Disassembly of section .text:

80000000 <\_start>:

80000000: 00200a13 addi x20,x0,2

80000004: 00000097 auipc x1,0x0

80000008: 03c08093 addi x1,x1,60 # 80000040 <\_x>

8000000c <lp>:

8000000c: 0000a103 lw x2,0(x1)

80000010: 0040a183 lw x3,4(x1)

80000014: 0080a203 lw x4,8(x1)

80000018: 00c0a283 lw x5,12(x1)

8000001c: 002f8fb3 add x31,x31,x2

80000020: 003f8fb3 add x31,x31,x3

80000024: 004f8fb3 add x31,x31,x4

80000028: 005f8fb3 add x31,x31,x5

8000002c: 01008093 addi x1,x1,16

80000030: fffa0a13 addi x20,x20,-1

80000034: fc0a1ce3 bne x20,x0,8000000c <lp>

80000038: 001f8f93 addi x31,x31,1

8000003c <lp2>:

8000003c: 0000006f jal x0,8000003c <lp2>

Disassembly of section .data:

80000040 <\_x>:

80000040: 0001 c.addi x0,0

80000042: 0000 unimp

80000044: 0002 0x2

80000046: 0000 unimp

80000048: 00000003 lb x0,0(x0) # 0 <elem\_sz-0x4>

8000004c: 0004 c.addi4spn x9,x2,0

8000004e: 0000 unimp

80000050: 0005 c.addi x0,1

80000052: 0000 unimp

80000054: 0006 0x6

80000056: 0000 unimp

80000058: 00000007 0x7

8000005c: 0008 c.addi4spn x10,x2,0

...

Трасса модифицированной программы представлена на рисунке 8.

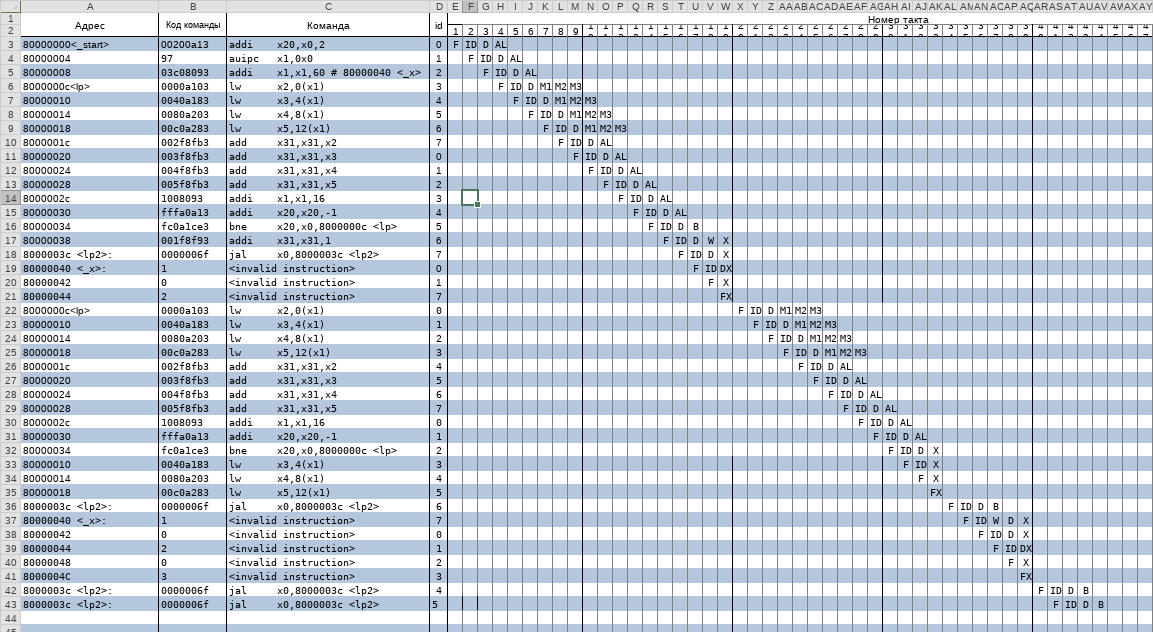


Рисунок 8 – трасса модифицированной программы

**Вывод:**

В ходе лабораторной работы произошло ознакомление с принципами функционирования, построения и особенностями архитектуры суперскалярных конвейерных микропроцессоров. Также получено представление о простейших методах оптимизации программ для таких микропроцессоров.