УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.04.04 Программная инженерия

Дисциплина «Системное программное обеспечение»

Лабораторная работа №3

Вариант 1

Студент
Алексеев М. Н.
P4116

Преподаватель

Кореньков Ю. Д.

Санкт-Петербург, 2023 г.

Задание лабораторной работы

Задание 1

Реализовать формирование линейного кода в терминах некоторого набора инструкций посредством
анализа графа потока управления для набора подпрограмм. Полученный линейный код вывести в
мнемонической форме в выходной текстовый файл.

Подготовка к выполнению по одному из двух сценариев:

1. Составить описание виртуальной машины с набором инструкций и моделью памяти по варианту
    a. Изучить нотацию для записи определений целевых архитектур
    b. Составить описание ВМ в соответствии с вариантом
        i. Описание набор регистров и банков памяти
        ii. Описать набор инструкций: для каждой инструкции задать структуру операционного кода, содержащего описание операндов и набор операций,
изменяющих состояние ВМ
            1. Описать инструкции перемещения данных и загрузки констант
            2. Описать инструкции арифметических и логических операций
            3. Описать инструкции условной и безусловной передачи управления
            4. Описать инструкции ввода-вывода с использованием скрытого регистра
в качестве порта ввода-вывода
        iii. Описать набор мнемоник, соответствующих инструкциям ВМ
    c. Подготовить скрипт для запуска ассемблированного листинга с использованием
описания ВМ:
        i. Написать тестовый листинг с использованием подготовленных мнемоник
инструкций
        ii. Задействовать транслятор листинга в бинарный модуль по описанию ВМ
        iii. Запустить полученный бинарный модуль на исполнение и получить результат работы
        iv. Убедиться в корректности функционирования всех инструкций ВМ
2. Выбрать и изучить прикладную архитектуру системы команд существующей ВМ
    a. Для выбранной ВМ:
    i. Должен существовать готовый эмулятор (например qemu)
    ii. Должен существовать готовый тулчейн (набор инструментов разработчика): компилятор Си, ассемблер и дизассемблер, линковщик, желательно отладчик

b. Согласовать выбор ВМ с преподавателем

c. Изучить модель памяти и набор инструкций ВМ

d. Научиться использовать тулчейн (собирать и запускать программы из листинга)

e. Подготовить скрипт для запуска ассемблированного листинга с использованием эмулятора

i. Написать тестовый листинг с использованием инструкций ВМ

ii. Задействовать ассемблер и компоновщик из тулчейна

iii. Запустить бинарный модуль на исполнение и получить результат его работы

Порядок выполнения:

1. Описать структуры данных, необходимые для представления информации об элементах образа программы (последовательностях инструкций и данных), расположенных в памяти

a. Для каждой инструкции – имя мнемоники и набор операндов в терминах данной ВМ

b. Для элемента данных – соответствующее литеральное значение или размер экземпляра типа данных в байтах

2. Реализовать модуль, формирующий образ программы в линейном коде для данного набора подпрограмм

a. Программный интерфейс модуля принимает на вход структуру данных, содержащую графы потока управления и информацию о локальных переменных и сигнатурах для набора подпрограмм, разработанную в задании 2 (п. 1.a, п. 2.b)

b. В результате работы порождается структура данных, разработанная в п. 1, содержащая описание образа программы в памяти: набор именованных элементов данных и набор именованных фрагментов линейного кода, представляющих собой алгоритмы подпрограмм

c. Для каждой подпрограммы посредством обхода узлов графа потока управления в порядке топологической сортировки (начиная с узла, являющегося первым базовым блоком алгоритма подпрограммы), сформировать набор именованных групп инструкций, включая пролог и эпилог подпрограммы (формирующие и разрушающие локальное состояние подпрограммы)

d. Для каждого базового блока в составе графа потока управления сформировать группу инструкций, соответствующих операциям в составе дерева операций 3

e. Использовать имена групп инструкций для формирования инструкций перехода между блоками инструкций, соответствующих узлам графа потока управления в соответствии с дугами в нём

3. Доработать тестовую программу, разработанную в задании 2 для демонстрации

работоспособности созданного модуля

a. Добавить поддержку аргумента командной строки для имени выходного файла, вывод информации о графах потока управления сделать опциональным

b. Использовать модуль, разработанный в п. 2 для формирования образа программы на основе информации, собранной в результате работы модуля, созданного в задании 2 (п. 2.b)

c. Для сформированного образа программы в линейном коде вывести в выходной файл ассемблерный листинг, содержащий мнемоническое представление инструкций и данных, как они описаны в структурах данных (п. 1), построенных разработанным модулем (пп. 2.c-e)

d. Проверить корректность решения посредством сборки сгенерированного листинга и запуска полученного бинарного модуля на эмуляторе ВМ (см. подготовка п. 1.c или п. 2.e)

4. Результаты тестирования представить в виде отчета, в который включить:

a. В части 3 привести описание разработанных структур данных

b. В части 4 описать программный интерфейс и особенности реализации разработанного модуля

c. В части 5 привести примеры исходных текстов, соответствующие ассемблерные листинги и примера вывода запущенных тестовых программ

Описание структур данных

```
enum codegen_asm_type {

    CODEGEN_ASM_TYPE_COMMENT = 0,
    CODEGEN_ASM_TYPE_LABEL,
    CODEGEN_ASM_TYPE_OP,
    CODEGEN_ASM_TYPE_DATA,
    CODEGEN_ASM_TYPE_LABEL_DATA,
};

enum codegen_asm_op_opcode {

    CODEGEN_ASM_OP_OPCODE_CONST = 0,
    CODEGEN_ASM_OP_OPCODE_LOAD,
    CODEGEN_ASM_OP_OPCODE_STORE,
    CODEGEN_ASM_OP_OPCODE_GET,
    CODEGEN_ASM_OP_OPCODE_SET,
    CODEGEN_ASM_OP_OPCODE_ZEXT,
    CODEGEN_ASM_OP_OPCODE_SEXT,
    CODEGEN_ASM_OP_OPCODE_TRUNC,
    CODEGEN_ASM_OP_OPCODE_ADD,
    CODEGEN_ASM_OP_OPCODE_SUB,
    CODEGEN_ASM_OP_OPCODE_MUL,
    CODEGEN_ASM_OP_OPCODE_DIV,
    CODEGEN_ASM_OP_OPCODE_REM,
    CODEGEN_ASM_OP_OPCODE_AND,
    CODEGEN_ASM_OP_OPCODE_OR,
    CODEGEN_ASM_OP_OPCODE_XOR,
    CODEGEN_ASM_OP_OPCODE_SHL,
    CODEGEN_ASM_OP_OPCODE_SHR,
    CODEGEN_ASM_OP_OPCODE_CMP,
    CODEGEN_ASM_OP_OPCODE_GOTO,
    CODEGEN_ASM_OP_OPCODE_IFZ,
    CODEGEN_ASM_OP_OPCODE_CALL,
    CODEGEN_ASM_OP_OPCODE_RET,
    CODEGEN_ASM_OP_OPCODE_NOP,
    CODEGEN_ASM_OP_OPCODE_HLT,
    CODEGEN_ASM_OP_OPCODE_IN,
    CODEGEN_ASM_OP_OPCODE_OUT,
};

enum codegen_asm_op_reg {

    CODEGEN_ASM_OP_REG_SP = 0,
    CODEGEN_ASM_OP_REG_FP = 1,
};

enum codegen_asm_op_cmp {

    CODEGEN_ASM_OP_CMP_EQ = 0,
    CODEGEN_ASM_OP_CMP_NE = 1,
    CODEGEN_ASM_OP_CMP_LT = 4,
    CODEGEN_ASM_OP_CMP_LE = 5,
    CODEGEN_ASM_OP_CMP_GT = 6,
    CODEGEN_ASM_OP_CMP_GE = 7,
};

struct codegen_asm_op {

    enum codegen_asm_op_opcode opcode;
```

```
    union {
        uint8_t imm8;
        uint8_t imm2;
        char * label;
        enum codegen_asm_op_reg reg;
        enum codegen_asm_op_cmp cmp;
    };
};

struct codegen_asm_data {

    size_t size;
    unsigned char * data;
};

struct codegen_asm {

    enum codegen_asm_type _type;

    union {

        char * comment;
        char * label;
        struct codegen_asm_op op;
        struct codegen_asm_data data;
        char * label_data;
    };
};

struct codegen_asm_list {

    size_t size;
    size_t capacity;
    struct codegen_asm * values;
}
```

Пример входных данных

Описание встроенных функций и утилит

```
char read();
write(char c);
write_str(string str);
int[] new_int_array(ulong size);
ulong get_int_array_size(int[] array);

byte ord(char c);
char chr(byte c);

char digit_to_char(byte d) {
    if (d == 0) '0';
    else if (d == 1) '1';
    else if (d == 2) '2';
    else if (d == 3) '3';
    else if (d == 4) '4';
    else if (d == 5) '5';
    else if (d == 6) '6';
    else if (d == 7) '7';
    else if (d == 8) '8';
    else if (d == 9) '9';
    else ' ';
}

write_ulong(ulong value) {
    if (value == 0) {
        write_str("0");
        break;
    }

    if (value / 10 != 0) {
        write_ulong(value / 10);
    }

    write(digit_to_char(0 + value % 10));
}

write_long(long value) {
    if (value < 0) {
        write('-');
        value = -value;
    }

    ulong z = 0;
    write_ulong(z + value);
}

ulong read_ulong() {
    ulong result = 0;

    while (true) {
        byte c = ord(read());

        ulong d;
        if (c >= ord('0') && c <= ord('9')) {
            d = c - ord('0');
        } else {
            break;
        }
```

```
        result = result * 10 + d;
    }

    result;
}
```

Программа считывает кол-во запросов и для каждого запроса выводит число Фибоначчи по данному номеру запоминая результаты в динамически аллоцируемом массиве

```
int[] increase_size(int[] x, ulong size) {
    ulong real_size = get_int_array_size(x);

    if (real_size >= size) {
        x;
        break;
    }

    int[] result = new_int_array(real_size * 2);

    ulong i = 0;
    while (i < real_size) {
        result[i] = x[i];
        ++i;
    }

    result;
}

main() {
    write_str("Number of queries: ");
    ulong cnt = read_ulong();

    int[] table = new_int_array(2);
    table[0] = 1;
    table[1] = 1;

    ulong filled = 2;

    while (cnt > 0) {
        write_str("Query: ");
        ulong q = read_ulong();

        table = increase_size(table, q + 1);

        while (filled <= q) {
            table[filled] = table[filled - 1] + table[filled - 2];
            ++filled;
        }

        write_str("Result: ");
        write_long(table[q]);
        write_str("\n");
        --cnt;
    }
}
```

Пример выходных данных

Результат компиляции

```
[section ram]
      const 4
      db 0, 0xf0, 0xff, 0xff
      set sp
      get sp
      set fp
      call main
      hlt
; test.in:12
digit_to_char:
      get sp
      const 4
      db 0x0, 0x0, 0x0, 0x0
      sub
      set sp
      get fp
      get sp
      const 4
      db 0x5, 0x0, 0x0, 0x0
      add
      set fp
      get fp
; 1: EXPR at 12:28
.node_1:
      goto .node_2
; 2: COND at 13:5
.node_2:
      get fp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      load 1
      zext 1
      const 4
      db 0x0, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      cmp eq
      ifz .node_4
      goto .node_3
; 3: EXPR at 13:17
.node_3:
      const 1
      db 0x30
      goto .leave
; 4: COND at 14:10
.node_4:
      get fp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      load 1
      zext 1
      const 4
      db 0x1, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      cmp eq
      ifz .node_6
```

```
      goto .node_5
; 5: EXPR at 14:22
.node_5:
      const 1
      db 0x31
      goto .leave
; 6: COND at 15:10
.node_6:
      get fp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      load 1
      zext 1
      const 4
      db 0x2, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      cmp eq
      ifz .node_8
      goto .node_7
; 7: EXPR at 15:22
.node_7:
      const 1
      db 0x32
      goto .leave
; 8: COND at 16:10
.node_8:
      get fp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      load 1
      zext 1
      const 4
      db 0x3, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      cmp eq
      ifz .node_10
      goto .node_9
; 9: EXPR at 16:22
.node_9:
      const 1
      db 0x33
      goto .leave
; 10: COND at 17:10
.node_10:
      get fp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      load 1
      zext 1
      const 4
      db 0x4, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      cmp eq
      ifz .node_12
      goto .node_11
; 11: EXPR at 17:22
```

```
.node_11:
      const 1
      db 0x34
      goto .leave
; 12: COND at 18:10
.node_12:
      get fp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      load 1
      zext 1
      const 4
      db 0x5, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      cmp eq
      ifz .node_14
      goto .node_13
; 13: EXPR at 18:22
.node_13:
      const 1
      db 0x35
      goto .leave
; 14: COND at 19:10
.node_14:
      get fp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      load 1
      zext 1
      const 4
      db 0x6, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      cmp eq
      ifz .node_16
      goto .node_15
; 15: EXPR at 19:22
.node_15:
      const 1
      db 0x36
      goto .leave
; 16: COND at 20:10
.node_16:
      get fp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      load 1
      zext 1
      const 4
      db 0x7, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      cmp eq
      ifz .node_18
      goto .node_17
; 17: EXPR at 20:22
.node_17:
      const 1
```

```
      db 0x37
      goto .leave
; 18: COND at 21:10
.node_18:
      get fp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      load 1
      zext 1
      const 4
      db 0x8, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      cmp eq
      ifz .node_20
      goto .node_19
; 19: EXPR at 21:22
.node_19:
      const 1
      db 0x38
      goto .leave
; 20: COND at 22:10
.node_20:
      get fp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      load 1
      zext 1
      const 4
      db 0x9, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      cmp eq
      ifz .node_22
      goto .node_21
; 21: EXPR at 22:22
.node_21:
      const 1
      db 0x39
      goto .leave
; 22: EXPR at 23:10
.node_22:
      const 1
      db 0x20
      goto .leave
.leave:
      store 1
      set fp
      get sp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      add
      set sp
      ret
; constants
; test.in:24
write_ulong:
      get sp
      const 4
      db 0x0, 0x0, 0x0, 0x0
```

```
      sub
      set sp
      get fp
      get sp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      add
      set fp
      get fp
; 1: EXPR at 26:26
.node_1:
      goto .node_2
; 2: COND at 27:5
.node_2:
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x0, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      cmp eq
      ifz .node_4
      goto .node_3
; 3: EXPR at 28:9
.node_3:
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      sub
      set sp
      const 4
      dd .const_1
      call write_str
      goto .leave
; 4: COND at 32:5
.node_4:
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0xa, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      div
      const 4
      db 0x0, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      cmp ne
      ifz .node_6
      goto .node_5
; 5: EXPR at 33:9
.node_5:
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      sub
```

```
      set sp
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0xa, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      div
      call write_ulong
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_6
; 6: EXPR at 36:5
.node_6:
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      sub
      set sp
      get sp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      set sp
      const 4
      db 0x0, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0xa, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      rem
      add
      trunc 1
      call digit_to_char
      call write
      goto .leave
.return_void:
      const 4
      db 0x0, 0x0, 0x0, 0x0
      trunc 2
.leave:
      store 2
      set fp
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      ret
```

```
; constants
.const_1:
      db 0x1, 0x0, 0x0, 0x0, 0x30
; test.in:37
write_long:
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      set sp
      get fp
      get sp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      add
      set fp
      get fp
; 1: EXPR at 39:24
.node_1:
      goto .node_2
; 2: COND at 40:5
.node_2:
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x0, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      cmp lt
      ifz .node_5
      goto .node_3
; 3: EXPR at 41:9
.node_3:
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      sub
      set sp
      const 1
      db 0x2d
      call write
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_4
; 4: EXPR at 42:9
.node_4:
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      const 4
      db 0x0, 0x0, 0x0, 0x0
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
```

```
      load 4
      sub
      store 4
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_5
; 5: EXPR at 45:11
.node_5:
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      const 4
      db 0x0, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      store 4
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_6
; 6: EXPR at 46:5
.node_6:
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      sub
      set sp
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      load 4
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      add
      call write_ulong
      goto .leave
.return_void:
      const 4
      db 0x0, 0x0, 0x0, 0x0
      trunc 2
.leave:
      store 2
```

```
      set fp
      get sp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      add
      set sp
      ret
; constants
; test.in:49
read_ulong:
      get sp
      const 4
      db 0x9, 0x0, 0x0, 0x0
      sub
      set sp
      get fp
      get sp
      const 4
      db 0xd, 0x0, 0x0, 0x0
      add
      set fp
      get fp
; 1: EXPR at 49:20
.node_1:
      goto .node_2
; 2: EXPR at 50:11
.node_2:
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      const 4
      db 0x0, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      store 4
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_3
; 3: COND at 52:5
.node_3:
      const 1
      db 0xff
      sext 1
      ifz .node_8
      goto .node_4
; 4: EXPR at 53:14
.node_4:
      get fp
      const 4
      db 0x5, 0x0, 0x0, 0x0
      sub
      get sp
      const 4
```

```
      db 0x1, 0x0, 0x0, 0x0
      sub
      set sp
      get sp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      set sp
      call read
      call ord
      store 1
      get fp
      const 4
      db 0x5, 0x0, 0x0, 0x0
      sub
      load 1
      get sp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_5
; 5: COND at 56:9
.node_5:
      get fp
      const 4
      db 0x5, 0x0, 0x0, 0x0
      sub
      load 1
      zext 1
      get sp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      set sp
      const 1
      db 0x30
      call ord
      zext 1
      cmp ge
      get fp
      const 4
      db 0x5, 0x0, 0x0, 0x0
      sub
      load 1
      zext 1
      get sp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      set sp
      const 1
      db 0x39
      call ord
      zext 1
      cmp le
      andb
      ifz .node_8
      goto .node_6
; 6: EXPR at 57:13
.node_6:
      get fp
```

```
      const 4
      db 0x9, 0x0, 0x0, 0x0
      sub
      get fp
      const 4
      db 0x5, 0x0, 0x0, 0x0
      sub
      load 1
      zext 1
      get sp
      const 4
      db 0x1, 0x0, 0x0, 0x0
      sub
      set sp
      const 1
      db 0x30
      call ord
      zext 1
      sub
      trunc 1
      zext 1
      store 4
      get fp
      const 4
      db 0x9, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_7
; 7: EXPR at 62:9
.node_7:
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0xa, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      mul
      get fp
      const 4
      db 0x9, 0x0, 0x0, 0x0
      sub
      load 4
      add
      store 4
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
```

```
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_3
; 8: EXPR at 65:5
.node_8:
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      goto .leave
.return_void:
      const 4
      db 0x0, 0x0, 0x0, 0x0
.leave:
      store 4
      set fp
      get sp
      const 4
      db 0x9, 0x0, 0x0, 0x0
      add
      set sp
      ret
; constants
; test1.in:2
increase_size:
      get sp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      set sp
      get fp
      get sp
      const 4
      db 0x18, 0x0, 0x0, 0x0
      add
      set fp
      get fp
; 1: EXPR at 2:42
.node_1:
      goto .node_2
; 2: EXPR at 3:11
.node_2:
      get fp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      set sp
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      call get_int_array_size
      store 4
      get fp
```

```
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_3
; 3: COND at 5:5
.node_3:
      get fp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      load 4
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      load 4
      cmp ge
      ifz .node_5
      goto .node_4
; 4: EXPR at 6:9
.node_4:
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      goto .leave
; 5: EXPR at 10:11
.node_5:
      get fp
      const 4
      db 0x10, 0x0, 0x0, 0x0
      sub
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      set sp
      get fp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x2, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      mul
      call new_int_array
      store 4
      get fp
      const 4
      db 0x10, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
```

```
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_6
; 6: EXPR at 12:11
.node_6:
      get fp
      const 4
      db 0x14, 0x0, 0x0, 0x0
      sub
      const 4
      db 0x0, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      store 4
      get fp
      const 4
      db 0x14, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_7
; 7: COND at 13:5
.node_7:
      get fp
      const 4
      db 0x14, 0x0, 0x0, 0x0
      sub
      load 4
      get fp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      load 4
      cmp lt
      ifz .node_10
      goto .node_8
; 8: EXPR at 14:9
.node_8:
      get fp
      const 4
      db 0x10, 0x0, 0x0, 0x0
      sub
      load 4
      get fp
      const 4
      db 0x14, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x2, 0x0, 0x0, 0x0
      mul
      add
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      get fp
```

```
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      get fp
      const 4
      db 0x14, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x2, 0x0, 0x0, 0x0
      mul
      add
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      load 2
      store 2
      get fp
      const 4
      db 0x10, 0x0, 0x0, 0x0
      sub
      load 4
      get fp
      const 4
      db 0x14, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x2, 0x0, 0x0, 0x0
      mul
      add
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      load 2
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_9
; 9: EXPR at 15:9
.node_9:
      get fp
      const 4
      db 0x14, 0x0, 0x0, 0x0
      sub
      get fp
      const 4
      db 0x14, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x1, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      add
      store 4
      get fp
      const 4
      db 0x14, 0x0, 0x0, 0x0
```

```
      sub
      load 4
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_7
; 10: EXPR at 18:5
.node_10:
      get fp
      const 4
      db 0x10, 0x0, 0x0, 0x0
      sub
      load 4
      goto .leave
.leave:
      store 4
      set fp
      get sp
      const 4
      db 0x14, 0x0, 0x0, 0x0
      add
      set sp
      ret
; constants
; test1.in:19
main:
      get sp
      const 4
      db 0x10, 0x0, 0x0, 0x0
      sub
      set sp
      get fp
      get sp
      const 4
      db 0x14, 0x0, 0x0, 0x0
      add
      set fp
      get fp
; 1: EXPR at 21:8
.node_1:
      goto .node_2
; 2: EXPR at 22:5
.node_2:
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      sub
      set sp
      const 4
      dd .const_1
      call write_str
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_3
; 3: EXPR at 23:11
.node_3:
      get fp
```

```
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      set sp
      call read_ulong
      store 4
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_4
; 4: EXPR at 25:11
.node_4:
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      set sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      call new_int_array
      store 4
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_5
; 5: EXPR at 26:5
.node_5:
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x0, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      const 4
      db 0x2, 0x0, 0x0, 0x0
```

```
      mul
      add
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      const 4
      db 0x1, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      trunc 2
      store 2
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x0, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      const 4
      db 0x2, 0x0, 0x0, 0x0
      mul
      add
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      load 2
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_6
; 6: EXPR at 27:5
.node_6:
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x1, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      const 4
      db 0x2, 0x0, 0x0, 0x0
      mul
      add
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      const 4
      db 0x1, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      trunc 2
      store 2
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
```

```
      load 4
      const 4
      db 0x1, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      const 4
      db 0x2, 0x0, 0x0, 0x0
      mul
      add
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      load 2
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_7
; 7: EXPR at 29:11
.node_7:
      get fp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      const 4
      db 0x2, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      store 4
      get fp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_8
; 8: COND at 31:5
.node_8:
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x0, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      cmp gt
      ifz .return_void
      goto .node_9
; 9: EXPR at 32:9
.node_9:
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      sub
      set sp
```

```
      const 4
      dd .const_2
      call write_str
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_10
; 10: EXPR at 33:15
.node_10:
      get fp
      const 4
      db 0x10, 0x0, 0x0, 0x0
      sub
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      set sp
      call read_ulong
      store 4
      get fp
      const 4
      db 0x10, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_11
; 11: EXPR at 35:9
.node_11:
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      set sp
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      load 4
      get fp
      const 4
      db 0x10, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x1, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      add
      call increase_size
      store 4
      get fp
```

```
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_12
; 12: COND at 37:9
.node_12:
      get fp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      load 4
      get fp
      const 4
      db 0x10, 0x0, 0x0, 0x0
      sub
      load 4
      cmp le
      ifz .node_15
      goto .node_13
; 13: EXPR at 38:13
.node_13:
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      load 4
      get fp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x2, 0x0, 0x0, 0x0
      mul
      add
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      load 4
      get fp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x1, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      sub
      const 4
      db 0x2, 0x0, 0x0, 0x0
      mul
```

```
      add
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      load 2
      sext 2
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      load 4
      get fp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x2, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      sub
      const 4
      db 0x2, 0x0, 0x0, 0x0
      mul
      add
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      load 2
      sext 2
      add
      trunc 2
      store 2
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      load 4
      get fp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x2, 0x0, 0x0, 0x0
      mul
      add
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      load 2
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_14
; 14: EXPR at 39:13
.node_14:
      get fp
      const 4
      db 0xc, 0x0, 0x0, 0x0
```

```
      sub
      get fp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x1, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      add
      store 4
      get fp
      const 4
      db 0xc, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_12
; 15: EXPR at 42:9
.node_15:
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      sub
      set sp
      const 4
      dd .const_3
      call write_str
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_16
; 16: EXPR at 43:9
.node_16:
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      sub
      set sp
      get fp
      const 4
      db 0x8, 0x0, 0x0, 0x0
      sub
      load 4
      get fp
      const 4
      db 0x10, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x2, 0x0, 0x0, 0x0
      mul
      add
      const 4
      db 0x4, 0x0, 0x0, 0x0
```

```
      add
      load 2
      sext 2
      call write_long
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_17
; 17: EXPR at 44:9
.node_17:
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      sub
      set sp
      const 4
      dd .const_4
      call write_str
      get sp
      const 4
      db 0x2, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_18
; 18: EXPR at 45:9
.node_18:
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      const 4
      db 0x1, 0x0, 0x0, 0x0
      trunc 1
      zext 1
      sub
      store 4
      get fp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      sub
      load 4
      get sp
      const 4
      db 0x4, 0x0, 0x0, 0x0
      add
      set sp
      goto .node_8
.return_void:
      const 4
      db 0x0, 0x0, 0x0, 0x0
      trunc 2
.leave:
      store 2
      set fp
      get sp
```

```
      const 4
      db 0x10, 0x0, 0x0, 0x0
      add
      set sp
      ret
; constants
.const_1:
      db 0x13, 0x0, 0x0, 0x0, 0x4e, 0x75, 0x6d, 0x62, 0x65, 0x72, 0x20,
0x6f, 0x66, 0x20, 0x71, 0x75, 0x65, 0x72, 0x69, 0x65, 0x73, 0x3a, 0x20
.const_2:
      db 0x7, 0x0, 0x0, 0x0, 0x51, 0x75, 0x65, 0x72, 0x79, 0x3a, 0x20
.const_3:
      db 0x8, 0x0, 0x0, 0x0, 0x52, 0x65, 0x73, 0x75, 0x6c, 0x74, 0x3a,
0x20
.const_4:
      db 0x1, 0x0, 0x0, 0x0, 0xa
; builtin: char read();
read:
      get sp
      in
      store 1
      ret
; builtin int write(char c);
write:
      out
      ret
; builtin int[] new_int_array(ulong size);
new_int_array:
      get fp
      get sp
      set fp
      get fp
      const 4
      db 8, 0, 0, 0
      add
      const 4
      dd heap_end
      load 4
      store 4
      const 4
      dd heap_end
      load 4
      get fp
      const 4
      db 4, 0, 0, 0
      add
      load 4
      store 4
      const 4
      dd heap_end
      const 4
      dd heap_end
      load 4
      get fp
      const 4
      db 4, 0, 0, 0
      add
      load 4
      const 4
      db 2, 0, 0, 0
      mul
      const 4
```

```
      db 4, 0, 0, 0
      add
      add
      store 4
      set fp
      get sp
      const 4
      db 4, 0, 0, 0
      add
      set sp
      ret
; builtin ulong get_int_array_size(int[] array);
get_int_array_size:
      load 4
      get sp
      const 4
      db 4, 0, 0, 0
      add
      get sp
      const 4
      db 4, 0, 0, 0
      add
      load 4
      store 4
      get sp
      const 4
      db 4, 0, 0, 0
      add
      set sp
      ret
; builtin int write_str(string str);
write_str:
      get sp
      load 4
      load 4
      get fp
      get sp
      set fp
      get fp
      const 4
      db 8, 0, 0, 0
      add
      get fp
      const 4
      db 8, 0, 0, 0
      add
      load 4
      const 4
      db 4, 0, 0, 0
      add
      store 4
      goto .loop_cond
.loop:
      get fp
      const 4
      db 8, 0, 0, 0
      add
      load 4
      load 1
      out
      get fp
      const 4
```

```
      db 8, 0, 0, 0
      add
      get fp
      const 4
      db 8, 0, 0, 0
      add
      load 4
      const 4
      db 1, 0, 0, 0
      add
      store 4
      get fp
      const 4
      db 4, 0, 0, 0
      add
      get fp
      const 4
      db 4, 0, 0, 0
      add
      load 4
      const 4
      db 1, 0, 0, 0
      sub
      store 4
.loop_cond:
      get fp
      const 4
      db 4, 0, 0, 0
      add
      load 4
      const 4
      db 0, 0, 0, 0
      cmp eq
      ifz .loop
      set fp
      get sp
      const 4
      db 8, 0, 0, 0
      add
      set sp
      ret
; builtin byte ord(char c);
ord:
      sext 2
      trunc 1
      ret
; builtin char chr(byte c);
chr:
      goto ord
; heap
heap_end:
      dd heap
heap:
```

Пример работы

```
Number of queries: 9
Query: 2
Result: 2
Query: 3
Result: 3
Query: 4
Result: 5
Query: 8
Result: 34
Query: 9
Result: 55
Query: 10
Result: 89
Query: 18
Result: 4181
Query: 19
Result: 6765
Query: 20
Result: 10946
```

Вывод

В ходе выполнения данной лабораторной работы было написано описание архитектуры для эмулятора, а также реализован транслятор в ассемблер для этой архитектуры.