ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

Факультет «Программной Инженерии и Компьютерных Технологий»

Отчет о лабораторной работе №3

«Задание 3»

по дисциплине

«Языки Программирования»

Вариант 3

Выполнил:

Студент группы P4119

Эр Мерт

Проверил:

Доцент факультета ПИиКТ, к.т.н.

Кореньков Ю. Д.

Задание:

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.

Описание разработанных структур данных

ProgramImage: Контейнер для всех секций программы. Хранит списки `ImageEntry` для кода и данных.

```
public record ProgramImage(
        List<ImageEntry> text,
        List<ImageEntry> rodata,
        List<ImageEntry> data,
        List<ImageEntry> bss,
        List<FunctionSig> externFunctions // функции без тела, но с typeRef
) {}
```

FunctionUnit / ProgramUnit: Структуры, объединяющие CFG, сигнатуры функций и таблицы локальных переменных.

```
public record FunctionUnit(
        CfgFunction cfg,
        FunctionSig sig,
        Map<String, TypeRef> locals,
        boolean hasBody) {
}

public record ProgramUnit(
        List<FunctionUnit> functions,
        List<FunctionSig> externFunctions) {
}
```

ImageEntry: Иерархия объектов, представляющих строки ассемблера: `LabeledInstr` (инструкция с меткой), `LabelOnly` (метка), `LabeledData` (директивы `.ascii`, `.word`).

Программный интерфейс и особенности реализации

- **Интерфейс:** Основной метод `MipsBuilder.build(ProgramUnit)` принимает структуру программы и возвращает объект `ProgramImage`, готовый к записи в файл.

```java
public ProgramImage build(ProgramUnit unit) {
    strId = 0;
    genLabelId = 0;
    strPool.clear();
    intBufLabel = null;
    inBufLabel = null;

    List<ImageEntry> text = new ArrayList<>();
    List<ImageEntry> rodata = new ArrayList<>();
    List<ImageEntry> data = new ArrayList<>();
    List<ImageEntry> bss = new ArrayList<>();

    boolean hasMain = unit.functions().stream().anyMatch(f -> f.hasBody() && f.sig().name().equals("main"));
    if (hasMain) {
        emitStart(text);
    }

    for (FunctionUnit fn : unit.functions()) {
        if (!fn.hasBody()) continue;
        emitFunction(text, rodata, bss, fn);
    }

    return new ProgramImage(
            List.copyOf(text),
            List.copyOf(rodata),
            List.copyOf(data),
            List.copyOf(bss),
            unit.externFunctions() == null ? List.of() : List.copyOf(unit.externFunctions()));
}
```

- **Генерация кода:**
  - **Управление стеком:** В прологе используется `addiu $sp, $sp, -frameSize`. Сохраняются старый `FP` и адрес возврата `RA`.
  - **Регистры:** Для вычислений используются временные регистры `$t0-$t3`. Аргументы функций передаются через `$a0-$a3`. Результат возвращается в `$v0`.
  - **Линеаризация:** Чтобы минимизировать количество переходов, блоки обходятся методом `reversePostOrder`. Если следующий в списке блок совпадает с целью безусловного перехода, инструкция `b` опускается.
  - **Пул строк:** Все строковые литералы автоматически выносятся в секцию `.rodata` с уникальными метками `.LCx`.

Ввод

```
function in() as int
end function

function out(a as int)
end function

function readInt() as int
    dim c, sign, n as int

    n = 0;
    sign = 1;

    c = in();

    ' boşluk / newline / tab atla
    while 1
        if c == 32 then
            c = in();
        else if c == 10 then
            c = in();
        else if c == 13 then
            c = in();
        else if c == 9 then
            c = in();
        else
            break
        end if
    wend

    ' işaret
    if c == 45 then
        sign = 0 - 1;     ' -1
        c = in();
    end if

    ' rakamlar: sadece 48..57 ise işle, değilse çık
    while 1
        if c >= 48 then
            if c <= 57 then
                n = (n * 10) + (c - 48);
```

```
            if c <= 57 then
                n = (n * 10) + (c - 48);
                c = in();
            else
                break
            end if
        else
            break
        end if
    wend

    n = n * sign;
    readInt = n;
end function

function writeInt(n as int)
    dim x, q, r as int

    x = n;

    if x < 0 then
        out(45);          ' '-'
        x = 0 - x;
    end if

    if x < 10 then
        out(48 + x);      ' tek basamak
    else
        ' q = x / 10 ve r = x % 10 (sadece çıkarma ile)
        r = x;
        q = 0;
        while r >= 10
            r = r - 10;
            q = q + 1;
        wend

        writeInt(q);
        out(48 + r);
    end if
end function
```

Вывод

```
PS C:\Users\Mert\task1> & c:/Users/Mert/task1/.venv/Scripts/Activate.ps1
(.venv) PS C:\Users\Mert\task1> $env:PYTHONPATH = "$PWD\src"
(.venv) PS C:\Users\Mert\task1> $ASSEMBLE_ID = & .\tools\ptp\Portable.RemoteTasks.Manager.exe `
>>   -ul 503291 -up "b551163b-6741-4c92-b016-01f99e8a9947" `
>>   -id -s Assemble -w `
>>   asmListing out3\fib_variant27_2addr.asm `
>>   definitionFile tools\ptp\variant27_2addr.target.pdsl `
>>   archName variant27_2addr
>>
>> $ASSEMBLE_ID
bbe52e78-430b-4bb4-b928-bd79e94471c6

(.venv) PS C:\Users\Mert\task1>  & .\tools\ptp\Portable.RemoteTasks.Manager.exe `
>>   -ul 503291 -up "b551163b-6741-4c92-b016-01f99e8a9947" `
>>   -g bbe52e78-430b-4bb4-b928-bd79e94471c6 -r out.ptptb -o out3\fib_variant27_2addr.ptptb
(.venv) PS C:\Users\Mert\task1> dir out3\fib_variant27_2addr.ptptb


    Directory: C:\Users\Mert\task1\out3


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        28.01.2026     19:52            810 fib_variant27_2addr.ptptb


(.venv) PS C:\Users\Mert\task1> "5`n" | Set-Content -Encoding ascii out3\in.txt
>>   -ul 503291 -up "b551163b-6741-4c92-b016-01f99e8a9947" `
>>   -id -s ExecuteBinaryWithInput -w `
>>   stdinRegStName rin `
>>   stdoutRegStName rout `
>>   inputFile out3\in.txt `
>>   definitionFile tools\ptp\variant27_2addr.target.pdsl `
>>   archName variant27_2addr `
>>   binaryFileToRun out3\fib_variant27_2addr.ptptb `
>>   codeRamBankName code `
>>   ipRegStorageName ip `
>>   finishMnemonicName hlt
>>
>> $RUNID
5f1f61cc-c00f-4ddd-8f8a-05e7e2a01751

(.venv) PS C:\Users\Mert\task1> & .\tools\ptp\Portable.RemoteTasks.Manager.exe `
>>   -ul 503291 -up "b551163b-6741-4c92-b016-01f99e8a9947" `
>>   -g 5f1f61cc-c00f-4ddd-8f8a-05e7e2a01751 -r stdout.txt -o out3\fib_stdout.txt
>>
>> Get-Content out3\fib_stdout.txt | Out-Host
8
```

Вывод:

В ходе выполнения практического задания №3 была разработана стековая виртуальная машина по варианту и реализован модуль трансляции, формирующий линейный код (в мнемонической форме) на основе графов потока управления, полученных на предыдущем этапе. Проведённое тестирование на наборе программ подтвердило корректность генерации кода для основных конструкций языка (арифметика, логика, ветвления, циклы, break, вызовы функций и рекурсия), а также работоспособность полученного результата при запуске.