

Портирование компилятора ВТРС64 на операционную систему macOS

Выполнена студентом группы ИУ9-72Б
Мамаевым Алексеем

Москва, 2021

ВТРС

“BeRo TinyPascal Compiler”

Компилятор подмножества языка Pascal

Написан в 2006 году для Windows x32 Бенджамином Россо

Самоприменим

Используется в лабораторных работах курса

“Конструирование компиляторов”

ВТРС64

Перенесенная версия ВТРС на Linux 64-bit

Портирована Антоном Беляевым в 2017 году

Сохранены основные принципы, в том числе и
самоприменимость

Постановка задачи

1. Изучить устройство существующих ВТРС и ВТРС64
2. **Портировать** ВТРС64.pas на платформу **macOS 64-bit**, сохранив основные принципы компилятора
3. **Перенести** ВТРС64 на macOS 64-bit
4. Протестировать, убедиться в самоприменимости

Устройство ВТРС64

Составляющие:

- низкоуровневая “библиотека” RTL
- высокоуровневый компилятор ВТРС64.pas

Устройство ВТРС64: RTL

RunTime Library – реализация основных системных функций:

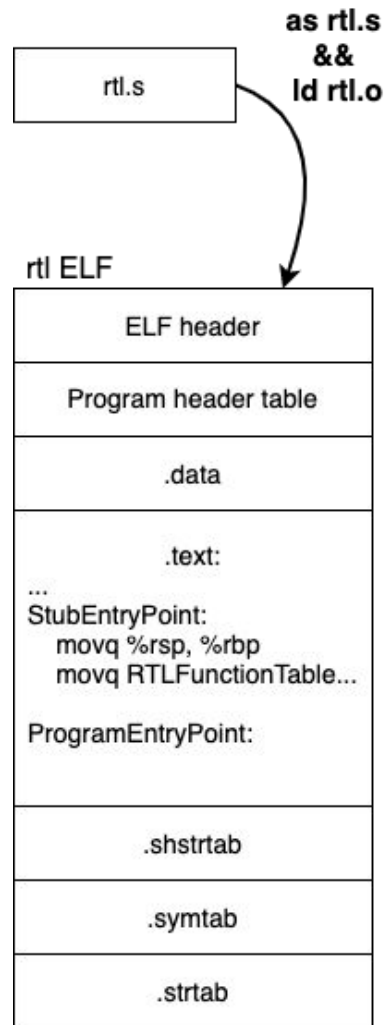
- чтение из `stdin` символов и чисел
- запись в `stdout` символов и чисел
- завершение работы программы

Устройство ВТРС64: RTL

Написана на языке ассемблера (синтаксис AT&T)

Точка входа указывает
на последнюю строку файла

После компиляции при запуске
происходит Segmentation fault



Устройство ВТРС64: компилятор

Написан на языке Pascal

Единый файл ВТРС64.pas: анализ и синтез

Входной поток преобразуется в байт-код

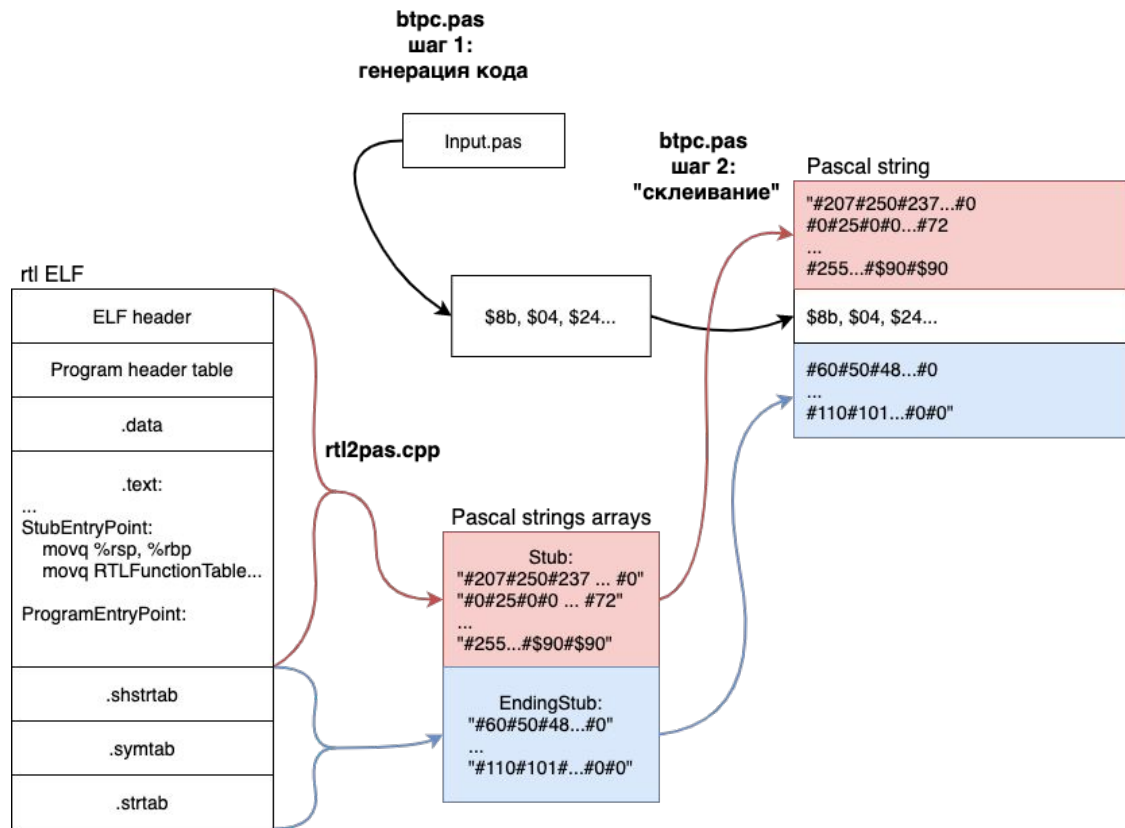
Байт-код преобразуется в ассемблерные инструкции:

вычисления на стеке, вызов функций RTL

Устройство ВТРС64: компилятор

Скомпилированный RTL
разбивается на 2 части

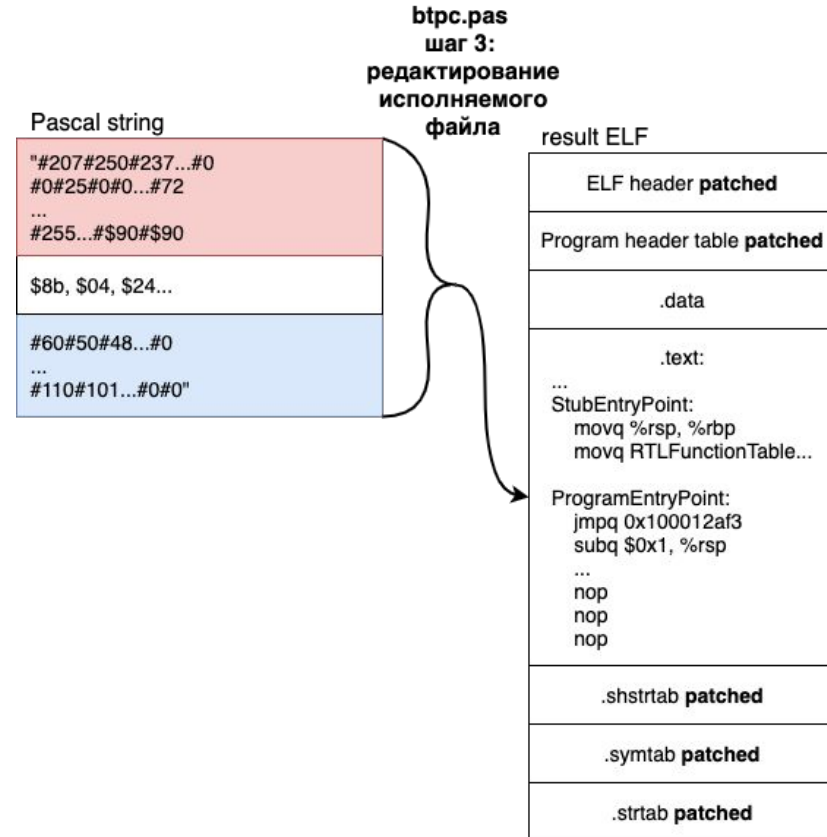
Происходит “инъекция”
сгенерированного кода



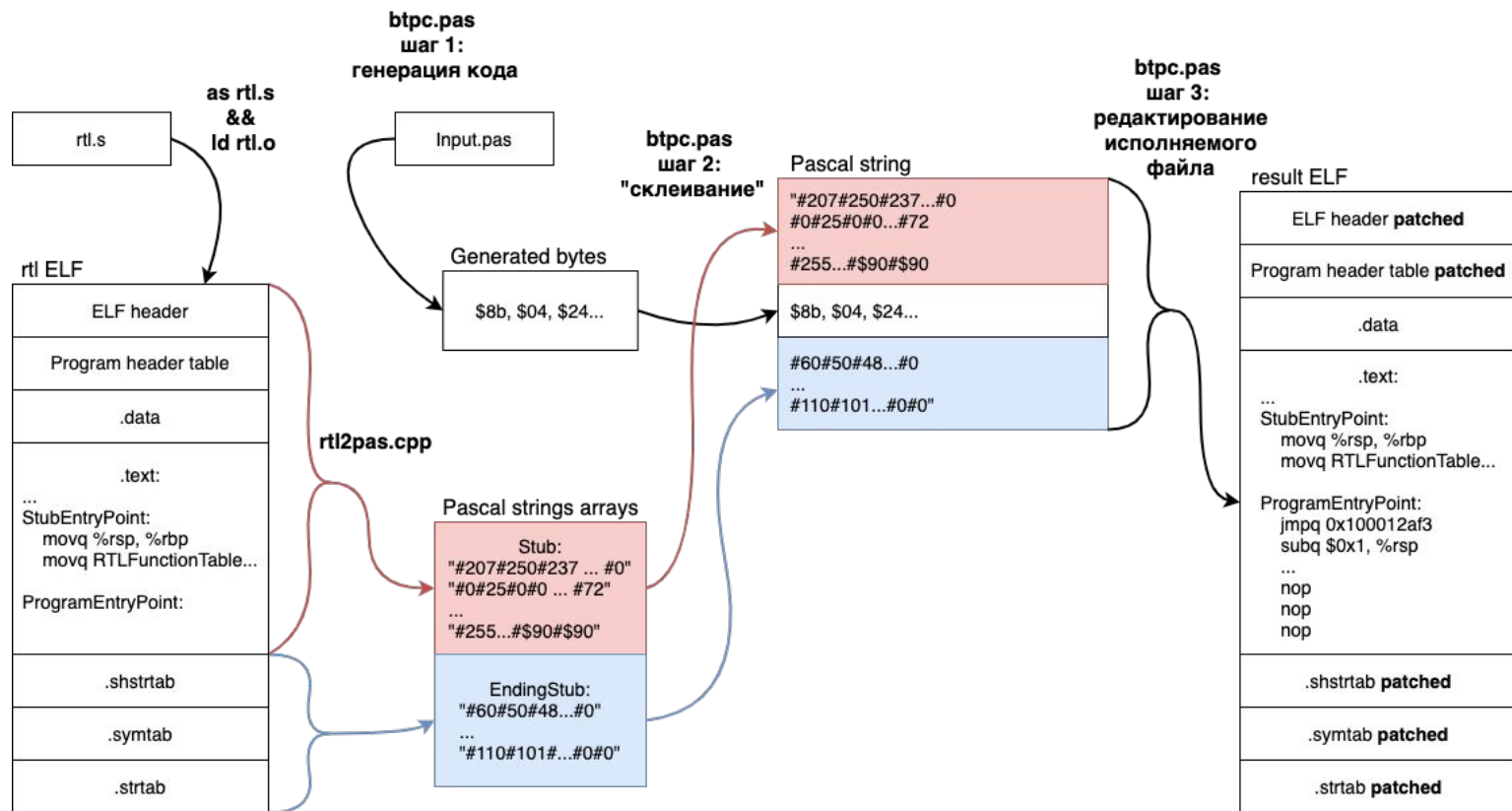
Устройство ВТРС64: компилятор

Получен ELF-файл с инструкциями RTL и порожденными компилятором инструкциями

Остается “пропатчить” заголовки ELF-файла



Устройство ВТРС64



Портирование: RTL

Синтаксис и разрядность сохраняются, но есть **особенности** ассемблерного кода на macOS.

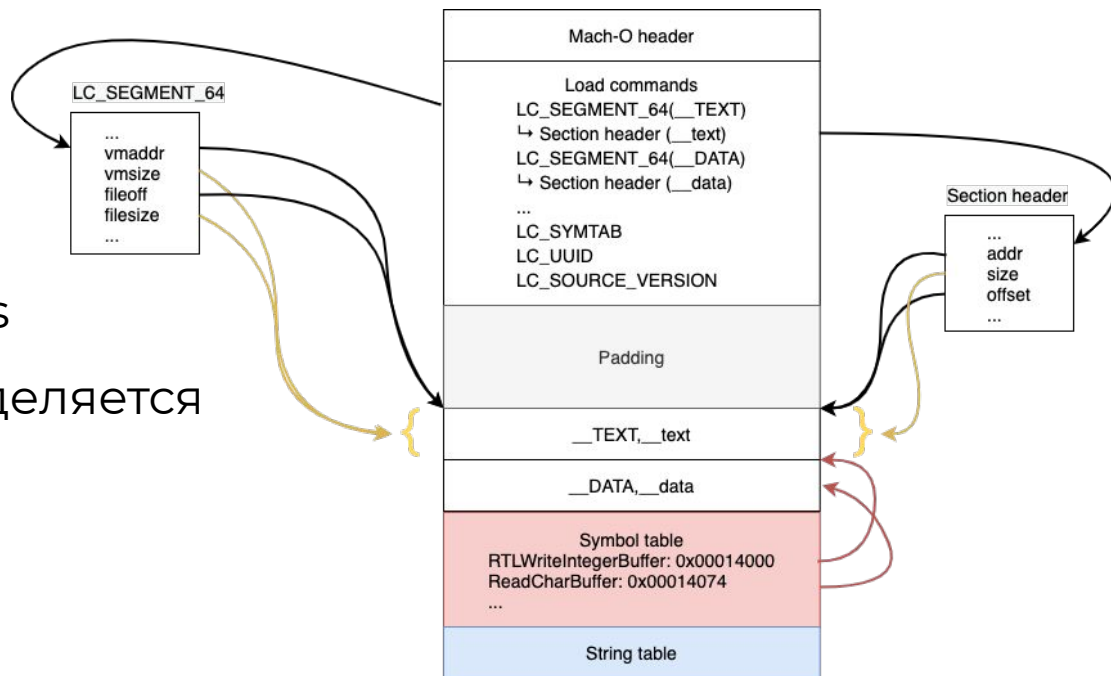
Требуется поддерживать другой формат исполняемых файлов – Mach-O.

```
232
233     movq $0x2000004,    %rax           # syscall 4 == Write
234     movq $1,    %rdi           # p1, write_to == 1 == stdout
235     movq RTLWriteIntegerBuffer@GOTPCREL(%rip), %rsi  # p2, write_from == buf addr
236     movq %rcx, %rdx           # p3, count
237
238     syscall
```

Портирование: Mach-O

Mach-O схож с ELF, но имеет отличия:

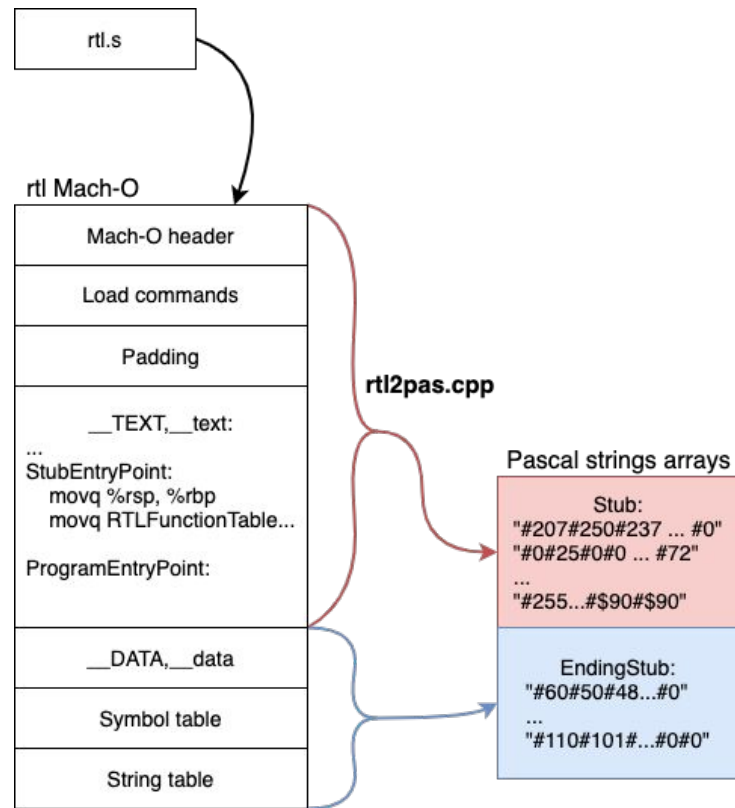
- другие структуры
- “отступ”
- явное задание сегментов
- единые load commands
- размер сегмента определяется в самом Mach-O



Портирование: парсинг RTL

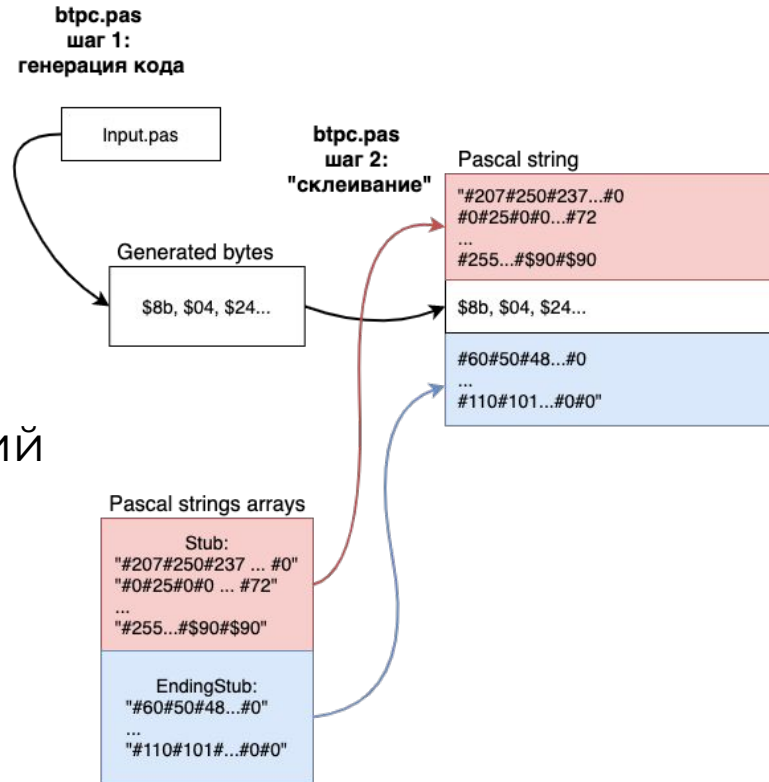
1. Вычитываются load commands
2. Вычисляется начало **секции** `__data`
3. Исполняемый файл побайтово разбивается на 2 Pascal-строки

Изменить порядок следования секций в Mach-O **НЕВОЗМОЖНО**



Портирование: компилятор

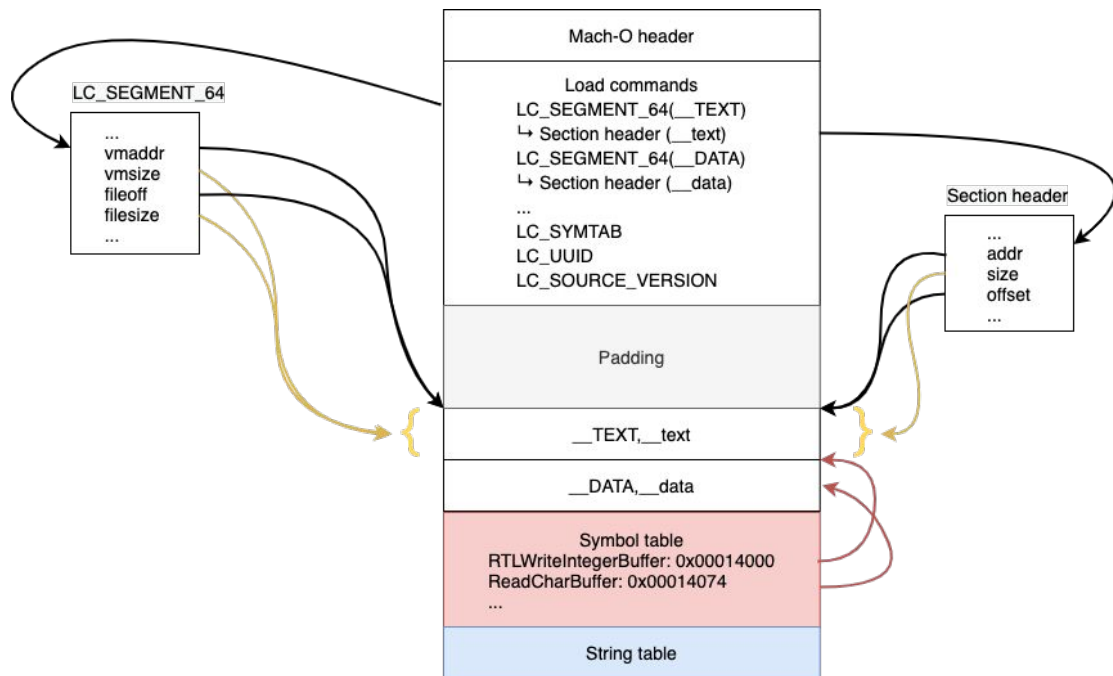
1. Задается шаблон выходного файла – “разрезанный” RTL
2. Генерируется код, вставляется в шаблон
3. Секция `__text` дополняется инструкциями NOP
4. Происходит редактирование значений в исполняемом Mach-O файле



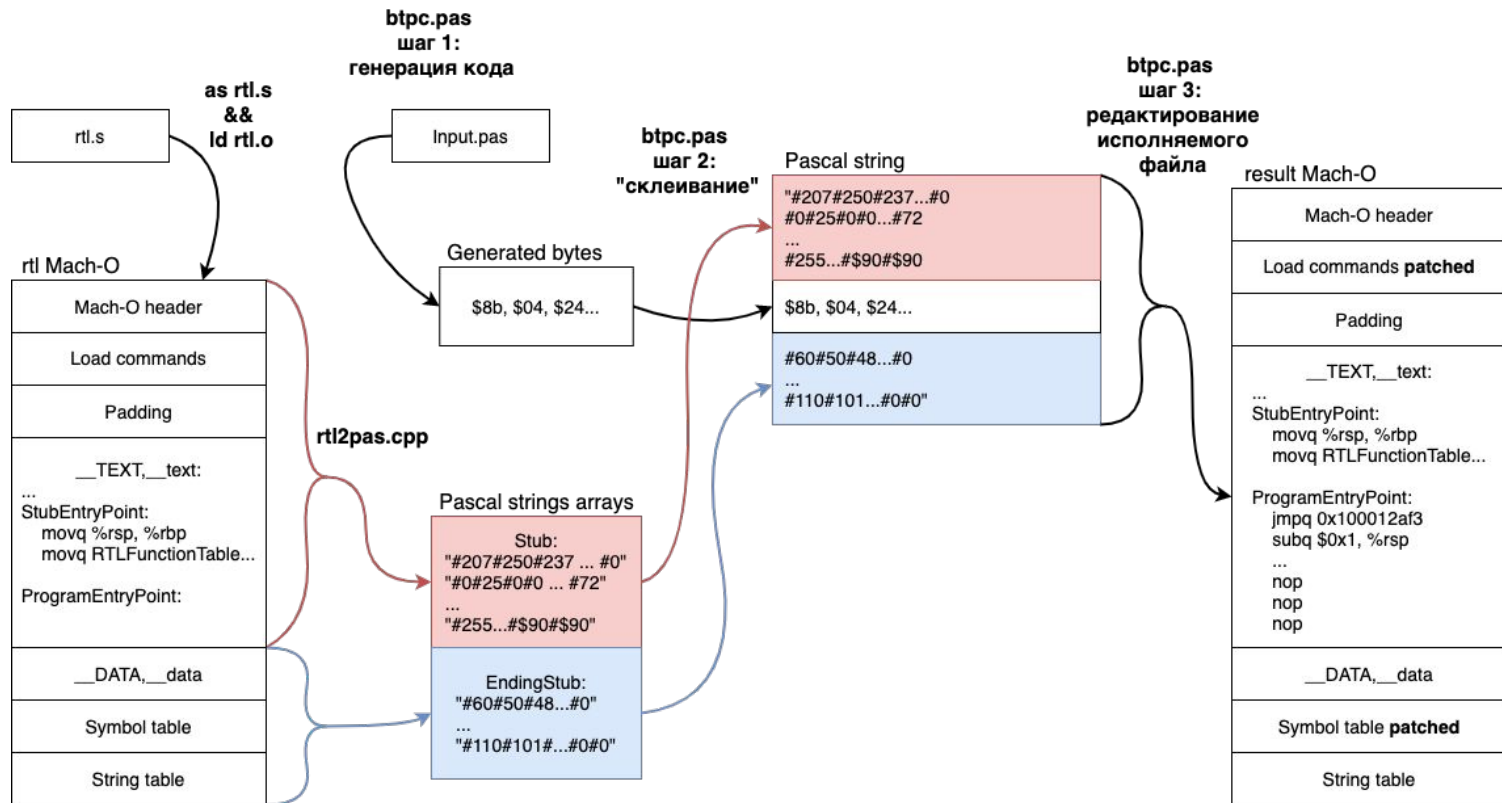
Портирование: патчевание

Требуется отредактировать:


- **размер** секции `__text` и сегмента `__TEXT`
- **адреса и смещения** остальных секций и смещений
- **адреса** буферов в таблице символов
- **адреса** в инструкциях



Устройство ВТРС64macOS



Технологии

- ASM – библиотека RTL
- Pascal – компилятор, тесты
- C++ – парсинг Mach-O файла
- htool, MachOViewer 
- Git – параллельная разработка на двух устройствах



Тестирование

1. “Поведенческие” тесты отдельных функций RTL
2. Программы малого размера для тестирования компиляторов

```
540  Test4:
541      pushq $-1234
542      pushq $6
543      call RTLWriteInteger
544      call RTLWriteLn
545      call RTLHalt
```

```
1  program RTLWriteTest;
2
3  type TSignature = array[1..5] of integer;
4
5  var a,i:integer;
6      m:TSignature;
7
8  begin
9
10     i:=1;
11     while i <= 3 do begin
12         Read(a);
13         WriteLn('->', a);
14         m[i]:=a;
15         i:=i+1;
16     end;
17
18     i:=1;
19     while i <= 3 do begin
20         WriteLn(m[i]);
21         i:=i+1;
22     end;
23 end.
```

Перенос

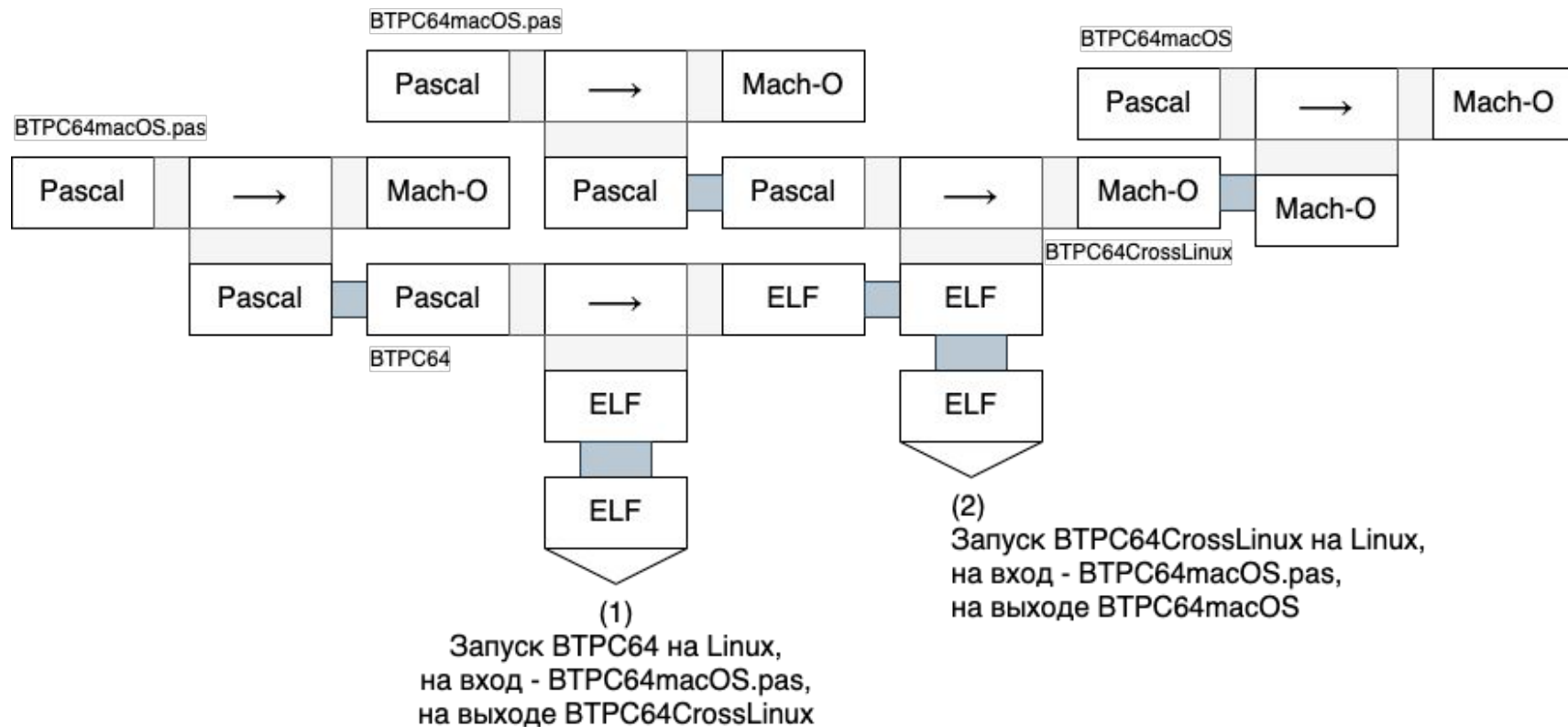
Получен компилятор ВТРС64macOS.pas, имеется раскрученный самоприменимый компилятор {ВТРС64.pas, ВТРС64}.

Для переноса требуется:

1. На исходной платформе скомпилировать кросскомпилятор ВТРС64CrossLinux
2. С его помощью на исходной платформе скомпилировать ВТРС64macOS

Получен раскрученный самоприменимый компилятор {ВТРС64macOS.pas, ВТРС64macOS}

Перенос



Заключение

- **Портирован** компилятор BTPC на платформу macOS 64-bit
- Изучено устройство **исполняемых файлов** PE32, ELF, Mach-O
- Исследована архитектура исходных версий компилятора
- На практическом примере рассмотрены процессы **компиляции**, **кросскомпиляции** и **раскрутки** компилятора

<https://github.com/AleksMa/A-Bauman-BTPC-macOS>

Спасибо за внимание