# Теория компиляторов

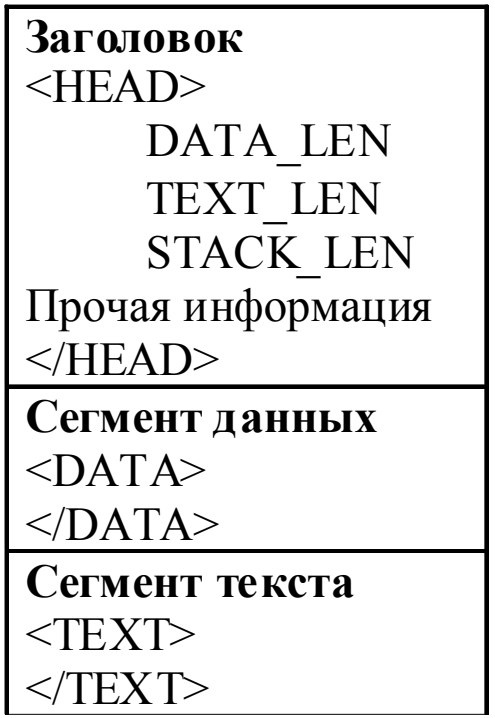
## Часть II

## Лекция 4. Объектный файл и виртуальная машина

# Формат объектного файла

3 основных сегмента:

- сегмент текста (исполняемые инструкции);
- сегмент данных (располагаются глобальные данные);
- сегмент стека (временные переменные).



# Сегменты

- **Заголовок**. <HEAD> </HEAD> Содержит необходимую управляющую информацию:
  - DATA_LEN – размер сегмента данных в словах.
  - TEXT_LEN – количество командных слов в сегменте текста.
  - STACK_LEN– размер сегмента стека в словах.
  - прочая служебная информация, определяющая конфигурацию ВМ: количество ФУ, размер регистрового файла и т.п. => гибкая система.
- **Сегмент данных.** <DATA> </DATA> . Содержит таблицу имен. Количество имен = DATA_LEN.

  Описание имени (тег):
  - Тип (число (0), строка (1))
  - Вид (константа (0), переменная (1))
  - Значение.

  Запись в сегменте данных – это тройка

  *(вид, тип, значение)*
- **Сегмент текста.** <TEXT> </TEXT>. Содержит командные слова (с произвольным количеством слогов).

  $(Тетрада_1; Тетрада_2; \ldots Тетрада_n)$

  $Тетрада_i=(OP, A_1, A_2, R)$

# Пример объектного файла

```
<HEAD>
  DATA_LEN = 6
  TEXT_LEN = 4
  STACK_LEN = 20
  ; Описание архитектуры ВМ
  FU_NUM = 3                    ; Количество функциональных устройств в ВМ
  RF_SIZE = 10                  ; Количество регистров в регистровом файле
</HEAD>
<DATA>
  (1,0,50)                      ; 000: переменная, число, 50
  (1,0,3.14)                    ; 001: переменная, число, 3.14
  (0,1,”qwerty”)                ; 002: константа, строка символов
  (1,0,1)                       ; 003: переменная, 1
  (1,1,”any string”)            ; 004: переменная, строка символов
  (0,0,0)                       ; 005: константа, число, 0
</DATA>
<TEXT>
  ((+,0,1,6))
  ((+,1,6,6);(*,3,5,7);(:=,1,5,))
  ((out,4,,);(+,7,3,1))
  ((out,1,,))
</TEXT>
```

# Образ программы в памяти

## Загрузчик

- **Размещение**. Выделение места для программы в оперативной памяти.
- **Перемещение**. Настройка всех величин в программе, зависящих от физических адресов в соответствии с адресами выделенной программе памяти.
- **Загрузка**. Фактическое размещение структур объектной программы в памяти и инициация ее выполнения.

| **Адрес** | **Тег** | |
|---|---|---|
| 000<br>001<br>002<br>…<br>DATA_LEN-1 | | **Сегмент данных**<br>DATA_LEN ячеек |
| DATA_LEN<br>DATA_LEN+1<br>DATA_LEN+2<br>…<br>DATA_LEN+STACK_LEN-1 | | **Сегмент стека**<br>STACK_LEN ячеек |
| DATA_LEN+STACK_LEN | | **Сегмент текста**<br>TEXT_LEN<br>командных слов |

# Архитектура виртуальной машины

## Особенности ВМ

1. Содержит N функциональных устройств.
2. Каждое ФУ имеет M регистров общего назначения и регистр флагов.
3. ВМ использует микропрограммное управление.
4. Организация памяти – теговая
5. Существует общий для всех ФУ регистровый файл.

# Базовые инструкции ВМ

| **Операции регистр-память** | | |
|---|---|---|
| load A, R | загрузка аргумента, находящегося по адресу A, в регистр R: A→R | |
| store R, A | сохранение регистра R в ячейке памяти по адресу A: R→A | |
| **Операции регистр-регистр** | | |
| set R1, R2 | R2 := R1 | |
| | | |
| **Операции память-память** | | |
| mov A1, A2 | A2 := A1 | |

| **Арифметические операции** | | |
|---|---|---|
| sum R1, R2, R3 | R3 := R1 + R2 | |
| sub R1, R2, R3 | R3 := R1 - R2 | |
| mul R1, R2, R3 | R3 := R1 * R2 | |
| div R1, R2, R3 | R3 := R1 / R2 | |
| **Операции ввода-вывода** | | |
| out R | вывод R на “терминал” | |
| in R | ввод значения R с “терминала” | |
| **Операции управления** | | |
| BR | | |
| BRZ | | |
| BRP | | |
| BRM | | |
| **Прочие операции** | | |
| exit | завершить выполнение программы | |
| nop | пустая инструкция | |
| | | |
| | | |

# Микрокоманды

```
mc “+” A1 A2 A3
  load A1,RA
  load A2,RB
  sum RA, RB, RC
  store RC, A3
#

(+, 101, 240, 012)
  load 101,RA
  load 240,RB
  sum RA, RB, RC
  store RC, 012
```