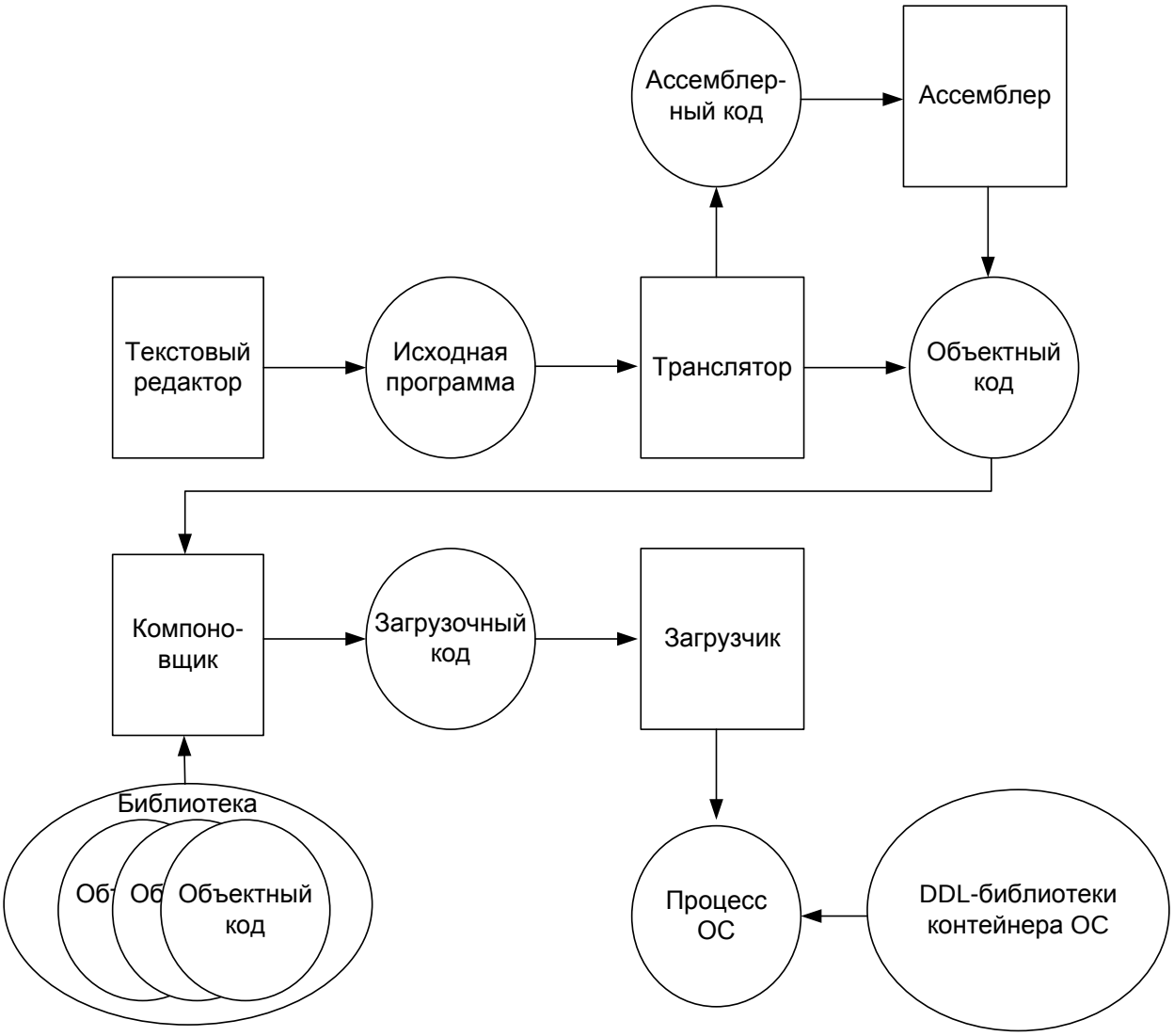
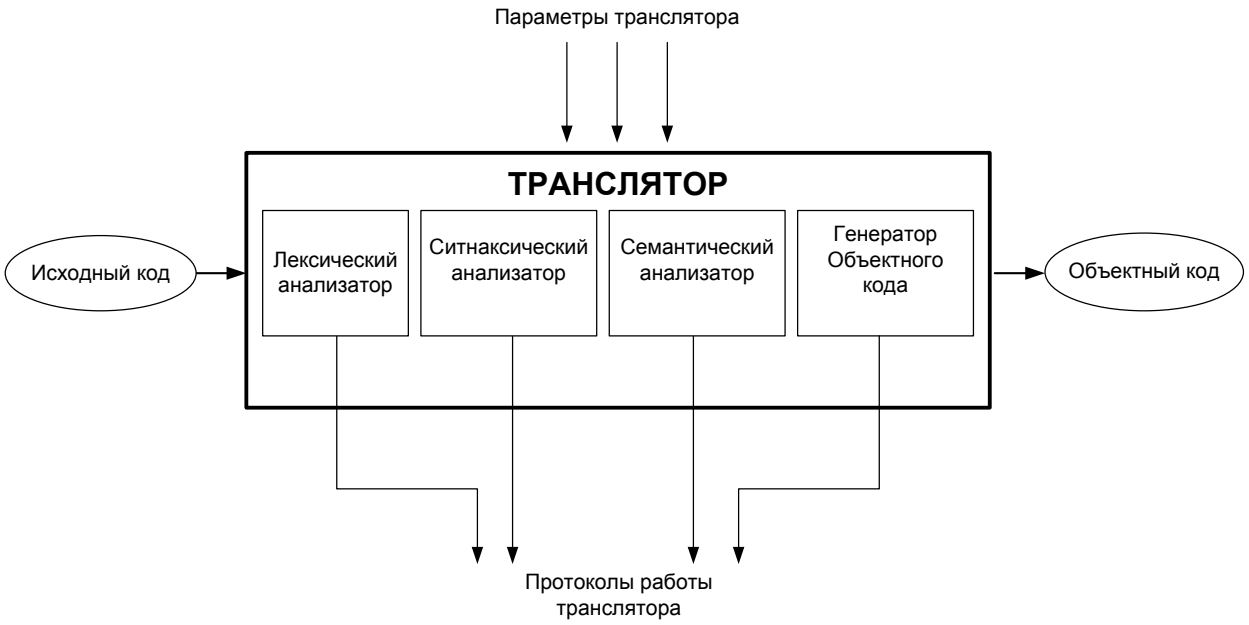
Лекция 23
БГТУ, ФИТ, ПОИТ, 3 семестр, Языки программирования

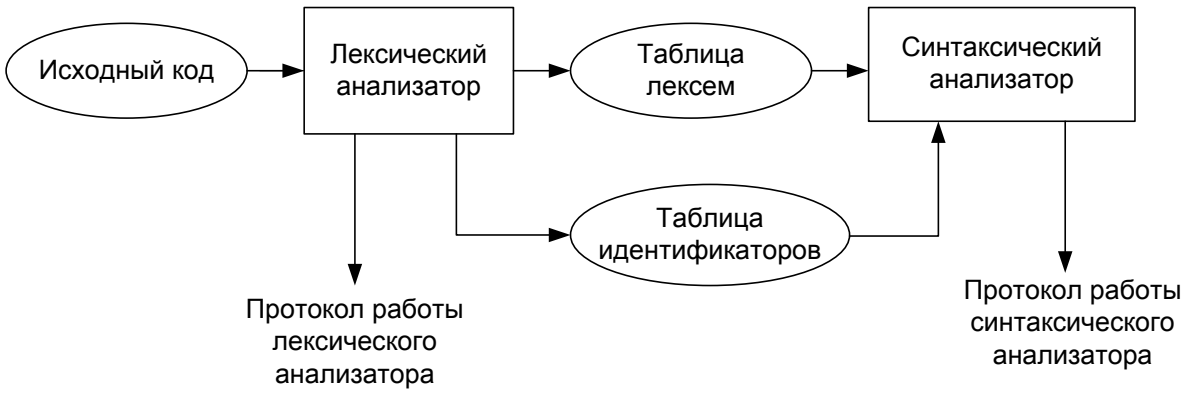
Генерация кода

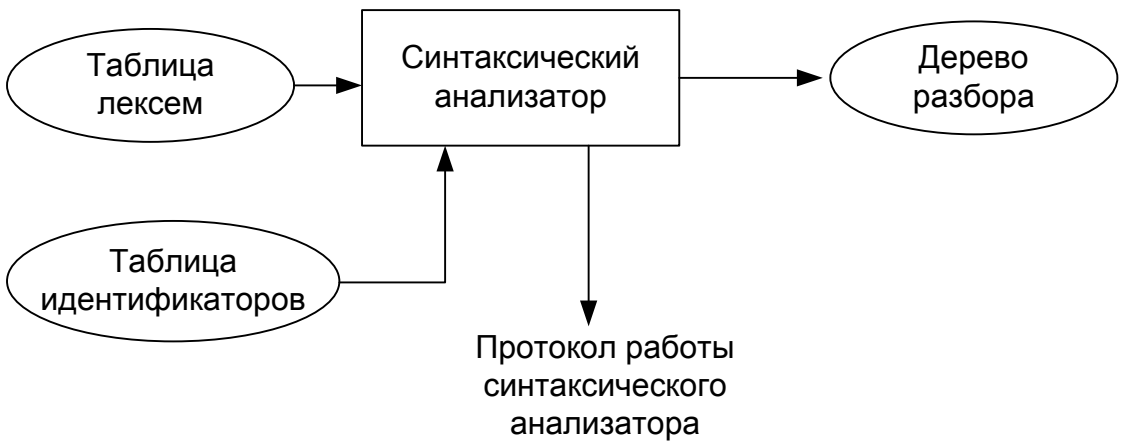
1. Система программирования

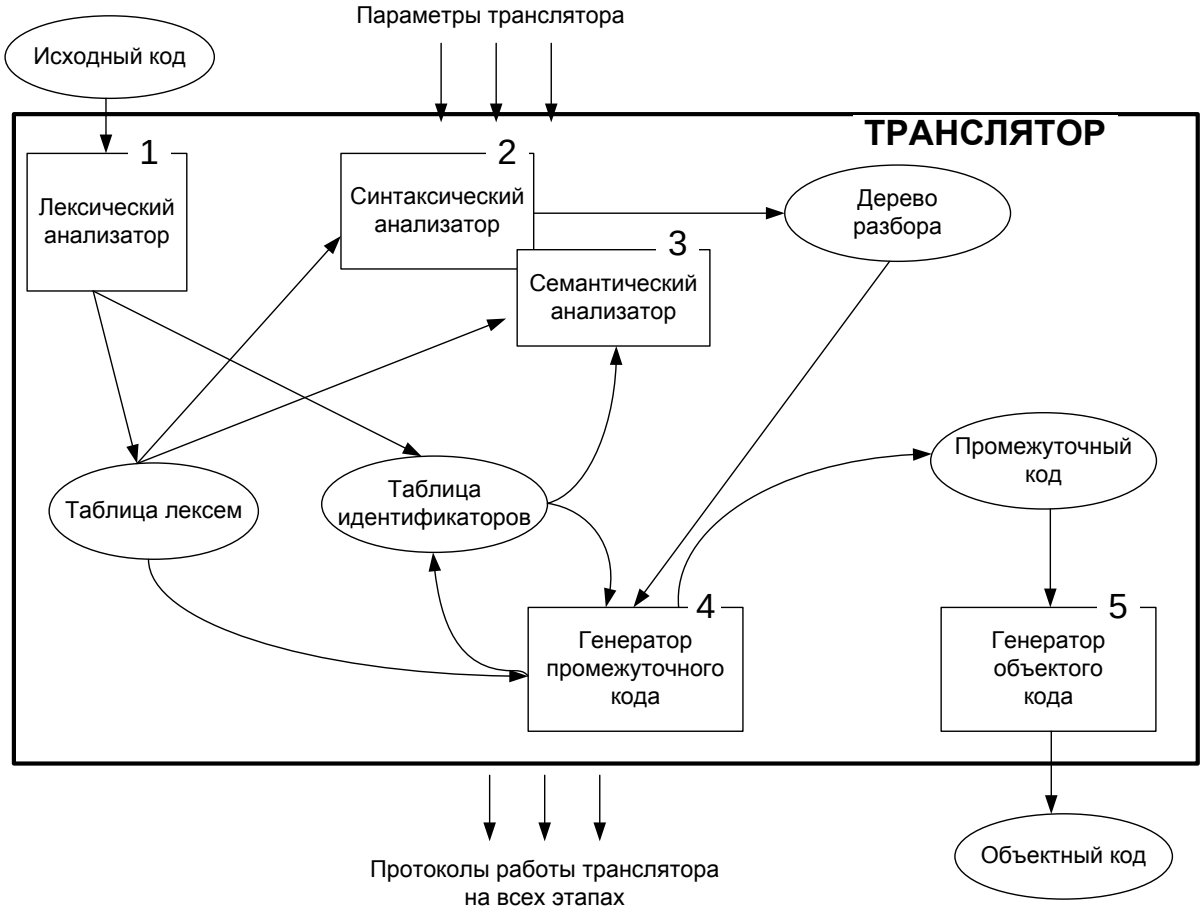


2. **Транслятор:** общая схема, фазы трансляции

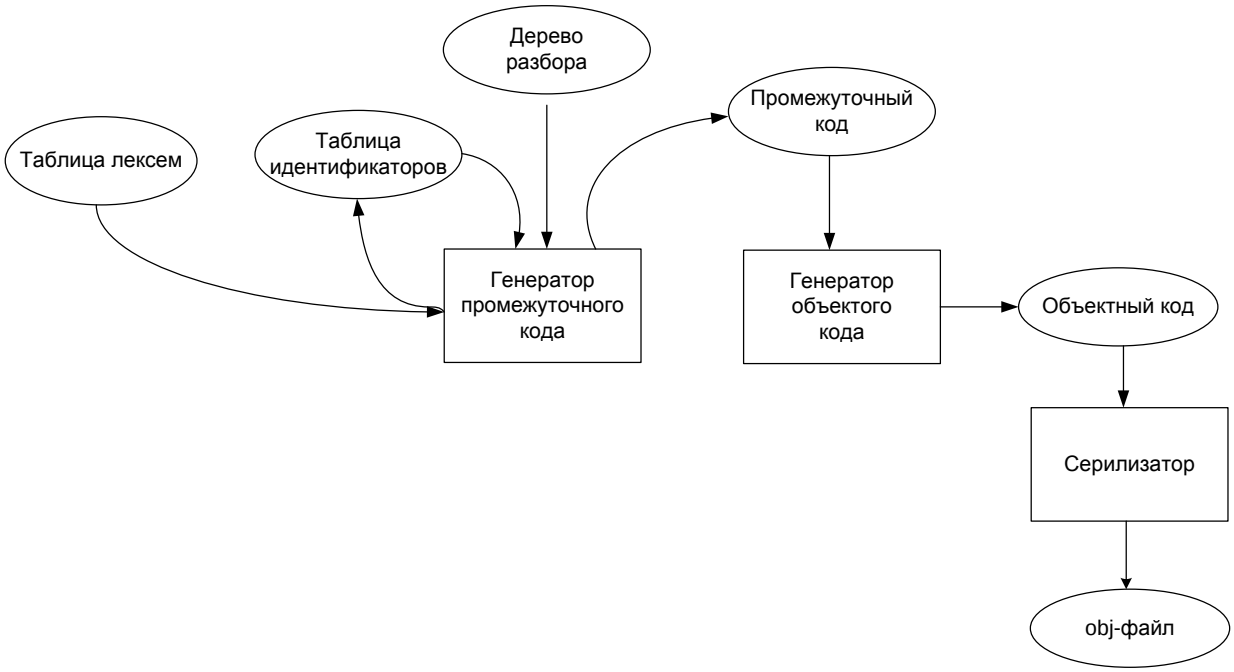




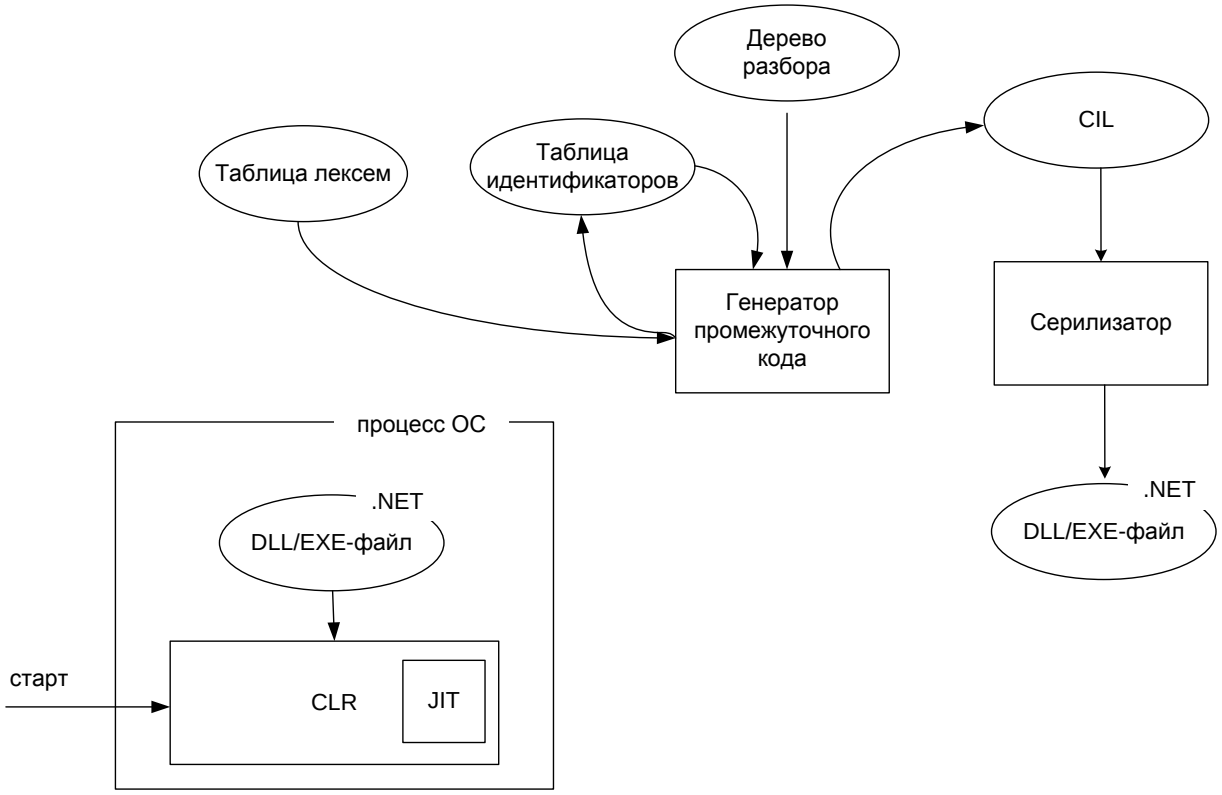




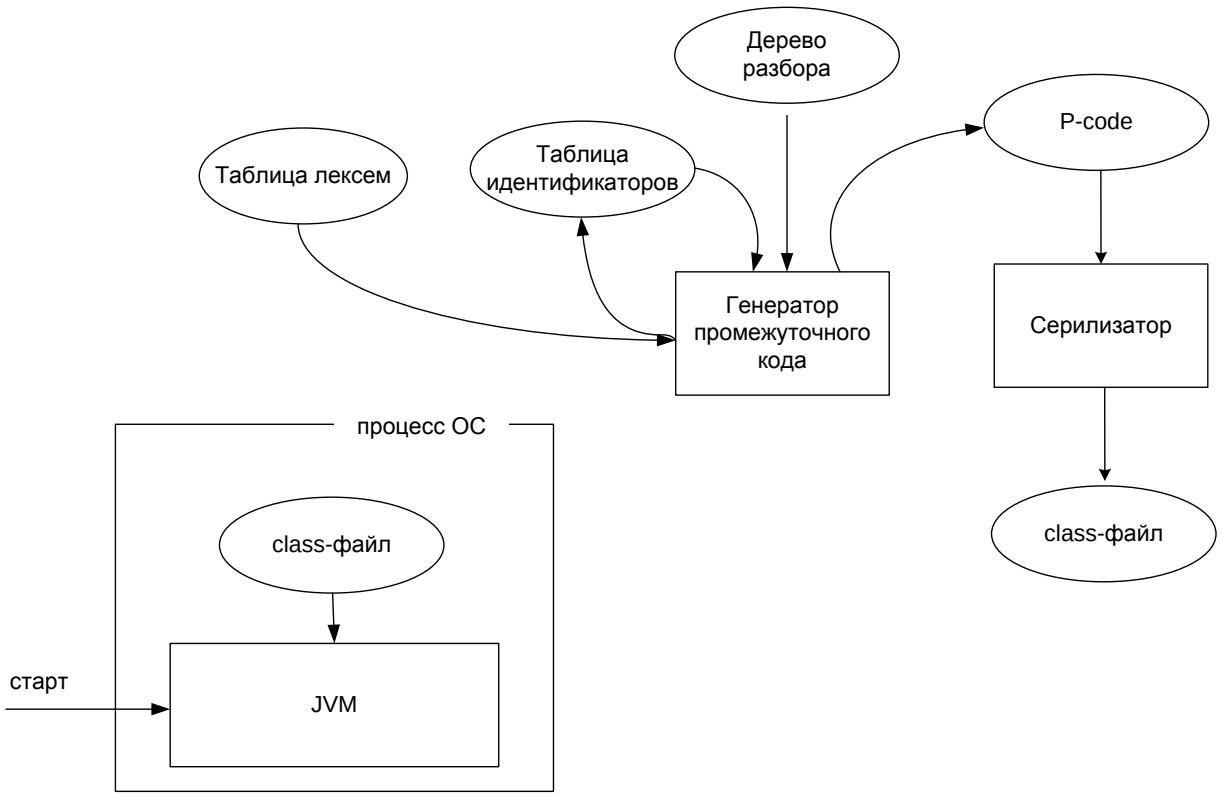
3. **Генерация кода C/C++**



4. **Генерация кода языка .NET (C#, VB.NET)**



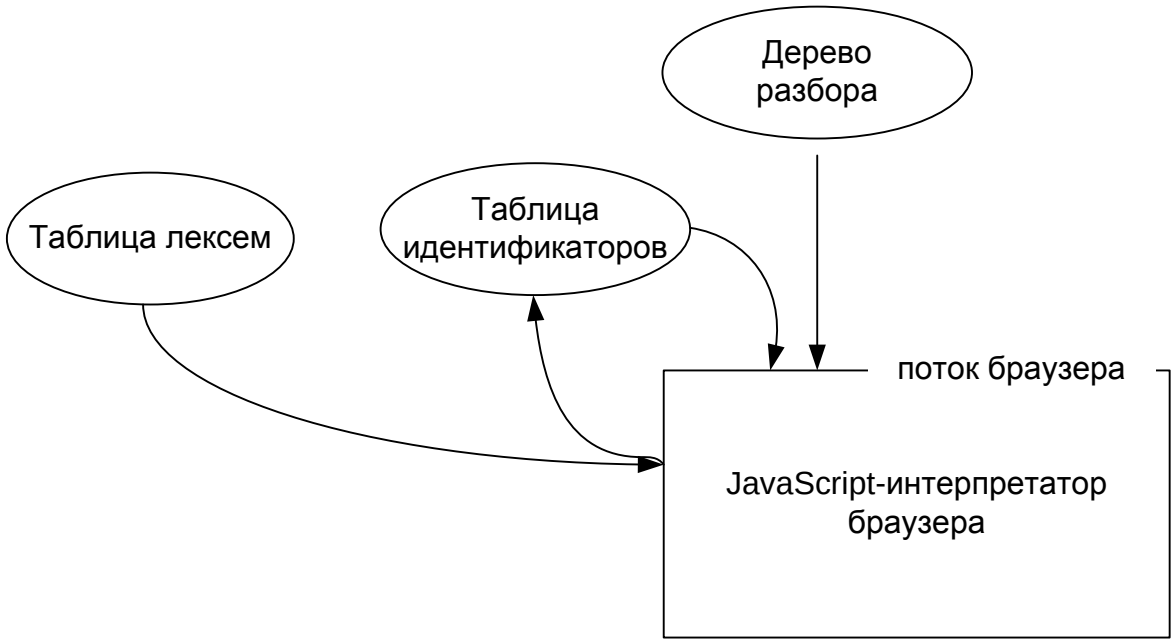
5. **Генерация кода языка Java**



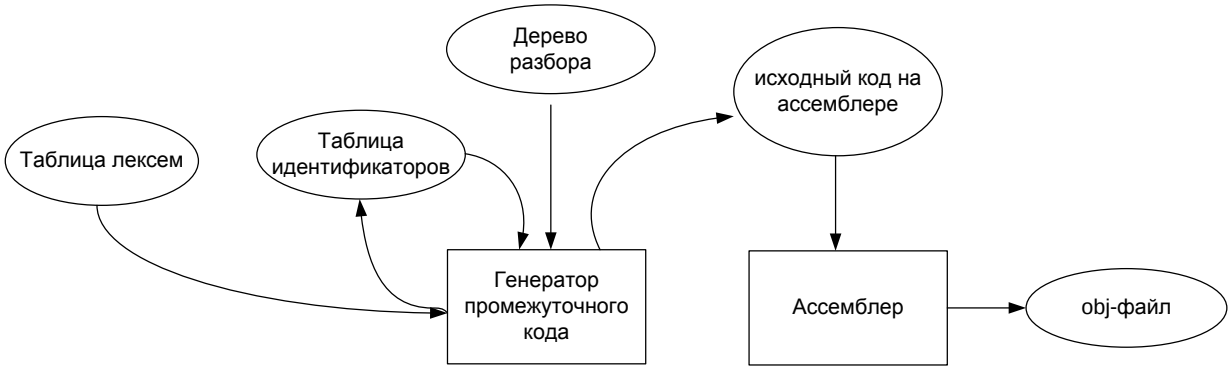
6. **Сериализация:** процесс преобразование структуры данных в последовательность битов.

7. **Десериализация:** процесс преобразования последовательности битов в структуру данных.

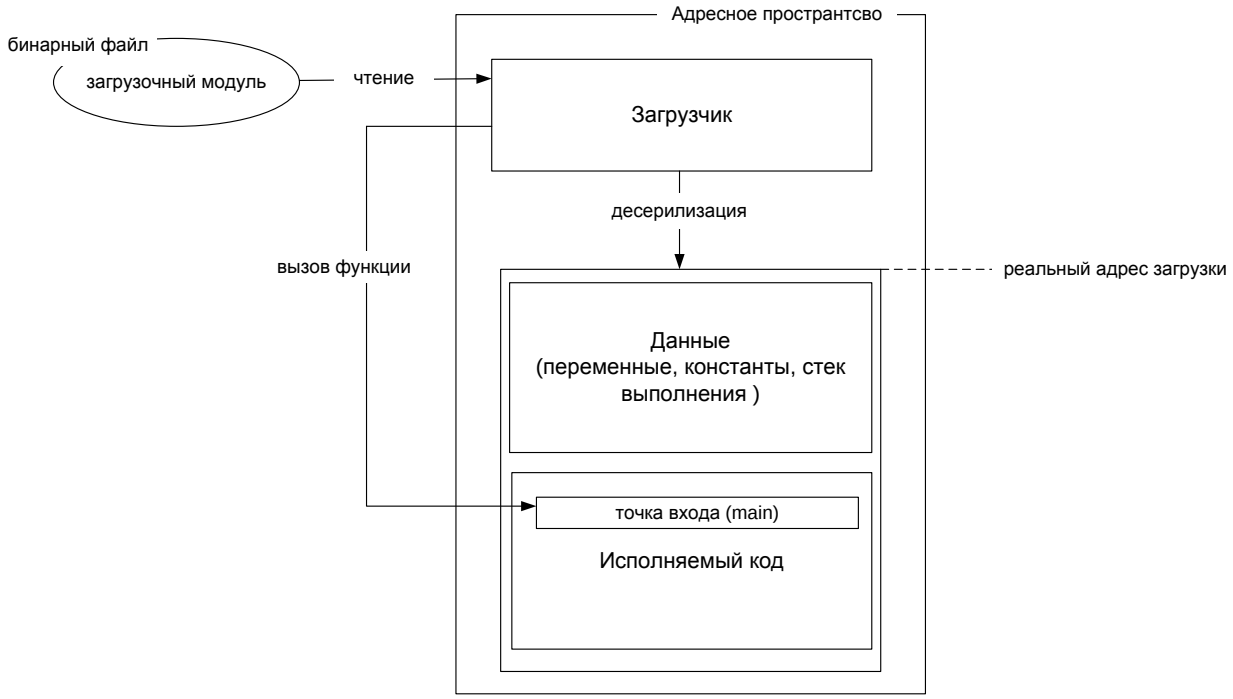
8. **Интерпретация кода JavaScript**



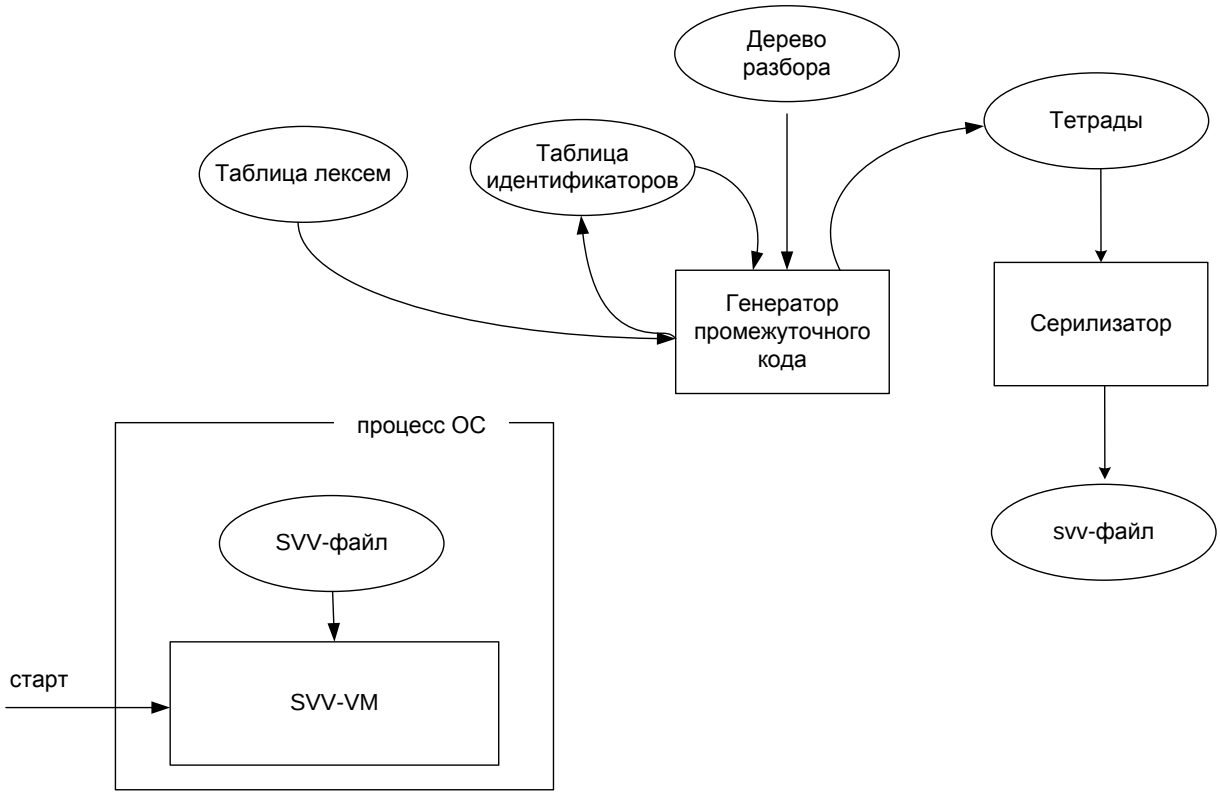
9. **Генерация исходного ассемблерного кода**



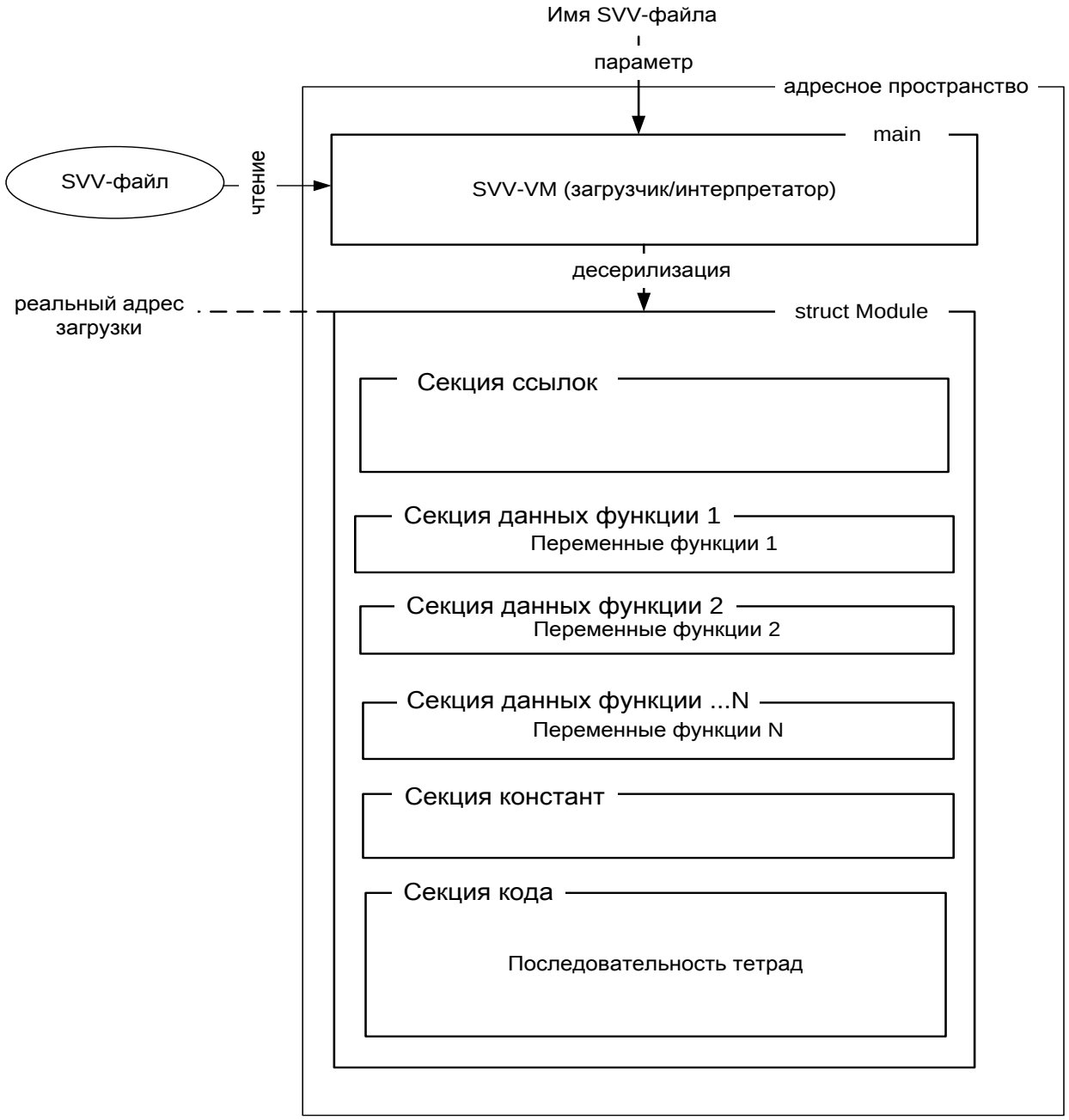
10. **Загрузка/десерилизация**



11. **Принцип реализации SVV-2015**

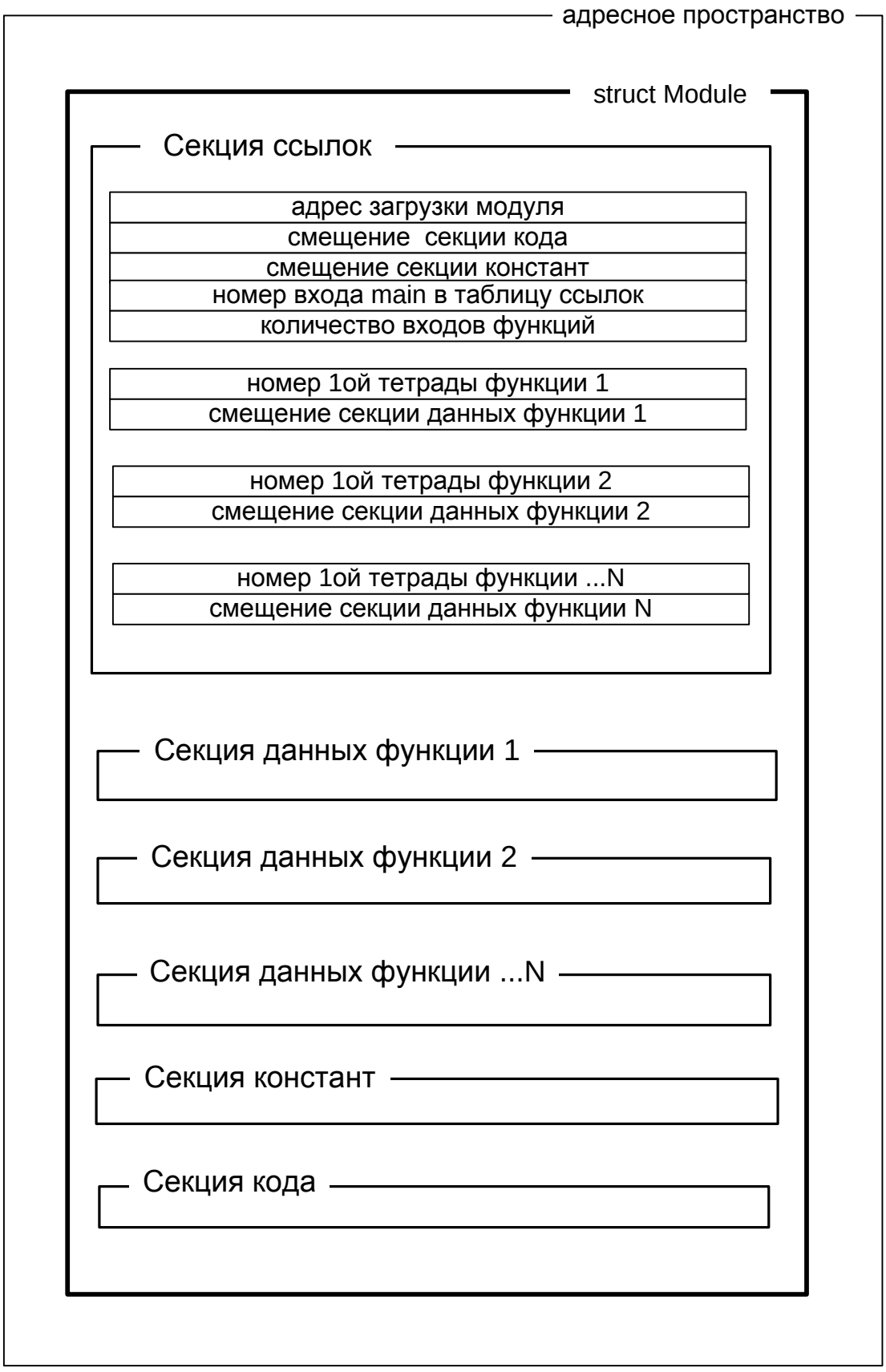


11.1 Модель памяти SVV-2015



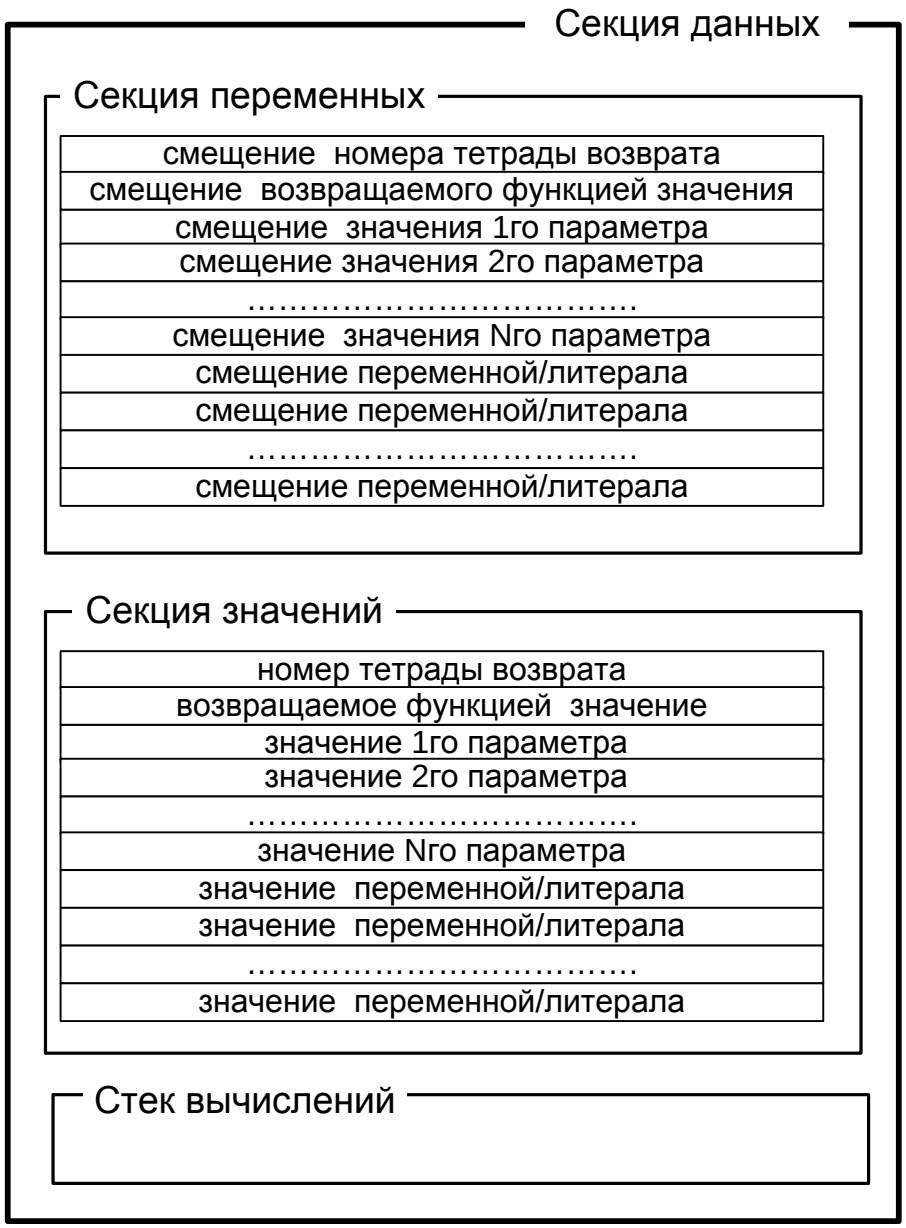

```
struct Module
{
     RefSection   refsection;                // секция ссылок
     DataSection  datasections[GEN_MAX_FUNC]; // секции данных в количестве refsection.functablesize
     ConstSection constsection;              // секция констант
     CodeSection  codesection;               // секция кода
     Module();
     void Serilization(char * filename){};   // серилизация                //
     void DeSerilization(char * filename){}; // десерилизация


     void zapvar(int offvar, int offval,  int offconst, int   initval);
     void zaptetrad(int offcode,  Tetrad t);

};
```

11.2 **Секция ссылок SVV-2015**



```
struct RefSection                  // секция ссылок
  {
    void* loadaddress;              // Run: адрес  структуры Module
    int   offcodesection;           // Gen: смещение секции кода
    int   offconstsection;          // Gen: смещение кода констант
    short mainentrynumber;          // Gen:  номер входа main
    short functablesize;            // Gen: = количеству функции из ТИ
    struct FuncSection              // Gen: входы для функций
    {
      short tfirst;                 // Gen: первая тетрада функции
      int   offdatasection;         // Gen: смещение секции данных функции
      FuncSection();
      FuncSection(short t, int ods);
    } functable[GEN_MAX_FUNC];      // Gen:  таблица функций
    RefSection();
    RefSection(Module* pmodule);
  };
```

11.3 **Секция данных**



```
struct VarSection    //  секция переменных
 {
   int  offtret;            //  Gen:  смещение номера тетрады возврата
   int  offrc;              //  Gen:  смещение возвращаемого значения
   int  offmem[GEN_MAX_ID]; // Gen: смещения параметров и переменных
   VarSection();
   VarSection(Module *pmodule);
   char* Serilization(){};       // серилизация                //
   char* DeSerilization(){};     // десерилизация
 };
struct ValSection     // секция значений
 {
    int  nt_ret;           // Run: тетрада для возврата
    char mem[GEN_MAX_VAL]; // Run: память для параметров и переменных
    char* Serilization(){};       // серилизация                //
    char* DeSerilization(){};     // десерилизация
 };

typedef std::stack<void*> CalcStackSection;
struct DataSection
{
    VarSection varsection;    // Gen   секция переменных
    ValSection valsection;    // Run   секция значений
    CalcStackSection calcstacksection; // Run
    DataSection();
    char* Serilization(){};       // серилизация                //
    char* DeSerilization(){};     // десерилизация
};
```

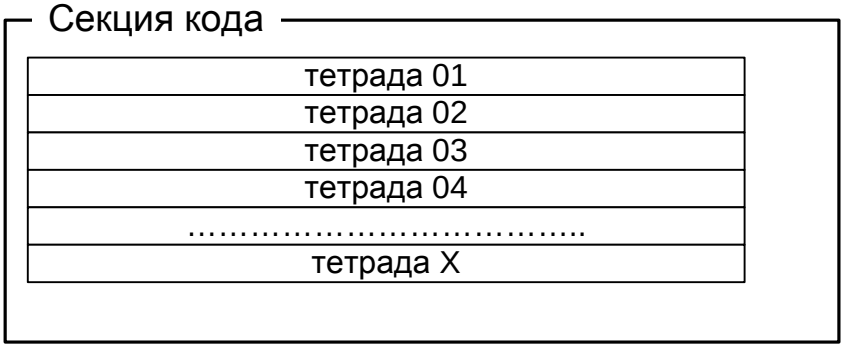
11.4 Секция констант

Секция констант

значение инициализации/литерала
значение инициализации/литерала
значение инициализации/литерала
.....
значение инициализации/литерала

```
struct ConstSection     // секция констант
{
   char mem[GEN_MAX_CONST];     //Gen
   char* Serilization(){};       // серилизация
   char* DeSerilization(){};     // десерилизация
};
```

11.5 Секция кода




```
enum TETRADETYPE  // типы тетерад
 {
    NILL = 0,    // пустая
    FUNC = 1,    // инициализация функции
    RETN = 2,    // return - записать код возврата и перейти по тетраде возврата
    PARM = 3,    // иниализация параметра функции
    DTI  = 4,    // инициализация переменной
    MAIN = 5,    // инициализация главной функции
    CALLPARM = 6, // записать  значение параметра перед вызовом
    CALL = 7   // вызов функции: записать номер тетрады возврата и перейти
};
struct Tetrad   // тетрада
{
   TETRADETYPE type;    // тип тетрады
   void* result;        // результат ???
   void* parm[4];       // параметры тетрады
   Tetrad() { ... }
   Tetrad(TETRADETYPE t, int p0, int p1, int p2, int p3) { ... }
   char* Serilization(){};       // серилизация
   char* DeSerilization(){};     // десерилизация
};
struct CodeSection   // секция кода
{
   int size;          // количество тетрад
   Tetrad  tetrad[GEN_MAX_TETR];     //Gen
   char* Serilization(){};       // серилизация
   char* DeSerilization(){};     // десерилизация
 };
```

11.6 Процесс генерации памяти

```
struct Genstate                     // состояние процесса генерации
{
   short size_funcdatasections;     // количество секций = количеству фцнкций
   struct FuncDataSection           // смещения  в секции данных
   {
     int next_varsection;           // смещение свободной памяти в секции переменных
     int next_valsection;           // смещение свободной памяти в секции значений
   } *funcdatasections;
   int  next_constsection;          // смещение свободной памяти в секции констант
   int  next_codesection;           // смещение свободной памяти в секции  кода
};
```

```
Module* generate(    //  генерация кода
                LEX::LEX& lex,                  // таблицы лексем и идентификаторов
                Mfst::Deducation deduct         // правила вывода
                );
```

11.7 Расширения таблицы идентификаторов

```
struct Entry      // строка таблицы идентификаторов
{
 short       idxfirstLE;         // индекс  первой строки в таблице лексем
 char        id[2*ID_MAXSIZE+1];     // идентификатор (автоматически усекается до  ID_MAXSIZE) + префикс_функции#
 IDDATATYPE  iddatatype;         // тип данных
 IDTYPE      idtype;             // тип идентикатора
 union { ... } value;        // значение идентификатора
 struct GenFuncExt   //заполняется при генерации функции
 {
  int offrcvar;   // смещение адреса возвращаемого значения
  int offrcval;   // смещение возвращаемого значения
  int offretvar;  // смещение адреса для тетрады возврата
  int offretval;  // смещение значения тетрады возврата
 } genfuncext;    // расширение, заполняемое при генерации кода

 struct GenVarExt  //заполняется при генерации переменных
 {
  int offvar;      // смещение адреса  значения
  int offval;      // смещение значения
  int offconst;    // смещение значения инициализации
 } genvarext;      // расширение, заполняемое при генерации кода

 Entry (char pid[2*ID_MAXSIZE+1], IDDATATYPE dt, IDTYPE it, short lef, int val) { ... }
 Entry (char pid[2*ID_MAXSIZE+1], IDDATATYPE dt, IDTYPE it, short lef, char* val) { ... }
 Entry () { ... }

};
```