Теория формальных языков и компиляторов

Лекций: 36 часов

Лабораторных работ: 36 часов (8 л.р.)

Курсовая работа +

самостоятельная работа: 108 часов

Проект: Разработка транслятора для учебного языка программирования

Сайт дисциплины:
http://vt.cs.nstu.ru/~malyavko/TFL&C/index.html

E-mail: a.malyavko@corp.nstu.ru

Малявко Александр Антонович

Литература

1. *Малявко А.А.* Формальные языки и компиляторы: учебное пособие для вузов. – М., Изд-во Юрайт, 2019
2. *Малявко А.А.* Формальные языки и компиляторы: учебник НГТУ. – Изд-во НГТУ, 2014, 004 М219, Id = 000184529
3. *Малявко А.А.* Системное программное обеспечение. Формальные языки и методы трансляции: Учеб. пособие. – Новосибирск: Изд-во НГТУ, 2010. – Ч.1, 004 М 219, Id = 143812
4. *Малявко А.А.* Системное программное обеспечение. Формальные языки и методы трансляции: Учеб. пособие. – Новосибирск: Изд-во НГТУ, 2011. – Ч.2, 004 М 219, Id=155235
5. *Малявко А.А.* Системное программное обеспечение. Формальные языки и методы трансляции: Учеб. пособие. – Новосибирск: Изд-во НГТУ, 2012. – Ч.3, 004 М 219, Id=170641
6. *Малявко А.А.* Системное программное обеспечение ЭВМ. Трансляторы / Методические указания. – Новосибирск: Изд-во НГТУ, 2006, Id=58442
7. *Ахо А., Сети Р., Ульман Д.* Компиляторы: принципы, технологии и инструменты. – М.: «Вильямс», 2001, Id=16803
8. *Карпов Ю.Г.* Теория и технология программирования. Основы построения трансляторов: учеб. пособие. – СПб.: БХВ-Петербург, 2005, Id=64347
9. *Свердлов С. З.* Языки программирования и методы трансляции : учебное пособие для вузов - СПб., 2007, Id=65534
10. *Гавриков М.М., ИванченкоА.Н., Гринченков Д.В.* Теоретические основы разработки и реализации языков программирования. – М.: Кнорус, 2010.

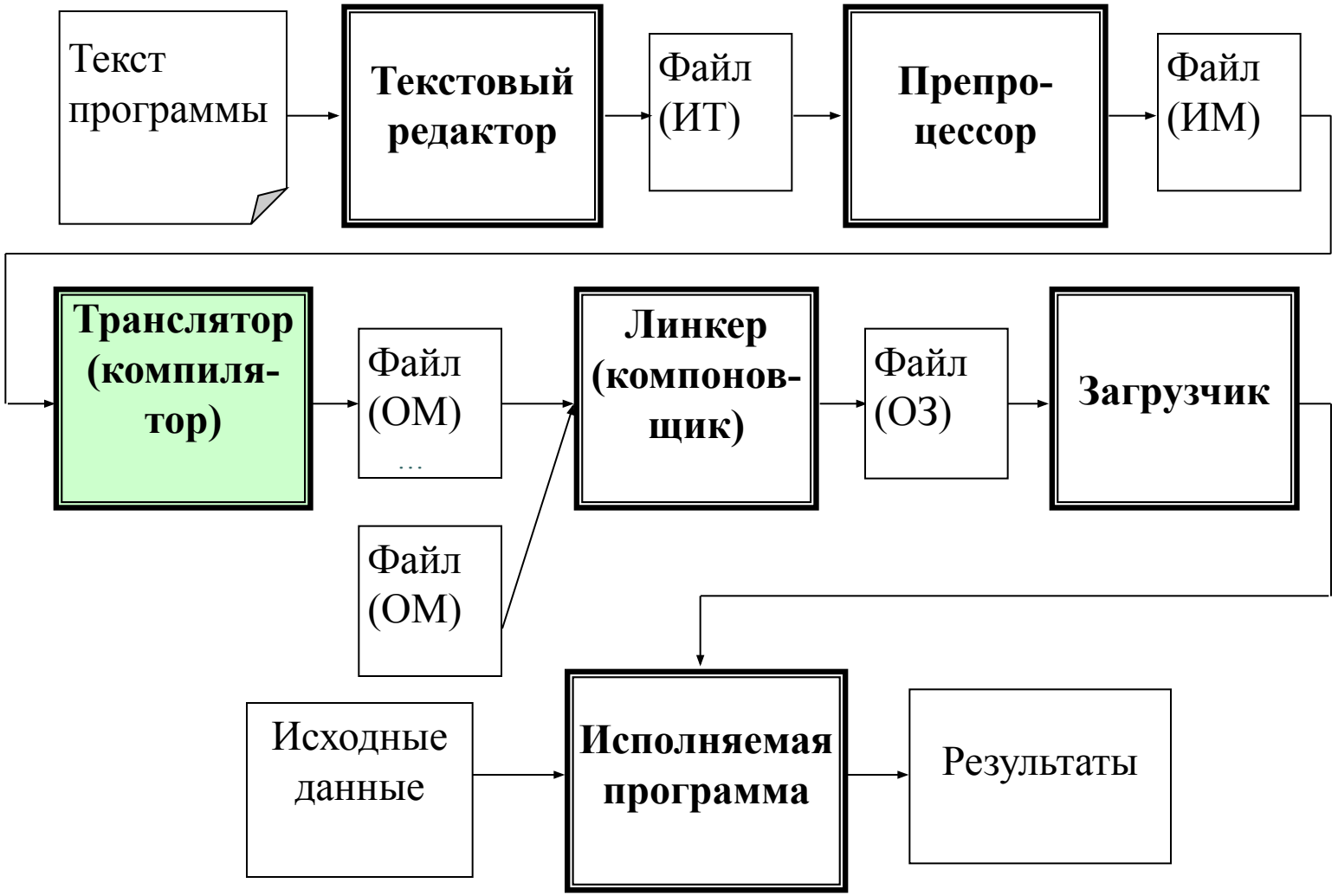
Балльно-рейтинговая система аттестации

№	Вид учебной работы	Диапазоны баллов
1	Лаб. работа №1	4 – 8
2	Лаб. работа №2	4 – 8
3	Лаб. работа №3	4 – 8
4	Лаб. работа №4	4 – 8
5	Лаб. работа №5	4 – 8
6	Лаб. работа №6	4 – 8
7	Лаб. работа №7	4 – 8
8	Лаб. работа №8	2 – 4
Итого по текущему рейтингу:		30 – 60
Экзамен:		20 – 40
Итого за семестр:		50 – 72 –> **3**, 73 – 86 –> **4**, 87 – 100 –> **5**

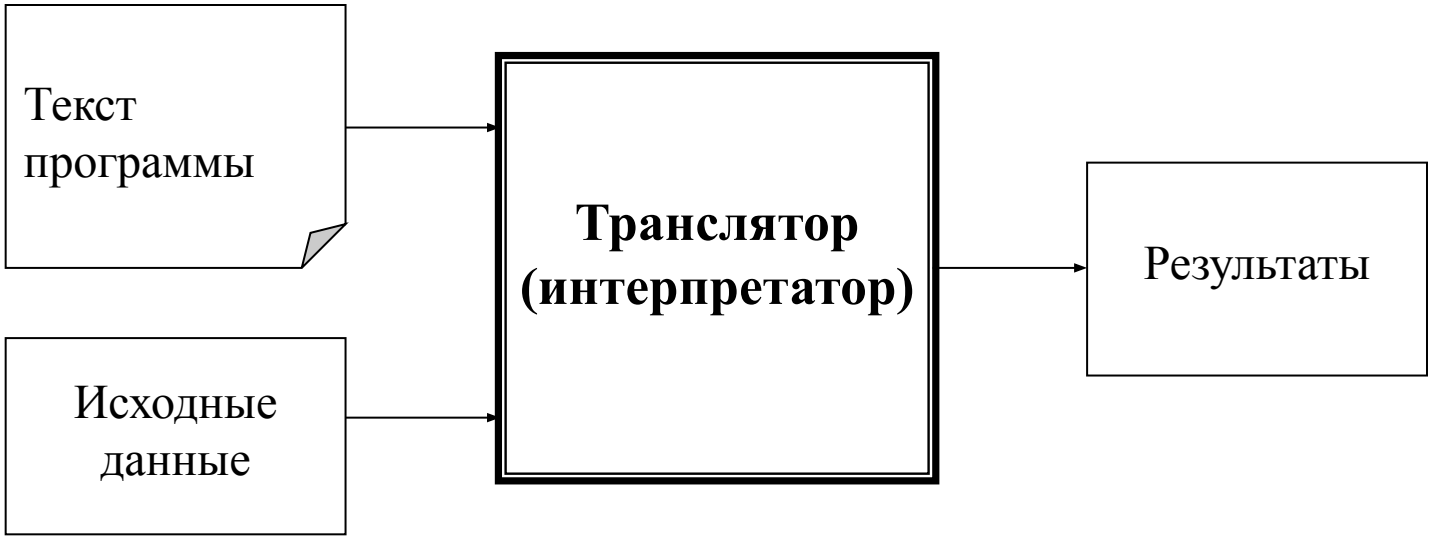
Балльно-рейтинговая система аттестации. Курсовая работа

Пункт задания	**Диапазоны баллов**
1	3 – 6
2	6 – 12
3	9 – 18
4	12 – 24
5	8 – 16
6	2 – 4
Итого по текущему рейтингу:	40 – 80
Защита:	10 – 20
Итого за семестр:	50 – 72 –> **3**, 73 – 86 –> **4**, 87 – 100 –> **5**

Введение
Технология компиляции

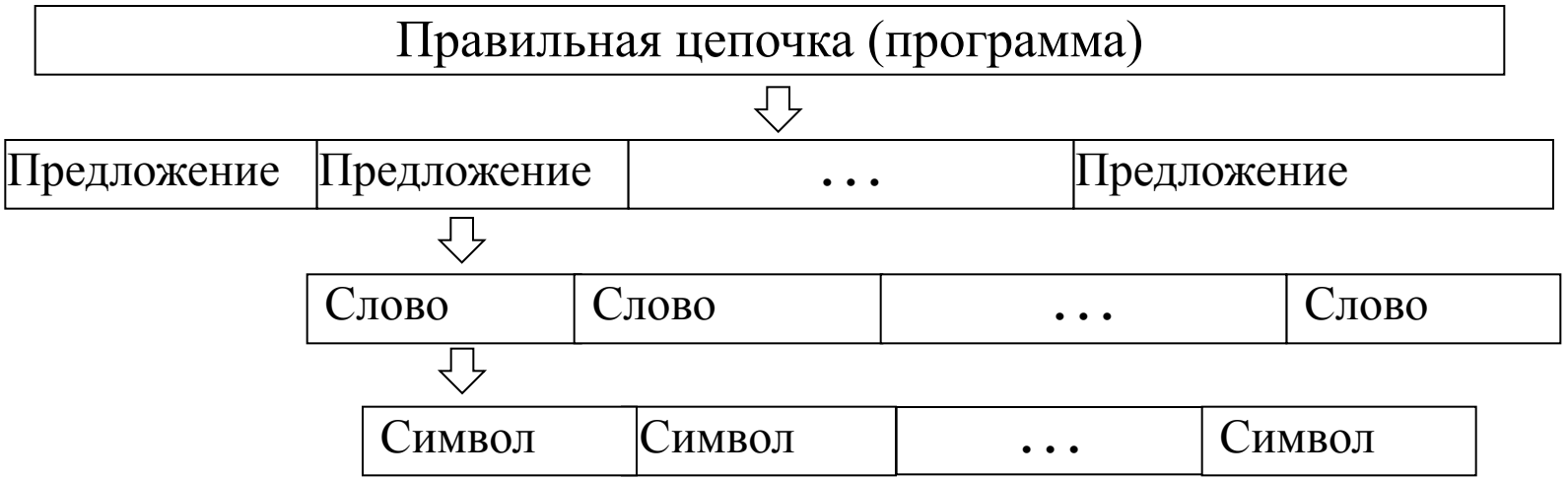


Технология интерпретации



Широко применяется в веб-браузерах и ряде других приложений

Элементарные понятия формальных языков



Пример:

Правильная цепочка на языке Java: *class SampleOfEmptyClass{ }*

Предложения: *class SampleOfEmptyClass* | *{ }*

Слова: *class* | | *SampleOfEmptyClass* | *{* | | *}*

Неправильные цепочки: *Class SampleOfClass{ }* | *class Sample{ char 's' = 3.14; }*

Элементарные понятия формальных языков

Формальный язык:

множество правильных цепочек (всех цепочек, построенных по некоторой системе правил)

Лексика: совокупность правил, определяющих способы образования слов из символов.

Синтаксис: совокупность правил, определяющих способы построения предложений из слов.

Семантика: совокупность правил, определяющих допустимость использования одних и тех же слов в разных предложениях.

Этапы процесса трансляции

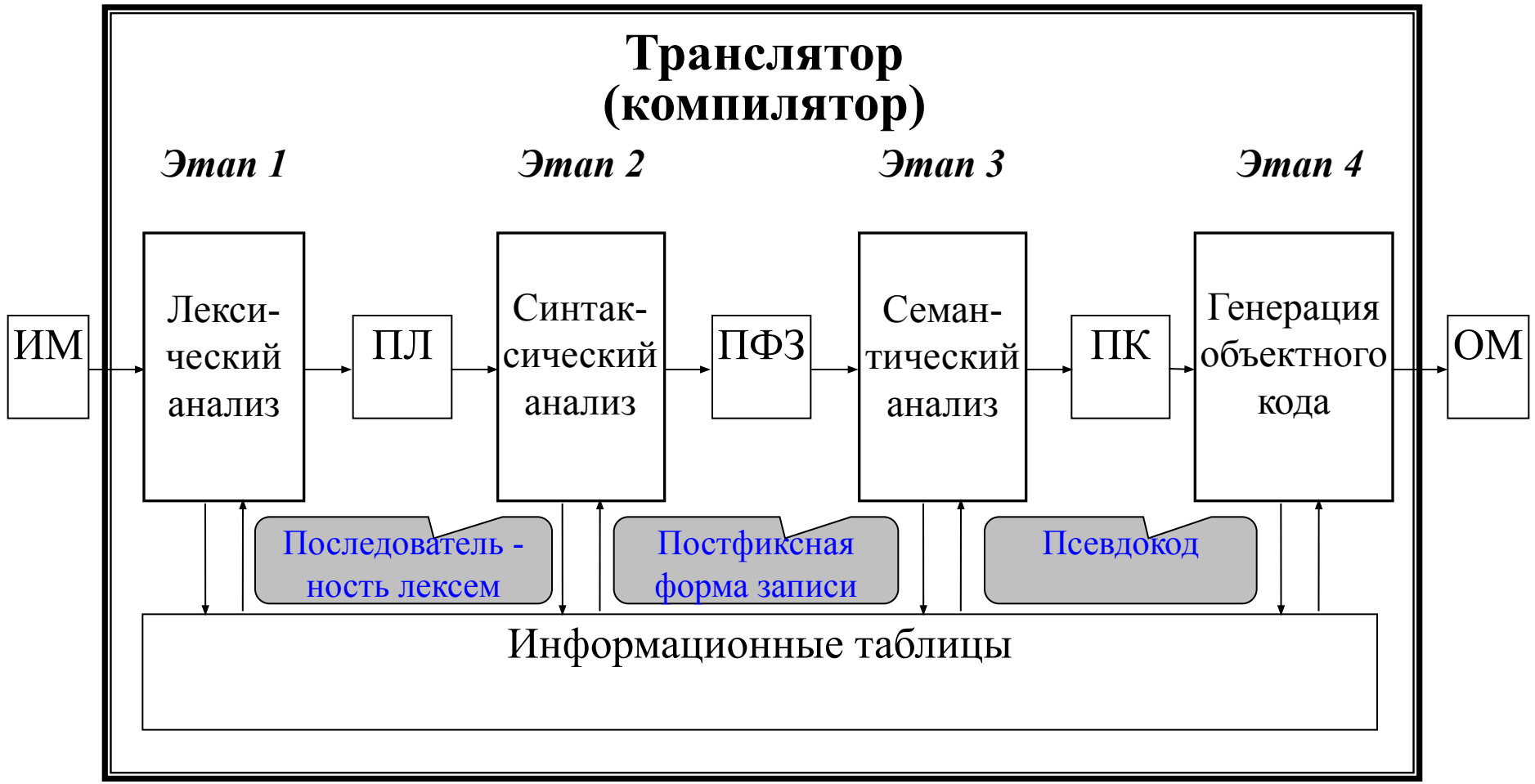


Иллюстрация процесса и результатов преобразований во время трансляции

ИМ (исходный модуль):

```
...
int nod( int first, int second ){    // вычисление НОД
    while ( first != second )  // пока два значения не равны
        if ( first < second )    // если первое меньше второго
            second –= first;   // вычитаем первое из второго
        else                  // если первое больше второго
            first –= second;   // вычитаем второе из первого
    return first;     // возвращаем значение НОД
}
...
```

Последовательность токенов (лексем)

Последовательность токенов/лексем функции nod:

Во внутреннем представлении (*{код токена, индекс слова}*):

{0,13} {5,52} {2,2} {0,13} {5,87} {4,2} {0,13} {5,33} {2,3} {2,0} {0,7} {2,2} {5,87} {1,9} {5,33} {2,3} {0,3} {2,2} {5,87} {1,6} {5,33} {2,3} {5,33} {1,2} {5,87} {4,0} {0,4} {5,87} {1,2} {5,33} {1,0} {0,15} {5,87} {1,0} {2,1}

С использованием имен групп слов в качестве расшифровки токенов:

{keyword,13} {ident,52} {bracket,2} {keyword,13} {ident,87} {delimiter,2} {keyword,13} {ident,33} {bracket,3} {bracket,0} {keyword,7} {bracket,2} {ident,87} {operation,9} {ident,33} {bracket,3} {keyword,3} {bracket,2} {ident,87} {operation,6} {ident,33} {bracket,3} {ident,33} {operation,2} {ident,87} {delimiter,0} {keyword,4} {ident,87} {operation,2} {ident,33} {delimiter,0} {keyword,15} {ident,87} {delimiter,0} {bracket,1}

В исходных терминах:

int nod (int first , int second) { while (first != second) if (first < second) second –= first ; else first –= second ; return first ; }

(**некоторые слова исчезли**)

Постфиксная запись

Синий цвет – для ключевых слов
Черный – для идентификаторов
Красным выделены знаки операций
Зеленым – метки (имена операций)
На голубом фоне – пояснения

заголовок функции:

nod int function first int argument second int argument

заголовок оператора цикла:

label_0_0: first second != label_0_1 jmpOnFalse

условный оператор:

first second < label_1_0 jmpOnFalse second first –= label_1_1 jmp label_1_0: first second –= label_1_1:

завершение оператора цикла:

label_0_0 jmp

возврат из функции:

label_0_1: first return

Еще кое-какие слова исчезли, но появились и новые слова

Псевдокод.
Последовательность **тетрад:**

defineLabel	label_0_0		
!=	second	first	push
jmpOnFalse	label_0_1	pop	
<	second	first	push
jmpOnFalse	label_1_0	pop	
–=	first	second	second
jmp	label_1_1		
defineLabel	label_1_0		
–=	second	first	first
defineLabel	label_1_1		
jmp	label_0_0		
defineLabel	label_0_1		
return	first		

Псевдокод. Последовательность **триад:**

defineLabel	label_0_0	
!=	second	first
jmpOnFalse	label_0_1	pop
<	second	first
jmpOnFalse	label_1_0	pop
–=	first	second
=	pop	second
jmp	label_1_1	
defineLabel	label_1_0	
–=	second	first
=	pop	first
defineLabel	label_1_1	
jmp	label_0_0	
defineLabel	label_0_1	
return	first	

Псевдокод.
Последовательность **пентад:**

label_0_0	**!=**	second	first	push
	jmpOnFalse	label_0_1	pop	
	<	second	first	push
	jmpOnFalse	label_1_0	pop	
	–=	first	second	second
	jmp	label_1_1		
label_1_0	**–=**	second	first	first
label_1_1	jmp	label_0_0		
label_0_1	return	First		

Объектный код. Без оптимизации

только эта колонка содержит результат трансляции

//int nod(int first, int second){				
00411B00	55	push	ebp	
00411B01	8B EC	mov	ebp,esp	
00411B03	83 EC 40	sub	esp,40h	//удаляется при оптимизации
00411B06	53	push	ebx	// удаляется при оптимизации
00411B07	56	push	esi	// удаляется при оптимизации
00411B08	57	push	edi	// удаляется при оптимизации
//while(first != second)				
00411B09	8B 45 08	mov	eax,dword ptr [first]	
00411B0C	3B 45 0C	cmp	eax,dword ptr [second]	
00411B0F	74 1E	je	nod+2Fh (411B2Fh)	// другое смещение
//if(first < second)				
00411B11	8B 45 08	mov	eax,dword ptr [first]	// удаляется при оптимизации
00411B14	3B 45 0C	cmp	eax,dword ptr [second]	// удаляется при оптимизации
00411B17	7D 0B	jge	nod+24h (411B24h)	// другое смещение
//second -= first;				
00411B19	8B 45 0C	mov	eax,dword ptr [second]	
00411B1C	2B 45 08	sub	eax,dword ptr [first]	
00411B1F	89 45 0C	mov	dword ptr [second],eax	

Объектный код
Без оптимизации

//else				
00411B22	EB 09	jmp	nod+2Dh (411B2Dh)	// на начало цикла
//first -= second;				
00411B24	8B 45 08	mov	eax,dword ptr [first]	// удаляется при оптимизации
00411B27	2B 45 0C	sub	eax,dword ptr [second]	
00411B2A	89 45 08	mov	dword ptr [first],eax	
00411B2D	EB DA	jmp	nod+9 (411B09h)	// на следующую команду
//return first;				
00411B2F	8B 45 08	mov	eax,dword ptr [first]	// удаляется при оптимизации
//}				
00411B32	5F	pop	edi	// удаляется при оптимизации
00411B33	5E	pop	esi	// удаляется при оптимизации
00411B34	5B	pop	ebx	// удаляется при оптимизации
00411B35	8B E5	mov	esp,ebp	
00411B37	5D	pop	ebp	
00411B38	C3	ret		

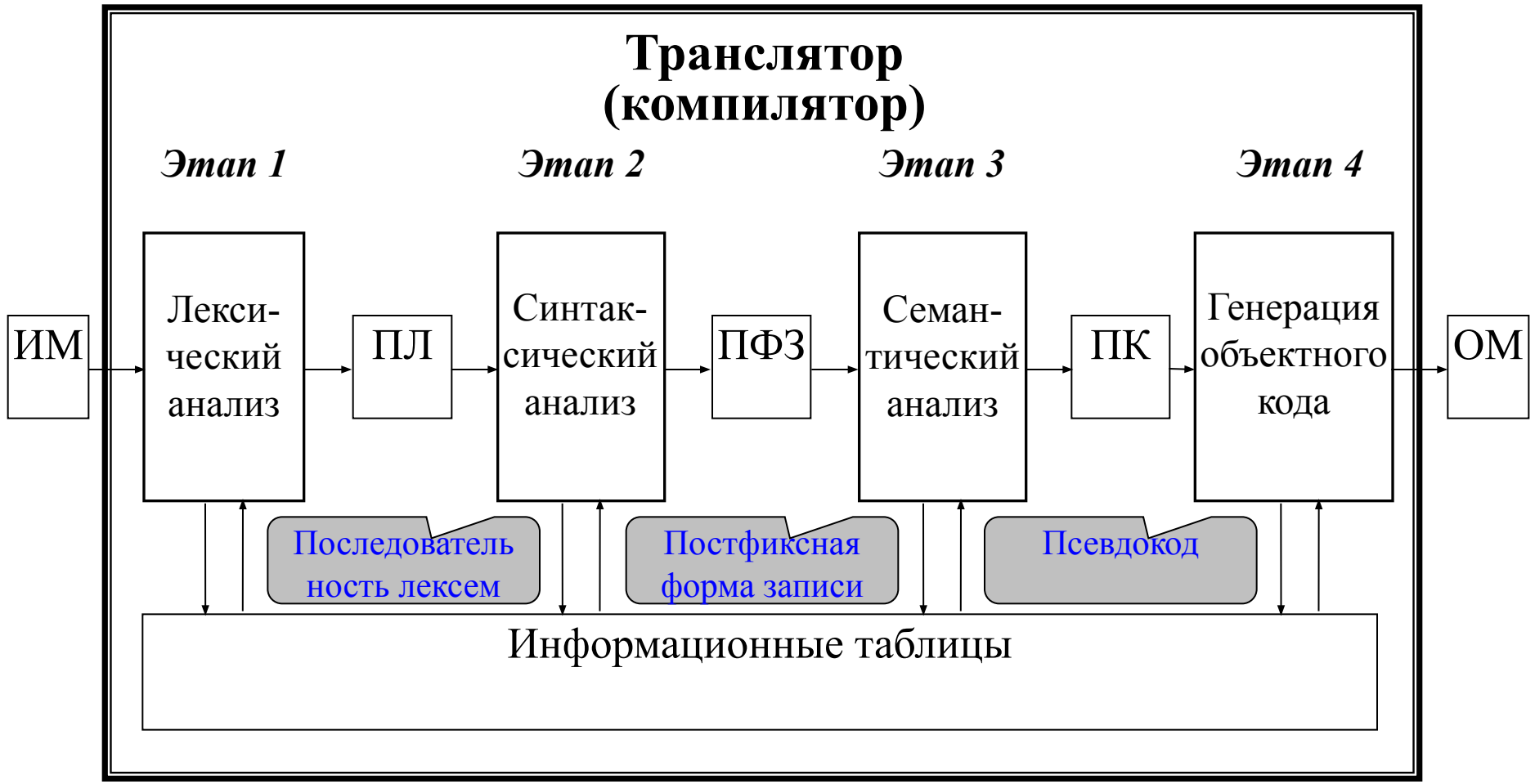
Объектный код Оптимизированный

00411B00	55	push	ebp
00411B01	8B EC	mov	ebp,esp
00411B03	8B 45 08	Mov	eax,dword ptr [first]
00411B06	3B 45 0C	cmp	eax,dword ptr [second]
00411B09	74 1E	je	nod+20h (411B2Fh)
00411B0B	7D 0B	jge	nod+18h (411B24h)
00411B0D	8B 45 0C	mov	eax,dword ptr [second]
00411B10	2B 45 08	sub	eax,dword ptr [first]
00411B13	89 45 0C	mov	dword ptr [second],eax
00411B16	EB 09	jmp	nod+3 (411B2Dh)
00411B18	2B 45 0C	sub	eax,dword ptr [second]
00411B1B	89 45 08	mov	dword ptr [first],eax
00411B1E	EB DA	jmp	nod+6 (411B09h)
00411B20	8B E5	mov	esp,ebp
00411B22	5D	pop	ebp
00411B23	C3	ret	

До оптимизации: команд – 27, байтов – 57

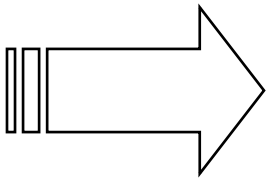
После: команд – 16, байтов – 36

Этапы процесса трансляции



Компиляция:

```
…
int nod( int first, int second ){
    while ( first != second )
     if ( first < second )
          second -= first;
     else
          first -= second;
    return first;
}
…
```

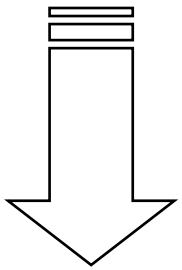


55
8B EC
8B 45 08
3B 45 0C
74 1E
7D 0B
8B 45 0C
2B 45 08
89 45 0C
EB 09
2B 45 0C
89 45 08
EB DA
8B E5
5D
C3

Интерпретация:

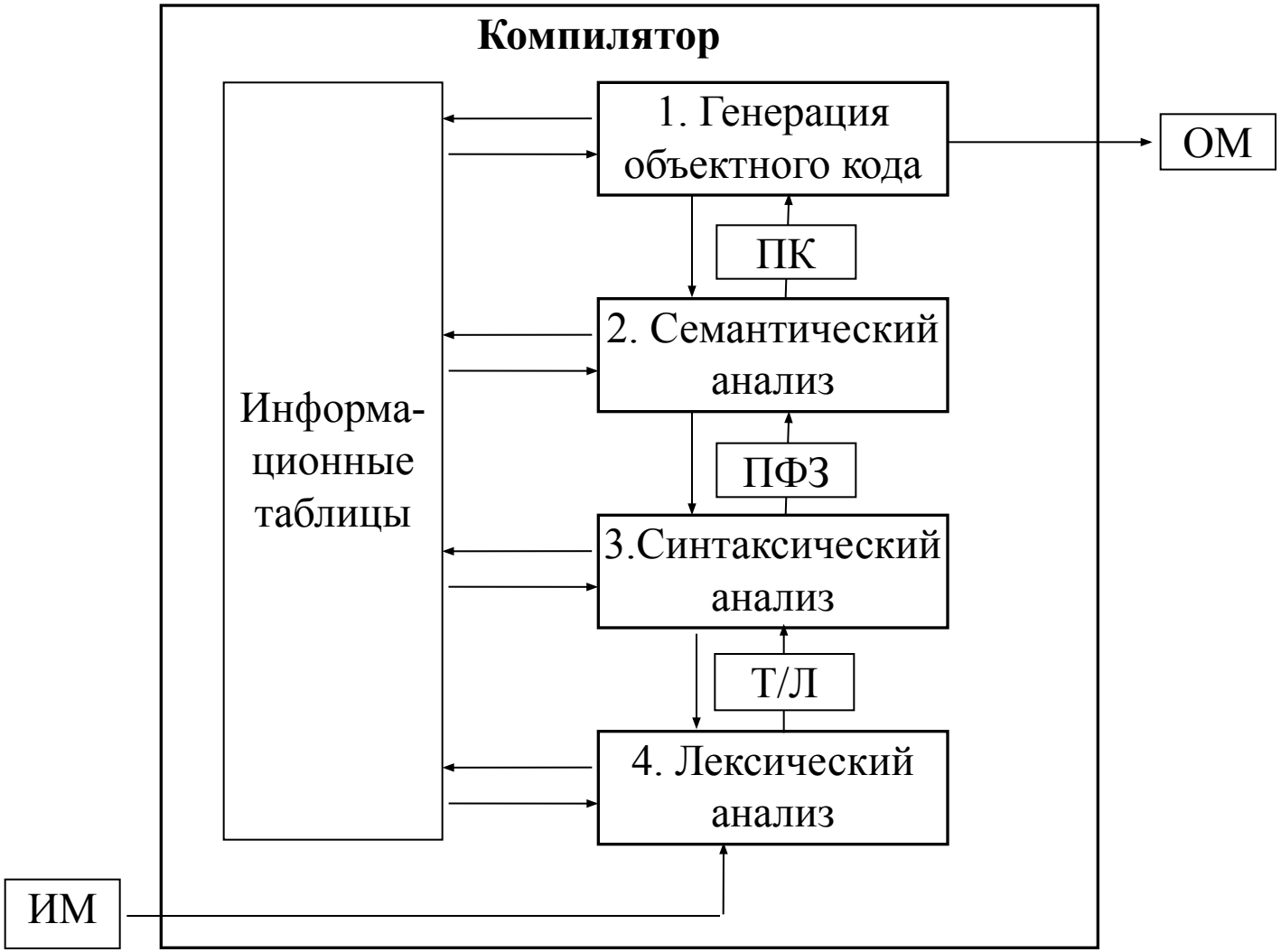
```
…
int nod( int first, int second ){
    while ( first != second )
     if ( first < second )
          second -= first;
     else
          first -= second;
    return first;
}
…
```

```
a = 2171;
x = nod( a, 949 );
```

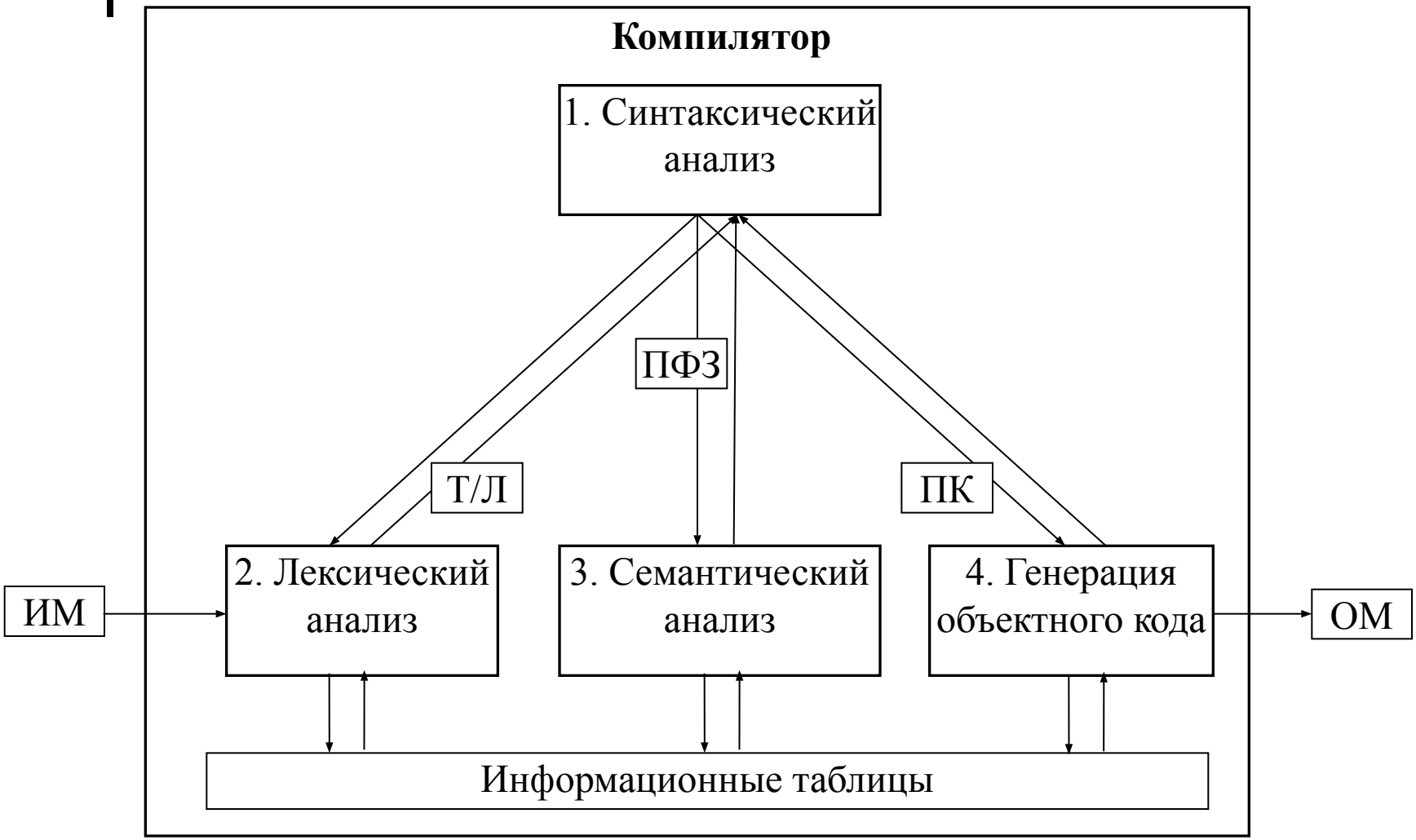


13

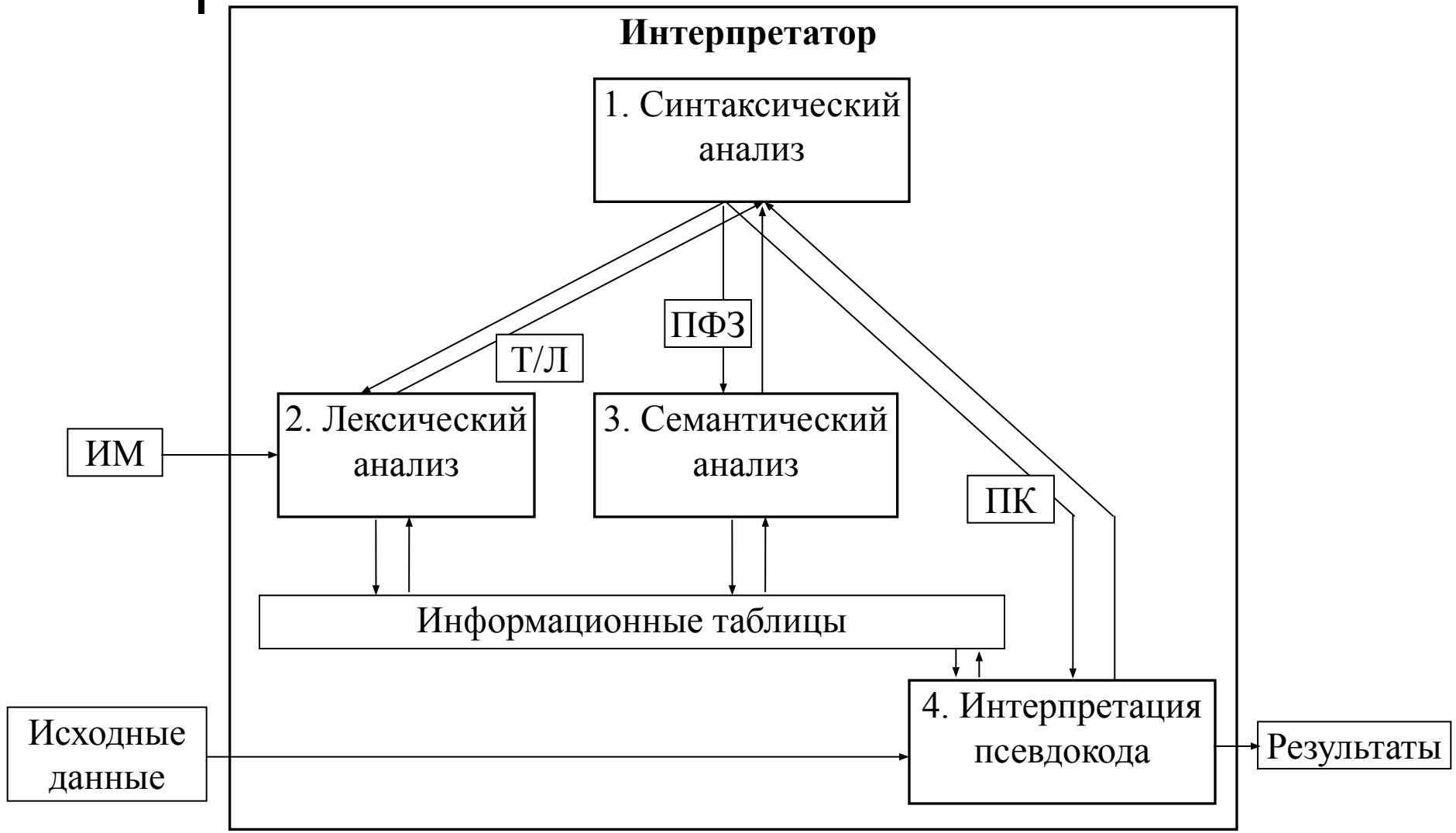
Возможная физическая последовательность этапов компиляции



Обычная последовательность этапов компиляции



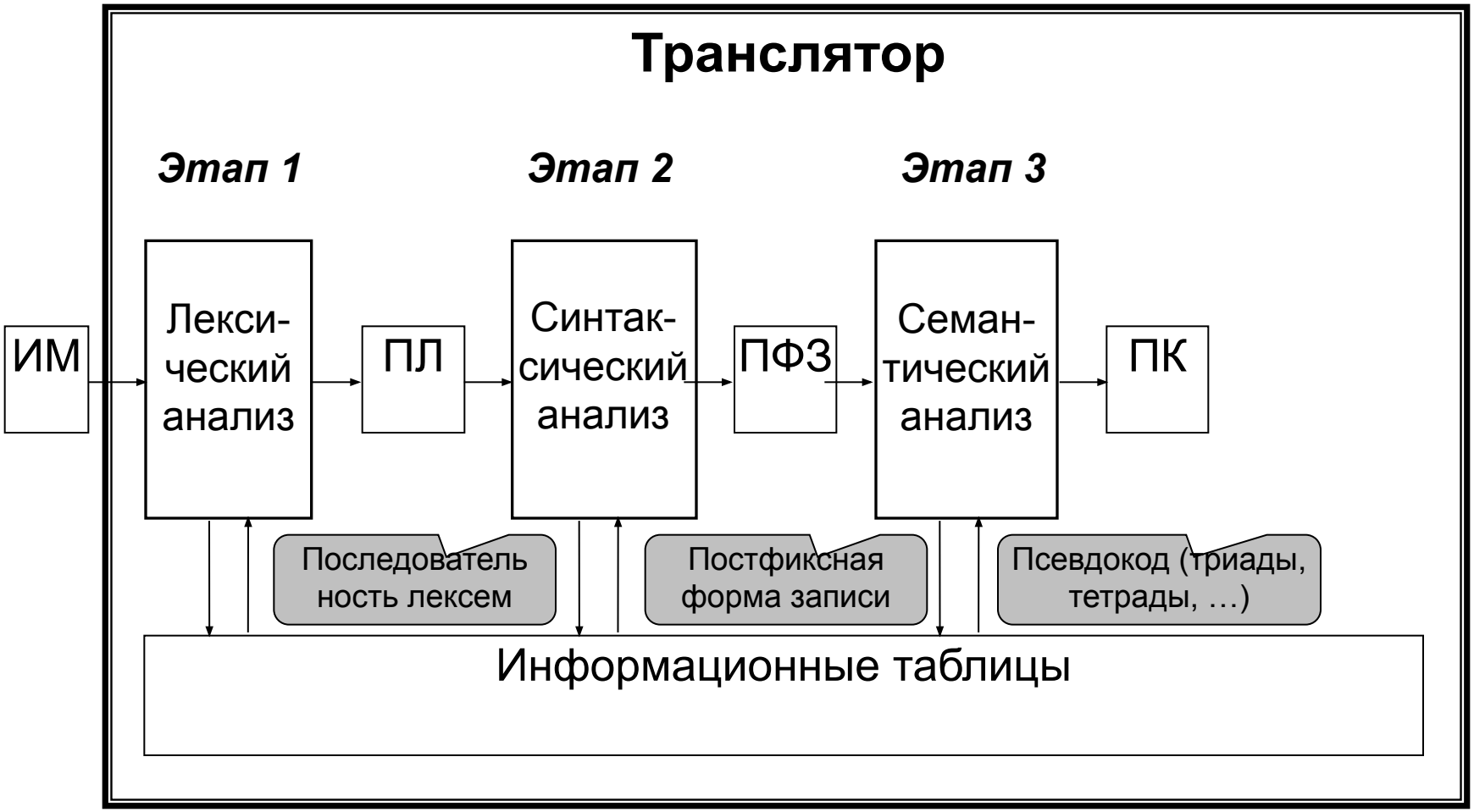
Обычная последовательность этапов интерпретации



Задание на курсовую работу

1. Разработать полное и точное описание лексики, синтаксиса и семантики заданного варианта языка. Написать несколько простых тестовых программ, содержащих все заданные элементы и управляющие конструкции языка. Эти программы использовать впоследствии для проверки элементов разрабатываемого транслятора.
2. Разработать систему регулярных выражений, определяющую лексику заданного варианта языка. Используя пакет Вебтранслаб, построить автоматную реализацию лексического анализатора на выбранном инструментальном языке (рекомендуется javascript), добиться его работоспособности.
3. Разработать формальную грамматику класса LL(1)
   или:
   разработать формальную грамматику класса не выше, чем LALR(1), определяющую синтаксис заданного языка. Используя пакет Вебтранслаб, построить автоматную реализацию синтаксического акцептора, добиться его работоспособности.
4. Разработать совокупность действий для расширения синтаксического акцептора, выполняющего преобразование входной последовательности лексем в постфиксную форму записи (ПФЗ) или в абстрактное синтаксическое дерево (АСД).
5. Разработать семантический анализатор, преобразователь ПФЗ (или АСД) в псевдокод заданного формата.
6. Оформить (в электронном виде) расчетно-пояснительную записку

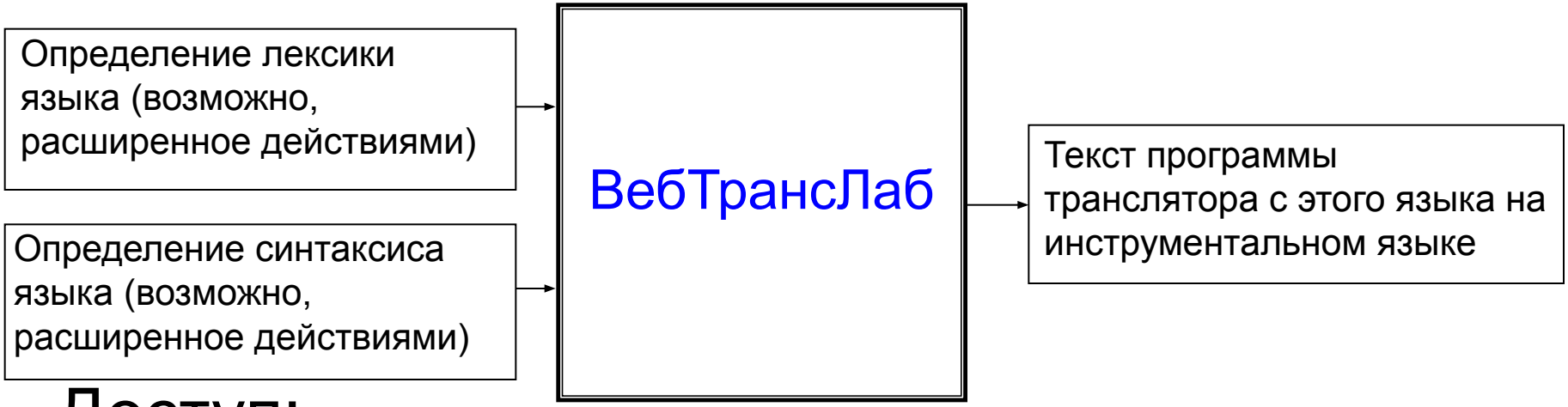
Задание на курсовую работу



Системы автоматизации проектирования трансляторов

Существует большое количество таких систем: Lex/Yacc, Flex/Bison, PCCTS, ANTLR, LLGEN, JavaCC …

На практических занятиях и при выполнении курсовой работы будет использоваться учебный пакет:



Доступ:

Извне http://vt.cs.nstu.ru:48095/wtl

Из 7-3xx http://172.16.7.18:8095/wtl