Методи на Транслация

Виртуални Изчислителни Машини.
Език. Метаезик на Бекус-Наур.

доц. д-р Александър Пенев

Език

Език

Езикът е система от съобщения, съставени от знаци, чрез който може да се обменя информация.

Езикът е система от изразни средства и правила за ползването им, чрез които може да се представя и обменя информация.

Език за програмиране (ЕП)

Език за програмиране (ЕП) се нарича всяка система от обозначения, предназначена за описание на алгоритми и структури от данни, която е реализирана на поне една изчислителна машина (ИМ).

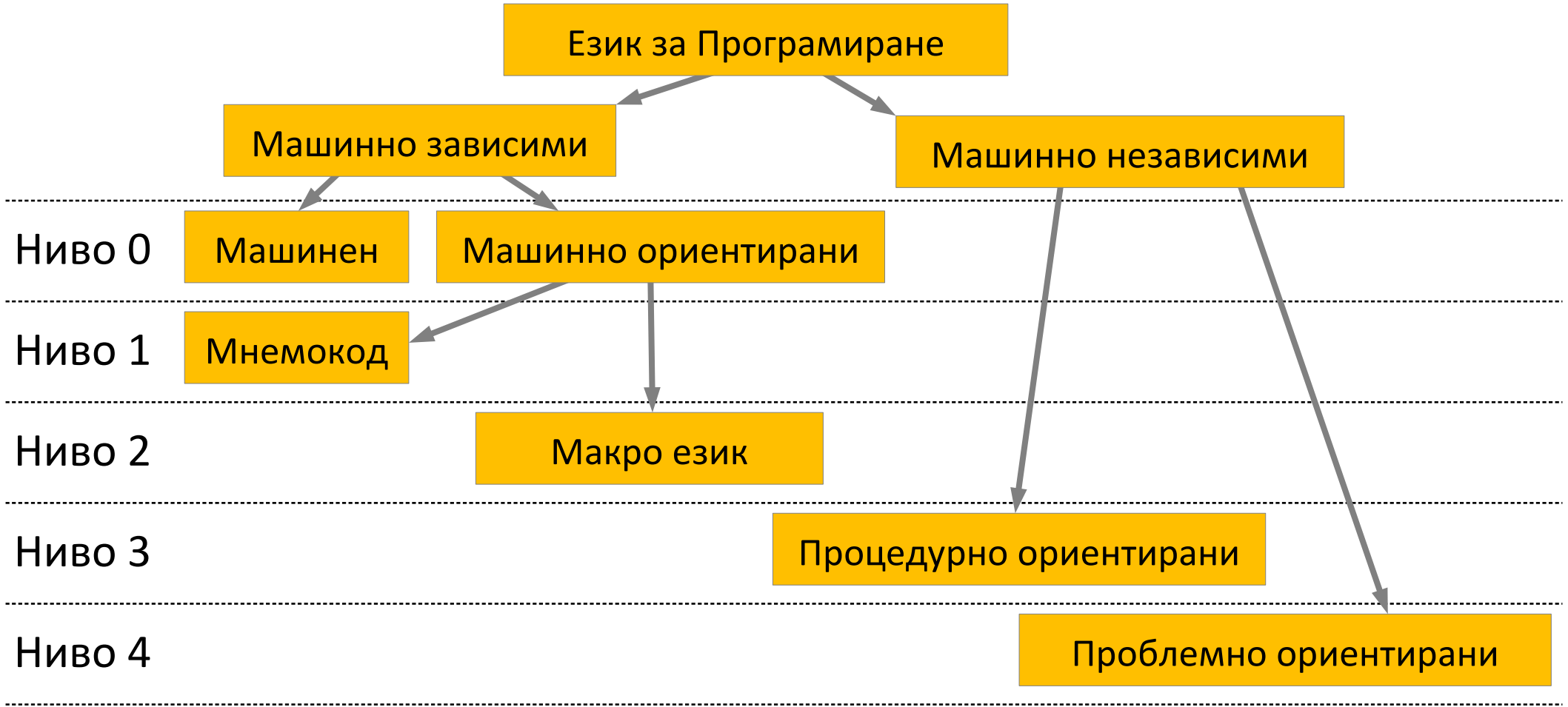
Ако езикът не е реализиран, то той се нарича **Алгоритмичен език**.

Как се реализират ЕП. Програма

Реализацията на ЕП се заключава в това да се създаде **транслатор**, превеждащ програмите написани този език в еквивалентни програми написани на езика на ИМ.

Програма наричаме алгоритъм и използваните от него структури от данни, описани на определен ЕП.

Нива на Езиците за програмиране (ЕП)



Граматики

- Автоматна (регулярна)
  Най-простия вид граматика (с най-много ограничения). Обикновено в ЕП се използва за описание на лексемите (лексикалната част от езика). A←a. A←aB. A←Ab.
- Контекстно свободна
  Описва основната част от синтаксисът на ЕП. A←a. A←aAc. A←BaAcB. A←ω.
- Контекстно зависима
  Описва частта от ЕП, която има контекстно зависими конструкции. Например ограничението в някои ЕП да трябва да декларираме променлива преди да я използваме и др. aAb←Ac. ψ←ω.

Описание на граматики

- Неформално – използва се естествен език и примери;
- Формално – използват се математическа нотация или мета езици;

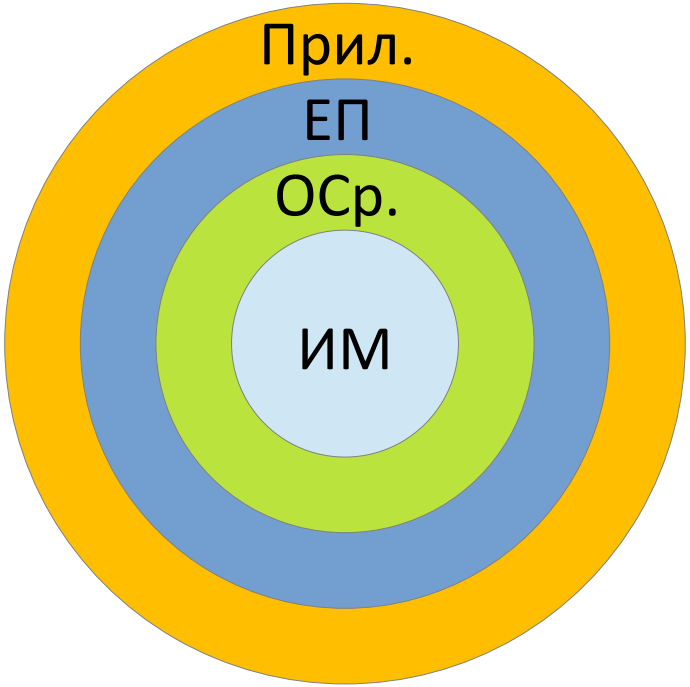
  Граматика е наредена четворка Г (N, Σ, P, S), където:
  - N – не терминални символи
  - Σ – терминали символи
  - P – множество от правила за извод от вида $\alpha \leftarrow \beta$
  - S – S принадлежи на N и е стартов символ за граматиката

Виртуални Изчислителни Машини

(ВИМ)

Изчислителна Машина (ИМ)

Изчислителна машина (ИМ) или изпълнител наричаме интегрираното множество от алгоритми, структури от данни, способни да съхранява и изпълнява програми.

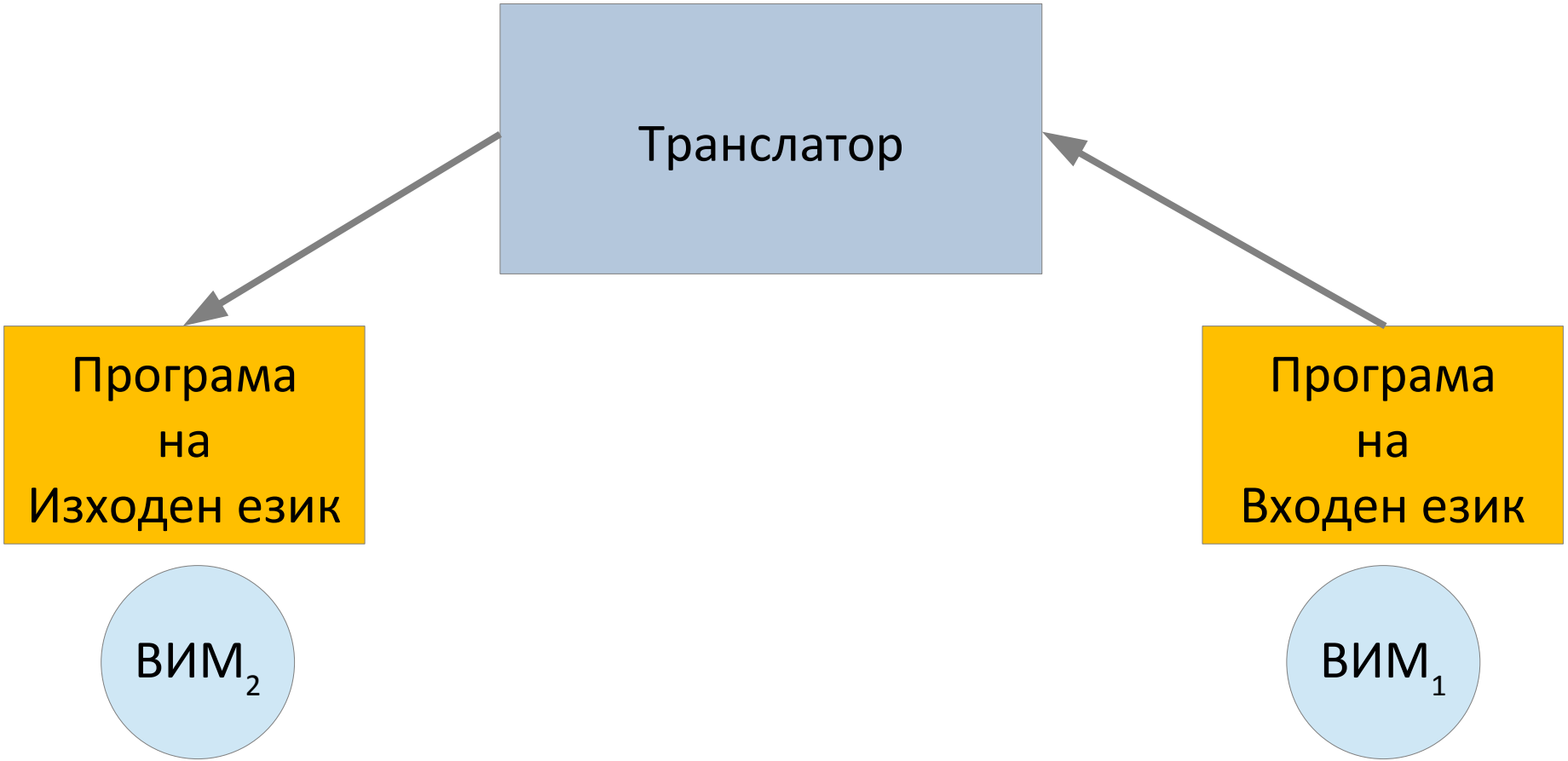


Реализация на ИМ

ИМ може да бъде построена по два начина:

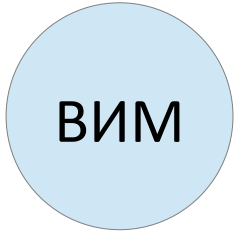
- Като реално физическо устройство (компютър);
- Като програма изпълнявана от някаква друга ИМ; **(ВИМ)**

Транслатор и ВИМ



ВИМ

- Данни;
- Операции;
- Управление на последователността от действия;
- Управление на данните;
- Управление на паметта;
- Операционна среда;



Данни

- Всяка ВИМ трябва да предоставя различни типове данни;
- Прости и съставни типове данни;
- Трябва да може да изпълнява определено количество операции над тях;
- Различни представяния на типовете в паметта;
- Особен тип данни са описващите операциите. Тези данни се наричат команди;

Операции

- Всяка ВИМ трябва да предоставя набор от операции (елементарни) за обработка на вградените типове данни;
- Елементарни е относително понятие (дали е елементарна зависи от конкретната ВИМ);
- Две операции в две ВИМ са еквивалентни ако реализациите им да тъждествени като алгоритми;
- Кога две елементарни операции са тъждествени?

Управление на послед. от действия

- Всяка универсална ВИМ трябва да предоставя средства за управление на последователността от действия;
- Ако това не е така тя ще може да изпълнява само линейни програми;
- В езиците от високо ниво имаме три механизма:
  - Управление в аритметичните изрази;
  - За управление на последователността от команди;
  - За управление на последователността при подпрограми;

Управление на данните

- Механизма за достъп на ВИМ до данните;
- Четене и запис на данни;
- Тези операции са достатъчни за постигане на произволен алгоритъм за управление на данните;

Управление на паметта

- Всяка ВИМ разполага със средства за съхранение на програми и данни;
- Този ресурс е краен, затова се налага той да бъде преразпределян по време на изпълнение;
- В класическите ВИМ такъв механизъм липсва и това налага съхранението да е на фиксирани места;

Операционна среда (ОСр.)

- Всяка ВИМ функционира във някаква външна (операционна) среда;
- Това е получаване и изпращане на данни от външни обекти;
- Тези външни обекти може да се разглеждат също като ВИМ;

Динамика на ВИМ

Динамика на ВИМ

- Разгледаните аспекти дават статична представа за организацията на ВИМ;
- За да се разбере динамиката на ВИМ, въвеждаме понятието състояние на ВИМ;
- Това е съвкупност от съхраняващата среда на ВИМ;
- Всяка изпълнявана от ВИМ операция променя състоянието на ВИМ;

Метаезик на Бекус-Наур

Метаезик

Метаезик се нарича всеки език, който служи за описание на друг език.

Азбука

- Собствени знаци

: :==	→	по дефиниция
|	→	или
< и >	→	ограничители

- Знаци за именуване на понятия – всички знаци, чрез които може да се изрази мисъл/понятие на български (или друг език).
- Терминални знаци – знаци за на описвания език. Те не трябва да имат имат сечение със собствените знаци на езика.

Лексика

- Литерал – низ от терминали. Стойността на литерала е самия низ. Например:

  `3.14` → 3.14

  `АБВ` → АБВ

- Понятие – низ от знаци за именуване на понятия, заграден в < и > като стойността на понятието е множество от литерали. Например:

  `<празно>`

  `<число>`

  `<програма>`

Изрази

- Прост израз – низ от лексеми. Стойността на простия израз е множество от терминални знаци, получено като резултат от декартовото произведение от множествата от стойностите на всички лексеми. Например:

  `<програма> 3.14 <край>` → {...} × {3.14} × {...}

  `<израз> + <израз>` → {вс.и.} × {+} × {вс.и.}

- Израз – прост израз или последователност от прости изрази (наречени алтернативи) разделени с вертикална черта `|`
  Стойността на израза е обединението на множествата на простите изрази, съставящи израза.
  Например:

  `<пр.израз> | <пр.израз>` → {...} U {...}

  `<буква> | <цифра>` → {а,б,в,...,я,0,1,2,...,9}

Оператори

- Оператор породен от тази граматика се нарича **мета лингвистична формула** и има вида:

  понятие : :== израз

- Смисъла на всяка мета лингвистична формула е, че с даденото понятие се свързва множеството от стойности породени от израза.

Примери

<буква> ::== а | б | в | … | я

Дефинира:
{а, б, …, я}

<дума> ::== <буква> | <дума> <буква>

Дефинира:
{а, б, …, я} U {а, б, …, я}×{а, б, …, я} U … =
{а, б, …, я, аа, аб, ав, …, ая, ба, бб, бв, …, бя, …, ааа, ааб, …}

Примери

```
<знак> ::== + | -
<цифра> ::== 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
<цяло осмично число> ::==
        <цифра> |
        <знак> <цифра> |
        <цяло осмично число> <цифра>
```

Дефинира:

{0, 1, ..., 7} U {+, –}×{0, 1, ..., 7} U {+, –}×{0, 1, ..., 7}×{0, 1, ..., 7} U ...

{0, 1, ..., 7, 00, 01, 02, ..., 07, 10, 11, 12, ..., 17, ...,
+0, +1, +2, ..., -0, -1, -2, ..., +00, +01, +02, ..., -00, -01, -02, ...}

Метаезик на Бекус-Наур (Разширен)

Разширен метаезик на Бекус-Наур (EBNF)

Символ	Значение	Примери
низ	описва терминали	'=', 'while'
име	описва не терминали	Ident, Statement
=	по дефиниция е	A = b c d .
.	край на правилото	A = b c d .
|	отделя алтернативи	a | b | c ≡ a или b или c
(…)	групира алтернативи	a (b | c) ≡ ab | ac
[…]	опционална част (нула или един път)	[a] b ≡ ab | b
{…}	повтаряща се част (нула или повече пъти)	{ a } b ≡ b | ab | aab | aaab | aaaab | …

Примери

```
буква = 'а' | 'б' | 'в' | … | 'я'.
дума = буква | дума буква.
```

```
буква = 'а' | 'б' | 'в' | … | 'я'.
дума = буква {буква}.
```

Примери

```
знак = '+' | '-'.
цифра = '0' | '1' | '2' | … | '7'.
цяло_осмично_число =
        цифра |
        знак цифра |
        цяло_осмично_число цифра.

цяло_осмично_число = [знак] цифра {цифра}.
```

Въпроси?

apenev@uni-plovdiv.bg