

ЛЕКЦИЯ 11

ТРАНСЛАТОР

ОТ ЕЗИКА АСЕМБЛЕР

-  **Входни данни**
-  **Изходни резултати**
-  **Променливи и таблици**
-  **Ограничения**
-  **Двупасов транслятор**
-  **Еднопасов транслятор**
-  **Многопасов транслятор**

ПРИНЦИПИ НА ПРЕВОДА

Транслаторът от Асемблер (наричан **също Асемблер**) трябва да **преведе** програмата от езика **Асемблер на машинен език**.

Асемблерите работят **само** на **компилативен принцип**, т. е. превеждат цялата програма.

Причините за този маниер на работа **са** **твърде прости** – езикът **Асемблер** за даден ЦП е **твърде близък до** неговия **МЕ**.

Съответствието между **инструкция** на МЕ и **оператор** на Асемблер е **1:1**, т. е. **няма никаква необходимост от интерпретация**.

ВХОДНИ ДАННИ

Основен вход (вх. данни) за транслятора е **текстът** на програмата **на** езика **Асемблер**.

При **Макроасемблерите** като **входни данни** може да бъде посочена **още и библиотека**, която съдържа, **познатите** на автора и често **използвани** от него **макродефиниции**.

Като **допълнителни** входни **данни** към превеждащата програма могат да бъдат посочени и **изборни възможности (options)**, които определят **как да протече преводът**.

ИЗХОДНИ РЕЗУЛТАТИ

В **результат** от работата на транслятора (изходните данни) се получават:

-  **програма** на машинен език;
-  **отпечатък** (листинг), **включващ** текста на **Асемблер**, текста на **МЕ** и **таблица** на символичните имена;
-  **съобщения** за откритите грешки.

Асемблерите **не блокират изпълнението** **при** откриване на **грешка**. Предполага се, че авторът ще отстрани грешките пряко на МЕ.

СХЕМА НА ПРЕВОДА

Преводът от Асемблер до машинен език протича по следната схема:



КА - 11

5/20

ПРИМЕРЕН ЛИСТИНГ

РЕД	БРП	СЪДЪРЖ.	ЕТИКЕТ	КОД	ОПЕР.	КОМЕНТАР
001	0000		*			Събиране на първите 15 цели числа.
002	0000			ORG	2000H	Започва от адрес 2000.
003	2000	4F	СТАРТ	CLRA		Нулева сума.
004	2001	B7 201B		STA	СУМА	
005	2004	B6 201C		LDA	БРОЙ	Начален брояч.
006	2007	B7 201A	ЦИКЪЛ	STA	БРОЯЧ	Запазване на брояча.
007	200A	BB 201B		ADDA	СУМА	Добавяне към сумата.
008	200D	B7 201B		STA	СУМА	
009	2010	B6 201A		LDA	БРОЯЧ	Актуализиране на брояча.
010	2013	8B FF		ADDA	#-1	
011	2015	26 F0		BNE	ЦИКЪЛ	Продължаваме ако не е 0.
012	2017	7E 1000		JMP	ОС	Възврат в ОС при край.
013	201A		ОС	EQU	1000H	Адрес за възврат в ОС.
014	201A		БРОЯЧ	RMB	1	Място за брояча.
015	201B		СУМА	RMB	1	Памет за сумата.
016	201C		НАЧСТ	EQU	15	Началната стойност е 15.
017	201C		БРОЙ	FCB	НАЧСТ	Запомня нач. стойност.
018	201D			END	СТАРТ	

ТАБЛИЦА НА ИМЕНАТА

СТАРТ	2000	ЦИКЪЛ	2007	ОС	1000	БРОЯЧ	201A
СУМА	201B	НАЧСТ	000F	БРОЙ	201C		

КА -11

6/20

ТИПИЧНИ ГРЕШКИ

Обичайните **грешки при Асемблер** са:

- ❶ **дублирано** име (остава **1 дефиниция**).
- ❷ **недефинирано** име (замества се с **0**).
- ❸ неизвестен **МнКОП** (**n** байта **NOP**).
- ❹ некоректна **адресация**.
- ❺ недостатъчен брой **операнди**.
- ❻ грешен **израз**.
- ❼ некоректна **константа** (@9, 2B).
- ❽ преход въвн от разрешения **обхват**.

НЕОБХОДИМИ ТАБЛИЦИ

При своята работа **транслаторът** от Асемблер трябва да **използва следните таблици**:

① мнемоничен код → машинен КОП:

- 🕯 изготвя се **предварително** и не се мени;
- 🕯 съдържа още **дължина** на МИ, **брой** на адресните **полета**, разрешени **адресации** и др.;
- 🕯 освен **1:1** може да бъде ***m:1*** и ***1:n***.

② ас. директива → адрес на ППГ за разбор;

③ символично име → адрес от ОП:

- 🕯 **изготвя се отново** при всеки превод;
- 🕯 използва се **много често**;
- 🕯 съдържа и **допълнителните характеристики**.

ПРОМЕНЛИВИ

За да се следи **докъде в ОП е стигнало разполагането на програмата** е необходима вътрешна променлива, наречена **Брояч за Разполагане на Програмата – БРП** (**Program Location Counter – PLC**).

Стойността на БРП определя от кой адрес в ОП ще се разположи преводът на оператор.

Стойността на БРП може да бъде използвана в изразите чрез **специално име (* или \$)**.

В началото БРП е 0. ORG задава БРП явно.

ОГРАНИЧЕНИЯ

Таблицата на символичните имена се попълва при превода на програмата.

При дефиниране на име (запис в етикетното поле), то се включва в таблицата, а при използването му в израз, таблицата посочва числовия еквивалент (обичайно адрес).

За да привърши преводът с едно прочитане на програмата използването трябва да следва дефинирането, т. е. поява в полето за етикет, и едва след това в полето за операнд.

ОГРАНИЧЕНИЯ (прод.)

Символични имена получават **два вида адреси** от ОП: на **данни** и на **МИ**.

При данните ограничението означава, **те** да бъдат описани **до използването** им **в МИ**.

Това е **възможно и полезно** при четене.

При МИ ограничението означава **забрана за преход** напред в програмата.

Това е **невъзможно изискване**, поради което **при направата на транслятор** трябва да се намери **решение без такова ограничение**.

ДВУПАСОВ ТРАНСЛАТОР

Транслаторът трябва да осъществи **2 неща**:

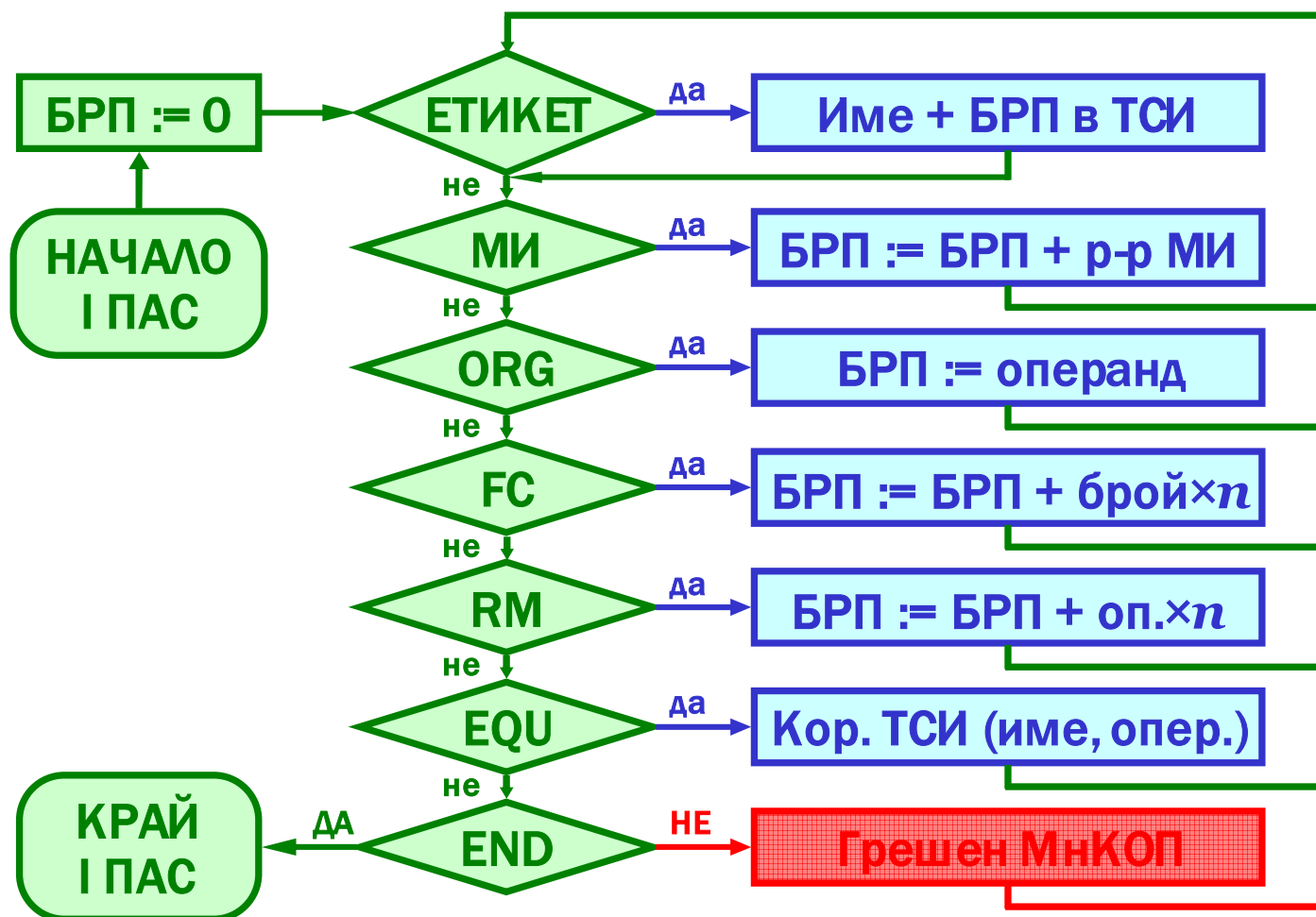
- ❶ да **попълва** таблицата с имена (**ТСИ**);
- ❷ да **генерира МИ** и **данни** за ОП.

Двете дейности **могат да се разделят** защото за попълване на **ТСИ** е **необходим само размерът**, но **не** и **съдържанието на МИ**.

При **първото прочитане** на програмата **се попълва само ТСИ** (**първи пас**).

МИ и данните в ОП се генерират при второ прочитане на програмата като се **използва** и **вече създадената ТСИ** (**втори пас**).

ПЪРВИ ПАС



КА - 11

13/20

МАКРОАСЕМБЛЕРИ

Макроапаратът леко усложнява направата на транслятор от Макроасемблер.

Ограничението всички макродефиници, да предшестват използването на макросите е естествено, поради което макроапаратът може да бъде съвместен с първия пас.

Транслаторът поддържа запомня текста на макродефинициите в допълнителна таблица.

При макроизвикване неговата дефиниция се извлича от тази таблица и след замяна параметрите (като низове) се получава крайният текст на макроразширението.

МАКРОАСЕМБЛЕРИ (прод.)

Поради наличие на **макроапарат** появата на **неизвестен текст** в полето за **МнКОП** не е основание за регистриране на **грешка**.

Този текст се третира като **име на макрос** и се търси в **таблицата** с макродефиниции.

При неуспех и наличие на **библиотека** с дефиниции **името** се търси и в **библиотеката**.
При успех текстът на съответната **дефиниция** се включва в **таблицата** с дефиниции.

Едва след неуспешно търсене в **библиотеката** **транслаторът издава** **съобщение за грешка**.

ВТОРИ ПАС

Вторият пас е доста по-прост.

Не е необходимо да **се проверява етикетното поле**, но някои «мъдри» транслятори правят проверка, защото често сами се объркват.

При **втория пас** се **генерират МИ** и става изчисляване на изразите в операндните полета за определяне на съответните адресни полета на МИ или на съдържанието на ОП.

При достигане на **директива END** **вторият пас завършва** с готова машинна програма.

ЕДНОПАСОВ АСЕМБЛЕР

За ускоряване на превода често **се създават** и **транслатори**, които прочитат програмата само един път (**еднопасови** транслатори).

При срещане на **обръщение напред** (**недефинирано** име) те **запомнят оператора** и **довършват** генерацията, **когато се появи дефиницията** на това необходимо име.

Изработването на подобни транслатори е по-трудно, а техните **действия** ще бъдат обяснени **в следващата лекция**.

ПРОБЛЕМИ

Директива **EQU** създава някои **проблеми**.

Да разгледаме следния **пример**:

Оператори	A(I)	A(II)	B(I)	B(II)	C(I)	C (II)
...	?	?	?	?!	?	5
A EQU B	?!	?!	?	?!	?	5
B EQU C	?!	?!	?!	5	?	5
C EQU 5	?!	?!	?!	5	5	5
...	?!	?!	?!	5	5	5

? – не е дефинирано, ?! – дефинирано, но без стойност.

МНОГОПАСОВ ТРАНСЛАТОР

Многопасовите транслатори **отстраняват** горепосочения преди това **проблем**.

По време на **първия пас** те **запомнят** всички неразрешени директиви **EQU**.

След това **многократно** преминават **през запомнените EQU** до невъзможност за нови определения в тях.

Накрая следва **стандартният втори пас** на транслатора **за генериране на МП**.

**БЛАГОДАРЯ ВИ
ЗА ВНИМАНИЕТО!**

**БЪДЕТЕ С МЕН И В
СЛЕДВАЩАТА ЛЕКЦИЯ,
КОЯТО ЩЕ НИ ОТВЕДЕ
В НЕВЕРОЯТНИЯ СВЯТ НА
АБСОЛЮТНИЯ
ОБЕКТЕН КОД**