

## ЛЕКЦИЯ 10 ЕЗИК АСЕМБЛЕР

- ⌚ Общи принципи на езика
- ⌚ Структура на оператор
- ⌚ Символични имена
- ⌚ Запис на константи
- ⌚ Изрази
- ⌚ Асемблерски директиви
- ⌚ Макроапарат
- ⌚ Условна трансляция

КА - 10

1/32

## МАШИНЕН ЕЗИК

- ① **МИ** се кодират с **поредачи от 0 и 1**, които се разделят на КОП и адресни полета.
- ② Макар, че машинните езици са различни, **разнообразието не е** чак толкова **голямо**.
- ③ **Числовите КОП са безлични** и се помнят трудно, макар че операциите са подобни.
- ④ **Числовите адреси също са безлични** и лесно се объркват един с друг.
- ⑤ **Кодирането на данните** е непривично.
- ⑥ **Корекциите** в програмата допълнително вгорчават живота, защото **изискват промяна на адресните полета** на МИ.

КА - 10

2/32

## АВТОМАТИЗАЦИЯ

Най-простият начин за автоматизиране на програмирането е **замяната на безличните числови елементи с мнемонични буквени:**

⌚ **КОП** ↔ **фиксиран** «говорещи» **съкращения**.

⌚ **Адреси** ↔ **измислени** «говорещи» **имена**.

**Втората стъпка** е **превод** на числовите данни от **привичната 10-ична ПБС** в **непривичната 2-ична ПБС** и **непривичния формат на ПЗ**.

През **1950** г. в Кембридж **Морис Уилкс** за **първи път** при програмиране на **EDSAC** **използва** такъв нов език: **Автокод (Асемблер)**.

КА - 10

3/32

## АСЕМБЛЕР И МЕ

### МАШИНЕН ЕЗИК

- ☹ разбираем за **ЦП**.
- ☹ програмата е **готова за изпълнение**.
- ☹ програмата се **пише трудно и бавно**.
- ☹ **четенето** е трудно.
- ☹ промяна е **сложна** и **грешките** се укриват.
- ☹ изисква **познания за особеностите на ЦП**.

### АСЕМБЛЕР

- ☺ разбираем за **хората**.
- ☺ програмата се **пише по-леко и по-бързо**.
- ☺ програмата се **чете и разбира лесно**.
- ☺ промяната е **лека** и **без някои грешки**.
- ☹ изисква **програма и време за превод**.
- ☹ изисква **познания за особеностите на МЕ**.

КА - 10

4/32

## ЕЗИК АСЕМБЛЕР

Обвързаността на езика Асемблер със съответния **МЕ** означава, че **всеки ЦП има свой собствен език Асемблер**.

**Често за един и същи ЦП се предоставят и няколко езика** със съответните транслатори.

**Превеждащата програма** от езика Асемблер по традиция **не се нарича транслатор** от езика Асемблер, **а също Асемблер**.

**Всички езици** от тип Асемблер си приличат по своите общи черти, което дава възможност те да бъдат изучавани заедно.

КА -10

5/32

## ОБЩИ ПРИНЦИПИ

Всеки **оператор** се пише **на отделен ред**.

Операторите имат **еднаква структура**.

**Един оператор** на езика поражда **една МИ**.

В езика има и **оператори за генериране на данни** и за **разпределяне на ОП**.

**Механизмите за именоване** на адресите и за **използване на тези символични имена** са **еднакви при всички езици** от тип Асемблер.

**Транслаторите** от Асемблер имат **еднакви входни данни** и **еднакви изходни резултати**.

КА -10

6/32

## АСЕМБЛЕР И ЕПВР

### АСЕМБЛЕР

☺ достъпен е **целият МЕ**.

☺ използват се **всички** видове **данни** на ЦП.

☺ **пълен контрол** над изпълнението (**бързо**).

☹ **бавно писане**.

☹ **пълният контрол** над КС е **опасен**.

☹ необходими са **познания за КС**.

### ЕПВР

☺ **бързо и леко писане**.

☺ **по-понятен** за хората.

☺ осигурява някои **АСД**.

☺ **привични термини**.

☺ **не са необходими познания** за КС.

☹ може да се използва **само част от МЕ**.

☹ **липсва контрол** над изпълнението.

КА -10

7/32

## ОПЕРАТОРИ

Всеки **оператор** на езика Асемблер се състои **от следните четири части**:

- ① поле за **етикет**;
- ② поле за **мнемоничен код** на операция;
- ③ поле за **операнди**;
- ④ поле за **коментар**.

**Разпределението на полетата** в реда може да бъде **фиксирано** (от позиция **до** позиция) или **свободно** (всяко поле започва или завършва с характерен **специален знак**).

КА -10

8/32

## СТРУКТУРА НА РЕД

1	6 8	12 14	28 30	80
етикет	МнКОП	операнди	коментар	
__:	__□	□__	;__	

Полето за **коментар** е на разположение на програмиста и **не е част от програмата**.

В **полето** за **МнКОП** може да бъде записано:  
 ♪ **име на МИ** и операторът ще породи тази МИ;  
 ♪ **име на асемблерска директива**.

В **полето** за **операнди** се записват **изрази**.  
 Техният брой **зависи от МнКОП** и **при МИ е равен на броя на техните адресни полета**.

КА-10

9/32

## АДРЕСАЦИЯ

**Съществен елемент** при формиране на адресното поле на МИ от съответния операнд на Асемблер **е посочването на желаня вид адресиране**.

Използват се **две системи**:

- ♪ **различни МнКОП (по-стара)**:  
**A** рег, **по база**; **BAL** рег, **памет по база**;  
**AR** рег, **рег**; **BALR** рег, **косвена регистрова**.
- ♪ **специален знак** преди или след операнда:  
**#...** – непосредствен операнд, **@...** – косвена,  
**(...)+** – автоувеличение, **-(...)** – автонамаление,  
**...(рег)**, **[...+рег]** – по база или с индексирание.

КА-10

10/32

## «МЕЧЕШКИ» УСЛУГИ

Често **«за улеснение»** на програмистите **адресацията се определя от транслятора**:

- ♪ инструкциите за **преход** използват **относителна** адресация, а **останалите** – **абсолютна** и това не се посочва явно.
  - ♪ не се отбелязва явно дали адресацията е **къса или пълна** (абсолютна, относителна).
- Втората услуга** води до **проблеми**, поради което **днес с дописване на знак към МнКОП се разрешава уточнение** къса или пълна.

КА-10

11/32

## РАЗМЕР НА ДАННИТЕ

Съвременните ЦП реализират еднакви **операции над думи с различна дължина**.

За посочване на **размера на данните към обичайния МнКОП се дописва буква**:  
**IBM360**: **A** – 32 бита, **AN** – 16 бита.  
**PDP-11**: **ADD** – 16 бита, **ADDB** – 8 бита.  
**M68000**: **ADD (ADD.W)** – 16 бита, **ADD.L** – 32 бита,  
**ADD.B** – 8 бита.

**Размерът** може да бъде посочен и неявно **чрез използвания акумулатор**:

**M6809**: **ADD A ...** – 8 бита, **ADD D ...** – 16 бита.  
**I80x86**: **ADD AL,...** – 8 бита, **ADD AX,...** – 16 бита.

КА-10

12/32

## СИМВОЛИЧНИ ИМЕНА

**Адресното поле** на МИ посочва **адрес от ОП**.  
 За да се облекчи програмирането **адресите от ОП**, които представляват **интерес**, се отбелязват чрез **символични имена**.  
**Имената се избират** от програмиста и са **еквивалентни на** отбелязания адрес от ОП.  
 Всички езици Асемблер осигуряват **еднакви механизми за дефиниране и използване** на избраните символичните имена.  
**Езиците Асемблер се различават само по броя на знаците** в имената (6, 8, 32).

КА -10

13/32

## ДЕФИНИРАНЕ НА ИМЕ

При превода всеки оператор ще бъде трансформиран в **МИ или други данни**, разположени **някъде в ОП** (напр. от адрес  $\alpha$ ).  
**Символично име**, еквивалентно на този начален адрес  $\alpha$ , се **дефинира** като се запише в **полето за етикет** на този оператор.  
 В етикетните полета на **МИ** е достатъчно да има **символично име само когато** към тези МИ ще се **осъществява преход**.  
**Всички асемблерски директиви за данни в ОП би трябвало да имат символично име в своето етикетно поле (за достъп до ОП)**.

КА -10

14/32

## ИЗПОЛЗВАНЕ НА ИМЕ

Символичните имена се **използват в полето за операнди**.  
 В него те **представят адреса** от ОП, на който **съответстват**.  
 Символичните **имена се заместват с техните еквивалентни числови адреси**.  
**След замяната операндното поле определя съдържанието на дадена клетка с данни в ОП или съответстващото адресно поле на МИ**, като се отчита и посочената адресация.

КА -10

15/32

## ЗАБЕЛЕЖКИ

- ① За «улеснение» на програмирането в някои езици **символичните имена имат и други характеристики** освен числовата си стойност: **размер на данни, област от ОП**.
- ② Регистрите на ЦП не са част от ОП, но **също имат (числови) адреси (номера)**.
- ③ **Посочването на регистър става чрез:**
  - ☞ специално **собствено име**: **AX, AH, AL, BP, DX** (I80x86), **A, B, X, Y** (M6809) и др.
  - ☞ **число**, евентуално предшествано от буква: **0, 1, 15** (IBM-360), **A0, A15, D2** (M68000) и др.

КА -10

16/32

## КОНСТАНТИ

Част от данните, с които оперира една програма (непосредствен операнд на МИ, клетка в ОП и др.), са константи.

Естествено е константите да се пишат в привичната за хората 10-ична ПБС.

Но тази БС е неудобна при МИ за логически операции, където 2-ичният запис е по-ясен.

Основата на БС се посочва по два начина:

☞ чрез представка (префикс): % – 2, @ – 8, \$ – 16  
[& – 10]: (\$C = @14 = %1100 = &12);

☞ чрез наставка (суфикс): B – 2, O – 8, H – 16:  
(12 = 1100B = 14O = 0CH, вместо CH – име).

КА -10

17/32

## ИЗРАЗИ

За да се намали броят на символичните имена и на възможностите за допускане на грешка в полето за операнди се записват изрази, състоящи се от константи и символични имена, свързани със знаци за операции (+, -, \*, /) и кръгли скоби.

Изразите се изчисляват по време на превод за да се попълни съответното АП на МИ.

A+1 е символичното име на клетката след A.

A+1 в ЕПВР изисква прочетената от ОП стойност на променлива да се увеличи с 1.

КА -10

18/32

## АСЕМБЛЕРСКИ ДИРЕКТИВИ

За правилното изпълнение на една програма в ОП трябва да бъдат разположени МИ, които ще изпълни ЦП, и данните, които програмата ще използва наготово при изпълнението си.

При изпълнението са необходими и работни полета в ОП за запис на междинни резултати.

Често трансляторът от Асемблер има нужда и от допълнителни указания за своята работа.

Тези цели се реализират чрез асемблерските директиви, наричани още и псевдооперации.

КА -10

19/32

## ОСНОВНИ ДИРЕКТИВИ

- ① **ORG** <израз> – начало в ОП (**ORiGin**).
- ② **END** [<израз>] – край на програмата.
- ③ **OPT** – управление на транслятора (**OPTion**).
- ④ **FC [DC]** – генерира данни в ОП (**Form [Define] Constants**).
- ⑤ **RM** <брой> – запазва памет (работно поле) за данни (**Reserve Memory**).
- ⑥ **ASSUME** – посочва очаквана стойност на регистри на ЦП (**допускам, че**).
- ⑦ **TITLE, PAGE** – украса на листинга.

КА -10

20/32

## ГЕНЕРИРАНЕ НА ДАННИ

ЦП разпознават и обработват **данни**, които се **различават по тип и дължина**.

Директива **FC (DC)** облекчава **попълването на клетки** в ОП с **начални данни**.

За отразяване на многообразието във вида на желаните данни се използват **две схеми**:

☞ **една директива** и посочване на **вида за всеки операнд**: **DC F'1',H'1'** – **32** и **16** бита.

☞ **различни директиви за всеки вид данни**: **FCB 1,2** – по **8** бита, **FCW 1,2** – по **16** бита, **FCC /текст/** – знакови **низове**.

КА -10

21/32

## ПОВТОРИТЕЛИ

**Видът на генерираните данни определя** характеристиката за **дължина** (ако има такава) **на символичното име** в етикетното поле.

Транслаторът е длъжен да осигури **правилно разполагане** на данните **в ОП**, когато ЦП има **изисквания към техните начални адреси**.

При попълване на **последователни клетки с еднакви данни** е по-удобно пред операнда да се записва **повторител**:

☞ **IBM-360**: DC **3F'5'** – **3** 32-битови **петици**;

☞ **IBM-360**: DB **10 DUP(2)** – **10** байта **двойки**.

КА -10

22/32

## СПЕЦИАЛНИ ИМЕНА

**Повторител 0** принудително **изравнява адреса** без да генерира данни (**DC OH**).

**За същата цел** може да има и **отделна директива**, например **.EVEN** в PDP-11.

Използването на **константи при ЦП без адресация непосредствен операнд** е **затруднено**: с **FC** генерираме **константа с име** и след това **в МИ използваме това име**.

Някои езици допускат **специални имена**, които задължават **транслатора сам да им определи място в ОП**: **AH 5,=H'15'** (IBM 360).

КА -10

23/32

## РАБОТНИ ПОЛЕТА

Използването на **повторител и специален операнд (?)** във **FC** решава проблема за **запазване на ОП** за бъдещо използване:

**DB 20 DUP(?)** – **20** байта с неясно съдържание.

**По-честата практика** е **отделни директиви** за всеки вид данни и посочване на брой:

**RMB 20** – запазва **20** байта ОП;

**RMW 10** – запазва **10** двойки байта ОП;

**RMD 5** – запазва **5** четворки байтове ОП.

**Символичното име** в етикетното поле на **FC** и **RM** **се свързва с адреса на първия байт**.

КА -10

24/32



## ДИРЕКТИВА EQU

Механизмът за дефиниране на символични имена има **съществен недостатък**: чрез него могат да бъдат **именовани само адресите** от ОП, които са **част от** написаната **програма**.

**За** да се осигури възможност за **именоване** и **на външни** за програмата **адреси** има **специална (допълнителна) директива**: име **EQU** израз [коментар]

Тази директива **дава възможност чрез име** **да се обозначават** не само адресите от ОП, но и **всяка желана** при писането **константа**:  
**BR EQU 5** ◀ ◃ **БУФ RMB BR** ◀ ◃ **LDA БУФ+BR-1**

КА -10

25/32

## РАБОТА С ДЪЛГИ ТЕКСТОВЕ

**Компютърните програми**, независимо от езика за програмиране, всъщност **са частен случай на** обикновените **текстове**.

**Естествена реакция при писане на твърде дълъг текст** и при **ограничения във времето** за писане, **е да се използват съкращения**.

Характерно за **текстове със съкращения** е:

- ☺ **по-бързо писане** (**не** се изписва **всичко**);
- ☺ **по-леко възприемане** (**по-малки** по обем);
- ☹ **съвсем непонятни** за този, който **не познава** смисъла на **използваните съкращения**.

КА -10

26/32

## РЕШЕНИЕ НА ПРОБЛЕМА

**Непонятен текст**, съдържащ съкращения, **би станал** напълно **разбираем**, стига към него да бъде добавен **списък на съкращенията**.

**Чрез** добавения **списък** всяко **съкращение** може да се замени с **нормален текст**.

**Замяната** може да бъде реализирана, както **оперативно (on-line)** по време на писане, така и **след края** на писането (**off-line**).

След като **програмите са** специфичен **текст** естествено е **програмистите първи** да се сетят за **програмното решение** на въпроса.

КА -10

27/32

## МАКРОАПАРАТ

Под **макроапарат** се разбират **правилата**, които **определят** как ще става **дефинирането** и **използването на съкращения** в текста.

**Определянето** на съкращение се нарича **макродефиниция**, **използването** му – **макроизвикване**, а **замеящият го пълен текст** е известен като **макроразширение**.

**Макрогенерация** е **процесът на замяна** на **съкратен запис** от текст **без съкращения**.

**Макроасемблер** е асемблер с макроапарат.

КА -10

28/32

## МАКРОАСЕМБЛЕР

Чрез директива **MACRO** с име в етикетното поле се отбелязва **началото на дефиницията**, а чрез **MEND (ENDM)** – нейният край.

**В операндното поле** на MACRO може да бъдат посочвани и **имена на формални параметри**.

За **използване** на дефинирано съкращение **в полето за МнКОП** се записва неговото **име**, а **в полето за операнди** – текстови низове, които са неговите **фактически параметри**.

**ЕПВР** рядко имат макроапарат. Изключения са езици като **PL/1**, **Си**, **Си++**, **ВБ** и др.

КА -10

29/32

## БИБЛИОТЕКИ

Твърде **досадно е** към всеки текст **да бъде добавян** и **списък** с всички съкращения.

**По-изгодна стратегия** при програмирането е често използваните макродефиниции се изнесат **вън от текста** на програмата и **за да могат да бъдат използвани многократно**.

**Такова обособяване** на макродефинициите е известно като **библиотека с дефиниции**.

При наличие на съкращение, без **дефиниция**, **макроапаратът** трябва да осигури **търсене на дефиниция** в посочената му **библиотека**.

КА -10

30/32

## УСЛОВНА ТРАНСЛАЦИЯ

Апаратът за **условна трансляция** предвижда **при определени условия част от написаното да бъде третирано като коментар**.

**Началото** на такъв участък **се отбелязва** с директива **IFxx <израз>**, а **краят** – с **ENDC**.

Когато **изчислената по време на превод стойност** на <израз> **отговаря** на условието **xx редовете до ENDC се превеждат**. В **противен случай** те **се игнорират като коментар**.

**Условията (xx)** биха могли да бъдат **EQ (=0)**, **NE (≠0)**, **GT (>0)**, **GE (≥0)**, **LT (<0)** и **LE (≤0)**.

КА -10

31/32

**БЛАГОДАРЯ ВИ  
ЗА ВНИМАНИЕТО!**

**БЪДЕТЕ С МЕН И В  
СЛЕДВАЩАТА ЛЕКЦИЯ,  
КОЯТО ЩЕ НИ ОТВЕДЕ  
В НЕВЕРОЯТНИЯ СВЯТ НА  
ТРАНСЛАТОРА  
ОТ АСЕМБЛЕР**