ЛЕКЦИЯ 14
МЕЖДУМОДУЛНИ ВРЪЗКИ

- Проблеми на единното програмиране
- Модулно програмиране
- Видове връзки
- Изменения в езика
- Изменения в обектния код

ПРОБЛЕМИ

Създаването на **големи програми от А до Я** поражда много **проблеми**:

1. **преводът** се забавя;
2. не е възможно **няколко програмиста**;
3. не е възможно **няколко компютъра**;
4. **производството** е доста бавно;
5. към края **се забравя** какво е началото.

Посочените проблеми личат **най-ярко при** писане на програми на **Асемблер**.

Някои ЕПВР (**Паскал**, **Алгол–60**) предлагат **само такава възможност**.

МОДУЛИ

За да се облекчи програмирането **големите програми** се разделят **на** обособени **части**, наречени (**програмни**) **модули**.

Всеки модул съдържа определен брой **ППГ и данни** (**константи и работни полета**).

Разделянето на модули дава възможност за привличане на **повече хора**, като **всеки** от тях програмира свой **отделен модул**.

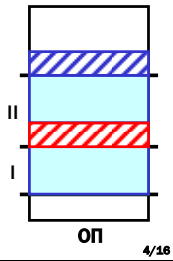
Модулното програмиране дава възможност да се **намали сложността**, но изисква от **транслаторите** да могат **да работят над** една **част**, известно като **разделна компилация**.

НОВИ ПРОБЛЕМИ

Основният проблем при модулната работа е **как** от отделно преведените модули **ще бъде сглобена** единна **обща програма**.

При **Асемблер** може да се постъпи така:

1. **ОП се дели на части** за всеки модул.
2. **„Чуждите адреси“** се описват **с EQU**.
3. **Превежда** се **първият** модул.
4. **Във II** се поставят **правилни адреси от I**.
5. **Превежда** се **вторият** модул.
6. Сега **в I** се поставят **правилните адреси**.
7. **Двата** модула се въвеждат **в ОП**.



ВИДОВЕ ВРЪЗКИ

Новите **проблеми** могат да бъдат **отстранени**, ако натоварим **транслатора** от Асемблер с **нова работа по осигуряване на връзка** между модулите.

Така ще се **елиминира** необходимостта от **многократен превод и разпределение**.

Междумодулните връзки могат да бъдат:

1. **По име** (използват се еднакви имена);
2. **Чрез синоними** за общите адреси (всеки сам си избира име за даден адрес).

ВРЪЗКА ПО ИМЕ

След като **транслаторът** бе лишен от знания за адреса на зареждане, той е изправен пред нов проблем: когато **в един модул се използва** име, което ще бъде **дефинирано в друг** модул, това име **няма да бъде записано в** нито едно **етикетно поле** на този модул.

Такива имена се наричат **външни**.

Новите асемблерски **директиви** са:

EXTERNAL (**XREF**) име1, име2, ...

GLOBAL (**XDEF**, **ENTRY**) име1, име2, ...

ВЪНШНИ ИМЕНА

При изчисляване на **израз външните имена** се заменят от транслатора с **0**, а **крайното изчисление** се реализира **при сглобяването**.

Външните имена могат да бъдат както **адреси**, така и **константи**, поради което могат да **се добавят** и **изваждат произволно**.

Израз, съдържащ външни имена, често се нарича **сложно преместваем**.

Транслаторът трябва да изведе **в обектния код** сведения за нужните **корекции и съответстващите адреси** при дефиниране.

ВРЪЗКА ЧРЕЗ СИНОНИМИ

За да се осигури възможност за синоними на **общите адреси**, всеки модул се разделя на **секции**, които имат свои **имена**.

Секциите биват два вида:

- **частни** – разполагат се **последователно в ОП**;
- **общи** – започват от **еднакъв адрес в ОП**.

Деленето на секции решава и проблемите на **неоднородната ОП**: **ИП** (RAM) и **ПП** (ROM), достъпна **с къс** или **само с пълен** адрес и др.

Символичните имена имат **отместване в** и **име на секцията**, в която са дефинирани.

Последна редакция 10 Септември 2005 г.

СЕКЦИИ

В езика се добавят още **нови директиви**:

Име **SECTION PRIVATE/COMMON**, вид ОП, ...

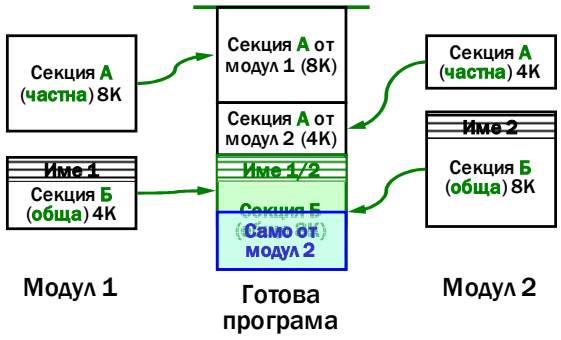
Име **ENDS** (начало и край на секция)

Транслаторът води **отделен брояч за разполагането** на всяка секция (**БРС**) **за** да може да определи **характеристиките на** дефинираните в нея **символични имена**.

Части от секция **могат да се записват** на **произволно** място в текста на програмата, като **при първо срещане** нейният **БРС е 0**.

КА-14 9/16

ПРИМЕР: ВИДОВЕ СЕКЦИИ



КА-14 10/16

ЗАБЕЛЕЖКИ

❶ По традиция (от езика Фортран) **всеки модул** може да съдържа точно **една обща секция без име**;

❷ **Безименната** секция **е обща за цялата програма** (именованите общи секции са общи само за част от модулите);

❸ В **безименната секция не** може да **се генерира код** (т. е. в нея може да се използват **само директиви RM**);

❹ Безименна обща секция **не се поддържа от всички езици** Асемблер.

КА-14 11/16

ОБЕКТЕН КОД

Преводът на парче изисква **към текста на модулите** да бъдат добавени **всички сведения за** правилното **довършване и сглобяване** на програмата.

Този обектен код често се нарича **език на свързващата програма**.

Той се използва от **всички компилатори**, поради което отделните **модули** могат да се пишат **на различни езици** за програмиране (**най-подходящите** за съответния алгоритъм).

КА-14 12/16

ЗАБЕЛЕЖКИ

Разделната компилация и еднотипният изход на езика за свързване **не са достатъчни** за да можем да се възползваме от възможността **да си избираме ЕП за всеки** отделен **модул**.

За сглобяването е **необходимо** още **и**:

☝ компилаторите да прилагат **еднакви съглашения** при предаването на фактически параметри;

✌ програмистите да знаят **съответствието** **между типовете от данни** на двата езика.

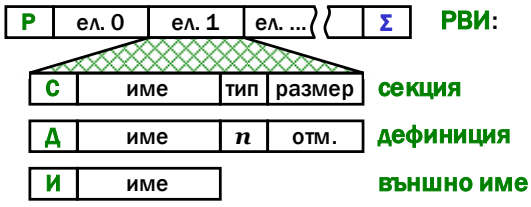
Предимствата на модулите са очевидни: **намаляване** на сложността **и** работа на **едро**.

КА-14 13/16

РЕЧНИК НА ИМЕНАТА

Модулите започват с **речник на външните имена**, съдържащ всички данни за модула.

Елементите на речника **се номерират за** да бъдат цитирани **по-лесно** в другите записи.



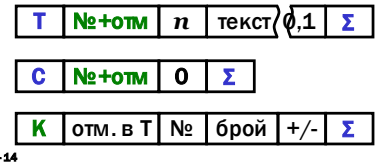
КА-14 14/16

ДРУГИ ПРОМЕНИ

Главната **промяна в другите записи** е, че **адресите** се посочват като **№+отместване**.

Записите за **корекция** са **основно** променени и следват текстовия, който се коригира.

Стартов адрес може и да **липсва**, тъй като той се **определя при сглобяването** на програмата.



КА-14 15/16

БЛАГОДАРЯ ВИ ЗА ВНИМАНИЕТО!

БЪДЕТЕ С МЕН И В ПОСЛЕДНАТА ЛЕКЦИЯ, КОЯТО ЩЕ НИ ОТВЕДЕ В НЕВЕРОЯТНИЯ СВЯТ НА СВЪРЗВАЩИЯ РЕДАКТОР