Лекція-3. Типи даних

- При програмуванні мовою ассемблера використовуються дані наступних типів:

1. ***Безпосередні*** дані, що представляють собою числові чи символьні значення, що є частиною команди.

2. Дані ***простого типа***, що описуються за допомогою обмеженого набору директив резервування пам'яті, і дозволяють виконувати самі елементарні операції по розміщенню й ініціалізації числової і символьної інформації.

3. Дані ***складного типу***, які були введені в мову асемблера з метою полегшення розробки програм.

Простий тип даних (фізична ***інтерпретація***)

- Поняття *простого* типу даних носить двоїстий характер. З погляду розмірності (***фізична інтерпретація***), мікропроцесор апаратно підтримує наступні основні типи даних
- *байт* — вісім послідовно розташованих бітів, пронумерованих від 0 до 7, при цьому біт 0 є самим молодшим значущим бітом;
- *слово* — послідовність із двох байт, що мають послідовні адреси. Розмір слова — 16 біт. ***Молодший байт завжди зберігається по меншій адресі. Адресою слова вважається адреса його молодшого байта. Адреса старшого байта може бути використаний для доступу до старшої половини слова***.
- *подвійне слово* — послідовність з чотирьох байт (32 біта), розташованих по послідовних адресах.
- *учетверенне слово* — послідовність з восьми байт (64 біта), розташованих по послідовних адресах. Подвійне слово, що містить нульовий біт, називається молодшим подвійним словом, а подвійне слово, що містить 63-й біт, — старшим подвійним словом.

Простий тип даних

(***фізична інтерпретація***)

Простий тип даних (логічна ***інтерпретація***)

Також мікропроцесор на рівні команд підтримує ***логічну інтерпретацію*** цих типів

- *Цілий тип зі знаком* — подвійне значення зі знаком, розміром 8, 16 чи 32 біта. Знак у цьому подвійному числі міститься в 7, 15 чи 31-м битці відповідно. Нуль у цих бітах в операндах відповідає позитивному числу, а одиниця — негативному. Негативні числа представляються в додатковому коді. Числові діапазони для цього типу даних наступні:
  - 8-розрядне ціле — від –128 до +127;
  - 16-розрядне ціле — від –32 768 до +32 767;
  - 32-розрядне ціле — від -2^{31} до $+2^{31}-1$.

Простий тип даних (логічна ***інтерпретація***)

- *Цілий тип без знака* — подвійне значення без знака, розміром 8, 16 чи 32 біта. Числовий діапазон для цього типу наступний:
  - байт — від 0 до 255;
  - слово — від 0 до 65 535;
  - подвійне слово — від 0 до $2^{32}–1$.

Простий тип даних (логічна ***інтерпретація***)

- *Вказівник на пам'ять двох типів:*
  - ближнього типу — 32-розрядна логічна адреса, що представляє собою відносний зсув у байтах від початку сегмента. Ці покажчики можуть також використовуватися в суцільний (плоскої) моделі пам'яті, де сегментні складові однакові;
  - далекого типу — 48-розрядна логічна адреса, що складається з двох частин: 16-розрядної сегментної частини — селектора, і 32-розрядного зсуву.

Простий тип даних (логічна ***інтерпретація***)

- *Ланцюг* — представляє собою деякий безупинний набір байтів, чи слів подвійних слів максимальної довжини до 4 Гбайт.

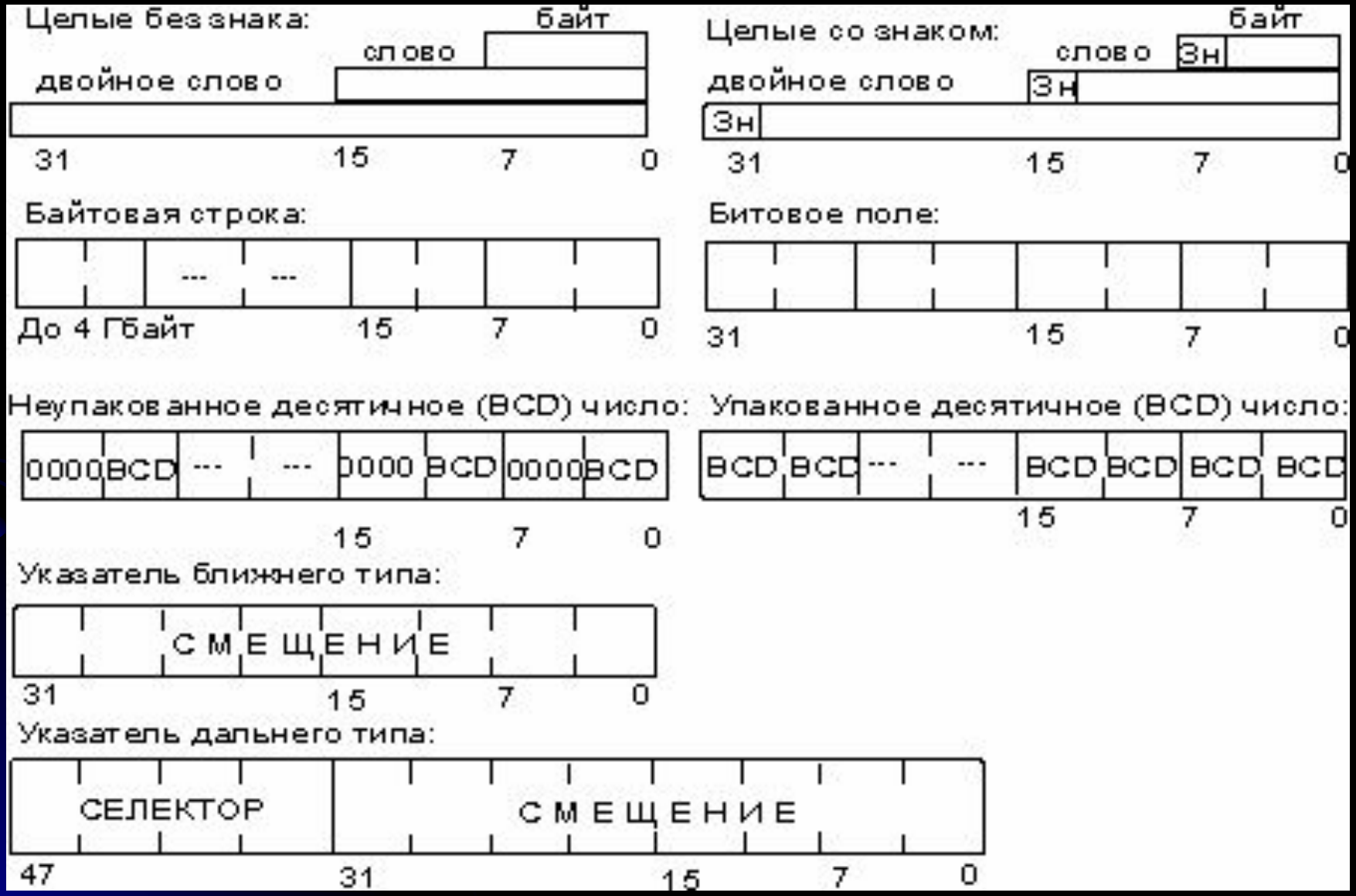
- *Бітове поле* являє собою безупинну послідовність біт, у якій кожен біт є незалежним і може розглядатися як окрема перемінна. Бітове поле може починатися з будь-якого біта будь-якого байта і містити до 32 біт.

Простий тип даних (логічна ***інтерпретація***)

- *Неупакований двоїчно-десятковий тип* — байтове представлення десяткової цифри від 0 до 9. Неупаковані десяткові числа зберігаються як байтове значення без знака по одній цифрі в кожнім байті. Значення цифри визначається молодшим напівбайтом.
- *Упакований двоїчно-десятковий тип* являє собою упаковане представлення двох десяткових цифр від 0 до 9 в одному байті. Кожна цифра зберігається у своєму напівбайті. Цифра в старшому напівбайті (біти 4–7) є старшою.

Простий тип даних

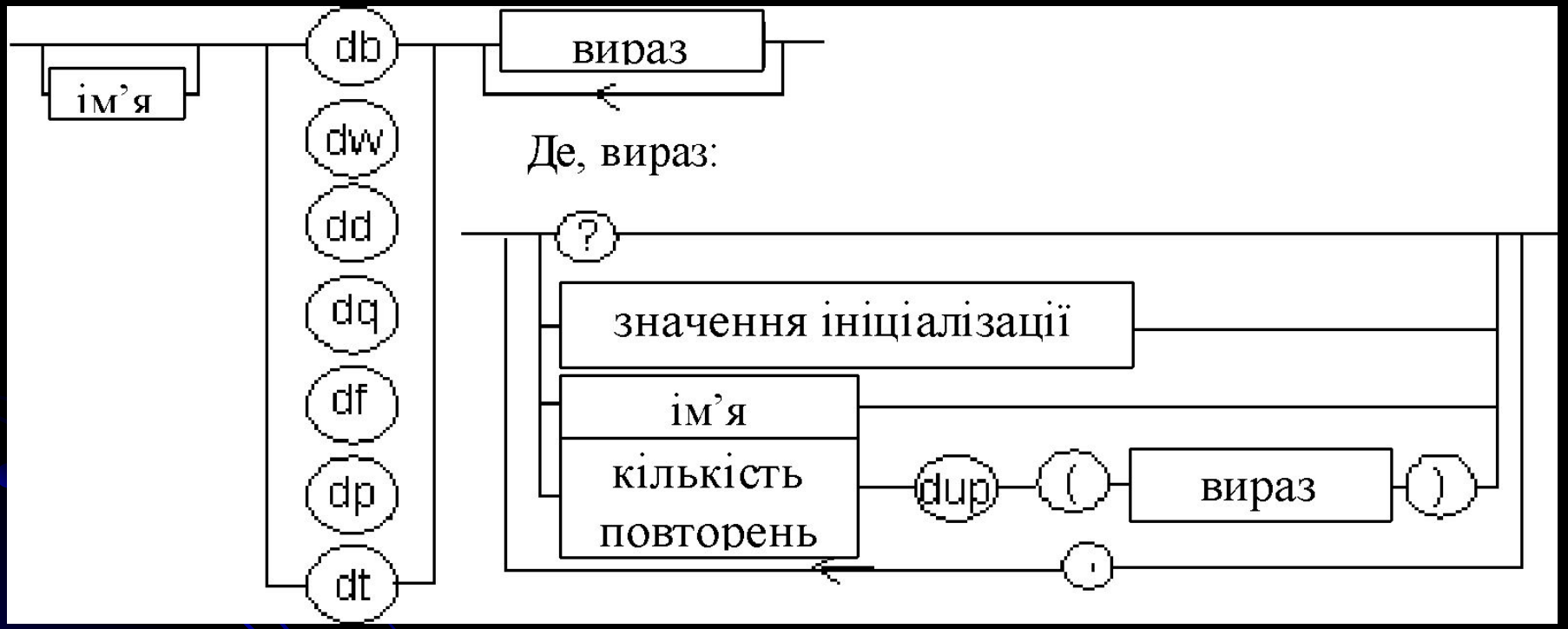
(***логічна інтерпретація***)



Описання простих типів даних

- Для описання простих типів даних у програмі використовуються *спеціальні директиви резервування й ініціалізації даних*, що, по суті, є вказівками транслятору на виділення визначеного обсягу пам'яті. Якщо проводити аналогію з мовами високого рівня, то директиви резервування й ініціалізації даних є визначеннями змінних.
- *Машинного еквівалента цим директивам немає; просто транслятор, обробляючи кожну таку директиву, виділяє необхідну кількість байт пам'яті і при необхідності ініціалізує цю область деяким значенням*.

Описання простих типів даних



Описання простих типів даних

- **?** - показує, що вміст поля не визначено, тобто при завданні директиви з таким значенням вираження уміст виділеної ділянки фізичної пам'яті змінюватися не буде. Фактично, створюється неініціалізована перемінна;
- **значення ініціалізації** — значення елемента даних, що буде занесено в пам'ять після завантаження програми.
- **вираз** — ітеративна конструкція із синтаксисом. Ця конструкція дозволяє повторити послідовне занесення у фізичну пам'ять виразу в дужках n раз.
- **ім'я** — деяке символічне ім'я чи мітки комірки пам'яті в сегменті даних, використовуване в програмі.

Описання простих типів даних

- **db** — резервування пам'яті для даних розміром 1 байт.
- Директивою **db** можна задавати наступні значення:
  - вираз чи константу, що приймає значення з діапазону:
    - для чисел зі знаком –128...+127;
    - для чисел без знака 0...255;
  - 8-бітове відносний вираз, що використовує операції HIGH і LOW;
  - символьний рядок з одного чи більш символів. Рядок полягає в лапки. У цьому випадку визначається стільки байт, скільки символів у рядку.

Описання простих типів даних

- **dw** — резервування пам'яті для даних розміром 2 байти.
- Директивою **dw** можна задавати наступні значення:
  - вираз чи константу, що приймає значення з діапазону:
    - для чисел зі знаком –32 768...32 767;
    - для чисел без знака 0...65 535;
  - вираз, що займає 16 чи менш біт, у якості якого може виступати зсув у 16-бітовому чи сегменті адреса сегмента;
  - 1- чи 2-байтовую рядок, укладений у лапки.

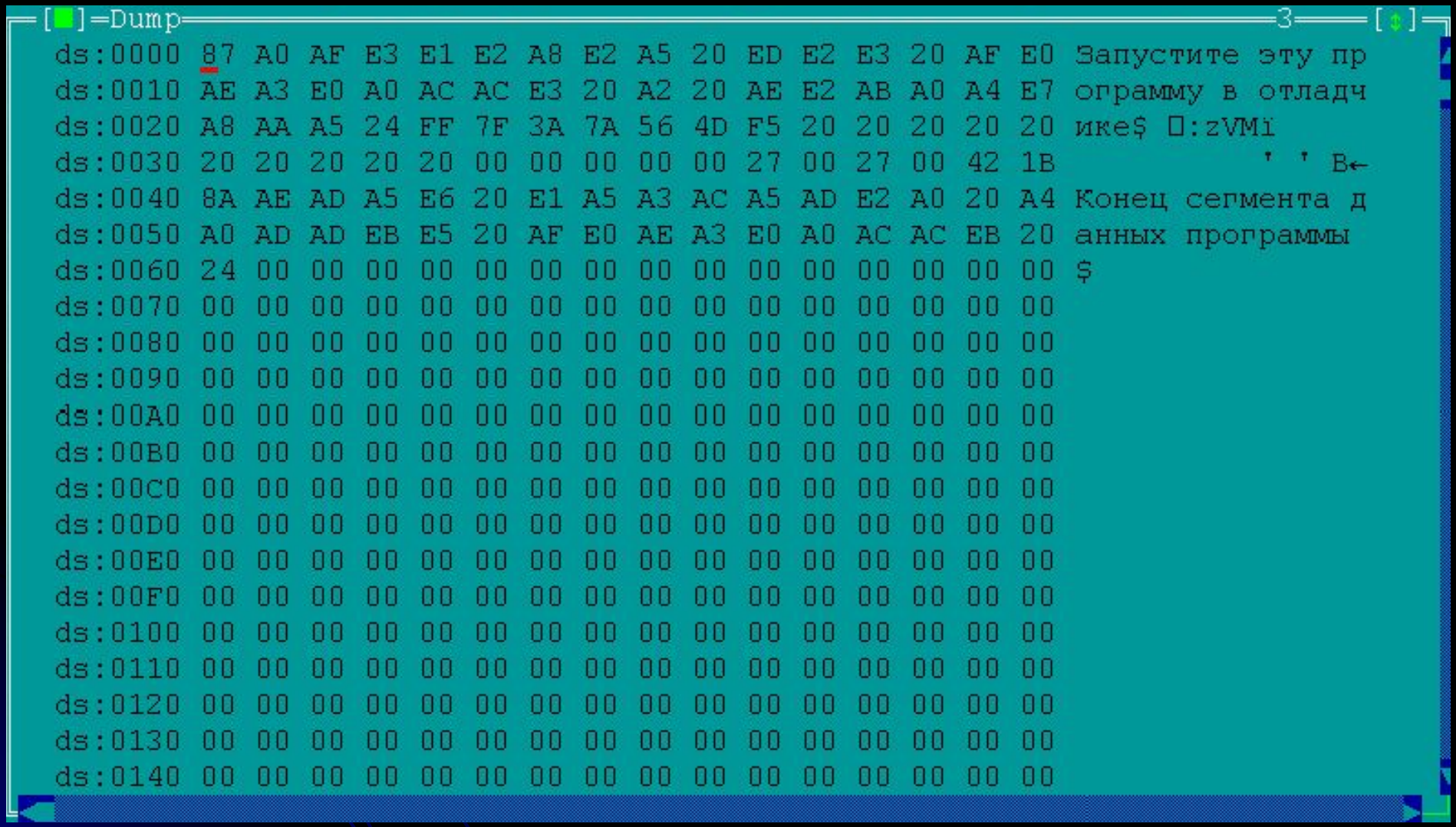
Описання простих типів даних

- **dd** — резервування пам'яті для даних розміром 4 байти.
- Директивою **dd** можна задавати наступні значення:
  - вираз чи константу, що приймає значення з діапазону:
    - для чисел зі знаком –2 147 483 648...+2 147 483 647;
    - для чисел без знака 0...4 294 967 295;
  - відносний чи адресний вираз, що складається з 16-бітової адреси сегмента і 16-бітового зсуву;
  - рядок довжиною до 4 символів, укладений у лапки.

Описання простих типів даних

- **df** — резервування пам'яті для даних розміром 6 байт;
- **dp** — резервування пам'яті для даних розміром 6 байт.
- **dq** — резервування пам'яті для даних розміром 8 байт.
- **dt** — резервування пам'яті для даних розміром 10 байт.
- Директивами можна задавати наступні значення:
  - для чисел зі знаком –2 147 483 648...+2 147 483 647;
  - для чисел без знака 0...4 294 967 295;

```
masm
model   small
.stack 100h
.data
message     db   'Запустите эту программу в отладчике',',$'
perem_1     db   0ffh
perem_2     dw   3a7fh
perem_3     dd   0f54d567ah
mas db 10   dup (' ')
pole_1  db   5 dup (?)
adr  dw  perem_3
adr_full dd   perem_3
fin db    'Кінець сегмента дані програми $'
.code
start:
    mov ax,@data
      mov      ds,ax
      mov      ah,09h
      mov      dx,offset message
      int   21h
      mov      ax,4c00h
      int   21h
    end   start
```


[■]=Dump 3 [↕]
ds:0000 87 A0 AF E3 E1 E2 A8 E2 A5 20 ED E2 E3 20 AF E0 Запустите эту пр
ds:0010 AE A3 E0 A0 AC AC E3 20 A2 20 AE E2 AB A0 A4 E7 ограмму в отладч
ds:0020 A8 AA A5 24 FF 7F 3A 7A 56 4D F5 20 20 20 20 20 ике$ □:zVMï
ds:0030 20 20 20 20 20 00 00 00 00 00 27 00 27 00 42 1B ' ' B←
ds:0040 8A AE AD A5 E6 20 E1 A5 A3 AC A5 AD E2 A0 20 A4 Конец сегмента д
ds:0050 A0 AD AD EB E5 20 AF E0 AE A3 E0 A0 AC AC EB 20 анных программы
ds:0060 24 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 $
ds:0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:00A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:00C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:00E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

- з зсувом **0000** розташовані символи, що входять у рядок *message*. Вона займає 34 байта. Після неї знаходиться байт, що має в сегменті даних символічне ім'я *perem_1*, значення цього байта **offh**.
- Тепер зверніть увагу на те, як розміщені в пам'яті байти, що входять у слово, позначене символічним ім'ям *perem_2*. Спочатку байт зі значенням **7fh**, а потім зі значенням **3ah**. Як бачите, у пам'яті дійсно спочатку розташований молодший байт значення, а потім старший.
- Зупинимося лише ще двох специфічних особливостях використання директив резервування й ініціалізації пам'яті. Мова йде про випадок використання в полі операндів директив **dw** і **dd** символічного імені з полючи ім'я цієї чи іншої директиви резервування й ініціалізації пам'яті. У нашому прикладі сегмента даних це директиви з іменами *adr* і *adr_full*.
- Коли транслятор зустрічає директиви опису пам'яті з подібними операндами, те він формує в пам'яті значення адрес тих перемінних, чиї імена були зазначені в якості операндов. У залежності від директиви, застосовуваної для одержання такої адреси, формується або повна адреса (директива **dd**) у виді двох байтів сегментної адреси і двох байтів зсуву, або тільки зсув (директива **dw**)