Datové typy

Tomáš Oberhuber

Faculty of Nuclear Sciences and Physical Engineering
Czech Technical University in Prague

Datové typy

- pomocí příkazu `DC` můžeme definovat proměnné
- `DS` se chová stejně jen neinicializuje paměť
- kompletní syntaxe má tento tvar
  - `[name] DC [d]t[L`*len*`]'value'`
  - `t` označuje typ, ten je povinný
  - `L` udává délku, není povinný
  - `d` je duplikační faktor, není povinný
  - `name` je jméno, není povinné
  - `value` hodnota, je povinná u `DC`

Datové typy

Jaké jsou možné datové typy?

Typ	Specifikátor	defaultní délka	defaultní zarovnání
Character	C	1	byte
Hexadecimal	X	1	byte
Binary	B	1	byte
Zoned decimal	Z	1	byte
Packed decimal	P	1	byte
Halfword	H	2	halfword
Fullword	F	4	fullword
Long floating point	D	8	doubleword
Doubleword	FD	8	doubleword

Zarovnání v paměti

Zarovnávání v paměti:

- na začátku má LC (location counter) hodnotu 0
- `A DC CL2'HI'`
- alokujeme 2 znaky (character), LC se posune na 2
- `B DC F'123'`
- alokujeme typ fullword, který musí být zarovnán na násobek 4
- LC se posune na 4, alokuje fullword a končí na pozici 8

- uvedeme-li délku, zarovnání se tím zruší
- `ALIGNED DC F'123'`
- `NONALIGNED DC FL4'123'`

Udání délky

- je-li délka uvedena, má vyšší prioritu před délkou napočítanou překladačem
  - `S DC C'HELLO'`
    - definuje `HELLO`
  - `S DC C4'HELLO'`
    - definuje `HELL`

Duplikační faktor

- udává, kolikrát se daný typ v paměti opakuje
- délka udává velikost typu, duplikační faktor má spíš význam definování pole
- `DATA DS 100PL2`
- často se používá duplikační faktor roven 0
- není pak alokováno žádné místo v paměti
- má to dvě uplatnění
  1. vynucení zarovnání
  2. definování struktur

Nulový duplikační faktor pro zarovnání v paměti

```
1                DS      0F
2  ALIGNED       DC      C'HELLO'
```

- zde je proměnná `ALIGNED` zarovnána na fullword

Nulový duplikační faktor pro definování struktur

```
1  STRUCT    DS    0CL30
2  NAME      DS    CL10
3  SURNAME   DS    CL20
```

- zde definujeme strukturu `STRUCT`, která obsahuje dvě položky `NAME` a `SURNAME`
- asembler si pamatuje délků proměnných (ta je dána pomocí `L`)
- lze tak snadno kopírovat celou strukturu nebo její položky

```
1          MVC  STRUCT,DATA1
2          MVC  NAME,DATA2
3          MVC  SURNAME,DATA3
```

- zde se překopíruje 30 bajtů na adresu `DATA1`, 10 bajtů do `DATA2` a 20 bajtů do `DATA3`

Rozdíl mezi duplikačním faktorem a délkou

```
1  X1          DS      CL100
2  X2          DS      100C
3  X3          DS      50CL2
```

- `MVC X1,DATA1` překopíruje 100 bajtů
- `MVC X2,DATA2` překopíruje 1 bajt
- `MVC X3,DATA3` překopíruje 2 bajty

Příklady

Definice	Výsledek
`C1 DC C'ABC'`	`C1C2C3`
`C2 DC CL5'ABC'`	`C1C2C34040` [1]
`C3 DC CL2'ABC'`	`C1C2` [2]
`C4 DC 2C'ABC'`	`C1C2C3C1C2C3`
`C5 DC CL6'ABC'`	`C1C2C3404040`
`X1 DC X'3C2F'`	`3C2F`
`X2 DC XL3'3C2F'`	`003C2F`
`X3 DC XL1'3C2F'`	`2F`
`X4 DC X'C2F'`	`0C2F`
`X5 DC 2X'C2F'`	`0C2F0C2F`
`X5 DC X'C2F,C2F,2F'`	`0C2F0C2F2F` [3]
`B1 DC B'10111'`	`17`
`B2 DC BL2'10111'`	`0017` [4]

[1] řetězec je zprava doplněn mezerami
[2] řetězec je useknutý na udanou délku
[3] udání více hodnot oddělených čárkou
[4] doplnění nulami zleva

Příklady

Definice	Výsledek
`Z1 DC Z'12345'`	`F1F2F3F4C5`
`Z2 DC ZL6'12345'`	`F0F1F2F3F4C5`
`Z3 DC ZL4'12345'`	`F2F3F4C5`
`Z4 DC Z'1,-2,3'`	`C1D2C3`
`P1 DC P'12345'`	`12345C`
`P2 DC PL4'12345'`	`0012345C`
`P3 DC PL2'12345'`	`345C`
`P4 DC P'1,37,-4892'`	`1C037C04892D`
`P5 DC P'1.37'`	`137C` [5]
`H1 DC H'10'`	`000A`
`H2 DC H'-5'`	`FFFB`
`H2 DC 2H'5'`	`00050005`
`F1 DC F'10'`	`0000000A`
`F2 DC F'-5'`	`FFFFFFFB`
`F2 DC 2F'5'`	`0000000500000005`

[5] desetinná čárka se ignoruje

Literály

Literály umožňují nahradit tento kód

```
1                AH    R5,ONE
2  ONE           DC    H'1'
```

tímto

```
1                AH    R5,=H'1'
```

- překladač sám vytvoří proměnou s hodnotou 1 a umístí jí na konec kontrolní sekce nebo v místě použití příkazu `LTORG`