Soubory

Tomáš Oberhuber

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

Práce se soubory

- práce se soubory probíhá pomocí maker
- nejprve poíšeme DCB zvoleného souboru
- pak lze použít makra `OPEN, CLOSE, GET PUT`

DCB - data-set control block

```
1  dcbname     DCB     DDNAME=FILENAME,     X
2                      DSORG=PS,            X
3                      RECFM=FB,            X
4                      MACRF=GM
```

- `DDNAME` je název data-setu
- `DSORG` je organizace data-setu
- `RECFM` je formát data-setu
- `MACRF` popisuje způsob práce se souborem

MACRF - způsob práce se souborem

- první znak je `PG` (get) nebo `P` (put) a značí čtení nebo zápis
- druhý znak je `M` (move) nebo `L` (locate) a značí zda se bude nebo nebude používat buffer
  - move - data jsou kopírována z nebo do programátorem alokovaného pole
  - locate - makro pro čtení vrací ukazatel na vztupní buffer, kde jsou data uložena
    - je to rychlejší ale komplikovanější

Popis souboru pro zápis

```
1  dcbname     DCB     DDNAME=FILENAME,     X
2                      DSORG=PS,            X
3                      RECFM=FBA,           X
4                      MACRF=PM,            X
5                      BLKSIZE=1330,        X
6                      LRECL=133
```

- `BLKSIZE` je velikost fyzického bloku
- `LRECL` je velikost logického záznamu
- priority při určování velikosti fyzického bloku a logického záznamu
  1. DCB v programu
  2. JCL
  3. parametry data-setu

Makra OPEN a CLOSE

- makro OPEN se používá ve tvaru
  `OPEN(dcbname,(mode))`
  - `mode` není povinný
- makro CLOSE se používá ve tvaru
  `CLOSE(dcbname,(option))`
  - `option` není povinné

Makra GET a PUT

- obě makra se používají ve tvaru
  - `GET dcb,area`
  - `GET dcb,area`

```
1                GET     INPUTDCB,INAREA
2                PUT     OUTPUTDCB,OUTAREA
3                 ...
4  INAREA        DS      CL80
5  OUTAREA       DS      CL133
```

Konec souboru

- k definici DCB lze přidat parametr `EODAD=label`
- `label` udává adresu, na kterou se provede skok po dosažení konce souboru

Příklad

```
1  COPY          SETUP
2                OPEN    (INFILE,(INPUT))
3                OPEN    (OUTFILE,(OUTPUT))
4  *
5  READLOOP      DS      0H
6                GET     INFILE,INREC
7                MVC     PRINTREC,=CL133' '
8                MVC     PRINTREC+11(80),INREC
9                PUT     OUTFILE,PRINTREC
10               B       READLOOP
11 *
12 ENDDATA       DS      0H
13               CLOSE   (INFILE)
14               CLOSE   (OUTFILE)
15               ENDIT
16 INFILE        DCB     DDNAME=SYSIN,          X
17                       DSORG=PS,              X
18                       MACRF=GM,              X
19                       EODAD=ENDDATA
20 OUTFILE       DCB     BLKSIZE=133,           X
21                       DDNAME=SYSPRINT,       X
22                       DSORG=PS,              X
23                       LRECL=133,             X
24                       MACRF=PM,              X
25                       RECFM=FBA
26 INREC         DS      CL80
27 PRINTREC      DS      CL133
28               END     COPY
```

Použité instrukce

- `B` – branch
  - provede skok na danou adresu
- `MVC` – move
- `MVC PRINTREC,=CL133' '` maže výstupní buffer
- lze to provést i takto:

```
1                MVI  PRINTREC,=C' '
2                MVC  PRINTREC+1(132),PRINTREC
```

- první způsob alokuje literál o velikosti 133 bajtů a celkem zabere 139 bajtů
- druhý zabírá 10 bajtů

Použité instrukce

- `MVC PRINTREC+11(80),INREC`
  - ve výstupní buferu se odsadí 11 znaků vlevo a překopíruje se celkem 80 znaků

Nulový duplikační faktor pro čtení/zápis tabulek

```
1  PRINTREC    DS    0CL133
2  ASA         DS    CL1
3              DS    CL10
4  INREC       DS    CL80
5              DS    CL42
```

- do `INREC` načteme vstupní data a do `ASA` zpíšeme kontrolní ASA znak
- tím máme nachystaný celý bufer a můžeme provést rovnou zápis

Nulový duplikační faktor pro čtení/zápis tabulek

1	CLIENT	**DS**	0CL133
2	**NAME**	**DS**	CL40
3	SURNAME	**DS**	CL40
4	ADRESS	**DS**	CL40
5	ACCOUNT	**DS**	CL13

- ze souboru můžeme načítat do `CLIENT` data klientů a okamžitě máme snadný přístup k jejich udajům

Konverze datových typů

Převod znaků na formát PACKED:

```
1                PACK   B,A
2  A             DC     C'589'
3  B             DS     PL4
```

- použité instrukce
  - `PACK D1(L1,B1),D2(L2,B2)`
- výsledek
  - `A = x'F5F8F9'`
  - `B = x'0000589F'`

Konverze datových typů

Převod z formátu PACKED do znaků:

- `UNPK D1(L1,B1),D2(L2,B2)` - unpack
- druhý operand převede z fomátu PACKED do znakoného zapísu a uloží do prvního operandu

Konverze datových typů

Převod z formátu PACKED do binární formy:

- `CVB R1,D2(X2,B2)` – convert to binary
  - druhý operand ve formátu PACKED o délce 8 bajtů je převedn do binární formy a zapsán do registru R1

Konverze datových typů

Převod z binárního formátu do formátu PACKED:

- `CVD R1,D2(X2,B2)` – convert to packed
  - první operand je přepsán z binárního tvaru do formátu PACKED o velikosti 8 bajtů
  - jde o další výjimku, kdy se výsledek zapisuje do druhého operandu

Příklad

Příklad je v EXER08.

Pokročilý převod z formátu PACKED

- instrukce ED umí převádět formát PACKED na znaky mnohem lépe
- `ED D1(L,B1),D2(B2)` – edit
- druhý operand obsahuje číslo ve formátu PACKED o délce `L` bajtů
- první operand obsahuje masku do níž je vepsáno číslo z druhého operandu v znakovém zápisu

Maska pro instrukci EDIT

Maska se skládá z:

- vyplňovací znak – fill byte
- znak pro zápis jedné číslice editovaného čísla – `x'20'` – digit selector
- znak pro začátek významných cifer – `x'21'` – significance starter
- restartovací znak pro výpis dalšího čísla – `x'22'` – field separator

Příklad

```
1          ED      A,B
2  A       DC      X'0001000C'
3  B       DC      X'4020202020202020'
```

- po zavolání `ED` bude `B` obsahovat `x'40404040F1F0F0F0'`
- to by se vypsalo jako `1000`

Příklad

```
1          ED    A,B
2  A       DC    X'0001000C'
3  B       DC    X'402020206B202020'
```

- po zavolání `ED` bude `B` obsahovat `x'40404040F16BF0F0F0'`
- to by se vypsalo jako `1,000`
- `6B` kóduje čárku

Příklad

```
1          ED      A,B
2  A       DC      X'0000999C'
3  B       DC      X'5C20206B2020204B2020'
```

- po zavolání `ED` bude `B` obsahovat `x'5C5C5C5C5C5CF94BF9F9'`
- to by se vypsalo jako `******9.99`
- `5C` kóduje hvězdičku
- `4B` kóduje tečku

Příklad

```
1          ED      A,B
2  A       DC      X'0000000C'
3  B       DC      X'4020202020202020'
```

- po zavolání `ED` bude `B` obsahovat `x'4040404040404040'`
- to by se vypsalo jako samé mezery

Příklad

```
1          ED      A,B
2  A       DC      X'0000000C'
3  B       DC      X'4020202020202120'
```

- po zavolání `ED` bude `B` obsahovat `x'4040404040404F0'`
- to by se vypsalo jako `0`

Příklad

```
1          ED      A,B
2  A       DC      X'0000000C'
3  B       DC      X'4020202020202120'
```

- po zavolání `ED` bude `B` obsahovat `x'4040404040404F0'`
- to by se vypsalo jako `0`

Příklad

```
1              ED      A,B
2  A           DC      X'0067890C'
3  B           DC      X'40204B202020202020'
```

- po zavolání `ED` bude `B` obsahovat `x'40404040F6F7F8F9F0'`
- to by se vypsalo jako `67890`
- pokud by `A` reprezentovalo číslo 0.06789, nemáme správný výsledek
- zde je potřeba použít significance starter

Příklad

```
1          ED      A,B
2  A       DC      X'0067890C'
3  B       DC      X'40214B202020202020'
```

- po zavolání `ED` bude `B` obsahovat `x'40404B40F6F7F8F9F0'`
- to by se vypsalo jako `.067890`

Příklad

```
1          ED    A,B
2  A       DC    X'000067890C'
3  B       DC    X'402021204B202020202020'
```

- po zavolání `ED` bude `B` obsahovat `x'404040F04B40F6F7F8F9F0'`
- to by se vypsalo jako `0.067890`

Příklad

```
1          ED    A,B
2  A       DC    X'000067890C'
3  B       DC    X'402021204B202020'
```

- po zavolání `ED` bude `B` obsahovat `x'404040F04B40F6F7'`
- to by se vypsalo jako `0.067`

Příklad

```
1          ED    A,B
2  A       DC    X'0001000D'
3  B       DC    X'40206B2020206B202020'
```

- po zavolání `ED` bude `B` obsahovat `x'4040404040F16BF0F0F0'`
- to by se vypsalo jako `1,000`

Příklad

```
1          ED    A,B
2  A       DC    X'0001000D'
3  B       DC    X'40206B2020206B20202060'
```

- po zavolání `ED` bude `B` obsahovat `x'4040404040F16BF0F0F060'`
- to by se vypsalo jako `1,000-`

Příklad

```
1          ED    A,B
2  A       DC    X'0001000C'
3  B       DC    X'40206B2020206B20202060'
```

- po zavolání `ED` bude `B` obsahovat `x'4040404040F16BF0F0F040'`
- to by se vypsalo jako `1,000`

Příklad

```
1          ED    A,B
2  A       DC    X'0001000D'
3  B       DC    X'40206B2020206B20202040D4C9D5
```

- po zavolání `ED` bude `B` obsahovat `x'4040404040F16BF0F0F040D4C9D5E4E2'`
- to by se vypsalo jako `1,000 MINUS`