

# Podprogramy

Tomáš Oberhuber

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

Videa na Youtube:

Přednáška

Cvičení

# Instrukce BAL

## Branch and link

BAL R1, D2 (X2, B2)

- adresa následující instrukce se uloží do registru R1
- provede se skok na danou adresu
- používá se při skocích do podprogramů, uložená adresa se použije pro skok zpět

BALR R1, R2

- adresa následující instrukce se uloží do registru R1
- provede se skok na adresu v registru R2
- je-li R2 = 0, neprovede se žádný skok

Pro uložení návratové adresy se standardně používá registr R14.

# Nastavení báze registru

- instrukce BALR se často používá pro nastavení báze registru

```
1          BALR    R12,0  
2          USING  *,R12
```

`USING base_adress,base_register`

- instrukce USING říká, jaký registr se má použít jako báze a s jakou adresou

## Volání podprogramů

- pokud chceme volat nějaký podprogram, je potřeba nejprve uložit obsah všech registrů
- místo v paměti (18 fullwordů), kam se uloží, se nazývá SAVEAREA
- **SAVEAREA je alokována volajícím programem**
- její adresa se předá v registru 13 volanému programu
- volaný program pak musí nejprve uložit všechny registry
- před svým ukončením je musí všechny obnovit

## Struktura SAVEAREA

Program 1 SA		
0	nepoužito	
4	předchozí SA	
8	následující SA	R13
12	návratová adresa	R14
16	Entry point adr.	R15
20	obsah R0	R0
24	obsah R1	R1
...	...	...
68	obsah R12	R12

Program 2 SA		
0	nepoužito	
4	předchozí SA	
8	následující SA	R13
12	návratová adresa	R14
16	Entry point adr.	R15
20	obsah R0	R0
24	obsah R1	R1
...	...	...
68	obsah R12	R12

# Struktura SAVEAREA

```

1  START      CSECT
2             STM      R14,R12,12(R13)  SAVE REGISTERS TO SAVEAREA
3             BALR     R12,0             SETUP THE BASE REGISTER
4             USING   *,R12            ESTABLISH ADDRESSING
5  *****
6  *          PREPARE FOR CALLING OTHER SUBPROGRAMS IF NECESSARY
7  *
8  *****
9             LA      R2,SAVEAREA        LOAD ADDRESS OF SAVEAREA TO R2
10            ST      R2,8( ,R13)        LINK THE PREVIOUS SAVEAREA TO THE CURRENT
11            ST      R13,4( ,R2)        LINK THE CURRENT SAVEAREA TO THE PREVIOUS
12            LR      R13,R2            LOAD THE ADDRESS OF THE CURRENT SAVEAREA TO
13  *****
14 *          PROCEED WITH THE MAIN CODE
15 *
16 *****
17            L      R13,4( ,R13)        RESTORE THE ADDRESS OF THE PREVIOUS SAVEAREA
18 SAVEAREA  DS      (R14,R12),T,RC=0  RESTORE REGISTERS AND BRANCH BACK
19            DS      18F

```

## Struktura SAVEAREA

Program 1 SA		
0		
4		
8		
12		
16		
20		
24		
...	...	...
68		

Program 2 SA		
0		
4		
8		
12		
16		
20		
24		
...	...	...
68		



## Struktura SAVEAREA

Program 1 SA		
0		
4		
8		
12	návratová adresa	R14
16	Entry point adr.	R15
20	obsah R0	R0
24	obsah R1	R1
...	...	...
68	obsah R12	R12

Program 2 SA		
0		
4		
8		
12		
16		
20		
24		
...	...	...
68		

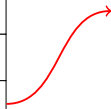
## Struktura SAVEAREA

Program 1 SA		
0		
4		
8		
12	návratová adresa	R14
16	Entry point adr.	R15
20	obsah R0	R0
24	obsah R1	R1
...	...	...
68	obsah R12	R12

Program 2 SA		
0		
4		
8		
12		
16		
20		
24		
...	...	...
68		

## Struktura SAVEAREA

Program 1 SA		
0		
4		
8	následující SA	R13
12	návratová adresa	R14
16	Entry point adr.	R15
20	obsah R0	R0
24	obsah R1	R1
...	...	...
68	obsah R12	R12



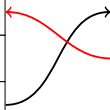
Program 2 SA		
0		
4		
8		
12		
16		
20		
24		
...	...	...
68		

1 **ST** R2,8(,R13)

## Struktura SAVEAREA

Program 1 SA		
0		
4		
8	následující SA	R13
12	návratová adresa	R14
16	Entry point adr.	R15
20	obsah R0	R0
24	obsah R1	R1
...	...	...
68	obsah R12	R12

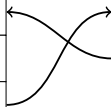
Program 2 SA		
0		
4	předchozí SA	
8		
12		
16		
20		
24		
...	...	...
68		

1 **ST** R13, 4( , R2)

## Struktura SAVEAREA

Program 1 SA		
0		
4		
8	následující SA	R13
12	návratová adresa	R14
16	Entry point adr.	R15
20	obsah R0	R0
24	obsah R1	R1
...	...	...
68	obsah R12	R12

Program 2 SA		
0		
4	předchozí SA	
8		
12		
16		
20		
24		
...	...	...
68		



# Struktura SAVEAREA

1 STM R14, R12, 12(R13)

- obsahy registrů R14, R15, R0, ..., R12 se uloží do SAVEAREA, jejíž adresa je uložena v R13, s offsetem 12

1 ST R2, 8(, R13)

- ulož obsah R2 na adresu danou R13 + 8 bajtů, tj. třetí položka současné SA

1 ST R13, 4(, R2)

- ulož obsah R13 na adresu danou R2 + 4 bajty, tj. druhá položka nové SA

# Konvence při volání podprogramů

## Stav registrů na konci:

- R13 - ukazuje na novou SA pro případ volání podprogramu
- R14 - obsahuje návratovou adresu
  - vzpomeňme na program IEBR14
- R15 - obsahuje adresu počátku současného podprogramu
  - R15 se používá pro skoky do podprogramů
  - při návratu obsahuje návratovou hodnotu

## Konvence při volání podprogramů

- makro RETURN obnoví obsah registrů R14 až R12 ze SAVEAREA, jejíž adresa je 8(R13) a provede skok na adresu v R14
- RC je návratový kód, který se uloží do R15



# Vytvoření nové SAVEAREA

- novou SAVEAREA lze alokovat dynamicky

```
1          GETMAIN      R, LV=72  
2          FREEMAIN     R, LV=72, A=(1)
```

- GETMAIN alokuje paměť a její adresu uloží do R1
- FREEMAIN podobně paměť uvolňuje

# Makro CALL

Makro CALL slouží k volání podprogramů s předáním parametrů:

```
1          CALL          SUMMARY, ( NUM1, NUM2, SUM) , VL
2          . . .
3 NUM1    DS          PL6
4 NUM2    DS          PL6
5 SUM     DS          PL6
```

## Makro CALL

```

1          CALL  SUMMARY, ( NUM1 , NUM2 , SUM ) , VL
2  +      B      *+8          BRANCH AROUND VCON
3  +IHB0001B DC  V(SUMMARY)  ENTRY POINT ADDRESS
4  +      LA     1 , IHB0003  LIST ADDRESS
5  +      B      IHB0003A     BYPASS LIST
6  +IHB0003 DS    0F
7  +      DC     A(NUM1)      PROB.PROG.PARAMETER
8  +      DC     A(NUM2)      PROB.PROG.PARAMETER
9  +      DC     A(SUM+X' 80000000' )
10 +IHB0003A EQU   *
11 +      L      15 , IHB0001B
12 +      BALR   14 , 15

```

## Makro CALL

2	+	B	*+8	BRANCH AROUND VCON
3	+	IHB0001B	DC	V(SUMMARY) ENTRY POINT ADDRESS
4	+	LA	1, IHB0003	LIST ADDRESS
5	+	B	IHB0003A	BYPASS LIST

- řádek 2 provádí skok na řádek 4, aby se přeskočila adresa rutiny, která hned následuje
- řádek 3 obsahuje adcon typu V, generuje adresu rutiny SUMMARY a pojmenuje ji návěštím IHB0001B
- řádek 4 do R1 načte adresu prvního parametru
- řádek 5 přeskočí seznam parametrů, který následuje

## Makro CALL

```
6 +IHB0003  DS    0F
7 +          DC    A(NUM1)      PROB.PROG.PARAMETER
8 +          DC    A(NUM2)      PROB.PROG.PARAMETER
9 +          DC    A(SUM+X'80000000')
```

- vytvoří návěští pro seznam parametrů
- pomocí adconu typu A generuje adresy (ukazatele) proměnných NUM1, NUM2, SUM
- poslední parametr se označí 32-bitem pomocí X'80000000'

## Makro CALL

```
10 +IHB0003A EQU *
11 +          L    15,IHB0001B
12 +          BALR 14,15
```

- řádek 10 generuje návěští pro přeskočení parametrů
- řádek 11 načte do R15 adresu rutiny SUMMARY s pomocí návěští IHB0001B
- řádek 12 provede skok na adresu v R15 a do R14 uloží adresu následující instrukce

## Podprogramy a rutiny

```
1  SUMMARY CSECT
2      ...
3      ZAP  MYSUM,=P' 0'
4  LOOP  DS  0H
5          L   R2,0( ,R1)
6          LTR R2,R2
7          BM  DONE
8          AP  MYSUM,0(6 ,R2)
9          AHI R1,4
10         B   LOOP
11  DONE  DS  0H
12         ZAP  0(6 ,R2) ,MYSUM
13         ...
14  MYSUM DS  PL6
```

# Podprogramy a rutiny

## 1 SUMMARY CSECT

- CSECT označuje spustitelnou kontrolní sekci, tedy sekci se spustitelným kódem
- každý program nebo podprogram musí být součástí spustitelné sekce



# Podprogramy a rutiny

```
4  LOOP    DS    0H  
5          L     R2, 0( , R1)  
6          LTR   R2, R2  
7          BM   DONE
```

- na řádce 5 načítáme do R2 adresu prvního parametru z adresy v R1
- na řádce 6 načítáme obsah R2 do R2 za účelem testování hodnoty v R2
- je-li záporná (řádek 7), je nastaven nejvyšší bit a my jsme tedy načítali poslední argument
  - v tom případě jde o proměnnou SUM a dál tedy nic neděláme

## Podprogramy a rutiny

```
8           AP   MYSUM,0(6,R2)
9           AHI  R1,4
10          B    LOOP
11  DONE    DS  0H
12          ZAP  0(6,R2),MYSUM
13          ...
14  MYSUM   DS  PL6
```

- na řádku 8 přičítáme hodnotu z adresy v R2 do MYSUM
  - R2 postupně obsahuje adresy jednotlivých parametrů
- na řádku 9 inkrementujeme adresu v R1 o 4, tj. posouváme se na ukazatel na další parametr
- nakonec máme v R2 adresu posledního parametru, do kterého máme uložit výsledek
- to uděláme na řádku 12 kopírováním 6 bajtů z MYSUM na adresu z R2

## Modifikace instrukcí

```
1  MAIN      SETUP
2           CALL PRINT , (MSGL) , VL
3           ENDIT
4  MSGL      DC AL2 (MSGEND-MSG)
5  MSG       DC C' IT_IS_TIME_FOR_A_BREAK'
6  MSGEND    EQU *
```

- chceme napsat rutinu PRINT, která vypíše do souboru zadanou zprávu
- zpráva nejprve obsahuje svou délku (max 256 znaků) a následně samotný řetězec

## Modifikace instrukcí

```

1 PRINT      SETUP
2           L      R3,0( ,R1)      LOAD ADDRESS OF MESSAGE
3           LH     R4,0( ,R3)      LOAD MESSAGE LENGTH
4           BCTR  R4,0             SUBTRACT ONE FROM R4 AND CONTINUE
5           STH   R4,LENGTH
6           MVC   MOVE+1(1),LENGTH+1
7 MOVE      MVC   PRINTREC+1,2(R3)
8           OPEN  (OUTPUT,(OUTPUT))
9           PUT   OUTPUT,PRINTREC
10          CLOSE (OUTPUT)
11          ENDIT
12 OUTPUT   DCB   BLKSIZE=133,
13          DDNAME=SYSPRINT,
14          DSORG=PS,
15          LRECL=133,
16          MACRF=PM,
17          RECFM=FBA
18 PRINTREC DC   CL133' _'
19 LENGTH   DS   H
20 END

```

# Modifikace instrukcí

1	PRINT	SETUP		
2		L	R3,0(,R1)	LOAD ADDRESS OF MESSAGE
3		LH	R4,0(,R3)	LOAD MESSAGE <b>LENGTH</b>
4		BCTR	R4,0	SUBTRACT ONE FROM R4 <b>AND</b> CONTINUE
5		STH	R4, <b>LENGTH</b>	

- na řádku 2 načteme do R3 adresu zprávy -tj. děláme dereferenci ukazatele v R1
- na řádku 3 načteme do R4 délku zprávy ve formě halfwordu
- na řádku 4 délku zmenšíme o 1 pro účely funkce MVC
- na řádku 5 si obsah R4 uložíme do pomocné proměnné LENGTH

## Modifikace instrukcí

```

6          MVC    MOVE+1(1),LENGTH+1
7  MOVE    MVC    PRINTREC+1,2(R3)
8          OPEN  (OUTPUT,(OUTPUT))
9          PUT   OUTPUT,PRINTREC
10         CLOSE (OUTPUT)
11        ENDIT

...

19 PRINTREC DC    CL133' _ '
20 LENGTH  DS    H
21 END

```

- na řádce 7, chceme použít instrukci MVC k přesunu vstupní zprávy do PRINTREC
- za tím účelem ale musíme do instrukce MVC zakódovat délku zprávy
- to uděláme na řádce 6, kde měníma druhý bajt instrukce MVC na řádce 7, který obsahuje délku kopírovaného řetězce
- řetězec kopírujeme z adresy dané dané hodnotou v R2 + 2 bajty pro přeskočení délky zprávy

# Modifikace instrukcí

- rutinu lze použít jako samostatný program
- v JCL pak použijeme předání parametru

```
1 //      PARM.G='message_text'
```