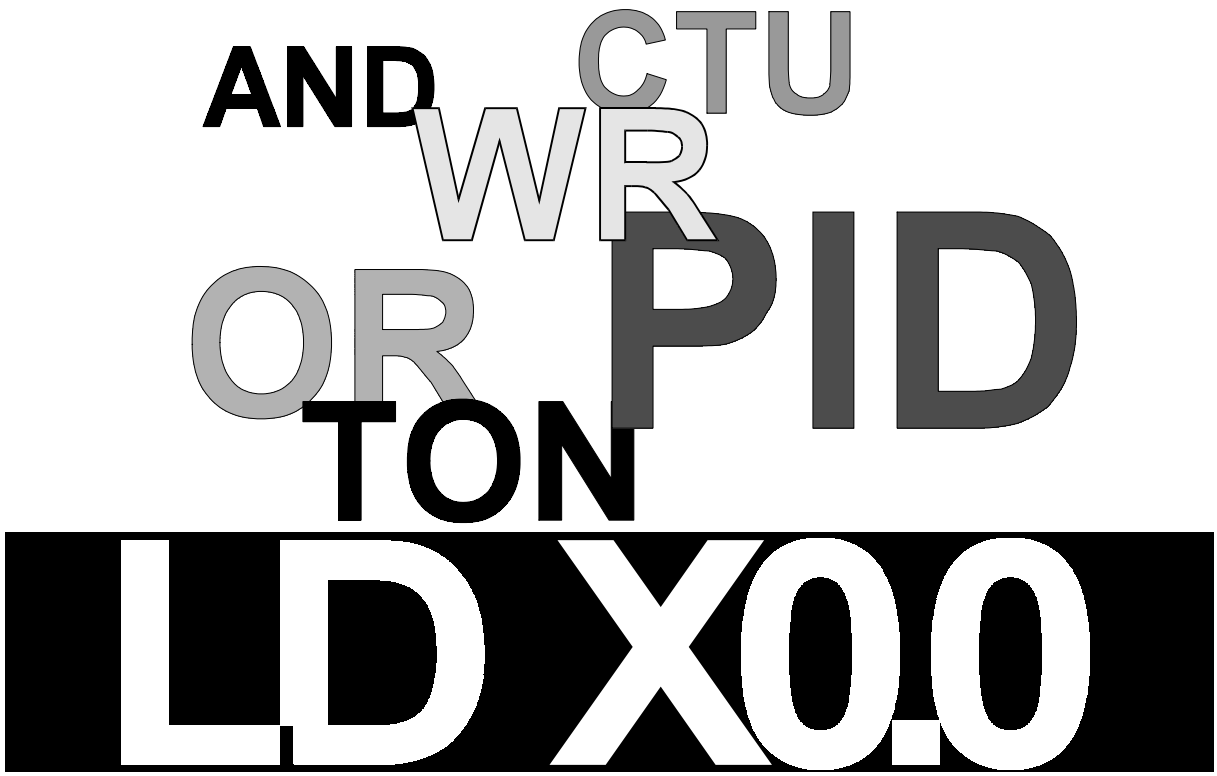


SOUBOR INSTRUKCÍ PLC TECOMAT
MODEL 16 BITŮ

SOUBOR INSTRUKCÍ PLC TECOMAT MODEL 16 BITŮ

8. vydání - září 2003

OBSAH

ÚVOD 5

1. INSTRUKCE PRO ČTENÍ A ZÁPIS DAT 8
LD, LDL, LDC 8
WR, WRC 11
WRA 14
PUT 16

2. LOGICKÉ INSTRUKCE 18
AND, ANL, ANC 18
OR, ORL, ORC 21
XOR, XOL, XOC 24
NEG, NGL 27
SET, RES 28
LET, BET 30
FLG 32
STK 34
ROL, ROR 35
SWP, SWL 37

3. ČÍTAČE, POSUVNÉ REGISTRY, ČASOVAČE, KROKOVÝ ŘADIČ 38
CTU, CTD, CNT 38
SFL, SFR 44
TON, TOF 46
RTO 50
IMP 53
STE 55

4. ARITMETICKÉ INSTRUKCE 57
ADD, ADX, ADL 57
SUB, SUX, SUL 59
MUL, MUD 61
DIV, DID 62
INR, DCR 64
EQ, LT, GT 66
CMP, CML 68
BIN, BIL, BCD, BCL 69

5. OPERACE SE ZÁSOBNÍKY 71
POP 71
CHG, CHGS, NXT, PRV 72
LAC, WAC 73

6. INSTRUKCE SKOKŮ A VOLÁNÍ 74
JMP, JMD, JMC, JMI 74
JZ, JNZ, JC, JNC, JS, JNS 75
CAL, CAD, CAC, CAI 77
RET, RED, REC 78
L 79

7. ORGANIZAČNÍ INSTRUKCE 80
P, E, ED, EC, EOC 80
NOP 82
BP 83
SEQ 84

8. TABULKOVÉ INSTRUKCE 85
LTB 85
WTB 88
FTB 91
FTM 93
FTS 95

9. BLOKOVÉ OPERACE 97
SRC, MOV 97
MTN, MNT 99
FIL 101

10. OPERACE SE STRUKTUROVANÝMI TABULKAMI 102
LDS 102
WRS 103
FIS, FIT 104
FNS, FNT 106

11. ARITMETICKÉ INSTRUKCE V PLOVOUCÍ ŘÁDOVÉ ČÁRCE 108
ADF, SUF 108
MUF, DIF 109
CMF 111
CEI, FLO, ABS 112
LOG, LN, EXP, POW, SQR, HYP 113
SIN, ASN, COS, ACS, TAN, ATN 115
UWF, IWF, ULF, ILF 117
UFW, IFW, UFL, IFL 118

12. INSTRUKCE REGULÁTORU PID 119
CNV 119
PID 129

13. OPERACE SE ZNAKY ASCII 141
BAS 141
ASB 142
STF 143
FST 145

14. SYSTÉMOVÉ INSTRUKCE 146
HPE, HPD 146
RDT, WRT 149
RDB, WDB, IDB 151
REI 153

PŘEHLED INSTRUKCÍ 154
Přehled instrukcí s přípustnými operandy 154
Abecední seznam instrukcí 158

ÚVOD

Zásady popisu instrukcí

V následujících kapitolách jsou popsány jednotlivé instrukce PLC. Velká část instrukcí připouští operandy různých typů z různých prostorů, nebo mohou být i bezoperandové. V zájmu přehlednosti popisu nebudeme podrobně popisovat všechny možné kombinace, ale pouze typické případy. Například přístup k operandům X, Y, S, D, R je vždy analogický. Popíšeme-li tedy chování instrukce LD %R12.3, budeme předpokládat, že instrukce LD %X1.7 se chová obdobně.

Přehledy instrukcí s přípustnými operandy a operačními časy pro jednotlivé typy centrálních jednotek jsou uvedeny v příloze.

V titulní hlavičce každé instrukce je uvedena její symbolická zkratka a název. Dále je uvedena tabulka znázorňující stav zásobníku a zápisníku před a po instrukci. Následují přípustné operandy (X, Y, S, D, R, #, T) a jejich typ (bit - b, byte - B, word - W, long - L, float - F) pro jednotlivé řady centrálních jednotek, popis funkce, ovlivňované příznaky a typické příklady chování.

Absolutní adresy jsou psány s uvozujícím znakem %, který sice není při programování centrálních jednotek se zásobníkem šířky 16 bitů povinný, ale doporučuje se používat s ohledem na přenositelnost uživatelských programů do centrálních jednotek se zásobníkem šířky 32 bitů.

Řady centrálních jednotek a model zásobníku

Centrální jednotky PLC TECOMAT a regulátorů TECOREG jsou rozděleny podle svých vlastností do následujících řad:

řada B - NS950 CPM-1B, CPM-2B
řada C - TC700 CP-7001, CP-7002
řada D - TR050, TR200, TR300, TC400, TC500, TC600, NS950 CPM-1D
řada E - NS950 CPM-1E
řada M - NS950 CPM-1M
řada S - NS950 CPM-1S, CPM-2S

PLC TECOMAT mají dva modely zásobníku, které se od sebe liší šířkou jedné vrstvy. Řady B, D, E, M a S mají jednotlivé vrstvy zásobníku široké 16 bitů, zatímco řada C má vrstvy zásobníku široké 32 bitů. Z toho plynou určité rozdíly mezi chováním jednotlivých modelů.

Tato příručka je věnována výhradně centrálním jednotkám se zásobníkem šířky 16 bitů. Instrukční soubor pro centrální jednotky se zásobníkem šířky 32 bitů je popsán v příručce Soubor instrukcí PLC TECOMAT TXV 004 01.01. Zde jsou popsány i rozdíly v chování obou modelů a přenos uživatelského programu mezi nimi.

Zásady zobrazení příkladů

V příkladech některých instrukcí jsou paměťové prostory a zásobník PLC zobrazeny graficky podle zásad odpovídajících použitému formátu. Malá písmena označují libovolnou nezměněnou hodnotu. Podrobnosti o formátech dat v paměťových prostorech a zásobníku jsou uvedeny v Příručce programátora PLC TECOMAT TXV 001 09.01.

V popisech instrukcí je vždy jako aktivní použit zásobník A, na jeho místě však může být kterýkoli další.

Stručný přehled souboru instrukcí

1. **Instrukce pro čtení a zápis dat**
   Čtení a zápis dat ve všech formátech, podmíněný zápis a zápis s alternací nejvyššího bitu.

2. **Logické instrukce**
   Logické instrukce AND, OR, XOR s přímými i negovanými operandy, negace, detekce náběžné hrany nebo obou hran, podmíněné nastavení nebo nulování proměnné, rotace vlevo i vpravo, logické sklopení zásobníku, záměna bytů vrcholu zásobníku, logické funkce vrcholu zásobníku.

3. **Čítače, posuvné registry, časovače, krokový řadič**
   Dopředný, zpětný a obousměrný čítač, posuvný registr vlevo i vpravo, časovač se zpožděným přítahem nebo odpadem, integrující časovač, impulz definované délky, krokový řadič.

4. **Aritmetické instrukce**
   Aritmetické instrukce v pevné řádové čárce (8, 16, 32 bitů), sčítání, odčítání, násobení, dělení, inkrementace, dekrementace, porovnání, převod z binární soustavy na BCD kód a opačně.

5. **Operace se zásobníky**
   Posun zásobníku, výměna zásobníků, přesun hodnot mezi zásobníky

6. **Instrukce skoků a volání**
   Přímé skoky, nepřímé skoky, podmíněné skoky, přímá volání podprogramu, nepřímá volání, podmíněná volání, návrat z podprogramu, podmíněný návrat z podprogramu, návěští.

7. **Organizační instrukce**
   Začátek a konec procesu, podmíněný konec procesu, konec cyklu, prázdná instrukce, ladicí bod, podmíněné přerušení procesu.

8. **Tabulkové instrukce**
   Čtení a zápis do tabulky nebo pole v zápisníku, hledání hodnoty.

9. **Blokové operace**
   Přesun bloku dat, přesun tabulky do zápisníku a opačně, plnění bloku konstantou.

10. **Operace se strukturovanými tabulkami**
    Čtení a zápis položky strukturované tabulky, hledání položky, plnění položky konstantou.

11. **Aritmetické instrukce v plovoucí řádové čárce**
    Sčítání, odčítání, násobení, dělení, porovnání, zaokrouhlování, absolutní hodnota, logaritmické, exponenciální a goniometrické funkce, převod mezi formáty s plovoucí a pevnou řádovou čárkou.

12. **Instrukce regulátoru PID**
    Převod měřených analogových hodnot na normalizované hodnoty s diagnostikou okrajových stavů, PID regulátor.

13. **Instrukce obsluhy terminálu a operace se znaky ASCII**
    Obsluha znakového displeje, převod čísel na ASCII řetězce a opačně, operace s ASCII řetězci.

14. Systémové instrukce

Ovládání odezvy zápisníkových komunikací, přístup k obvodu reálného času, zápis do a čtení z přídavné paměti DataBox, ovládání periferního systému.

1. INSTRUKCE PRO ČTENÍ A ZÁPIS DAT

LD, LDL	**Čtení dat**
LDC	**Čtení negovaných dat**

Instrukce	**Vstupní parametry**									**Výsledek**								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
LD [b B W]									*a*	A6	A5	A4	A3	A2	A1	A0	*a*	*a*
LD [L F]									*a*	A5	A4	A3	A2	A1	A0	*a*		*a*
LDL									*a*	A5	A4	A3	A2	A1	A0	*a*		*a*
LDC [b B W]									*a*	A6	A5	A4	A3	A2	A1	A0	$\overline{a}$	*a*
LDC [L]									*a*	A5	A4	A3	A2	A1	A0	$\overline{a}$		*a*

Operandy

		bit	byte	word	long	float
LD	X Y S D R	B D S M E	B D S M E	B D S M E	B D	B D
LD	U		B D S M	B D S M		
LD	#			B D S M E		
LDL	#				B D	B D
LDC	X Y S D R	B D S M E	B D S M E	B D S M E	B D	
LDC	#			B D S M E		

Funkce

LD - čtení dat na vrchol zásobníku
LDL - čtení 32-bitové konstanty na vrchol zásobníku
LDC - čtení negovaných dat na vrchol zásobníku

Popis

Instrukce **LD** a **LDL** přečte data z adresovaného místa a beze změny ji uloží na vrchol zásobníku, instrukce **LDC** přečtená data neguje a pak ji uloží na vrchol zásobníku. Obsah zdrojového místa je nezměněn.

Instrukce s operandem typu **bit** posunou zásobník o jednu úroveň vpřed a nastaví shodně všech 16 bitů vrcholu zásobníku A0.

Instrukce s operandem typu **byte** posunou zásobník o jednu úroveň vpřed a zapíší do dolního bytu vrcholu zásobníku A0. Horní byte vrcholu je vynulován.

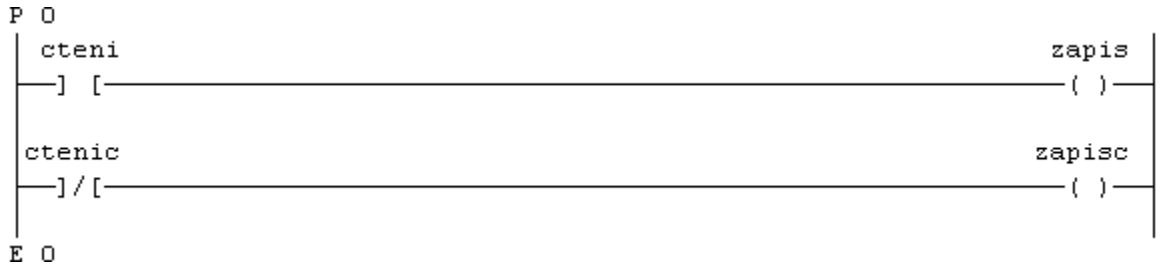
Instrukce s operandem typu **word** posunou zásobník o jednu úroveň vpřed a zapíší na celý vrchol zásobníku A0.

Instrukce s operandem typu **long** a **float** posunou zásobník o dvě úrovně vpřed a zapíší na vrchol zásobníku A01.

Příklad

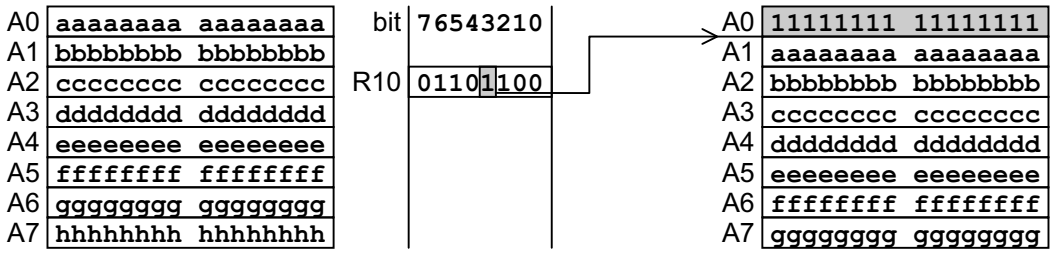
```
#def cteni   %X0.0
#def ctenic  %X0.1
#def zapis   %Y0.1
#def zapisc  %Y0.7
;
```

```
P 0
     LD    cteni
     WR    zapis
     LDC   ctenic
     WR    zapisc
E 0
```

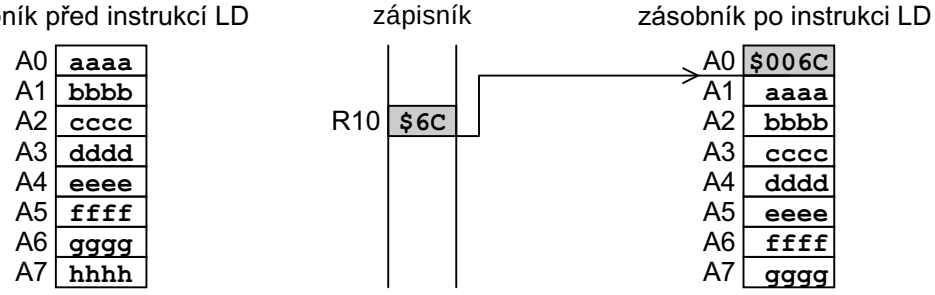


Schéma

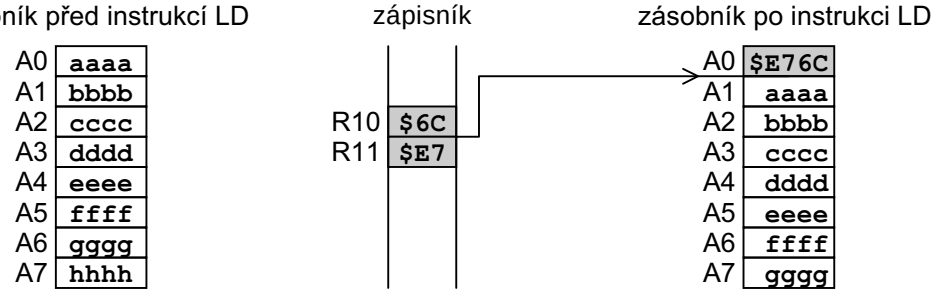
LD %R10.3



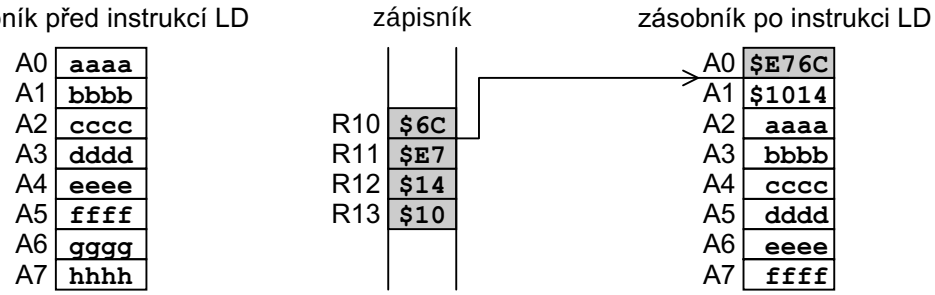
LD %R10



LD %RW10

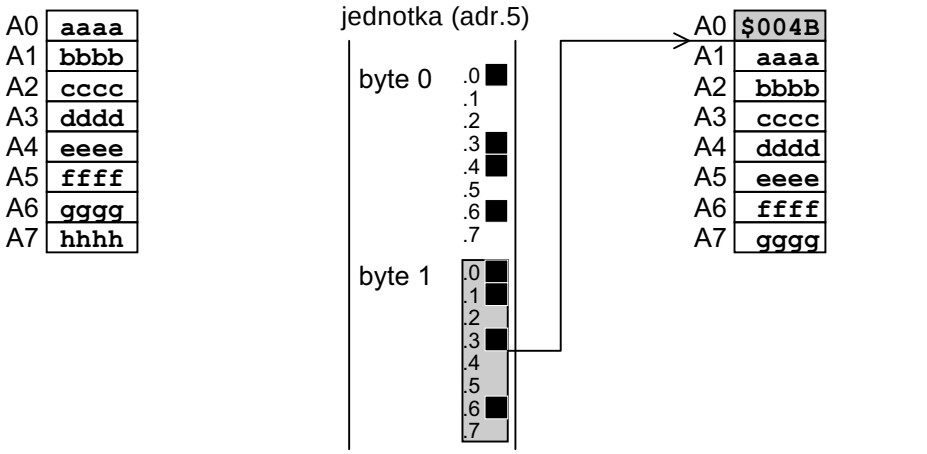


LD %RL10



LD %U$9501

(příklad pro NS950)



WR	**Zápis dat**
WRC	**Zápis negovaných dat**

Instrukce	**Vstupní parametry**									**Výsledek**								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
WR [b B W]								*a*									*a*	*a*
WR [L F]							*a*									*a*		*a*
WRC [b B W]								*a*									*a*	$\overline{a}$
WRC [L]							*a*									*a*		$\overline{a}$

Operandy

		bit	byte	word	long	float
WR	X Y S R	B D S M E	B D S M E	B D S M E	B D	B D
WR	U		B D S M	B D S M		
WRC	X Y S R	B D S M E	B D S M E	B D S M E	B D	

Funkce

WR - zápis dat z vrcholu zásobníku
WRC - zápis negovaných dat z vrcholu zásobníku

Popis

Instrukce **WR** přečte hodnotu vrcholu zásobníku a beze změny ji uloží do adresovaného místa, instrukce **WRC** přečtenou hodnotu neguje a pak ji uloží do adresovaného místa. Obsah celého zásobníku zůstává nezměněn.

Instrukce s operandem typu **bit** provede logický součet (OR) všech bitů vrcholu zásobníku A0 a jeho hodnotu uloží do adresovaného bitu, bitová instrukce **WRC** ukládá negovanou hodnotu tohoto součtu (NOR). Je-li tedy A0 = 0, pak instrukce **WR** zapisuje hodnotu log.0 a **WRC** hodnotu log.1, v ostatních případech (A0 $\neq$ 0) zapisuje instrukce **WR** hodnotu log.1 a instrukce **WRC** hodnotu log.0.

Upozornění: Bitová instrukce **WRC** zapisuje negovanou hodnotu logického součtu všech bitů A0, tedy funkci NOR. Její výsledek není totožný s výsledkem, který bychom obdrželi logickým sečtením negovaných bitů A0.

Instrukce s operandem typu **byte** pracují pouze s dolním bytem vrcholu zásobníku A0. Horní byte vrcholu není bytovými instrukcemi zpracován.

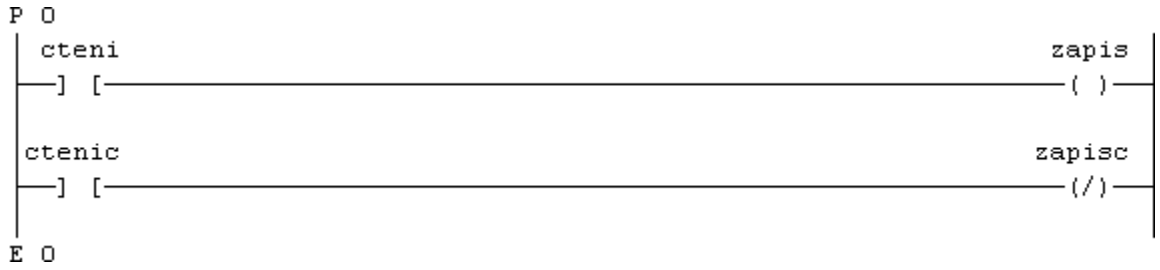
Instrukce s operandem typu **word** pracují s celým vrcholem zásobníku A0.

Instrukce s operandem typu **long** a **float** pracují s vrcholem zásobníku tvořeným dvojvrstvou A01.

Příklad

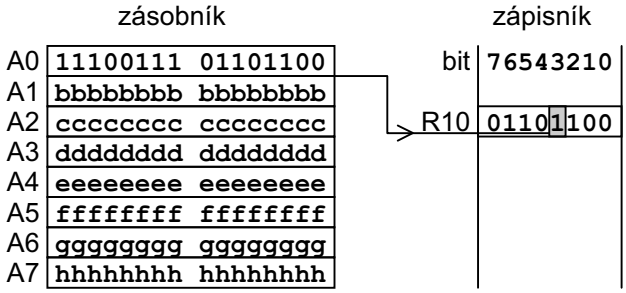
```
#def cteni   %X0.0
#def ctenic  %X0.1
#def zapis   %Y0.1
#def zapisc  %Y0.7
;
```

```
P 0
     LD   cteni
     WR   zapis
     LD   ctenic
     WRC  zapisc
E 0
```

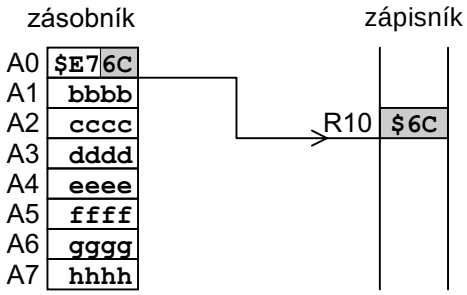


Schéma

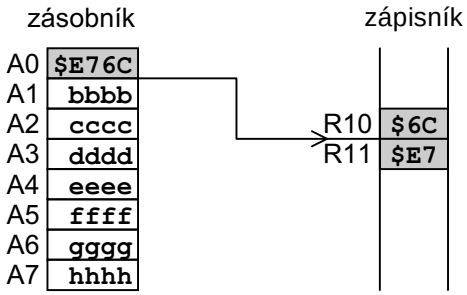
WR %R10.3



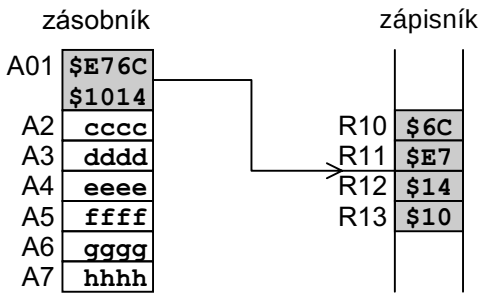
WR %R10



WR %RW10

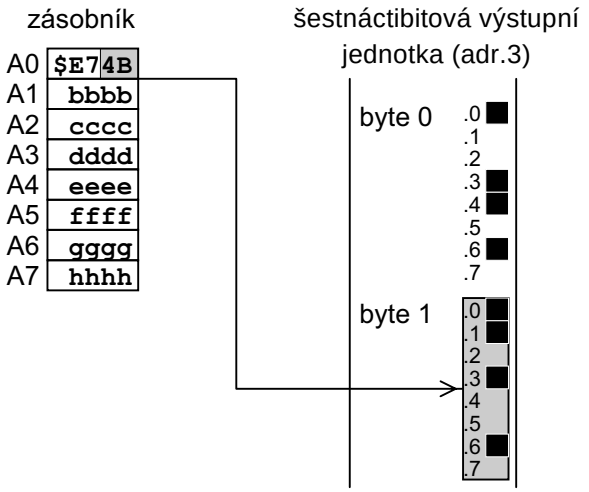


WR %RL10



WR %U$9381

(příklad pro NS950)



WRA Zápis dat s alternací

Instrukce	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								operand
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	
WRA [B W]								*a*	*b*								*a*	$(\overline{b_{max}})a$
WRA [L]							*a*		*b*							*a*		$(\overline{b_{max}})a$

Operandy

		byte	word	long
WRA	X Y S R	B D	B D	B D

Funkce

WRA - zápis dat z vrcholu zásobníku s alternací nejvyššího bitu

Popis

Instrukce **WRA** čte hodnotu z vrcholu zásobníku, vymaskuje nejvyšší bit a uloží ji do adresovaného místa. Pak provede negaci stávajícího nejvyššího bitu adresovaného místa (alternaci). Obsah celého zásobníku zůstává nezměněn. Tuto instrukci lze s výhodou použít při ovládání inteligentních periferií, které vyžadují alternaci nejvyššího bitu při předávání parametrů (např. obsluha sériového kanálu v režimu **uni**, nebo jednotek GT-41, SC-11, CD-01, CD-02 v PLC TECOMAT NS950).

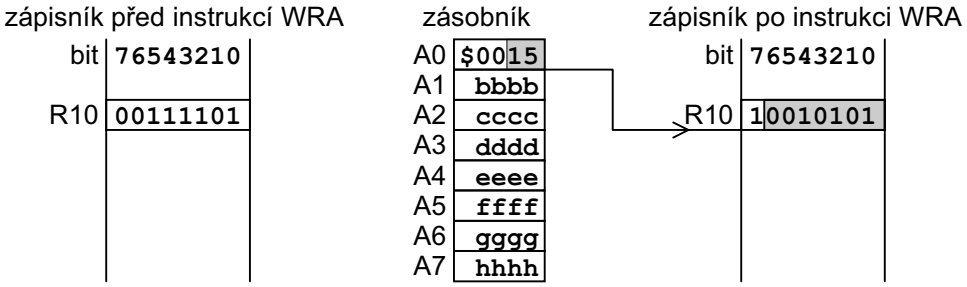
Instrukce s operandem typu **byte** pracuje pouze s dolním bytem vrcholu zásobníku A0. Horní byte vrcholu není instrukcí zpracován.

Instrukce s operandem typu **word** pracuje s celým vrcholem zásobníku A0.

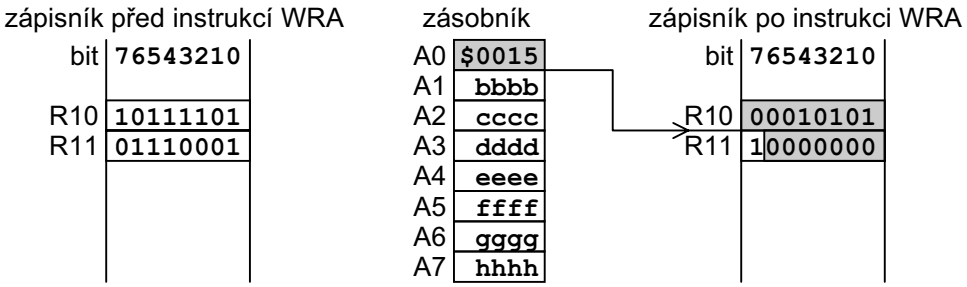
Instrukce s operandem typu **long** pracuje s vrcholem zásobníku představovaným dvojvrstvou A01.

Schéma

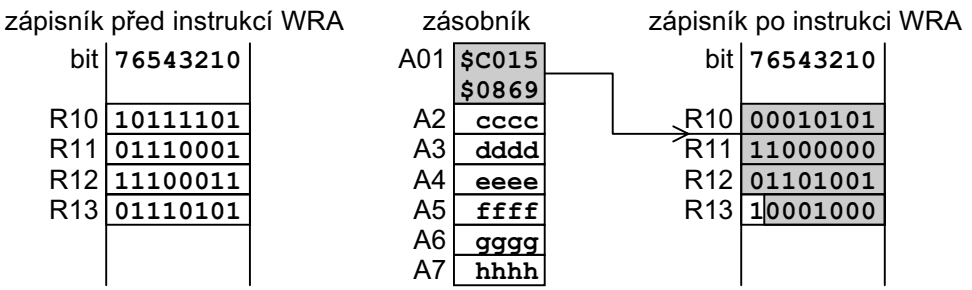
WRA %R10



WRA %RW10



WRA %RL10


zápisník před instrukcí WRA
zásobník
zápisník po instrukci WRA
bit 76543210
R10 10111101
R11 01110001
R12 11100011
R13 01110101
A01 $C015
$0869
A2 cccc
A3 dddd
A4 eeee
A5 ffff
A6 gggg
A7 hhhh
bit 76543210
R10 00010101
R11 11000000
R12 01101001
R13 10001000

PUT Podmíněný zápis dat

Instrukce	Vstupní parametry									Výsledek								
	zásobník								S1.0	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	rand
PUT [b B W]								*a*	1								*a*	*a*
								a	0								*a*	
PUT [L F]							*a*		1							*a*		*a*
							a		0							*a*		

Operandy

		bit	byte	word	long	float
PUT	X Y S R	B D S M E	B D S M E	B D S M E	B D	B D

Funkce

PUT - zápis dat z vrcholu zásobníku podmíněný hodnotou log.1 bitu S1.0

Popis

Instrukce **PUT** je obdobou instrukce **WR**, která se však provede pouze tehdy, je-li S1.0 = log.1. Při S1.0 = log.0 neprovede žádnou činnost. Instrukce **PUT** otestuje bit S1.0 a pokud je roven log.1, čte hodnotu vrcholu zásobníku A0 a beze změny ji uloží do adresovaného místa. Obsah celého zásobníku i příznakových registrů zůstává nezměněn.

Instrukce s operandem typu **bit** v případě S1.0 = log.1 provede logický součet (OR) všech bitů vrcholu zásobníku A0 a jeho hodnotu uloží do adresovaného bitu. Je-li tedy A0 = 0, pak instrukce zapisuje hodnotu log.0, v ostatních případech (A0 $\neq$ 0) zapisuje instrukce hodnotu log.1.

Instrukce s operandem typu **byte** pracují pouze s dolní částí vrcholu zásobníku A0L. Horní část vrcholu A0H není bytovými instrukcemi zpracovávána.

Instrukce s operandem typu **word** pracují s celým vrcholem zásobníku A0.

Instrukce s operandem typu **long** a **float** pracují s vrcholem zásobníku tvořeným dvojvrstvou A01.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	S

S1.0 (S) - vstupní podmínka instrukce
0 - instrukce se neprovede
1 - instrukce se provede v plném rozsahu

Příklad

```
#def cteni     %X0.0
#def podminka  %X0.2
#def zapis     %Y0.1
;
P 0
     LD    podminka
     WR    %S1.0
     LD    cteni
     PUT   zapis
E 0
```

Schéma

Pokud má S1.0 hodnotu log.1, je schéma instrukce **PUT** totožné s instrukcí **WR**. Pokud má S1.0 hodnotu log.0, instrukce se chovají jako prázdné.

2. LOGICKÉ INSTRUKCE

AND, ANL	**Funkce AND**
ANC	**Funkce NAND**

Instrukce	Vstupní parametry									Výsledek								
	zásobník								operand	zásobník								operand
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
AND								a	b								$a \cdot b$	b
AND bez op.							a	b		b	A7	A6	A5	A4	A3	A2	$a \cdot b$	
ANL							a		b							$a \cdot b$		b
ANL bez op.					a		b			b		A7	A6	A5	A4	$a \cdot b$		
ANC								a	b								$a \cdot \bar{b}$	b

Operandy

		bit	byte	word	long
AND	X Y S R D	B D S M E	B D S M	B D	
AND	#			B D S M	
AND	bez operandu			B D S M	
ANL	#				B D
ANL	bez operandu				B D
ANC	X Y S R D	B D S M E	B D S M	B D	

Funkce

AND - logický součin vrcholu zásobníku s operandem
ANL - logický součin vrcholu zásobníku s operandem (long)
ANC - logický součin vrcholu zásobníku s negovaným operandem

Popis

Funkce logického součinu (AND) nabývá hodnoty log.1, pokud jsou oba její operandy log.1, jinak má hodnotu log.0. Ve výrokovém počtu jí odpovídá spojení vyjadřující současnost („a“, „i“, „současně“). V reléových schématech jí odpovídá sériové řazení kontaktů. Funkce je patrná z pravdivostní tabulky:

Vstupní parametry		Výsledek	
a	b	$a \cdot b$	$a \cdot \bar{b}$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	0

Operandové instrukce **AND**, **ANL** sejmou obsah adresovaného místa a provedou jeho logický součin s vrcholem zásobníku. Ten je přepsán výsledkem operace. Instrukce **ANC** provádí logický součin negace sejmutého obsahu adresovaného místa s vrcholem zásobníku. Obsah zdrojového místa je nezměněn.

Instrukce s operandem typu **bit** zpracují celý vrchol zásobníku A0 tak, že s každým jeho bitem provedou určenou operaci. Výsledek těchto 16 operací instrukce uloží zpět na vrchol zásobníku A0.

Instrukce s operandem typu **byte** zpracují dolní byte vrcholu zásobníku A0 jako 8 bitových operací mezi odpovídajícími bity zásobníku a operandu. Výsledek uloží do dolního

bytu vrcholu zásobníku A0L. Horní byte vrcholu A0H je vynulován (provedena operace AND 0).

Instrukce s operandem typu **word** zpracují vrchol zásobníku A0 jako 16 bitových operací mezi odpovídajícími bity zásobníku a operandu. Výsledek uloží na vrchol zásobníku A0.

Instrukce s operandem typu **long** zpracují vrchol zásobníku A01 jako 32 bitových operací mezi odpovídajícími bity zásobníku a operandu. Výsledek uloží na vrchol zásobníku A01.

Instrukce **AND**, **ANC** bez operandu provedou 16 bitových operací mezi odpovídajícími bity vrstev A0 a A1 zásobníku. Pak posunou zásobník o jednu úroveň zpět a výsledek operace zapíš9 na nový vrchol zásobníku A0.

Instrukce **ANL** bez operandu provede 32 bitových operací mezi odpovídajícími bity dvojvrstev A01 a A23 zásobníku. Pak posunou zásobník o dvě úrovně zpět a výsledek operace zapíší na nový vrchol zásobníku A01.

Příklady

Logický součin $y = a \cdot \overline{b} \cdot c$

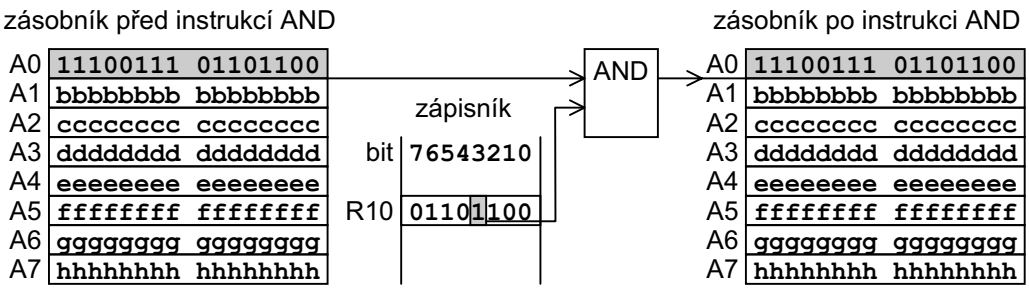
```
#def va      %X0.0
#def vb      %X0.3
#def vc      %X1.4
#def vystup  %Y0.4
;
P 0
     LD    va
     ANC   vb
     AND   vc
     WR    vystup
E 0
```

Logický součin $y = a \cdot b$

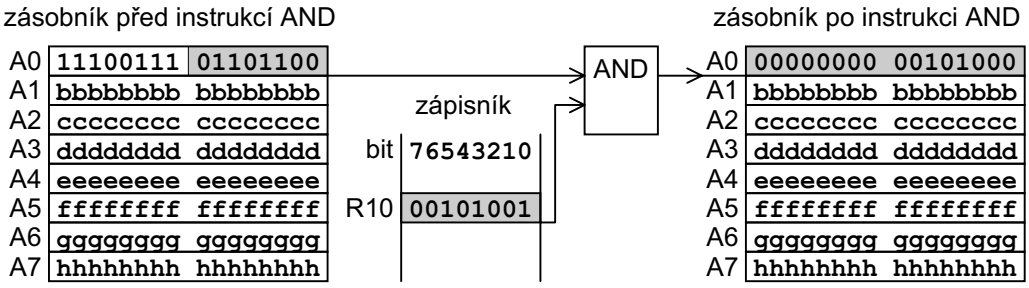
```
#def va      %X0.1
#def vb      %X0.5
#def vystup  %Y0.2
;
P 0
     LD    va
     LD    vb
     AND
     WR    vystup
E 0
```

Schéma

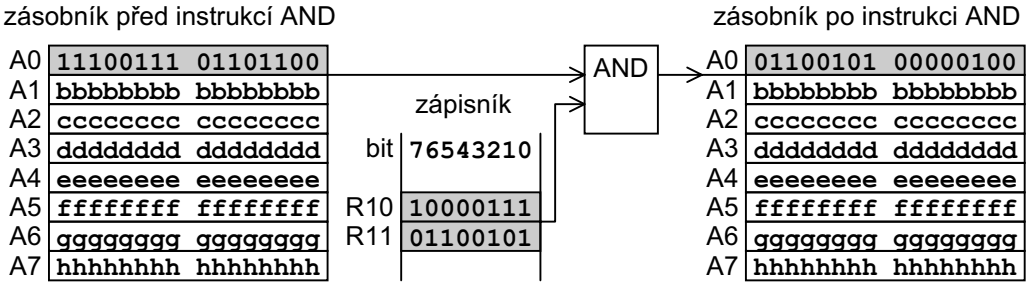
```
LD   $E76C
AND  %R10.3
```



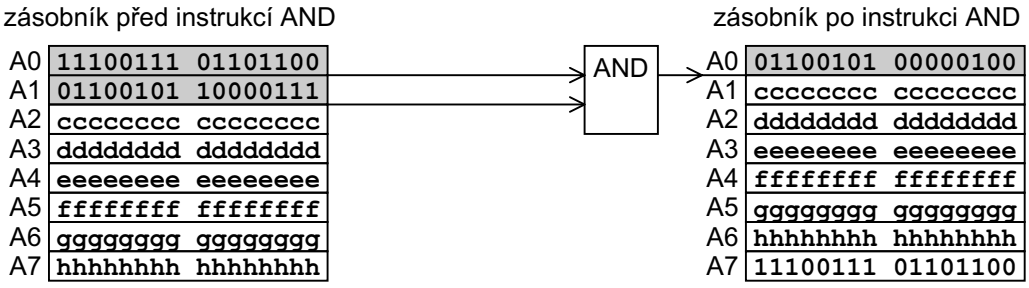

```
LD   $E76C
AND  %R10
```



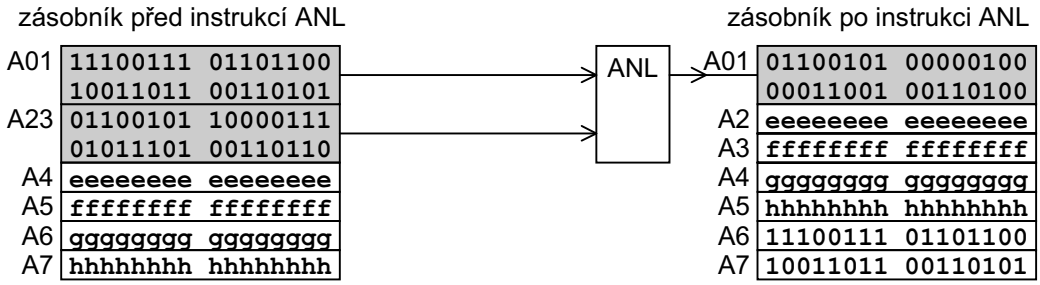

```
LD   $E76C
AND  %RW10
```




```
LD   $6587
LD   $E76C
AND
```




```
LDL  $5D366587
LDL  $9B35E76C
ANL
```



OR, ORL	**Funkce OR**
ORC	**Funkce NOR**

Instrukce	**Vstupní parametry**									**Výsledek**								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
OR								a	b								$a+b$	b
OR bez op.							a	b		b	A7	A6	A5	A4	A3	A2	$a+b$	
ORL							a		b							$a+b$		b
ORL bez op.					a		b			b		A7	A6	A5	A4	$a+b$		
ORC								a	b								$a+\bar{b}$	b

Operandy

		bit	byte	word	long
OR	X Y S R D	B D S M E	B D S M	B D	
OR	#			B D S M	
OR	bez operandu			B D S M	
ORL	#				B D
ORL	bez operandu				B D
ORC	X Y S R D	B D S M E	B D S M	B D	

Funkce

OR - logický součet vrcholu zásobníku s operandem
ORL - logický součet vrcholu zásobníku s operandem (long)
ORC - logický součet vrcholu zásobníku s negovaným operandem

Popis

Funkce logického součtu (OR) nabývá hodnoty log.1, pokud je aspoň jeden z jejích operandů log.1, jinak má hodnotu log.0. Ve výrokovém počtu jí odpovídá spojka „nebo“. V reléových schématech jí odpovídá paralelní řazení kontaktů. Funkce je patrná z pravdivostní tabulky:

Vstupní parametry		**Výsledek**	
a	b	$a+b$	$a+\bar{b}$
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	1

Operandové instrukce **OR**, **ORL** sejmou obsah adresovaného místa a provedou jeho logický součet s vrcholem zásobníku. Ten je přepsán výsledkem operace. Instrukce **ORC** provádí logický součet negace sejmutého obsahu adresovaného místa s vrcholem zásobníku. Obsah zdrojového místa je nezměněn.

Instrukce s operandem typu **bit** zpracují celý vrchol zásobníku A0 tak, že s každým jeho bitem provedou určenou operaci. Výsledek těchto 16 operací instrukce uloží zpět na vrchol zásobníku A0.

Instrukce s operandem typu **byte** zpracují dolní byte vrcholu zásobníku A0 jako 8 bitových operací mezi odpovídajícími bity zásobníku a operandu. Výsledek uloží do dolního bytu vrcholu zásobníku A0L. Horní byte vrcholu A0H je vynulován (provedena operace AND 0).

Instrukce s operandem typu **word** zpracují vrchol zásobníku A0 jako 16 bitových operací mezi odpovídajícími bity zásobníku a operandu. Výsledek uloží na vrchol zásobníku A0.

Instrukce s operandem typu **long** zpracují vrchol zásobníku A01 jako 32 bitových operací mezi odpovídajícími bity zásobníku a operandu. Výsledek uloží na vrchol zásobníku A01.

Instrukce **OR** bez operandu provede 16 bitových operací mezi odpovídajícími bity vrstev A0 a A1 zásobníku. Pak posune zásobník o jednu úroveň zpět a výsledek operace zapíše na nový vrchol zásobníku A0.

Instrukce **ORL** bez operandu provede 32 bitových operací mezi odpovídajícími bity dvojvrstev A01 a A23 zásobníku. Pak posune zásobník o dvě úrovně zpět a výsledek operace zapíše na nový vrchol zásobníku A01.

Příklady

Logický součet $y = a + b + \overline{c}$

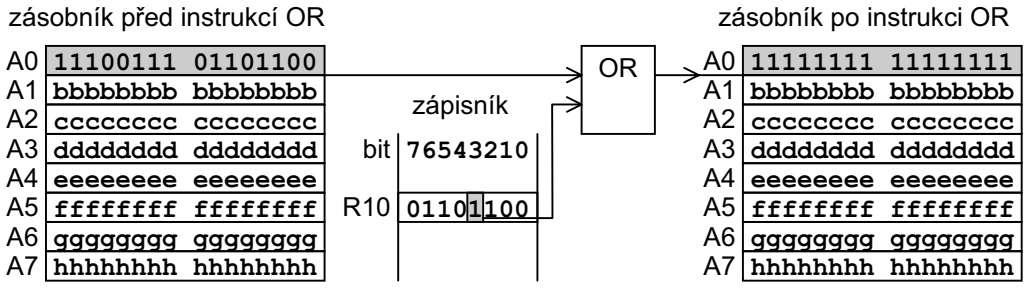
```
#def va      %X0.1
#def vb      %X0.2
#def vc      %X0.4
#def vystup  %Y0.3
;
P 0
      LD    va
      OR    vb
      ORC   vc
      WR    vystup
E 0
```

Logický součet $y = a + b$

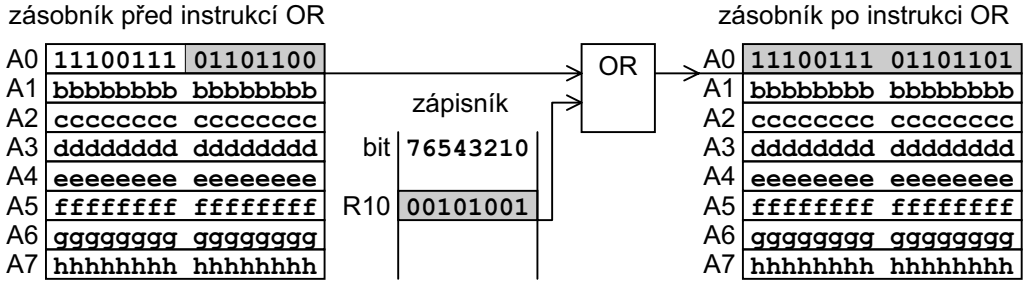
```
#def va      %X0.0
#def vb      %X0.3
#def vystup  %Y0.4
;
P 0
      LD    va
      LD    vb
      OR
      WR    vystup
E 0
```

Schéma

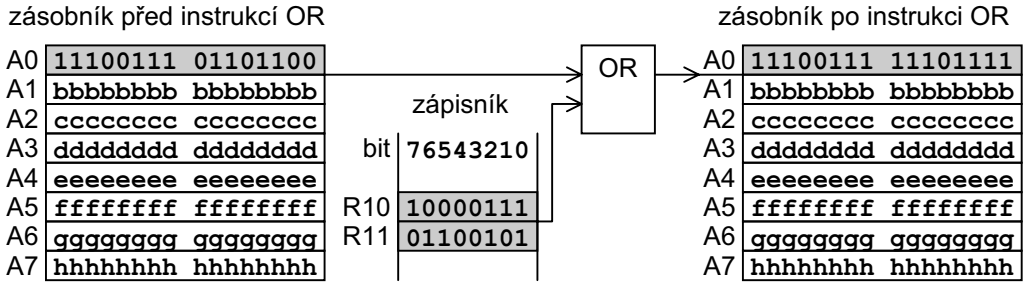
```
LD   $E76C
OR   %R10.3
```



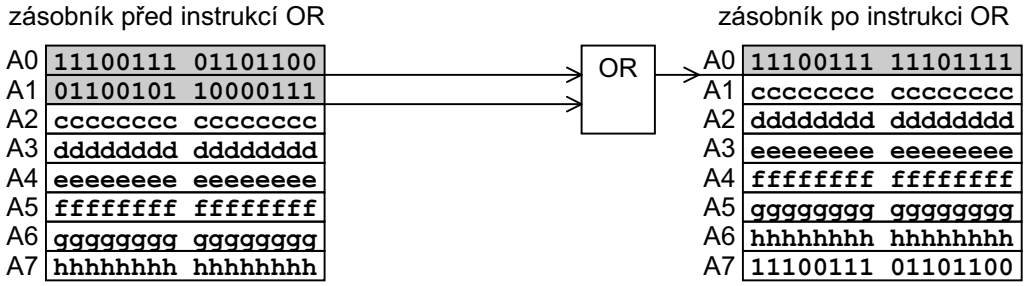

```
LD   $E76C
OR   %R10
```



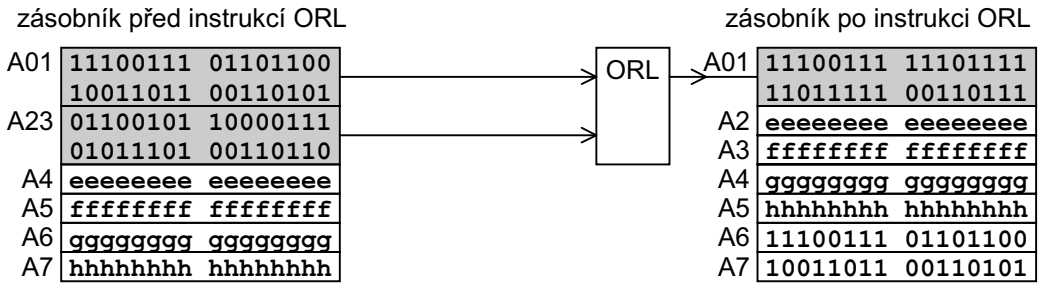

```
LD   $E76C
OR   %RW10
```




```
LD   $6587
LD   $E76C
OR
```




```
LDL  $5D366587
LDL  $9B35E76C
ORL
```



XOR, XOL	**Funkce Exclusive OR**
XOC	**Funkce Exclusive NOR**

Instrukce	**Vstupní parametry**									**Výsledek**								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
XOR								a	b								$a \oplus b$	b
XOR bez op.							a	b		b	A7	A6	A5	A4	A3	A2	$a \oplus b$	
XOL							a		b							$a \oplus b$		b
XOL bez op.					a		b			b		A7	A6	A5	A4	$a \oplus b$		
XOC								a	b								$a \oplus \bar{b}$	b

Operandy

		bit	byte	word	long
XOR	X Y S R D	B D S M E	B D S M	B D	
XOR	#			B D S M	
XOR	bez operandu			B D S M	
XOL	#				B D
XOL	bez operandu				B D
XOC	X Y S R D	B D S M E	B D S M	B D	

Funkce

XOR - výlučný logický součet vrcholu zásobníku s operandem
XOL - výlučný logický součet vrcholu zásobníku s operandem (long)
XOC - výlučný logický součet vrcholu zásobníku s negovaným operandem

Popis

Funkce výlučného logického součtu (XOR) nabývá hodnoty log.1, pokud je právě jeden její operand log.1, jinak má hodnotu log.0. Ve výrokovém počtu jí odpovídá spojení „buď..., anebo“. Pro dvě proměnné je funkce XOR totožná s funkcemi nerovnosti, součtu modulo 2 a liché parity. Pro větší počet vstupů však tato totožnost již neplatí. Dvouvstupovou funkci XOR můžeme tedy vysvětlit i jako neshodu - je rovna log.1, pokud jsou oba operandy navzájem různé. Funkce je patrná z pravdivostní tabulky:

Vstupní parametry		**Výsledek**	
a	b	$a \oplus b$	$a \oplus \bar{b}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Operandové instrukce **XOR**, **XOL** sejmou obsah adresovaného místa a provedou jeho výlučný logický součet s vrcholem zásobníku. Ten je přepsán výsledkem operace. Instrukce **XOC** provádí výlučný logický součet negace sejmutého obsahu adresovaného místa s vrcholem zásobníku. Obsah zdrojového místa je nezměněn.

Instrukce s operandem typu **bit** zpracují celý vrchol zásobníku A0 tak, že s každým jeho bitem provedou určenou operaci. Výsledek těchto 16 operací instrukce uloží zpět na vrchol zásobníku A0.

Instrukce s operandem typu **byte** zpracují dolní byte vrcholu zásobníku A0 jako 8 bitových operací mezi odpovídajícími bity zásobníku a operandu. Výsledek uloží do dolního

bytu vrcholu zásobníku A0L. Horní byte vrcholu A0H je vynulován (provedena operace AND 0).

Instrukce s operandem typu **word** zpracují vrchol zásobníku A0 jako 16 bitových operací mezi odpovídajícími bity zásobníku a operandu. Výsledek uloží na vrchol zásobníku A0.

Instrukce s operandem typu **long** zpracují vrchol zásobníku A01 jako 32 bitových operací mezi odpovídajícími bity zásobníku a operandu. Výsledek uloží na vrchol zásobníku A01.

Instrukce **XOR** bez operandu provede 16 bitových operací mezi odpovídajícími bity vrstev A0 a A1 zásobníku. Pak posune zásobník o jednu úroveň zpět a výsledek operace zapíše na nový vrchol zásobníku A0.

Instrukce **XOL** bez operandu provede 32 bitových operací mezi odpovídajícími bity dvojvrstev A01 a A23 zásobníku. Pak posune zásobník o dvě úrovně zpět a výsledek operace zapíše na nový vrchol zásobníku A01.

Příklady

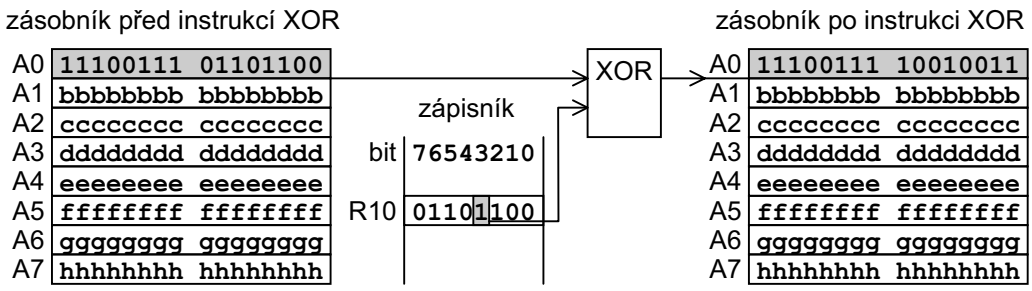
Logický výlučný součet $y = a \cdot \overline{b} + \overline{a} \cdot b$

```
#def va      %X0.1
#def vb      %X0.3
#def vystup  %Y0.2
;
P 0
      LD    va
      XOR   vb
      WR    vystup
E 0
```

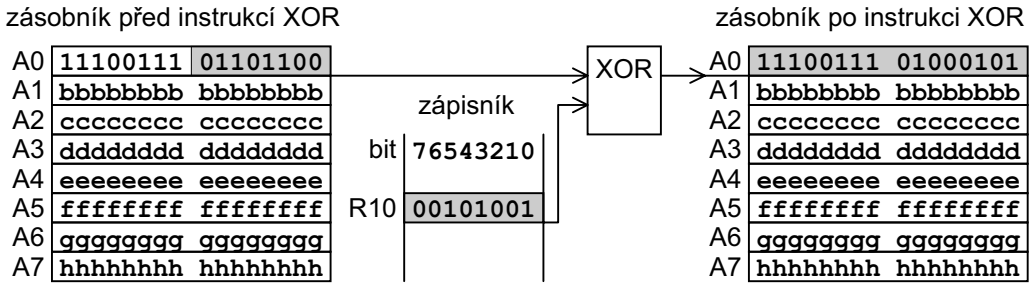

```
#def va      %X0.1
#def vb      %X0.3
#def vystup  %Y0.5
;
P 0
      LD    va
      LD    vb
      XOR
      WR    vystup
E 0
```

Schéma

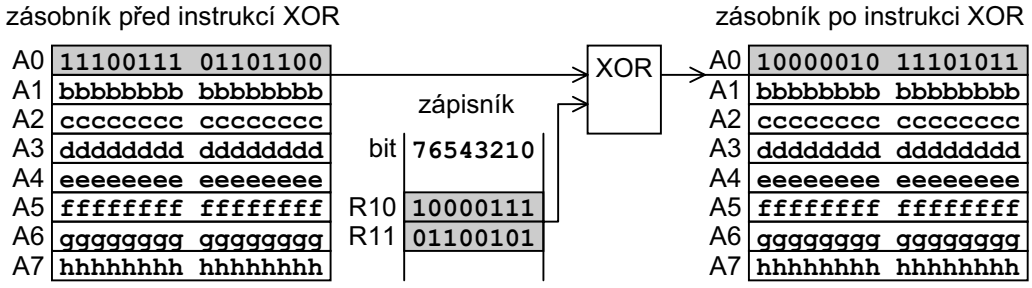
```
LD   $E76C
XOR %R10.3
```



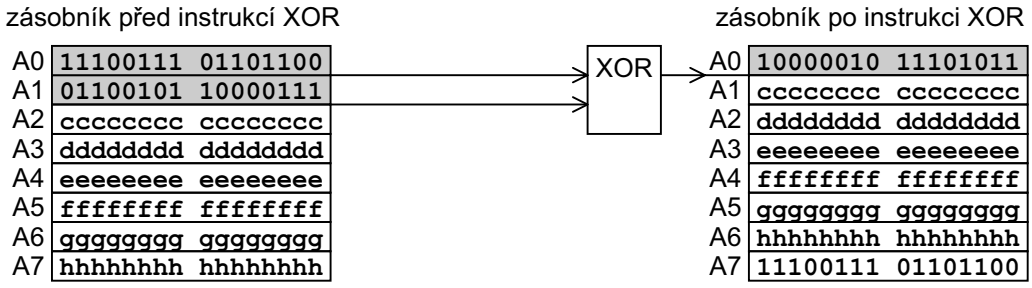

```
LD   $E76C
XOR %R10
```



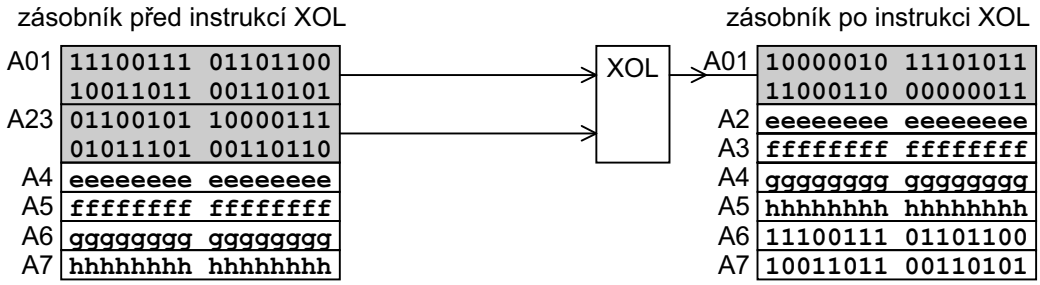

```
LD   $E76C
XOR %RW10
```




```
LD   $6587
LD   $E76C
XOR
```




```
LDL  $5D366587
LDL  $9B35E76C
XOL
```



NEG, NGL Negace

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
NEG								a									$\overline{a}$	
NGL							a									$\overline{a}$		

Operandy

		word	long
NEG	bez operandu	B D S M	
NGL	bez operandu		B D

Funkce

NEG - negace vrcholu A0 zásobníku
NGL - negace vrcholu A01 zásobníku

Popis

Instrukce provedou negaci všech bitů vrcholu zásobníku představovaného vrstvou A0, resp. dvojvrstvou A01. Ostatní úrovně zásobníku se nemění.

Schéma

```
LD   $E76C
NEG
```

zásobník před instrukcí NEG

A0	11100111 01101100
A1	bbbbbbbb bbbbbbbb
A2	cccccccc cccccccc
A3	dddddddd dddddddd
A4	eeeeeeee eeeeeeee
A5	ffffffff ffffffff
A6	gggggggg gggggggg
A7	hhhhhhhh hhhhhhhh

NEG

zásobník po instrukci NEG

A0	00011000 10010011
A1	bbbbbbbb bbbbbbbb
A2	cccccccc cccccccc
A3	dddddddd dddddddd
A4	eeeeeeee eeeeeeee
A5	ffffffff ffffffff
A6	gggggggg gggggggg
A7	hhhhhhhh hhhhhhhh

```
LD   $9B35E76C
NGL
```

zásobník před instrukcí NGL

A01	11100111 01101100
	10011011 00110101
A2	cccccccc cccccccc
A3	dddddddd dddddddd
A4	eeeeeeee eeeeeeee
A5	ffffffff ffffffff
A6	gggggggg gggggggg
A7	hhhhhhhh hhhhhhhh

NGL

zásobník po instrukci NGL

A01	00011000 10010011
	01100100 11001010
A2	cccccccc cccccccc
A3	dddddddd dddddddd
A4	eeeeeeee eeeeeeee
A5	ffffffff ffffffff
A6	gggggggg gggggggg
A7	hhhhhhhh hhhhhhhh

SET	**Podmíněné nastavení**
RES	**Podmíněné nulování**

Instrukce	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
SET								a	b								a	$a+b$
RES								a	b								a	$\overline{a}\cdot b$

Operandy

		bit	byte	word
SET	X Y S R	B D S M E	B D S M	B D
RES	X Y S R	B D S M E	B D S M	B D

Funkce

SET - podmíněný zápis log.1 do paměti, nastavení klopného obvodu typu R - S
RES - podmíněný zápis log.0 do paměti, nulování klopného obvodu typu R - S

Popis

Instrukce **SET** provádí podmíněný zápis log.1 do adresovaného místa, instrukce **RES** podmíněný zápis log.0. Pokud obě instrukce pracují nad společným paměťovým místem, pak jeho obsah můžeme chápat jako analogii klopného obvodu typu R - S nebo jiného typu klopného obvodu s asynchronními vstupy R a S. Instrukce nemění obsah zásobníku.

Funkce SET nastavuje obsah adresovaného místa na log.1 pouze tehdy, pokud má řídící proměnná načtená z vrcholu zásobníku hodnotu log.1, jinak se obsah místa nemění. Funkce RES nuluje obsah adresovaného místa pouze tehdy, pokud má řídící proměnná hodnotu log.1, jinak se obsah nemění. Souhrnně lze tedy říci, že funkce SET a RES jsou aktivní (mění obsah adresovaného místa) pouze tehdy, pokud má řídící proměnná hodnotu log.1 a v tomto případě provádí funkce SET zápis log.1 a RES zápis log.0. Má-li řídící proměnná hodnotu log.0, pak se obsah paměťového místa nezmění po SET ani po RES (pamatuje si minulý obsah). Funkce SET a RES můžeme popsat pravdivostní tabulkou:

Vstupní parametry		Výsledek	
a	b	$a+b$ (SET)	$\overline{a}\cdot b$ (RES)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	1	0

Pro instrukce s operandem typu **bit** je řídící proměnná rovna logickému součtu (OR) všech šestnácti bitů vrcholu zásobníku A0. Je-li tedy obsah A0 nenulový ($A0 \neq 0$), pak instrukce **SET** nastavuje adresovaný bit na log.1 a instrukce **RES** za této podmínky zapisuje log.0. Je-li obsah úrovně A0 nulový (A0 = 0), pak žádná z instrukcí obsah adresovaného místa nemění.

Instrukce s operandem typu **byte** provádějí naráz 8 bitových operací pro stejnolehlé bity nejnižšího bytu vrcholu zásobníku A0 (soubor 8 řídících proměnných a) a adresovaného místa (soubor 8 stavových proměnných b).

Instrukce s operandem typu **word** provádějí naráz 16 bitových operací pro stejnolehlé bity vrcholu zásobníku A0 (soubor 16 řídících proměnných a) a adresovaného místa (soubor 16 stavových proměnných b).

Poznámka

Technicky je možné adresovat libovolné místo, do kterého lze zapisovat. Z funkčního hlediska však některé možnosti nemají smysl, nebo nezaručují správnou činnost instrukcí (např. obsazené vstupy X, aktivní systémové registry S).

Během jednoho cyklu uživatelského programu smí být aktivováno více instrukcí **SET** nebo **RES**, adresujících společnou stavovou proměnnou. Při aktivaci stejných instrukcí (buď pouze **SET** nebo pouze **RES**) je výsledek po poslední instrukci stejný, jako bychom provedli jedinou operaci se součtovou řídící proměnnou (sečtenou funkcí OR). Pokud se nad společnou proměnnou provedou aktivní instrukce v pořadí **SET** a **RES** (s řídící proměnnou hodnoty log.1), zůstane v platnosti stav po instrukci **RES** (paměť s převažujícím nulováním). Při opačném pořadí instrukcí (tedy **RES** a pak **SET**) zůstane platným stav po instrukci **SET** (paměť s převažujícím nastavením) - převahu má vždy poslední aktivní instrukce.

LET	**Impulz od náběžné hrany**
BET	**Impulz od libovolné hrany**

Instrukce	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
LET								a	b								$a \cdot \overline{b}$	a
BET								a	b								$a \oplus b$	a

Operandy

		bit	byte	word
LET	X Y S R	B D S M E	B D S M	B D
BET	X Y S R	B D	B D	B D

Funkce

LET - generování spouštěcího impulzu od náběžné hrany
BET - generování spouštěcího impulzu od libovolné hrany

Popis

Instrukce nastavují svou adresovanou stavovou proměnnou podle stejných pravidel jako instrukce **WR**. Navíc porovnají původní a nově zapsaný obsah stavové proměnné (před a po provedení zápisu).

Instrukce **LET** nastaví výsledek na vrcholu zásobníku na log.1 jen tehdy, pokud dojde ke změně stavové proměnné z hodnoty log.0 na log.1 (náběžná hrana), jinak jej nulují.

Instrukce **BET** nastaví výsledek na vrcholu zásobníku na log.1 jen tehdy, pokud dojde ke změně stavové proměnné z hodnoty log.0 na log.1 nebo z hodnoty log.1 na log.0 (libovolná hrana), jinak jej nulují.

Instrukce nemění obsah zásobníku.

Logické funkce LET a BET (hodnota nastavovaná na vrchol zásobníku) lze definovat pravdivostní tabulkou:

Vstupní parametry		Výsledek	
a	b	$a \cdot \overline{b}$ (LET)	$a \oplus b$ (BET)
0	0	0	0
0	1	0	1
1	0	1	1
1	1	0	0

Instrukce s operandem typu **bit** provedou logický součet (OR) všech šestnácti bitů vrcholu A0 a hodnota tohoto součtu bude po testu a nastavení vrcholu zásobníku uložena v adresovaném bitu. Výsledek porovnání na vrcholu zásobníku je stejný ve všech šestnácti bitech. Nastavení vrcholu zásobníku na log.1 tedy představuje hodnota 65 535 (samé jedničky).

Instrukce s operandem typu **byte** provádějí naráz 8 bitových operací pro stejnolehlé bity nejnižšího bytu vrcholu zásobníku A0 (soubor 8 řídících proměnných a) a adresovaného místa (soubor 8 stavových proměnných b). Výsledky porovnání jsou uloženy v dolním bytu vrcholu zásobníku A0 (soubor 8 výsledků). Horní byte A0 je vynulován.

Instrukce s operandem typu **word** provádějí naráz 16 bitových operací pro stejnolehlé bity vrcholu zásobníku A0 (soubor 16 řídících proměnných a) a adresovaného místa

(soubor 16 stavových proměnných *b*). Výsledky porovnání jsou uloženy na vrcholu zásobníku A0 (soubor 16 výsledků).

Poznámka

Pro správné fungování instrukcí **LET**, **BET** je nezbytně nutné, aby zápis do stavové proměnné prováděla pouze jediná instrukce **LET**, **BET** (jednou v každém cyklu) a aby na jejím obsahu nepracoval systémový program.

Pokud výstup instrukcí **LET**, **BET** zpracujeme pouze ve vnitřních proměnných, může impulz trvat kratší dobu. Časový odstup mezi náběžnými hranami musí být spolehlivě delší, než je dvojnásobek doby cyklu (v jednom cyklu nelze vyhodnotit náběžnou a sestupnou hranu, pokud ji nevyhodnocujeme v přerušujícím procesu). Pokud je však řídící proměnná odvozena od vnitřních proměnných uživatelského programu (nikoliv od vstupů nebo systémových proměnných), lze v rámci jednoho cyklu vyhodnotit i více náběžných hran.

Při prvé aktivaci systémového programu (po restartu) mohou instrukce **LET**, **BET** náhodně vygenerovat falešné informace. Tomu lze předejít buď ignorováním výsledků prvního cyklu, který bude chápán jako ustálení přechodového děje, nebo před prvým cyklem v procesu ošetření restartu nastaví uživatel všechny stavové proměnné do jedniček pro instrukce **LET** resp. do stavu, který odpovídá klidovému ustálenému stavu, pro instrukce **BET**.

FLG Logické příznaky vrcholu zásobníku

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								S0
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
FLG								VAL		A6	A5	A4	A3	A2	A1	VAL	N4	NFLG

VAL zpracovaná hodnota
N4 - logický součin AND celého vrcholu zásobníku A0 (viz popis)
NFLG - soubor logických funkcí nad vrcholem zásobníku A0 (viz popis)

Operandy

		word
FLG	bez operandu	B D S M

Funkce

FLG - logické AND a příčné funkce bytů vrcholu zásobníku v S1

Popis

Instrukce **FLG** zpracuje obsah A0, posune zásobník vpřed a provede následující operace:

- Určí počet jedničkových bitů původního vrcholu A0. Toto číslo N nabývá hodnoty 0 až 16, dvojkově je lze zapsat na pěti bitech. Čtyři spodní bity N3 až N0 jsou uloženy v dolní polovině systémového registru S1. Nejvyšší bit N4, který má současně význam podélného logického součinu (AND) všech bitů A0, je uložen ve všech bitech nového vrcholu zásobníku A0.
  Údaj N lze výhodně využít k realizaci symetrických funkcí (parita, majorita, prahové funkce, apod.), např.:
  N > 0 (N $\neq$ 0) - logický součet OR
  N0 = S1.0 - lichá parita, součet nad 2
  N4 = A0 - logický součin AND celého vrcholu zásobníku A0
  N3 = S1.3 - pokud byl horní byte vrcholu zásobníku A0 nulový, logický součin AND dolního bytu A0
  N = 2 - prahová funkce F_{16}^{2} nebo F_{n}^{2}
  N = k - prahová funkce F_{16}^{k} nebo F_{n}^{k}
  N = 1 - právě 1 z 16 (1 z n), funkce „buď, anebo“, „výlučný součet“
  N = {soubor čísel} - libovolná symetrická funkce definovaná souborem čísel
- Odděleně provede funkce logického součtu (OR) a součinu (AND) pro oba dolní byty původního vrcholu zásobníku A0 a výsledky uloží do registru S1.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	ORH	ORL	ANH	ANL	N3	N2	N1	N0

S1.3 až S1.0 (N3 až N0)
- Spolu s hodnotou bitu N4, který je uložen ve všech bitech nového vrcholu zásobníku A0 tvoří pětibitové číslo N, které udává počet jedničkových bitů v původním vrcholu zásobníku A0.

S1.0 (N0) - lichá parita původního vrcholu zásobníku

S1.4 (ANL) - podélný logický součin všech bitů dolního bytu původního vrcholu zásobníku
S1.5 (ANH) - podélný logický součin všech bitů horního bytu původního vrcholu zásobníku
S1.6 (ORL) - podélný logický součet všech bitů dolního bytu původního vrcholu zásobníku
S1.7 (ORH) - podélný logický součet všech bitů horního bytu původního vrcholu zásobníku

STK Sklopení zásobníku

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
STK	*h*	*g*	*f*	*e*	*d*	*c*	*b*	*a*		*h*	*g*	*f*	*e*	*d*	*c*	*b*	NSTK	

NSTK - logické součty všech vrstev zásobníku (viz popis)

Operandy

		long
STK	bez operandu	B D S M

Funkce

STK - sklopení logických hodnot 8 úrovní zásobníku do A0

Popis

Instrukce **STK** provede pro každou vrstvu zásobníku A0 až A7 logický součet (OR) všech šestnácti bitů úrovně a pak tento „sloupec“ osmi bitových hodnot „sklopí“ do vrstvy A0 podle následujícího schématu:

A0.7	A0.6	A0.5	A0.4	A0.3	A0.2	A0.1	A0.0
OR7	OR6	OR5	OR4	OR3	OR2	OR1	OR0

OR0 až OR7 jsou hodnoty logických součtů jednotlivých vrstev A0 až A7.
Horní byte vrcholu zásobníku je vynulován, ostatní úrovně zásobníku se nemění.

ROL	**Rotace čísla vlevo**
ROR	**Rotace čísla vpravo**

Instrukce	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	
ROL n								a	n								$a\langle\langle n$	
ROR n								a	n								$a\rangle\rangle n$	

Operandy

		word
ROL	n	B D S M
ROR	n	B D

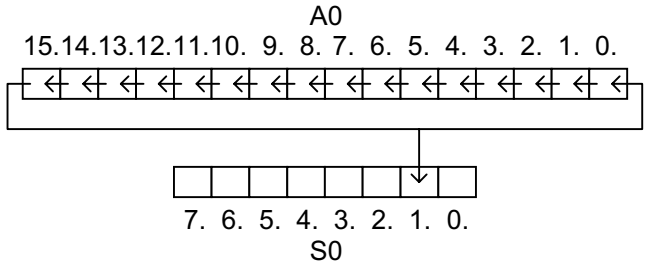
Funkce

ROL - n-násobná rotace hodnoty vrcholu zásobníku A0 vlevo
ROR - n-násobná rotace hodnoty vrcholu zásobníku A0 vpravo

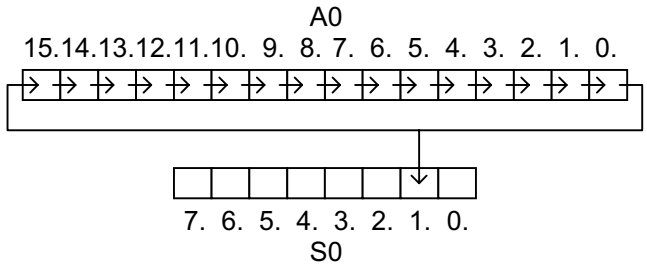
Popis

Instrukce **ROL** provádí kruhový posun spodních šestnácti bitů vrcholu zásobníku A0 vlevo. Instrukce **ROR** provádí totéž vpravo. Parametr instrukce určuje počet jednotkových posunů, tj. o kolik bitů je obsah posunut. Je-li operand větší než 15, přepočítává se modulo 16, takže se nikdy neprovádí víc než 15 posunů. Při nulové hodnotě parametru se žádný posun neprovádí, pouze se nastaví příznaky.

Schématické znázornění instrukce **ROL** n:



Schématické znázornění instrukce **ROR** n:



Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	-	-	-	-	≤	CO	ZR

S0.0 (ZR) - nulovost výsledku
1 - výsledek je 0

S0.1 (CO) - výstupní přenos
1 - hodnota posledního přenášeného bitu v kruhu (z nejvyššího bitu do nejnižšího při **ROL**, resp. z nejnižšího bitu do nejvyššího při **ROR** se přenáší log.1)

S0.2 (≤) - logický součet S0.0 OR S0.1

SWP	**Záměna bytů vrcholu zásobníku**
SWL	**Záměna vrstev A0 a A1 zásobníku**

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
SWP								*ab*									*ba*	
SWL							*ab*	*cd*								*cd*	*ab*	

Operandy

		word	long
SWP	bez operandu	B D S M	
SWL	bez operandu		B D

Funkce

SWP - vzájemná záměna obou bytů vrcholu zásobníku
SWL - vzájemná záměna vrstev A0 a A1 zásobníku

Popis

Instrukce **SWP** zamění obsahy obou bytů vrcholu zásobníku A0, instrukce **SWL** zamění obsahy vrstev A0 a A1 zásobníku. Ostatní úrovně zásobníku se nemění.

3. ČÍTAČE, POSUVNÉ REGISTRY, ČASOVAČE, KROKOVÝ ŘADIČ

CTU	**Dopředný čítač**
CTD	**Zpětný čítač**
CNT	**Obousměrný čítač**

Instr.	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
CTU							UP	RES	VAL0	A6	A5	A4	A3	A2	UPC	RES	VAL	VAL
CTD							DWN	SET	VAL0	A6	A5	A4	A3	A2	DNC	SET	VAL	VAL
CNT						UP	DWN	RES	VAL0	A6	A5	A4	A3	UPC	DNC	RES	VAL	VAL

UP - řídící proměnná pro čítání nahoru (typ bit)
DWN - řídící proměnná pro čítání dolů (typ bit)
RES - nulovací proměnná čítače (typ bit)
SET - nastavovací proměnná čítače (typ bit)
VAL0 - číselná hodnota čítače před instrukcí (typ word)
UPC - přenos čítání nahoru do vyšší kaskády (typ bit)
DNC - přenos čítání dolů do vyšší kaskády (typ bit)
VAL - aktuální číselná hodnota čítače (typ word)

Operandy

		word
CTU	R	B D S M E
CTD	R	B D S M E
CNT	R	B D S M

Funkce

CTU - dopředný čítač
CTD - zpětný čítač
CNT - obousměrný čítač

Popis

Instrukce **CTU** otestuje, jestli se hodnota UP změnila oproti stavu UP po posledně aktivované instrukci **CTU** nebo **CNT** z log.0 na log.1 (náběžná hrana). Pokud ano, pak se obsah čítače, který je adresován instrukcí, zvýší o 1. Pokud ne, obsah čítače se nemění. Současně instrukce **CTU** posune zásobník vpřed a na nový vrchol uloží aktuální obsah čítače. Pokud při čítání došlo k přenosu (změna obsahu čítače z maximální hodnoty na 0), je do proměnné UPC uložena hodnota log.1 (samé jedničky). Pokud k přenosu nedošlo, je UPC = log.0. Proměnná RES zůstává nedotčena.

Pokud je proměnná RES = log.1, vynuluje se obsah čítače. Pokud je současně vyhodnocena náběžná hrana, má přednost nulování a informace o hraně se ztratí. Nulování však neruší mechanizmus vyhodnocování náběžných hran, takže po odeznění požadavku na RES je prvá náběžná hrana UP normálně zpracována.

Instrukce **CTD** otestuje, jestli se hodnota DWN změnila oproti stavu DWN po posledně aktivované instrukci **CTD** nebo **CNT** z log.0 na log.1 (náběžná hrana), pak se obsah čítače, který je adresován instrukcí, sníží o 1. Pokud ne, obsah čítače se nemění. Součas-

ně instrukce **CTD** posune zásobník vpřed a na nový vrchol uloží obsah čítače. Pokud při čítání došlo k přenosu (změna obsahu čítače z 0 na maximální hodnotu), je do proměnné DNC uložena hodnota log.1 (samé jedničky). Pokud k přenosu nedošlo, je DNC = log.0. Proměnná SET zůstává nedotčena.

Pokud je proměnná SET = log.1, nastaví se obsah čítače na maximální hodnotu. Pokud je současně vyhodnocena náběžná hrana, má přednost nastavení a informace o hraně se ztratí. Nastavení však neruší mechanismus vyhodnocování náběžných hran, takže po odeznění požadavku na SET je prvá náběžná hrana DWN normálně zpracována.

Instrukce **CNT** otestuje vstupy UP a DWN. Pokud se hodnota UP změnila oproti stavu UP po posledně aktivované instrukci **CTU** nebo **CNT** z log.0 na log.1 (náběžná hrana), pak se obsah čítače, který je adresován instrukcí, zvýší o 1. Pokud se hodnota DWN změnila oproti stavu DWN po posledně aktivované instrukci **CTD** nebo **CNT** z log.0 na log.1 (náběžná hrana), pak se obsah slova čítače, který je adresován instrukcí, sníží o 1. Při současném výskytu obou náběžných hran se obsah čítače nemění (vzájemná eliminace).

Současně instrukce **CNT** posune zásobník vpřed a na nový vrchol uloží obsah čítače. Pokud při čítání došlo k přenosu nahoru (změna obsahu čítače z maximální hodnoty na 0), je do proměnné UPC uložena hodnota log.1 (samé jedničky). Pokud došlo k přenosu dolů (změna obsahu čítače z 0 na maximální hodnotu), je do proměnné DNC uložena hodnota log.1 (samé jedničky). Pokud k přenosu nedošlo, jsou obě proměnné nulové. Proměnná RES zůstává nedotčena.

Pokud je proměnná RES = log.1, vynuluje se obsah čítače. Pokud je současně vyhodnocena náběžná hrana, má přednost nulování a informace o hraně se ztratí. Nulování však neruší mechanismus vyhodnocování náběžných hran, takže po odeznění požadavku na RES je prvá náběžná hrana UP nebo DWN normálně zpracována.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	-	-	-	-	≤	CO	ZR

S0.0 (ZR) - nulovost výsledku
1 - hodnota čítače je nulová
S0.1 (CO) - výstupní přenos
1 - hodnota čítače překročila maximum
S0.2 (≤) - logický součet S0.0 OR S0.1

Poznámka

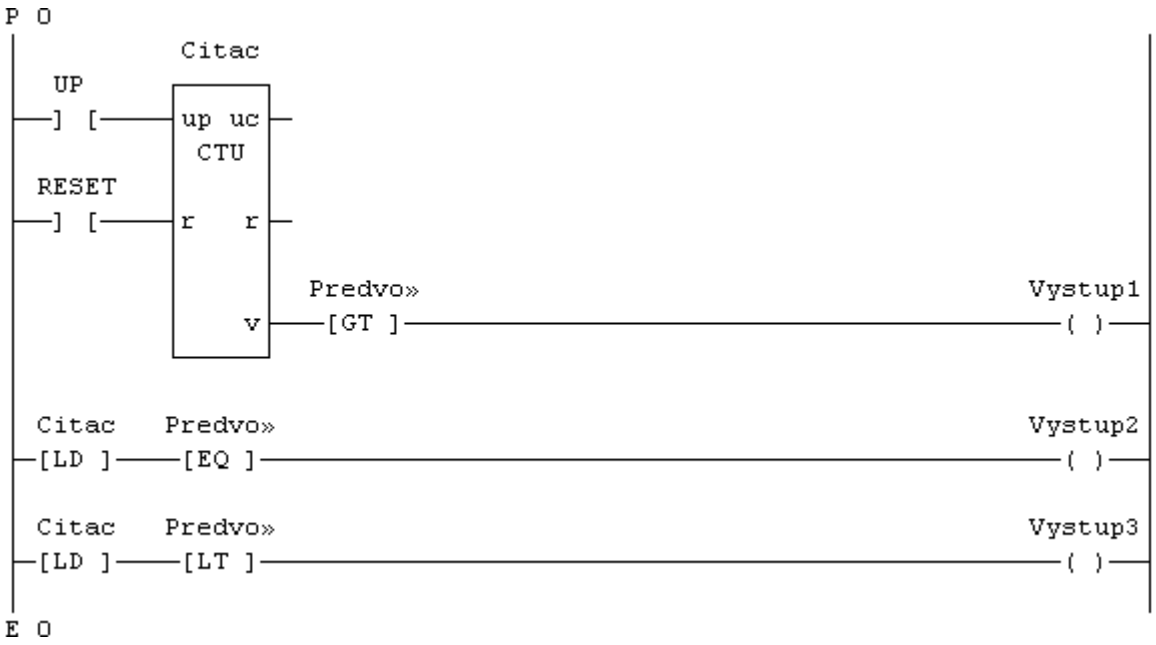
Při prvé aktivaci čítače (po zapnutí nebo při změně režimu časovače) je do paměti minulé hodnoty řídící veličiny XT uložena log.1, takže čítání zahájí prvá skutečně vyhodnocená náběžná hrana, nikoliv náhodný přechod.

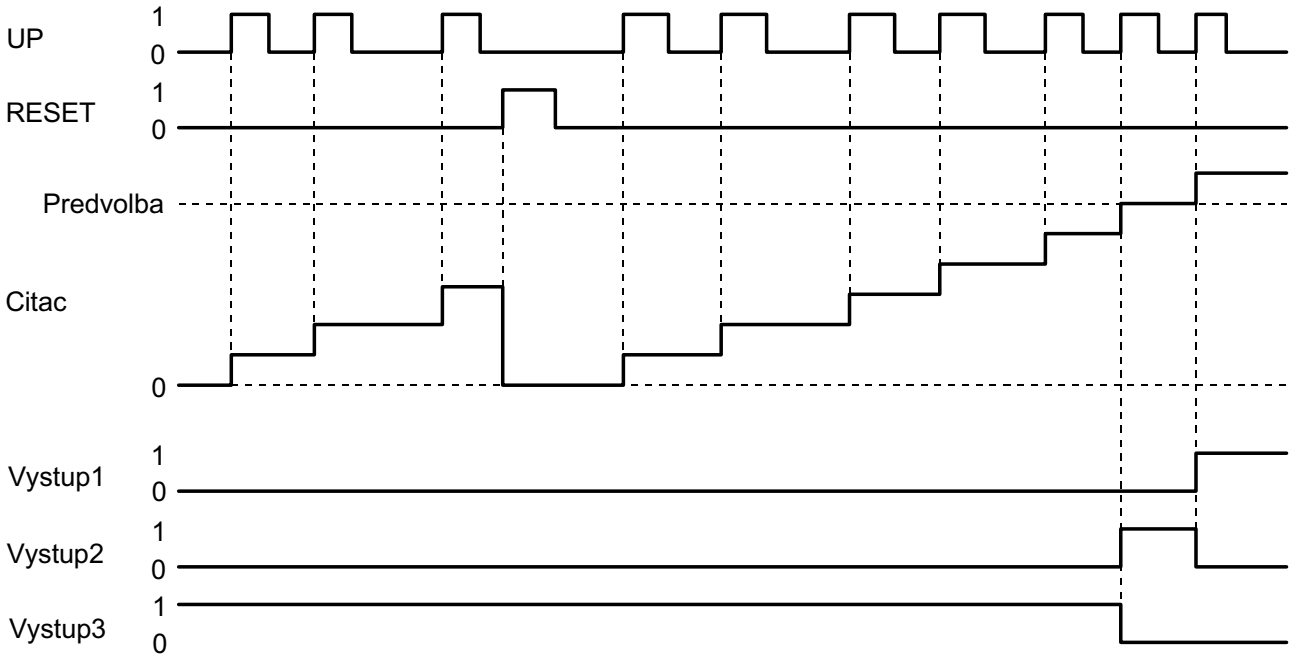
Nad jedním objektem mohou pracovat libovolné z instrukcí **CTU**, **CTD**, **CNT**, **SFL** a **SFR**, přičemž změna typu instrukce nevyvolá inicializaci. Je však třeba zajistit, aby v jednom cyklu proběhla maximálně jedna z těchto operací pro stejný směr čítání (nelze použít v jednom cyklu např. dvakrát **CTU**, nebo **CTD** a **CNT**, apod.).

Příklady

Realizujme dopředný čítač

```
#reg word Citac
#def UP         %X0.0
#def RESET      %X0.5
#def Vystup1    %Y0.0
#def Vystup2    %Y0.1
#def Vystup3    %Y0.2
#def Predvolba  50
;
P 0
      LD    UP
      LD    RESET
      CTU   Citac
      GT    Predvolba
      WR    Vystup1
      LD    Citac
      EQ    Predvolba
      WR    Vystup2
      LD    Citac
      LT    Predvolba
      WR    Vystup3
E 0
```

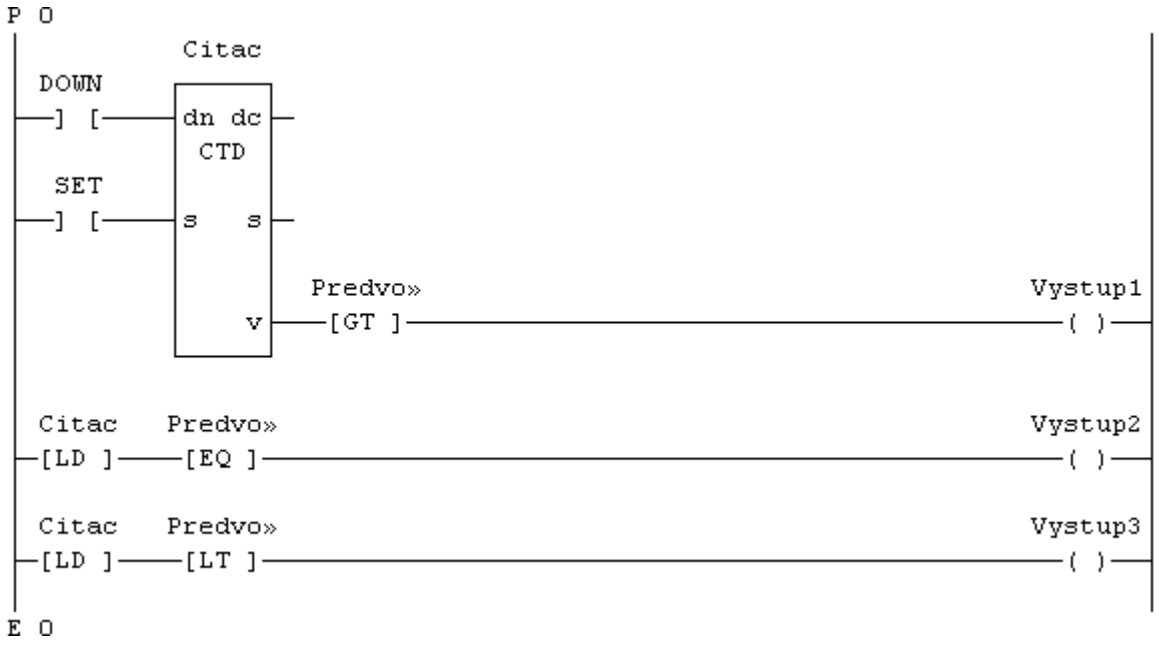


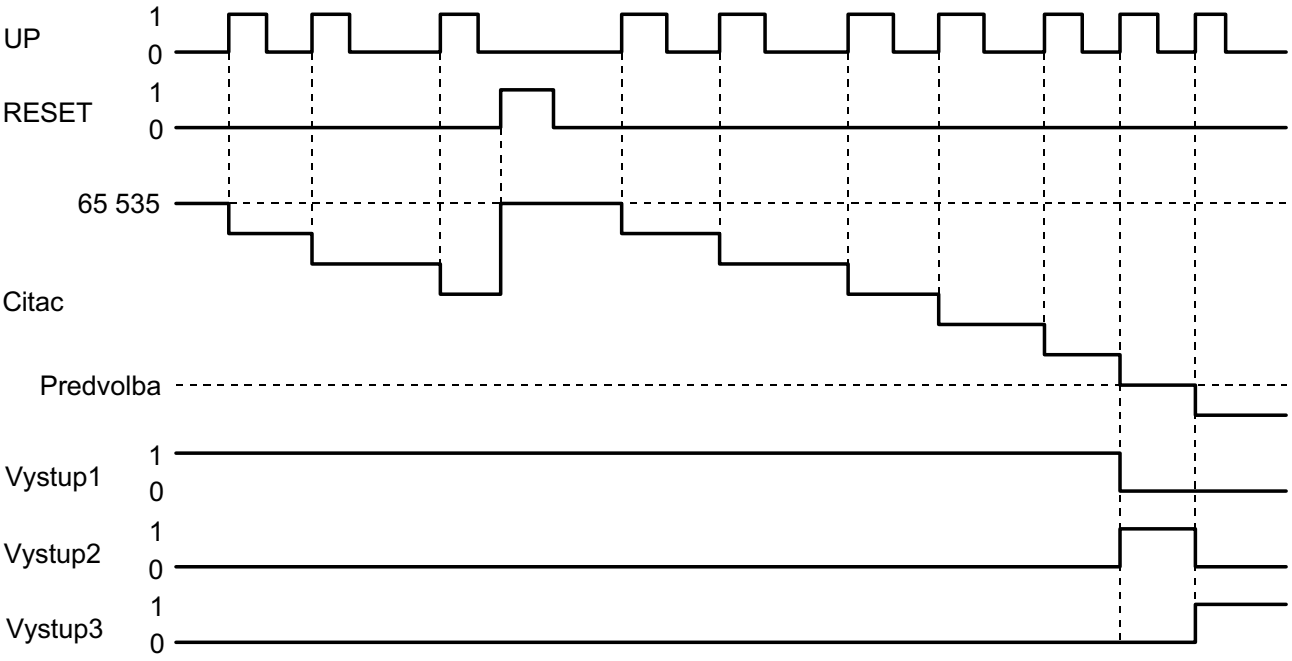


Chování dopředného čítače podle příkladu

Realizujme zpětný čítač

```
#reg word Citac
#def DOWN       %X0.0
#def SET        %X0.5
#def Vystup1    %Y0.0
#def Vystup2    %Y0.1
#def Vystup3    %Y0.2
#def Predvolba  65500
;
P 0
      LD    DOWN
      LD    SET
      CTD   Citac
      GT    Predvolba
      WR    Vystup1
      LD    Citac
      EQ    Predvolba
      WR    Vystup2
      LD    Citac
      LT    Predvolba
      WR    Vystup3
E 0
```



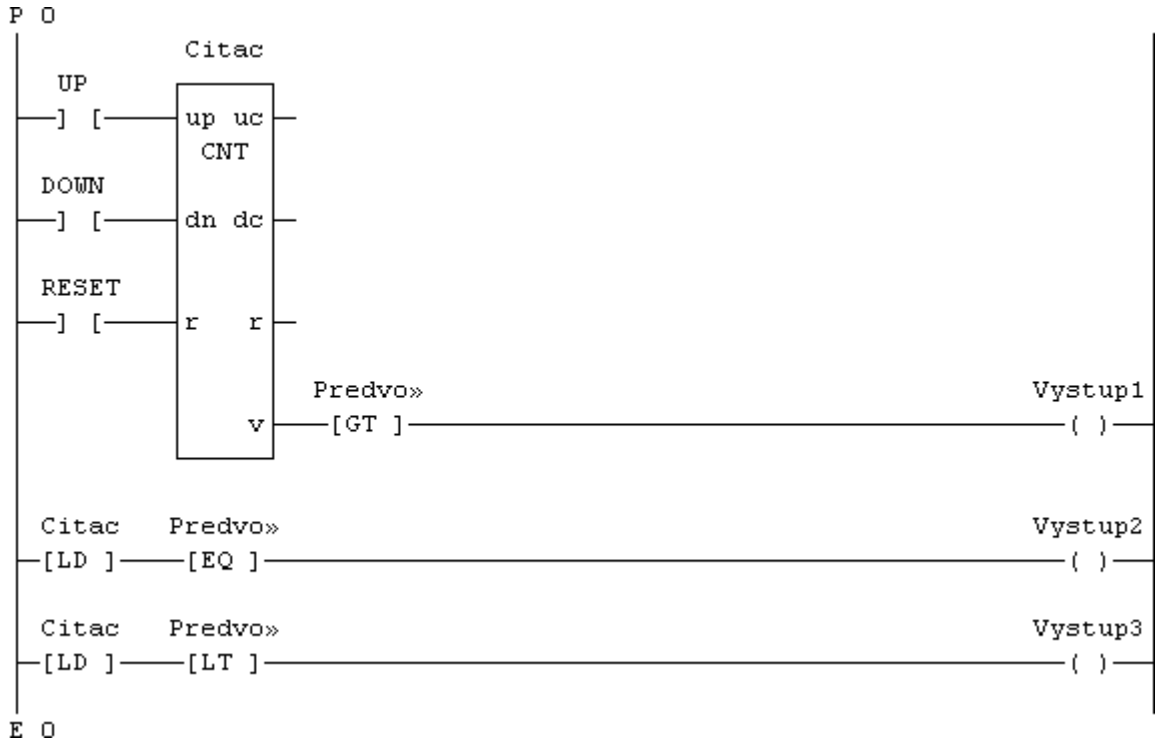


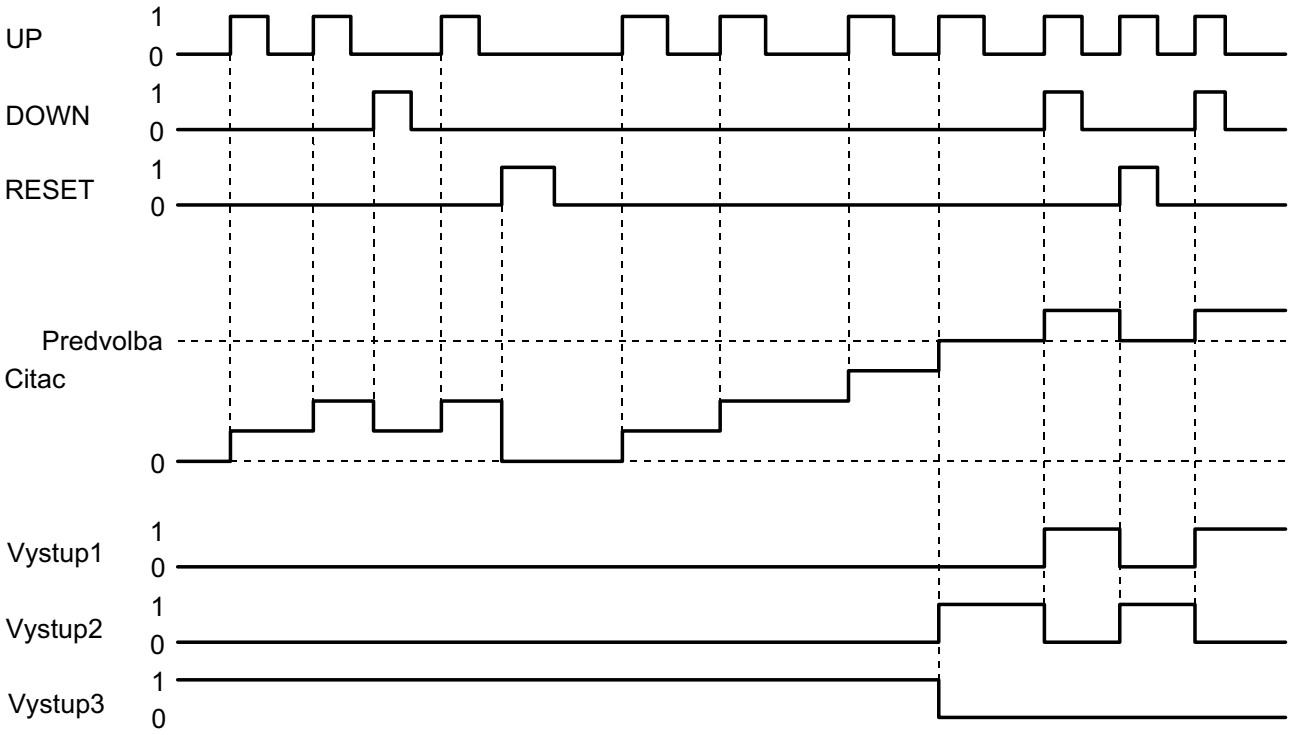
Chování zpětného čítače podle příkladu

Realizujme obousměrný čítač

```
#reg word Citac
#def DOWN       %X0.0
#def UP         %X0.1
#def RESET      %X0.5
#def Vystup1    %Y0.0
#def Vystup2    %Y0.1
#def Vystup3    %Y0.2
#def Predvolba  50
;
P 0
     LD    UP
     LD    DOWN
     LD    RESET
```

```
      CNT   Citac
      GT    Predvolba
      WR    Vystup1
      LD    Citac
      EQ    Predvolba
      WR    Vystup2
      LD    Citac
      LT    Predvolba
      WR    Vystup3
E 0
```





Chování obousměrného čítače podle příkladu

SFL	**Posuvný registr vlevo**
SFR	**Posuvný registr vpravo**

Instr.	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
SFL							CLC	DATAI	VAL0	A6	A5	A4	A3	A2	CLC	DATAO	VAL	VAL
SFR							CLC	DATAI	VAL0	A6	A5	A4	A3	A2	CLC	DATAO	VAL	VAL

CLC - řídící proměnná pro posun (typ bit)
DATAI - hodnota vsouvaného bitu (typ bit)
VAL0 - číselná hodnota registru před instrukcí (typ word)
DATAO- hodnota vysouvaného bitu (typ bit)
VAL - aktuální číselná hodnota registru (typ word)

Operandy

		word
SFL	R	B D S M
SFR	R	B D S M

Funkce

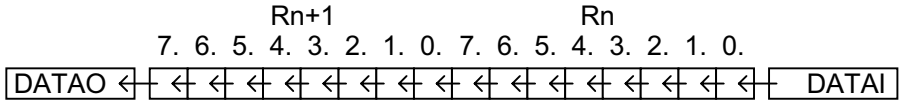
SFL - posun hodnoty registru vlevo
SFR - posun hodnoty registru vpravo

Popis

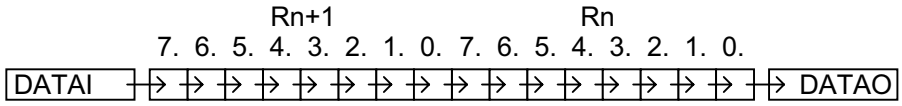
Pokud se hodnota CLC změnila oproti stavu CLC po posledně aktivované instrukci **SFL** nebo **SFR** z log.0 na log.1 (náběžná hrana), pak se celý obsah posuvného registru posune o 1 bit. Po instrukci **SFL** se adresovaný registr posune o 1 bit vlevo, na pozici nejnižšího bitu se nasune obsah proměnné DATAI a z pozice nejvyššího bitu se vysune obsah do proměnné DATAO. Po instrukci **SFR** se adresovaný registr posune o 1 bit vpravo, na pozici nejvyššího bitu se nasune obsah proměnné DATAI a z pozice nejnižšího bitu se vysune obsah do proměnné DATAO.

Pokud nebyla náběžná hrana vyhodnocena, obsah registru se nemění. Současně instrukce posune zásobník vpřed a na nový vrchol uloží aktuální obsah registru. Proměnná CLC zůstává nedotčena.

Schématické znázornění instrukce **SFL**:



Schématické znázornění instrukce **SFR**:



Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	-	-	-	-	≤	CO	ZR

S0.0 (ZR) - nulovost výsledku
1 - hodnota registru je nulová
S0.1 (CO) - vysunutá hodnota
S0.2 (≤) - logický součet S0.0 OR S0.1

Poznámka

Nad jedním objektem mohou pracovat libovolné z instrukcí **CTU**, **CTD**, **CNT**, **SFL** a **SFR**, přičemž změna typu instrukce nevyvolá inicializaci. Je však třeba zajistit, aby v jednom cyklu proběhla maximálně jedna z těchto operací pro stejný směr čítání, resp. posunu (nelze použít v jednom cyklu např. dvakrát **SFL**, apod.).

TON	**Časovač (zpožděný přítah)**
TOF	**Časovač (zpožděný odpad)**

Instrukce	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
TON							XT	VAL	TIM							XT	YT	TIM
TOF							XT	VAL	TIM							XT	YT	TIM

XT - řídící proměnná (typ bit)
VAL - číselná hodnota předvolby (typ word)
*TIM - číselná hodnota časovače (typ word) - jednotky dané parametrem **k***
YT - výstupní proměnná, výsledek porovnání aktuální hodnoty časovače s předvolbou (typ bit)

Operandy

		word
TON	R.k	B D S M E
TOF	R.k	B D S M

k - kód časové jednotky (není-li zadán, bere se k = 0)
k = 0 - 10 ms, 1 - 100 ms, 2 - 1 s, 3 - 10 s

Funkce

TON - časování od sepnutí vstupu (posunutá náběžná hrana)
TOF - časování od rozepnutí vstupu (posunutá sestupná hrana)

Popis

Instrukce **TON** otestuje řídící proměnnou XT. Je-li XT = log.0, je časovač pasivní. Je-li XT = log.1, je aktivní. Pasivní časovač je vynulován a jsou nulovány i příznaky S0.4 a S0.5. Pokud je předvolba nenulová, je vynulován celý registr S0. Aktivní časovač aktualizuje časový údaj a na vrchol zásobníku ukládá výsledek porovnání s předvolbou. Není-li dosaženo předvolby, je YT = log.0. Je-li předvolba dosažena nebo překročena, je YT = log.1 (samé jedničky). Přetečení rozsahu časovače vyvolá nastavení bitů S0.4 a S0.5.

Instrukce **TOF** otestuje řídící proměnnou XT. Je-li XT = log.1, je časovač pasivní. Je-li XT = log.0, je aktivní. Pasivní časovač je vynulován a jsou nulovány i příznaky S0.4 a S0.5. Pokud je předvolba nenulová, jsou vynulovány i příznaky S0.2 až S0.0 a výstup časovače YT je nastaven na hodnotu log.1 (vrchol zásobníku A0). Aktivní časovač aktualizuje časový údaj a na vrchol zásobníku ukládá výsledek porovnání s předvolbou. Není-li dosaženo předvolby, je YT = log.1 (samé jedničky). Je-li předvolba dosažena nebo překročena, je YT = log.0. Přetečení rozsahu časovače vyvolá nastavení bitů S0.4 a S0.5.

Poznámka

Nad jedním objektem smí být provozován jediný typ instrukce časovače s jedinou časovou jednotkou. Při jakékoliv změně typu instrukce nebo časové jednotky se provede inicializace - časovač se vynuluje.

Nad jedním objektem smí být aktivní jen jediná instrukce časovače. Časoměrné systémové proměnné jsou aktualizovány pouze v otočce cyklu (čas běží po skocích). Během cyklu mají stále stejnou hodnotu, takže je lhostejné, na kterém místě v programu je instrukce časovače umístěna. Pokud však dojde v jednom cyklu k vynechání instrukce časovače, nedojde v následující otočce cyklu k jeho aktualizaci - časovač přestane

časovat. Časovat začne opět průchodem programu instrukcí časovače, ovšem s tím, že jeho hodnota je poznamenaná odpovídajícím časovým výpadkem.

Pokud má předvolba VAL hodnotu 0, je výstupní proměnná YT shodná s řídící proměnnou XT. Stav systémových příznaků S0 není definován.

Pokud je časová jednotka **k** přibližně stejná nebo menší než doba cyklu PLC, je funkce příznaků S0.0 a S0.5 nespolehlivá (hodnota časovače narůstá po větších skocích a předvolba, resp. rozsah časovače, je rovnou překročena, takže její dosažení nemusí být detekováno). Příznaky S0.0 a S0.5 lze nahradit testem náběžné hrany příznaků S0.2 a S0.4.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	-	OC	OV	-	≤	CO	ZR

S0.0 (ZR) - dosažení předvolby
1 - předvolba dosažena v tomto cyklu

S0.1 (CO) - překročení předvolby
1 - předvolba překročena

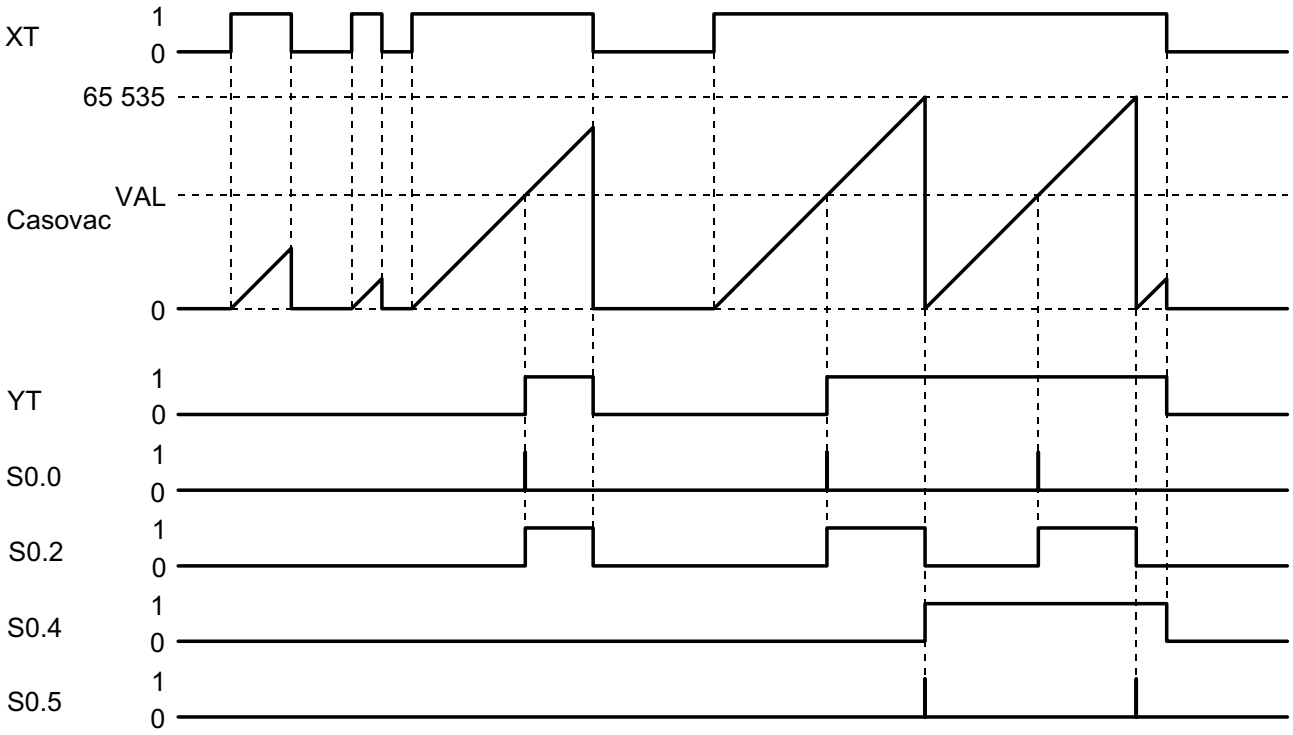
S0.2 (≤) - logický součet S0.0 OR S0.1
1 - předvolba dosažena nebo překročena

S0.4 (OV) - překročení maximálního rozsahu časovače
1 - překročen rozsah časovače během poslední aktivace

S0.5 (OC) - kaskádování časovače
1 - překročen rozsah časovače v tomto cyklu

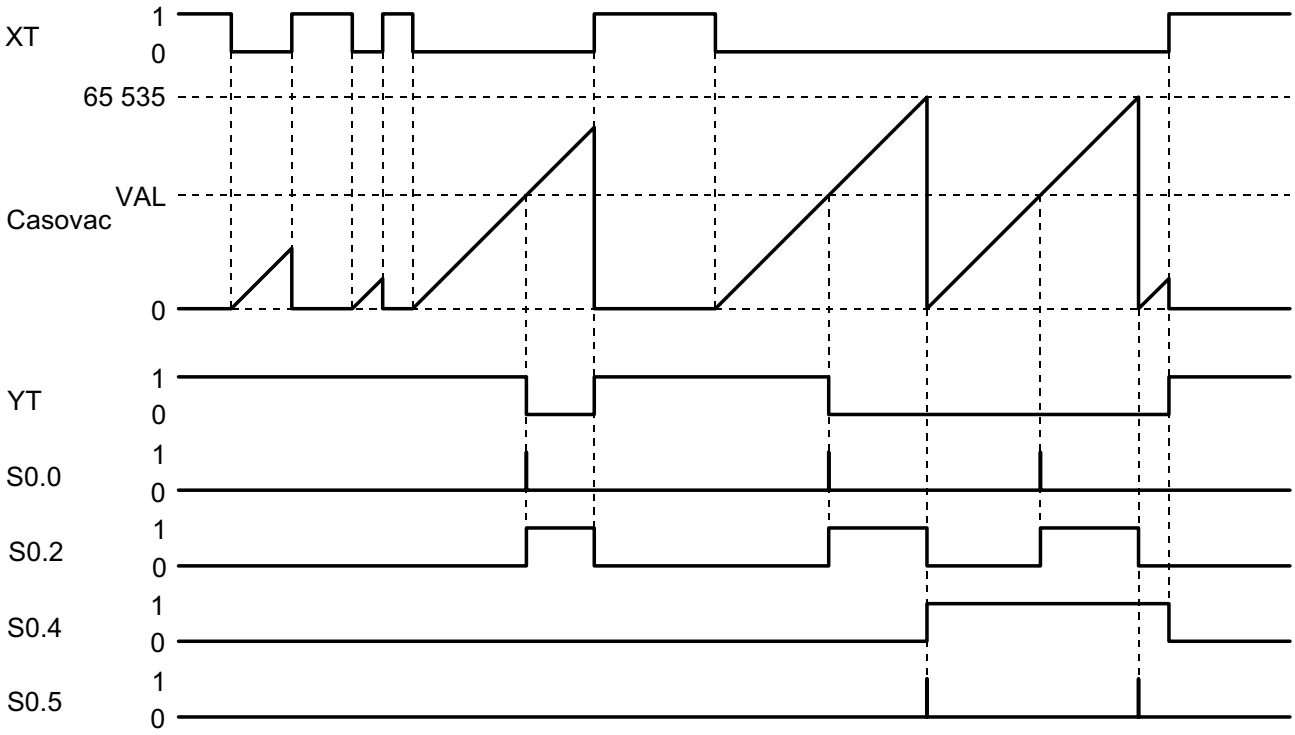
Příklady

```
#reg word casovac
#def XT   %X0.5
#def YT   %Y0.2
#def VAL  5
#def sek  2
;
P 0
      LD    XT
      LD    VAL
      TON   casovac.sek
      WR    YT
E 0
```



Časový diagram časovače TON

```
#reg word casovac
#def XT   %X0.5
#def YT   %Y0.2
#def VAL  5
#def sek  2
;
P 0
      LD    XT
      LD    VAL
      TOF   casovac.sek
      WR    YT
E 0
```



Časový diagram časovače TOF

RTO Integrující časovač

Instrukce	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
RTO						XT	RT	VAL	TIM						YC	RT	YT	TIM

XT - řídící proměnná (typ bit)
RT - nulovací proměnná (typ bit)
VAL - číselná hodnota předvolby (typ word)
*TIM - číselná hodnota časovače (typ word) - jednotky dané parametrem **k***
YC - přetečení přes maximální rozsah (typ bit)
YT - výstupní proměnná, výsledek porovnání aktuální hodnoty časovače s předvolbou (typ bit)

Operandy

		word
RTO	R.k	B D S M

k - kód časové jednotky (není-li zadán, bere se k = 0)
k = 0 - 10 ms, 1 - 100 ms, 2 - 1 s, 3 - 10 s

Funkce

RTO - integrující časovač

Popis

Je-li nulovací proměnná RT = log.1, je časovač pasivní. Pasivní časovač nuluje výstup YT na vrcholu zásobníku A0, přenos z časovače YC a jsou nulovány i příznaky S0.

Je-li nulovací proměnná RT = log.0 a řídící proměnná XT = log.1, je časovač aktivní. Aktivní časovač aktualizuje časový údaj a na vrchol zásobníku A0 ukládá výsledek porovnání s předvolbou. Není-li dosaženo předvolby, je YT = log.0. Je-li předvolba dosažena nebo překročena, je YT = log.1.

Přetečení rozsahu časovače vyvolá nastavení bitů S0.4 a S0.5. Bit S0.5 je kopírován do všech bitů vrstvy A2 (přenos YC). Je roven log.1 jen v cyklu, ve kterém k přetečení došlo.

Je-li nulovací proměnná RT = log.0 a řídící proměnná XT = log.0, je časovač v čekacím stavu. Časovač v čekacím stavu nečasuje, ani se nenuluje, provádí se však porovnání s předvolbou a nastavení příznaků v S0.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	-	OC	OV	-	≤	CO	ZR

S0.0 (ZR) - dosažení předvolby
1 - předvolba dosažena v tomto cyklu
S0.1 (CO) - překročení předvolby
1 - předvolba překročena
S0.2 (≤) - logický součet S0.0 OR S0.1
1 - předvolba dosažena nebo překročena
S0.4 (OV) - překročení maximálního rozsahu časovače
1 - překročen rozsah časovače během poslední aktivace
S0.5 (OC) - kaskádování časovače
1 - překročen rozsah časovače v tomto cyklu

Poznámka

Nad jedním objektem smí být provozován jediný typ instrukce časovače s jedinou časovou jednotkou. Při jakékoliv změně typu instrukce nebo časové jednotky se provede inicializace - časovač se vynuluje.

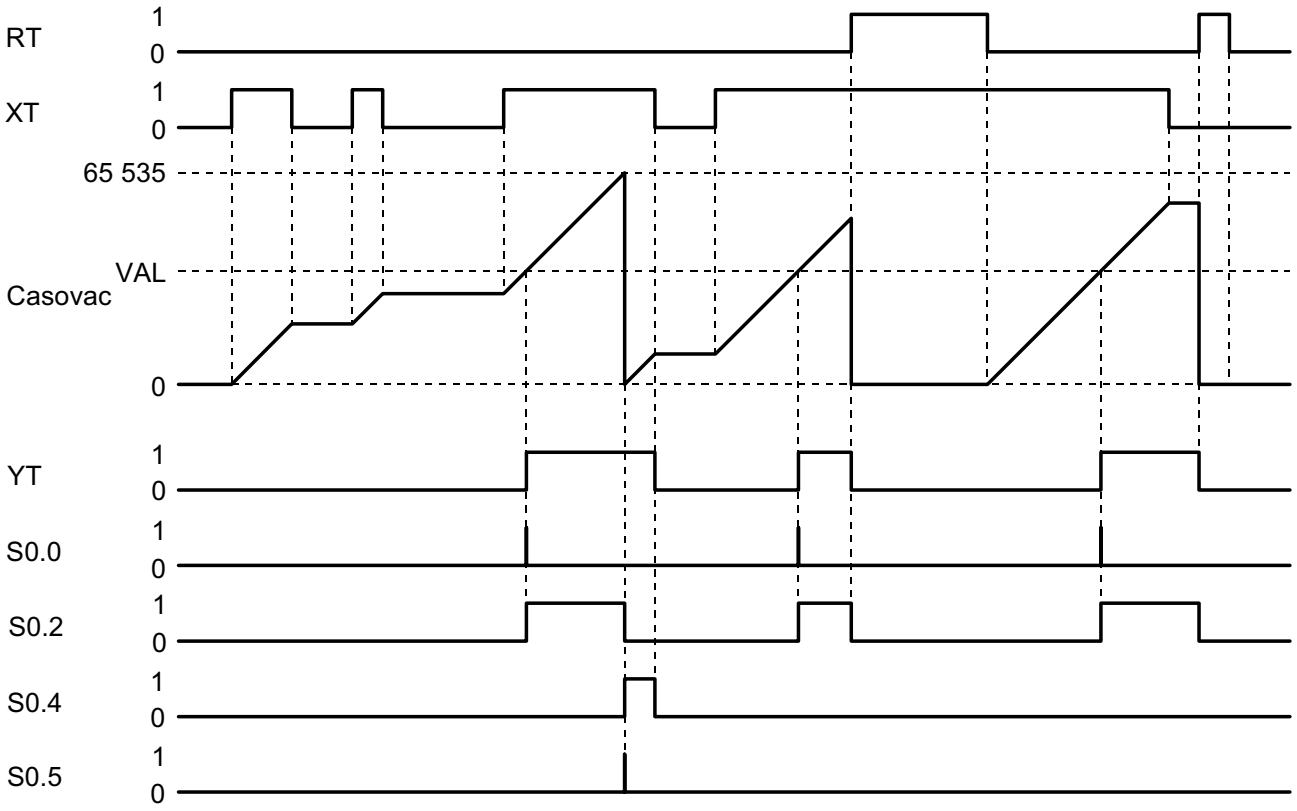
Nad jedním objektem smí být aktivní jen jediná instrukce časovače. Časoměrné systémové proměnné jsou aktualizovány pouze při otočce cyklu (čas běží po skocích). Během cyklu mají stále stejnou hodnotu, takže je lhostejné, na kterém místě v programu je instrukce časovače umístěna. Pokud však dojde v jednom cyklu k vynechání instrukce časovače, nedojde v následující otočce cyklu k jeho aktualizaci - časovač přestane časovat. Časovat začne opět průchodem programu instrukcí časovače, ovšem s tím, že jeho hodnota je poznamenaná odpovídajícím časovým výpadkem.

Pokud má předvolba VAL hodnotu 0, je výstupní proměnná YT stále log.1, pouze v případě RT = log.1, je YT = log.0. Stav systémových příznaků S0 není definován.

Pokud je časová jednotka **k** přibližně stejná nebo menší než doba cyklu PLC, je funkce příznaků S0.0 a S0.5 nespolehlivá (hodnota časovače narůstá po větších skocích a předvolba, resp. rozsah časovače, je rovnou překročena, takže její dosažení nemusí být detekováno). Příznaky S0.0 a S0.5 lze nahradit testem náběžné hrany příznaků S0.2 a S0.4.

Příklad

```
#reg word casovac
#def XT   %X0.5
#def RT   %X0.6
#def YT   %Y0.2
#def VAL  5
#def sek  2
;
P 0
      LD    XT
      LD    RT
      LD    VAL
      RTO   casovac.sek
      WR    YT
E 0
```



Časový diagram časovače RTO

IMP Impulz

Instrukce	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
IMP							XT	VAL	TIM							XT	YT	TIM

XT - řídící proměnná (typ bit)
VAL - číselná hodnota předvolby (typ word)
*TIM - číselná hodnota časovače (typ word) - jednotky dané parametrem **k***
YT - výstupní proměnná, výsledek porovnání aktuální hodnoty časovače s předvolbou (typ bit)

Operandy

		word
IMP	R.k	B D S M

k - kód časové jednotky (není-li zadán, bere se k = 0)
k = 0 - 10 ms, 1 - 100 ms, 2 - 1 s, 3 - 10 s

Funkce

IMP - generátor impulzů od náběžné hrany

Popis

Po inicializaci je časovač pasivní. Pasivní časovač je vynulován a jsou nulovány i příznaky S0.4 a S0.5. Pokud je předvolba nenulová, jsou vynulovány příznaky S0.2 až S0.0.

Časovač je aktivní od příchodu první náběžné hrany proměnné XT (přechod z log.0 na log.1). Aktivní časovač aktualizuje časový údaj a na vrchol zásobníku A0 ukládá výsledek porovnání s předvolbou. Není-li dosaženo předvolby, je YT = log.1 (samé jedničky). Je-li předvolba dosažena, je YT = log.0, časovač se opět stává pasivním a čeká na novou náběžnou hranu proměnné XT.

Délku impulzu není možno měnit. Časovač lze předčasně zastavit jedině vyvoláním inicializace (restart systému nebo změna režimu časovače - viz poznámka).

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	-	-	-	-	≤	-	ZR

S0.0 (ZR) - dosažení předvolby
1 - předvolba dosažena v tomto cyklu
S0.2 (≤) - logický součet S0.0 OR S0.1

Poznámka

Nad jedním objektem smí být provozován jediný typ instrukce časovače s jedinou časovou jednotkou. Při jakékoliv změně typu instrukce nebo časové jednotky se provede inicializace - časovač se vynuluje.

Nad jedním objektem smí být aktivní jen jediná instrukce časovače. Časoměrné systémové proměnné jsou aktualizovány pouze při otočce cyklu (čas běží po skocích). Během cyklu mají stále stejnou hodnotu, takže je lhostejné, na kterém místě v programu je instrukce časovače umístěna. Pokud však dojde v jednom cyklu k vynechání instrukce časovače, nedojde v následující otočce cyklu k jeho aktualizaci - časovač přestane

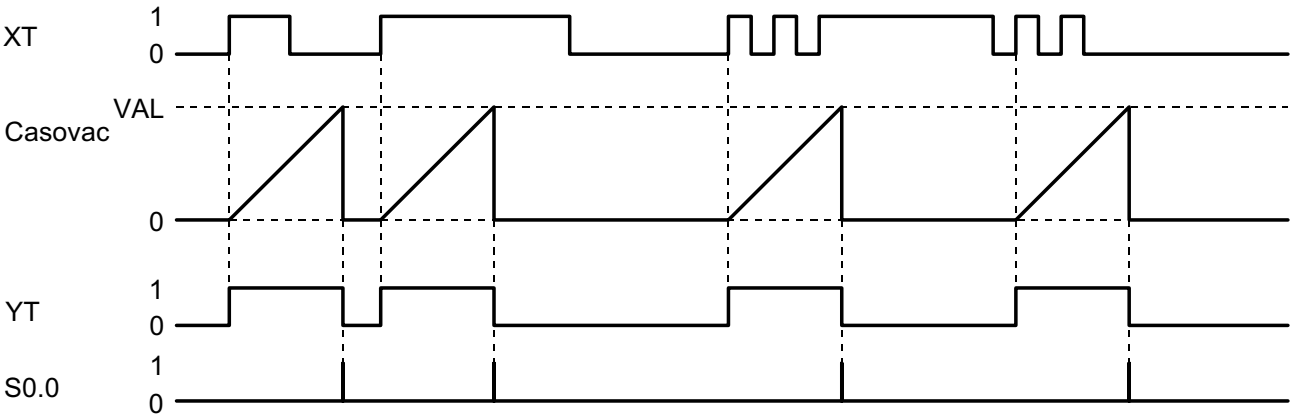
časovat. Časovat začne opět průchodem programu instrukcí časovače, ovšem s tím, že jeho hodnota je poznamenaná odpovídajícím časovým výpadkem.

Pokud má předvolba VAL hodnotu 0, je výstupní proměnná YT stále log.0 (impulz nulové délky). Stav systémových příznaků S0 není definován.

Pokud je časová jednotka **k** přibližně stejná nebo menší než doba cyklu PLC, je funkce příznaku S0.0 nespolehlivá (hodnota časovače narůstá po větších skocích a předvolba je rovnou překročena, takže její dosažení nemusí být detekováno). Příznak S0.0 lze nahradit příznakem S0.2, který detekuje i překročení předvolby.

Příklad

```
#reg word casovac
#def XT   %X0.5
#def YT   %Y0.2
#def VAL  5
#def sek  2
;
P 0
      LD    XT
      LD    VAL
      IMP   casovac.sek
      WR    YT
E 0
```



Časový diagram časovače IMP

STE Krokový řadič

Instrukce	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
STE								VEC	STP0								VAL	STP

VEC - podmínkový vektor - soubor podmínek pro rotaci stavové masky (typ podle operandu)
STP0 - stav krokového řadiče před instrukcí
VAL - výsledná hodnota stavové masky (typ word)
STP - aktuální stav krokového řadiče

Operandy

		word
STE	R	B D S M

k - kód časové jednotky (není-li zadán, bere se k = 0)
k = 0 - 10 ms, 1 - 100 ms, 2 - 1 s, 3 - 10 s

Funkce

STE - krokový (sekvenční) řadič

Popis

Instrukce **STE** provádí následující komplex činností. Dolní byte adresovaného registru (nižší adresa) má význam stavu krokového řadiče. Význam mají pouze spodní čtyři bity (hodnoty 0 až 15). Ostatní čtyři jsou ignorovány. Hodnota spodních čtyř bitů je převedena na masku 1 z 16 (stavová maska):

stav (bity 3 - 0)	bitová maska
0	00000000 00000001
1	00000000 00000010
:	:
14	01000000 00000000
15	10000000 00000000

Po převedení čísla stavu na stavovou masku se testuje, zda podmínkový vektor má na pozici odpovídající pozici jedničky v masce i jedničkový podmínkový bit. Pokud ano, pak se zvýší číslo stavu v dolním bytu adresovaného registru o 1, stavová maska se posune o 1 bit vlevo v kruhu (hodnota nejvyššího bitu přechází do nejnižšího) a nastaví se příznak S1.0. Pokud došlo k přenosu (změna stavu z 15 na 0), nastaví se i příznak S1.1. Pokud odpovídající bit v podmínkovém vektoru je nulový, pak se nemění ani číslo stavu ani stavová maska a registr S1 = 0. Aktualizovaná hodnota stavové masky je zapsána na vrchol zásobníku A0. Aktualizované číslo stavu je uloženo do dolního bytu adresovaného registru (nižší adresa). Hodnota není korigována modulo 16, ale nabývá velikosti až 255. Do horního bytu adresovaného registru (vyšší adresa) je uložen obsah horních čtyř bitů z dolního bytu (hodnota 0 až 15) ve významu přenosu nebo přetečení. Je zde tedy uložen počet „protočení“ stavové masky.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	OM	ST

S1.0 (ST) - přechod v řadiči
1 - došlo ke změně stavu a stavové masky

S1.1 (OM) - přenos v řadiči
1 - došlo k „protočení“ stavové masky (jednička přešla z nejvyššího bitu na bit 0)

Poznámka

Pokud je podmínkový vektor stále nulový, pracuje instrukce **STE** jako dekodér číslo → maska „1 z n“.

Obsahuje-li podmínkový vektor samé jedničky, provádí instrukce **STE** rotaci masky a současnou inkrementaci čísla stavu.

4. ARITMETICKÉ INSTRUKCE

ADD	**Sčítání s přenosem**
ADX, ADL	**Sčítání**

Instrukce	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								op.
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	
ADD								a	b								$a+b+CI$	b
ADD bez op.							a	b		b	A7	A6	A5	A4	A3	A2	$a+b+CI$	
ADX [B W]								a	b								$a+b$	b
ADX, ADL [L]							a		b								$a+b$	b
ADL bez op.					a		b			b		A7	A6	A5	A4		$a+b$	

Operandy

		byte	word	long
ADD	X Y S D R		B D S M	
ADD	#		B D S M	
ADD	bez operandu		B D S M	
ADX	X Y S D R	B D	B D	B D
ADL	#			B D
ADL	bez operandu			B D

Funkce

ADD - sčítání s přenosem
ADX - sčítání
ADL - sčítání

Popis

Instrukce **ADD** s operandem přičte k vrcholu zásobníku A0 obsah zadaného operandu a obsah přenosu zdola (CI). Obsah ostatních vrstev zásobníku se nemění. Instrukce nastavuje příznaky v registru S0.

Instrukce **ADX** a **ADL** s operandem přičte k vrcholu zásobníku obsah zadaného operandu. Obsah ostatních úrovní zásobníku se nemění. Instrukce nenastavuje žádné příznaky.

Instrukce **ADD** bez operandu posune zásobník o jednu úroveň zpět a k vrcholu zásobníku (původně A1) přičte původní obsah vrcholu A0 a obsah přenosu zdola (CI). Instrukce nastavuje příznaky v registru S0.

Instrukce **ADL** bez operandu sečte obsahy dvojvrstev A23 a A01. Pak posune zásobník o dvě úrovně zpět a na nový vrchol zásobníku A01 zapíše výsledek. Instrukce nenastavuje žádné příznaky.

Příznaky instrukce ADD

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	-	-	-	CI	≤	CO	ZR

S0.0 (ZR) - nulovost výsledku
1 - výsledek je 0
S0.1 (CO) - výstupní přenos
1 - výsledek překročil maximální hodnotu 65 535
S0.2 (≤) - logický součet S0.0 OR S0.1
S0.3 (CI) - vstupní přenos určený ke kaskádování instrukcí, před instrukcí **ADD** je nutné jej nastavit na hodnotu přenosu z předchozí kaskády (přesun CO → CI), jinak se instrukce provede bez přenosu zdola (příznak CI je po instrukci **ADD** vynulován)

Příklad

Realizace výrazu $d = a + (b - c)$

```
#reg long va, vb, vc, vd
;
P 0
      LD    vb
      SUX   vc          ;(b - c)
      ADX   va          ;a + ( )
      WR    vd
E 0
```

SUB	**Odčítání s přenosem**
SUX, SUL	**Odčítání**

Instrukce	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								op.
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	
SUB								a	b								$a-b-CI$	b
SUB bez op.							a	b		b	A7	A6	A5	A4	A3	A2	$a-b-CI$	
SUX [B W]								a	b								$a-b$	b
SUX, SUL [L]							a		b								$a-b$	b
SUL bez op.					a		b			b		A7	A6	A5	A4		$a-b$	

Operandy

		byte	word	long
SUB	X Y S D R		B D S M	
SUB	#		B D S M	
SUB	bez operandu		B D S M	
SUX	X Y S D R	B D	B D	B D
SUL	#			B D
SUL	bez operandu			B D

Funkce

SUB - odčítání s přenosem
SUX - odčítání
SUL - odčítání

Popis

Instrukce **SUB** s operandem odečte od vrcholu zásobníku A0 obsah zadaného operandu a obsah přenosu zdola (CI). Obsah ostatních vrstev zásobníku se nemění. Instrukce nastavuje příznaky v registru S0.

Instrukce **SUX** a **SUL** s operandem odečte od vrcholu zásobníku obsah zadaného operandu. Obsah ostatních úrovní zásobníku se nemění. Instrukce nenastavuje žádné příznaky.

Instrukce **SUB** bez operandu posune zásobník o jednu úroveň zpět a od vrcholu zásobníku (původně A1) odečte původní obsah vrcholu A0 a obsah přenosu zdola (CI). Instrukce nastavuje příznaky v registru S0.

Instrukce **SUL** bez operandu odečte obsah dvojvrstvy A01 od obsahu dvojvrstvy A23. Pak posune zásobník o dvě úrovně zpět a na nový vrchol zásobníku A01 zapíše výsledek. Instrukce nenastavuje žádné příznaky.

Příznaky instrukce SUB

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	-	-	-	CI	≤	CO	ZR

S0.0 (ZR) - nulovost výsledku
1 - výsledek je 0, shodná hodnota obou parametrů
S0.1 (CO) - výstupní přenos
1 - výsledek je záporný, odečítání většího čísla od menšího
S0.2 (≤) - logický součet S0.0 OR S0.1

S0.3 (CI) - vstupní přenos určený ke kaskádování instrukcí, před instrukcí **SUB** je nutné jej nastavit na hodnotu přenosu z předchozí kaskády (přesun CO → CI), jinak se instrukce provede bez přenosu zdola (příznak CI je po instrukci **SUB** vynulován)

Příklad

Realizace výrazu $d = a + (b - c)$

```
#reg long va, vb, vc, vd
;
P 0
      LD    vb
      SUB   vc          ;(b - c)
      ADD   va          ;a + ( )
      WR    vd
E 0
```

MUL, MUD Násobení

Instrukce	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
MUL								a	b								$a \cdot b$	b
MUL bez op.							a	b		b	A7	A6	A5	A4	A3	A2	$a \cdot b$	
MUD								a	b	A6	A5	A4	A3	A2	A1	$a \cdot b$		b
MUD bez op.							a	b								$a \cdot b$		

Operandy

		byte	word
MUL	X Y S D R	B D S M	
MUL	#	B D S M	
MUL	bez operandu	B D S M	
MUD	X Y S D R		B D
MUD	#		B D
MUD	bez operandu		B D

Funkce

MUL - násobení (byte x byte = word)
MUD - násobení (word x word = long)

Popis

Instrukce **MUL** s operandem vynásobí dolní byte vrcholu zásobníku A0 obsahem zadaného operandu. Výsledek uloží na vrchol zásobníku A0. Obsah ostatních úrovní zásobníku se nemění.

Instrukce **MUD** s operandem vynásobí obsah vrcholu zásobníku A0 obsahem zadaného operandu. Pak posune zásobník o jednu úroveň vpřed a výsledek uloží na vrchol zásobníku A01.

Instrukce **MUL** bez operandu vynásobí dolní byte vrstvy A1 s dolním bytem vrstvy A0. Pak posune zásobník o jednu úroveň zpět a na nový vrchol zásobníku A0 zapíše výsledek.

Instrukce **MUD** bez operandu vynásobí obsah vrstvy A1 s obsahem vrstvy A0. Na vrchol zásobníku A01 zapíše výsledek. Obsah ostatních úrovní zásobníku se nemění.

Příklad

Realizace výrazu $d = a + (b \cdot c)$

```
#reg byte vb, vc
#reg word va, vd
;
P 0
     LD   vb
     MUL  vc          ;(b . c)
     ADX  va          ;a + ( )
     WR   vd
E 0
```

DIV, DID Dělení se zbytkem

Instrukce	Vstupní parametry									Výsledek									
	zásobník								ope-	zásobník									op.
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0		
DIV								*a*	*b*								*M*	*a* / *b*	*b*
DIV bez op.							*a*	*b*		*b*	A7	A6	A5	A4	A3	A2	*M*	*a* / *b*	
DID							*a*		*b*	A6	A5	A4	A3	A2	*M*	*a* / *b*			*b*
DID bez op.						*a*		*b*							*M*	*a* / *b*			

M - *zbytek dělení (a % b)*

Operandy

		byte	word / long
DIV	X Y S D R	B D S M	
DIV	#	B D S M	
DIV	bez operandu	B D S M	
DID	X Y S D R		B D
DID	#		B D
DID	bez operandu		B D

Funkce

DIV - dělení se zbytkem (byte / byte = byte)
DID - dělení se zbytkem (long / word = long)

Popis

Instrukce **DIV** s operandem vydělí dolní byte vrcholu zásobníku A0 obsahem zadaného operandu. Celočíselný podíl ukládá v dolním bytu A0, zbytek ukládá v horním bytu A0. Obsah ostatních úrovní zásobníku se nemění.

Instrukce **DID** s operandem vydělí obsah vrcholu zásobníku A01 obsahem zadaného operandu. Pak posune zásobník o jednu úroveň vpřed a na nový vrchol zásobníku A01 zapíše celočíselný podíl, zbytek uloží do vrstvy A2.

Instrukce **DIV** bez operandu vydělí dolní byte vrstvy A1 dolním bytem vrstvy A0. Pak posune zásobník o jednu úroveň zpět a na nový vrchol zásobníku zapíše celočíselný podíl do dolního bytu A0, zbytek do horního bytu A0.

Instrukce **DID** bez operandu vydělí obsah dvojvrstvy A12 obsahem vrstvy A0. Na vrchol zásobníku A01 pak zapíše výsledek, zbytek uloží do vrstvy A2. Obsah ostatních úrovní zásobníku se nemění.

Pokud dojde k dělení nulou, nastaví se bit S0.0 na log.1 a do registru S34 se zapíše chyba 16. Vrchol zásobníku obsahuje samé jedničky (maximální hodnota podle formátu).

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	-	-	-	-	-	-	ZR

S0.0 (ZR) - dělení nulou
1 - došlo k dělení nulou, výsledek je neplatný

S34 = 16 ($10) chyba dělení nulou

Příklad

Realizace výrazu $d = a + \frac{b}{c}$

```
#reg word vc
#reg long va, vb, vd
;
P 0
      LD    vb
      DID   vc          ;(b / c)
      ADX   va          ;a + ( )
      WR    vd
E 0
```

INR	**Inkrementace**
DCR	**Dekrementace**

Instrukce	Vstupní parametry									Výsledek								
	zásobník								op.	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	rand
INR									a									$a+1$
INR bez op.								a									$a+1+CI$	
DCR									a									$a-1$
DCR bez op.								a									$a-1-CI$	

Operandy

		byte	word	long
INR	X Y S R	B D	B D	B D
INR	bez operandu		B D S M	
DCR	X Y S R	B D	B D	B D
DCR	bez operandu		B D S M	

Funkce

INR - zvýšení obsahu o 1
DCR - snížení obsahu o 1

Popis

Instrukce **INR** s operandem zvýší obsah operandu o 1. Obsah zásobníku se nemění. Instrukce nenastavuje žádné příznaky.

Instrukce **INR** bez operandu přičte k obsahu vrcholu zásobníku hodnotu 1 a obsah přenosu zdola (CI). Nastavuje příznaky výsledku. Úroveň zásobníku se nemění.

Instrukce **DCR** s operandem sníží obsah operandu o 1. Obsah zásobníku se nemění. Pokud je obsah operandu po odečtení 1 roven 0, je nastaven příznak S0.0 (ZR). Ve spojení s instrukcemi **JZ** a **JNZ** lze tak snadno realizovat programový cyklus.

Instrukce **DCR** bez operandu odečte od obsahu vrcholu zásobníku 1 a obsah přenosu zdola (CI). Nastavuje příznaky výsledku. Úroveň zásobníku se nemění.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	-	-	-	CI	≤	CO	ZR

S0.0 (ZR) - nulovost výsledku (nenastavuje instrukce **INR** s operandem)
1 - výsledek je 0

S0.1 (CO) - výstupní přenos (jen pro bezoperandové instrukce)
1 - výsledek překročil maximální hodnotu 65 535

S0.2 (≤) - logický součet S0.0 OR S0.1(jen pro bezoperandové instrukce)

S0.3 (CI) - vstupní přenos určený ke kaskádování bezoperandových instrukcí, před instrukcí je nutné jej nastavit na hodnotu přenosu z předchozí kaskády (přesun CO → CI), jinak se instrukce provede bez přenosu (příznak CI je po instrukci vynulován)

Příklad

Proveďme 5x stejný úsek programu

```
#reg byte Pocitadlo
;
P 0
     LD   5
     WR   Pocitadlo
smycka:                  ;začátek cyklu
     :                   ;opakovaný program
     :
     DCR  Pocitadlo      ;test počtu opakování
     JNZ  smycka
                         ;konec cyklu, Pocitadlo = 0
E 0
```

EQ	**Porovnání (rovnost)**
LT	**Porovnání (menší než)**
GT	**Porovnání (větší než)**

Instrukce	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								op.
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	
EQ								*a*	*b*								$a = b + CI$?	*b*
EQ bez op.							*a*	*b*		*b*	A7	A6	A5	A4	A3	A2	$a = b + CI$?	
LT								*a*	*b*								$a < b + CI$?	*b*
LT bez op.							*a*	*b*		*b*	A7	A6	A5	A4	A3	A2	$a < b + CI$?	
GT								*a*	*b*								$a > b + CI$?	*b*
GT bez op.							*a*	*b*		*b*	A7	A6	A5	A4	A3	A2	$a > b + CI$?	

Operandy

		word
EQ	X Y S D R	B D S M
EQ	#	B D S M
EQ	bez operandu	B D S M E
LT	X Y S D R	B D S M
LT	#	B D S M
LT	bez operandu	B D S M
GT	X Y S D R	B D S M
GT	#	B D S M
GT	bez operandu	B D S M

Funkce

EQ - porovnání hodnot s testem na rovnost
LT - porovnání hodnot s testem na menší než ...
GT - porovnání hodnot s testem na větší než ...

Popis

Instrukce **EQ**, **LT**, **GT** s operandem jsou si vnitřně rovnocenné. Porovnají obsah vrcholu zásobníku s operandem, nastaví příznaky v S0 a potom zapíší na vrchol zásobníku pravdivostní výsledek testu - log.1 (samé jedničky), pokud je podmínka testu splněna, nebo log.0, pokud podmínka splněna není.

Porovnání se provádí vnitřním odečtením, které je rovnocenné instrukci **SUB**. Od vrcholu A0 je odečten operand a vstupní přenos CI a jsou nastaveny příznaky výsledku. Hodnota rozdílu je však přepsána výsledkem testu.

Instrukce **EQ**, **LT**, **GT** bez operandu jsou si vnitřně rovnocenné. Porovnají obsah vrstvy A1 s obsahem vrcholu zásobníku A0, nastaví příznaky v S0, posunou zásobník o jednu úroveň zpět a potom zapíší na nový vrchol zásobníku pravdivostní výsledek testu - log.1 (samé jedničky), pokud je podmínka testu splněna, nebo log.0, pokud podmínka splněna není.

Porovnání se provádí vnitřním odečtením, které je rovnocenné instrukci **SUB**. Od vrstvy A1 je odečtena vrstva A0 a vstupní přenos CI a jsou nastaveny příznaky výsledku. Hodnota rozdílu je však přepsána výsledkem testu.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	-	-	-	-	≤	CO	ZR

S0.0 (ZR) - porovnání na shodu
0 - platí $a \neq b$
1 - platí $a = b$

S0.1 (CO) - výstupní přenos
0 - platí $a \geq b$
1 - platí $a < b$

S0.2 (≤) - logický součet S0.0 OR S0.1
0 - platí $a > b$
1 - platí $a \leq b$

CMP, CML Porovnání

Instrukce	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
CMP [B W]								*a*	*b*								*a*	*b*
CMP, CML [L]							*a*		*b*							*a*		*b*
CMP bez op.							*a*	*b*								*a*	*b*	
CML bez op.					*a*		*b*							*a*		*b*		

Operandy

		byte	word	long
CMP	X Y S D R	B D	B D	B D
CMP	#		B D	
CMP	bez operandu		B D	
CML	#			B D
CML	bez operandu			B D

Funkce

CMP - porovnání hodnot
CML - porovnání hodnot (long)

Popis

Instrukce **CMP**, **CML** s operandem porovnají obsah vrcholu zásobníku s operandem a nastaví příznaky v S0.

Instrukce **CMP** bez operandu porovná obsah vrstvy A1 s obsahem vrcholu zásobníku A0 a nastaví příznaky v S0.

Instrukce **CML** bez operandu porovná obsah dvojvrstvy A23 s obsahem vrcholu zásobníku A01 a nastaví příznaky v S0.

Všechny tyto instrukce nemění obsah zásobníku. Pro vyhodnocení příznaků nastavených v registru S0 lze s výhodou využít skokové instrukce **JZ**, **JNZ**, **JC** a **JNC**.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	-	-	-	-	≤	CO	ZR

S0.0 (ZR) - porovnání na shodu
0 - platí $a \neq b$
1 - platí $a = b$

S0.1 (CO) - výstupní přenos
0 - platí $a \geq b$
1 - platí $a < b$

S0.2 (≤) - logický součet S0.0 OR S0.1
0 - platí $a > b$
1 - platí $a \leq b$

BIN , BIL Převod z BCD formátu do binárního
BCD, BCL Převod z binárního formátu do BCD

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
BIN								NBCD									NBIN	
BIL							NBCD			-	A7	A6	A5	A4	A3	NBIN		
BCD								NBIN									NBCD	
BCL							NBIN			A6	A5	A4	A3	A2		NBCD		

NBCD - desítkové číslo ve formátu BCD
*(**BIN** - rozsah 0 až 9999; **BCD** - čtyři číslice; **BIL**, **BCL** - rozsah 0 až 4 294 967 295)*
*NBIN - číslo v binárním formátu (**BIN**, **BCD** - word; **BIL**, **BCL** - long)*
NBCD5- nejvyšší pátá číslice desítkového čísla v bitech S0.6, .5, .4

Operandy

		word	long
BIN	bez operandu	B D S M	
BIL	bez operandu		B D
BCD	bez operandu	B D S M	
BCL	bez operandu		B D

Funkce

BIN - převod desítkového čísla v BCD kódu do binárního formátu (word)
BIL - převod desítkového čísla v BCD kódu do binárního formátu (long)
BCD - převod čísla v binárním formátu na desítkové v BCD kódu (word)
BCL - převod čísla v binárním formátu na desítkové v BCD kódu (long)

Popis

Instrukce **BIN** chápe vrchol zásobníku A0 jako čtyřmístné desítkové číslo ve formátu BCD (každá číslice je kódována dvojkově na čtyřech bitech), převádí jej do dvojkové soustavy a ukládá zpět do A0. Hodnoty ostatních vrstev zásobníku se nemění. Číselný rozsah převáděných čísel je 0 až 9999.

Instrukce **BIL** zpracovává vrstvy A2, A1, A0 jako desetimístné desítkové číslo ve formátu BCD, převede jej do dvojkové soustavy, posune zásobník o jednu úroveň zpět a výsledek uloží na vrchol zásobníku A01. Číselný rozsah převáděných čísel je 0 až 4 294 967 295.

Instrukce **BCD** chápe vrchol zásobníku A0 jako dvojkové číslo šířky word, převede jej na desítkové číslo v BCD kódu a jeho dolní čtyři číslice uloží na vrchol zásobníku A0. Nejvyšší číslici uloží do registru S0 na bity S0.6 až S0.4. Hodnoty ostatních úrovní zásobníku se nemění. Rozsah převáděných čísel je 0 až 65 535.

Instrukce **BCL** zpracovává vrchol zásobníku A01 jako dvojkové číslo šířky long, převede jej na desítkové číslo v BCD kódu, posune zásobník o jednu úroveň vpřed a výsledek uloží do vrstev A2, A1 a A0. Rozsah převáděných čísel je 0 až 4 294 967 295.

Příznaky instrukce BCD

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	D5.2	D5.1	D5.0	-	-	-	-

S0.6 až S0.4 (D5.2 až D5.0) - nejvyšší číslice převedeného čísla v BCD kódu (max. 6)

Příklady

Převod čísla v BCD na binární číslo

```
#reg word Deset, Binar
;
P 0
      LD    Deset
      BIN
      WR    Binar
E 0

#reg byte DesetH      ;nejvyšší dvě číslice (10. a 9.)
#reg long DesetL      ;dalších 8 číslic (8. až 1.)
#reg long Binar
;
P 0
      LD    DesetH
      LD    DesetL
      BIL
      WR    Binar
E 0
```

Převod binárního čísla do BCD

```
#reg byte DesetH      ;nejvyšší číslice (5.)
#reg word DesetL      ;další 4 číslice (4. až 1.)
#reg word Binar
;
P 0
      LD    Binar
      BCD
      WR    DesetL
      LD    %S0
      ROL   12
      AND   $0007
      WR    DesetH
E 0

#reg byte DesetH      ;nejvyšší dvě číslice (10. a 9.)
#reg long DesetL      ;dalších 8 číslic (8. až 1.)
#reg long Binar
;
P 0
      LD    Binar
      BCL
      WR    DesetL
      POP   2
      WR    DesetH
E 0
```

5. OPERACE SE ZÁSOBNÍKY

POP	**Posun zásobníku**

Instrukce	**Vstupní parametry**									**Výsledek**								
	zásobník								ope-rand	zásobník								ope-rand
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
POP									*n*		→	→	→	*n x*	→	→		*n*

Operandy

		word
POP	n	B D S M

n - počet posunutí zásobníku (–7 až 7)

Funkce

POP - n-násobný zpětný posun zásobníku

Popis

Instrukce **POP** posunuje zásobník o zadaný počet úrovní zpět. Instrukce provádí zpětnou rotaci zásobníku, takže je kdykoli možno se k hodnotě vysunuté z vrcholu A0 opět vrátit. Pokud potřebujeme zásobník posunout vpřed, zadáme počet otočení se znaménkem –.

CHG, CHGS	**Výměna aktivního zásobníku**
NXT	**Aktivace následujícího zásobníku**
PRV	**Aktivace předchozího zásobníku**

Operandy

		long
CHG	n	C
CHGS	n	C
NXT	bez operandu	C
PRV	bez operandu	C

n - označení vybraného zásobníku (0 až 7)

Funkce

CHG - aktivace vybraného zásobníku
CHGS - aktivace vybraného zásobníku se zálohováním S0 a S1
NXT - aktivace následujícího zásobníku v řadě se zálohováním S0 a S1
PRV - aktivace předcházejícího zásobníku v řadě se zálohováním S0 a S1

Popis

Instrukce **CHG** aktivuje vybraný zásobník určený parametrem n, který nabývá hodnot 0 až 7, což přestavuje zásobníky A až H. Instrukce **CHGS** navíc provádí i současné ukládání a vybírání stavu systémových registrů S0 a S1. Hodnoty těchto registrů jsou uloženy u právě opuštěného zásobníku a registry S0 a S1 v zápisníku jsou přepsány hodnotami, které byly uloženy u právě aktivovaného zásobníku.

Instrukce **NXT** a **PRV** aktivují zásobník podle následující tabulky:

Aktivní zásobník před instrukcí	Aktivní zásobník po instrukci NXT	Aktivní zásobník po instrukci PRV
A (0)	B (1)	H (7)
B (1)	C (2)	A (0)
C (2)	D (3)	B (1)
D (3)	E (4)	C (2)
E (4)	F (5)	D (3)
F (5)	G (6)	E (4)
G (6)	H (7)	F (5)
H (7)	A (0)	G (6)

Instrukce **NXT** a **PRV** provádějí ukládání a vybírání stavu systémových registrů S0 a S1.

Příznaky

Instrukce **CHGS**, **NXT** a **PRV** ukládají hodnoty S0 a S1 k právě opouštěnému zásobníku a registry S0 a S1 přepíší hodnotami uloženými u právě aktivovaného zásobníku.

LAC	**Načtení hodnoty z vrcholu vybraného zásobníku**
WAC	**Zápis hodnoty na vrchol vybraného zásobníku**

Instrukce	**Vstupní parametry**									**Výsledek**								
	zásobník								ope-rand	zásobník								ope-rand
LAC	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
									n	A6	A5	A4	A3	A2	A1	A0	*a*	*n*
	m7	m6	m5	m4	m3	m2	m1	m0		m7	m6	m5	m4	m3	m2	m1	m0	
								a		*a*	m7	m6	m5	m4	m3	m2	m1	
WAC	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
								a	*n*								*a*	*n*
	m7	m6	m5	m4	m3	m2	m1	m0		m7	m6	m5	m4	m3	m2	m1	m0	
										m6	m5	m4	m3	m2	m1	m0	*a*	

n - *číslo zásobníku (0 až 7)*
m - *označení zásobníku (A až H)*

Operandy

		long
LAC	n	C
WAC	n	C

n - označení vybraného zásobníku (0 až 7)

Funkce

LAC - načtení hodnoty z vrcholu vybraného zásobníku a jeho posun
WAC - zápis hodnoty na vrchol vybraného zásobníku a jeho posun

Popis

Instrukce **LAC** načte hodnotu vrcholu zásobníku určeného parametrem n, který nabývá hodnot 0 až 7, což přestavuje zásobníky A až H, na vrchol aktivního zásobníku. Vůči aktivnímu zásobníku se instrukce chová stejně jako instrukce **LD**, tedy před zápisem hodnoty na jeho vrchol provede posun zásobníku o jednu úroveň vpřed. Vybraný zásobník se po operaci posune o jednu úroveň zpět a na jeho vrcholu je tak připravena nová hodnota k přečtení.

Ve spojení se zapisovací instrukcí **WAC** se vybraný zásobník chová jako odkládací zásobník typu LIFO (last in, first out), tedy hodnota, která se instrukcí **WAC** zapíše jako poslední, se instrukcí **LAC** první načte.

Instrukce **WAC** zapíše hodnotu vrcholu aktivního zásobníku na vrchol zásobníku určeného parametrem n, který nabývá hodnot 0 až 7, což přestavuje zásobníky A až H. Vůči aktivnímu zásobníku se instrukce chová stejně jako instrukce **WR**, tedy nemění jeho obsah. Vůči vybranému zásobníku se instrukce chová stejně jako instrukce **LD**, tedy před zápisem hodnoty na jeho vrchol provede posun zásobníku o jednu úroveň vpřed.

Ve spojení se čtecí instrukcí **LAC** se vybraný zásobník chová jako odkládací zásobník typu LIFO (last in, first out), tedy hodnota, která se instrukcí **WAC** zapíše jako poslední, se instrukcí **LAC** první načte.

Instrukci **WAC** lze použít také pro přípravu parametrů pro instrukce, které zpracovávají více vrstev zásobníku. Pokud hodnoty těchto parametrů získáváme během rozsáhlého programu na jeho různých místech, můžeme je postupně odkládat na vybraný zásobník a pak pouhým přepnutím zásobníků jsou parametry připraveny ke zpracování.

6. INSTRUKCE SKOKŮ A VOLÁNÍ

JMP	**Skok**
JMD	**Skok podmíněný nenulovostí vrcholu zásobníku**
JMC	**Skok podmíněný nulovostí vrcholu zásobníku**
JMI	**Nepřímý skok**

Operandy

JMP	Ln	B D S M E
JMD	Ln	B D S M E
JMC	Ln	B D S M
JMI	bez operandu	B D S M

Funkce

JMP - nepodmíněný skok na návěští L n
JMD - skok na návěští L n podmíněný nenulovostí vrcholu zásobníku A0
JMC - skok na návěští L n podmíněný nulovostí vrcholu zásobníku A0
JMI - nepodmíněný skok na návěští L n, jehož číslo n udává vrchol zásobníku A0

Popis

Instrukce **JMP** nepodmíněně převede program na instrukci **L** n.

Instrukce **JMD** se zachová jako instrukce **JMP** pouze v případě, že vrchol zásobníku A0 není 0 (logický součet OR všech 16 bitů A0 je log.1). Pokud tato podmínka není splněna, je instrukce ignorována a program pokračuje ve vykonávání další bezprostředně následující instrukce.

Instrukce **JMC** se zachová jako instrukce **JMP** pouze v případě, že vrchol zásobníku A0 je 0 (logický součet OR všech 16 bitů A0 je log.0). Pokud tato podmínka není splněna, je instrukce ignorována a program pokračuje ve vykonávání další bezprostředně následující instrukce.

Instrukce **JMI** nepodmíněně převede program na instrukci **L** n, jejíž číslo n obsahuje vrchol zásobníku A0.

JZ	**Skok podmíněný nenulovostí příznaku ZR**
JNZ	**Skok podmíněný nulovostí příznaku ZR**
JC	**Skok podmíněný nenulovostí příznaku CO**
JNC	**Skok podmíněný nulovostí příznaku CO**
JS	**Skok podmíněný nenulovostí příznaku S1.0**
JNS	**Skok podmíněný nulovostí příznaku S1.0**

Operandy

JZ	Ln	B D
JNZ	Ln	B D
JC	Ln	B D
JNC	Ln	B D
JS	Ln	B D
JNS	Ln	B D

Funkce

JZ - skok na návěští L n podmíněný nenulovostí příznaku rovnosti ZR (S0.0)
JNZ - skok na návěští L n podmíněný nulovostí příznaku rovnosti ZR (S0.0)
JC - skok na návěští L n podmíněný nenulovostí příznaku přenosu CO (S0.1)
JNC - skok na návěští L n podmíněný nulovostí příznaku přenosu CO (S0.1)
JS - skok na návěští L n podmíněný nenulovostí příznaku S1.0
JNS - skok na návěští L n podmíněný nulovostí příznaku S1.0

Popis

Instrukce **JZ**, **JNZ**, **JC** a **JNC** jsou určeny především pro snadné vyhodnocení výsledků porovnání instrukcemi **CMP**, **CML**. Instrukce **JS**, **JNS** jsou určeny především pro snadné vyhodnocení výsledků tabulkových instrukcí a všech dalších instrukcí, které příznak S1.0 využívají jako příznak korektnosti provedené operace.

Instrukce **JZ** se zachová jako instrukce **JMP** pouze v případě, že příznak rovnosti ZR (S0.0) je log.1.

Instrukce **JNZ** se zachová jako instrukce **JMP** pouze v případě, že příznak rovnosti ZR (S0.0) je log.0.

Instrukce **JC** se zachová jako instrukce **JMP** pouze v případě, že příznak přenosu CO (S0.1) je log.1.

Instrukce **JNC** se zachová jako instrukce **JMP** pouze v případě, že příznak přenosu CO (S0.1) je log.0.

Instrukce **JS** se zachová jako instrukce **JMP** pouze v případě, že příznak S1.0 je log.1.

Instrukce **JNS** se zachová jako instrukce **JMP** pouze v případě, že příznak S1.0 je log.0.

Pokud příslušná podmínka není splněna, je instrukce ignorována a program pokračuje ve vykonávání další bezprostředně následující instrukce.

Příklady

Porovnejme obsahy registrů *hodnota1* a *hodnota2* a proveďme skok v případě, že obsah *hodnota1* bude roven obsahu *hodnota2*

```
      LD    hodnota1
      CMP   hodnota2
      JZ    skok
       :              ;hodnota1 ≠ hodnota2
skok:
       :              ;hodnota1 = hodnota2
```

Proveďme 6x tutéž část programu

```
      LD    6
      WR    index      ;index = 6
skok:
       :              ;tělo cyklu
      DCR   index      ;index = index - 1
      JNZ   skok       ;index = 0 ?
       :              ;ano, cyklus ukončen
```

Porovnejme obsahy registrů *hodnota1* a *hodnota2* a proveďme skok v případě, že obsah *hodnota1* nebude větší než obsah *hodnota2*

```
      LD    hodnota1
      CMP   hodnota2
      JC    skok
       :              ;hodnota1 > hodnota2
skok:
       :              ;hodnota1 ≤ hodnota2
```

Hledejme položku s obsahem 4 v tabulce *Tab* a proveďme skok v případě, že byla položka nalezena

```
      LD    4
      FTB   Tab        ;hledáme položku s obsahem 4
      JS    skok
       :              ;položka s tímto obsahem nebyla nalezena
skok:
       :         ;položka byla nalezena, index je na vrcholu zásobníku
```

CAL	**Volání podprogramu**
CAD	**Volání podmíněné nenulovostí vrcholu zásobníku**
CAC	**Volání podmíněné nulovostí vrcholu zásobníku**
CAI	**Nepřímé volání podprogramu**

Operandy

CAL	Ln	B D S M
CAD	Ln	B D S M
CAC	Ln	B D S M
CAI	bez operandu	B D S M

Funkce

CAL - nepodmíněné volání podprogramu označeném návěštím L n

CAD - volání podprogramu označeném návěštím L n podmíněné nenulovostí vrcholu zásobníku A0

CAC - volání podprogramu označeném návěštím L n podmíněné nulovostí vrcholu zásobníku A0

CAI - nepodmíněné volání podprogramu označeném návěštím L n, jehož číslo n udává vrchol zásobníku A0

Popis

Instrukce **CAL** nepodmíněně zavolá podprogram začínající na instrukci **L** n.

Instrukce **CAD** se zachová jako instrukce **CAL** pouze v případě, že vrchol zásobníku A0 není 0 (logický součet OR všech 16 bitů A0 je log.1). Pokud tato podmínka není splněna, je instrukce ignorována a program pokračuje ve vykonávání další bezprostředně následující instrukce.

Instrukce **CAC** se zachová jako instrukce **CAL** pouze v případě, že vrchol zásobníku A0 je 0 (logický součet OR všech 16 bitů A0 je log.0). Pokud tato podmínka není splněna, je instrukce ignorována a program pokračuje ve vykonávání další bezprostředně následující instrukce.

Instrukce **CAI** nepodmíněně zavolá podprogram začínající na instrukci **L** n, jejíž číslo n obsahuje vrchol zásobníku A0.

Poznámka

Každý volaný podprogram musí končit instrukcí **RET**, která vrací program na instrukci bezprostředně následující po instrukci volání podprogramu. V případě nesplnění této podmínky PLC zastaví chod programu a vyhlásí chybu. Počet vnoření podprogramů (volání podprogramu v rámci jiného podprogramu) je maximálně 8.

RET	**Návrat z podprogramu**
RED	**Návrat podmíněný nenulovostí vrcholu zásobníku**
REC	**Návrat podmíněný nulovostí vrcholu zásobníku**

Operandy

RET	bez operandu	B D S M
RED	bez operandu	B D S M
REC	bez operandu	B D S M

Funkce

RET - nepodmíněný návrat z podprogramu
RED - návrat z podprogramu podmíněný nenulovostí vrcholu zásobníku A0
REC - návrat z podprogramu podmíněný nulovostí vrcholu zásobníku A0

Popis

Instrukce **RET** nepodmíněně ukončí podprogram a vrátí řízení na instrukci bezprostředně následující za instrukcí volání, kterou byl podprogram vyvolán.

Instrukce **RED** se zachová jako instrukce **RET** pouze v případě, že vrchol zásobníku A0 není 0 (logický součet OR všech 16 bitů A0 je log.1). Pokud tato podmínka není splněna, je instrukce ignorována a program pokračuje ve vykonávání další bezprostředně následující instrukce.

Instrukce **REC** se zachová jako instrukce **RET** pouze v případě, že vrchol zásobníku A0 je 0 (logický součet OR všech 16 bitů A0 je log.0). Pokud tato podmínka není splněna, je instrukce ignorována a program pokračuje ve vykonávání další bezprostředně následující instrukce.

L Návěští

Operandy

L	n	B D S M E

Funkce

L - návěští číslo n

Popis

Instrukce **L** označuje místo v programu, které slouží jako cíl instrukcím skoků a volání. Návěštím může být označeno libovolné místo v programu, pokud je to potřebné v zájmu lepší přehlednosti při prohlížení, monitorování nebo ladění uživatelského programu. Z hlediska programu se instrukce **L** chová jako prázdná, nevykonává se žádná činnost.

Poznámka

V programu se nesmí víckrát opakovat návěští se stejným parametrem. Na číselném pořadí parametru instrukcí **L** v programu nezáleží.

7. ORGANIZAČNÍ INSTRUKCE

P	**Začátek procesu**
E	**Konec procesu**
ED	**Konec procesu podmíněný nenulovým vrcholem zásobníku**
EC	**Konec procesu podmíněný nulovým vrcholem zásobníku**
EOC	**Mimořádný konec cyklu**

Operandy

P	n	B D S M E
E	n	B D S M E
ED	bez operandu	B D S M
EC	bez operandu	B D S M
EOC	bez operandu	B D S M

n - číslo procesu (0 až 64)

Funkce

P - začátek procesu Pn
E - konec procesu Pn
ED - konec aktivního procesu podmíněný nenulovostí vrcholu zásobníku A0
EC - konec aktivního procesu podmíněný nulovostí vrcholu zásobníku A0
EOC - mimořádný konec cyklu

Popis

Instrukce **P** označuje místo v programu, na kterém začíná příslušný proces. Slouží k jeho vyhledání systémovým programem jako počáteční zarážka procesu.

Instrukce **E** označuje místo v programu, na kterém končí příslušný proces Pn. Slouží k předání řízení systémovému programu, který rozhodne o aktivaci dalšího procesu, a dále slouží jako koncová zarážka procesu.

Instrukce **ED** se zachová jako instrukce **E** (neslouží však jako zarážka procesu) pouze v případě, že vrchol zásobníku A0 není 0 (logický součet OR všech 16 bitů A0 je log.1). Pokud tato podmínka není splněna, je instrukce ignorována a program pokračuje ve vykonávání další bezprostředně následující instrukce.

Instrukce **EC** se zachová jako instrukce **E** (neslouží však jako zarážka procesu) pouze v případě, že vrchol zásobníku A0 je 0 (logický součet OR všech 16 bitů A0 je log.0). Pokud tato podmínka není splněna, je instrukce ignorována a program pokračuje ve vykonávání další bezprostředně následující instrukce.

Instrukce **EOC** nepodmíněně přeruší posloupnost procesů naplánovaných k vykonání v daném cyklu a okamžitě vykoná všechny operace spojené s otočkou cyklu (nastavení výstupů, sejmutí vstupů, aktualizace času). Zbytek procesu za instrukcí **EOC** ani žádný z plánovaných procesů se již neaktivuje. Po otočce cyklu se budou aktivovat procesy podle plánu pro nový cyklus.

Poznámka

Parametr n nabývá hodnot pouze v rozsahu čísel přípustných procesů. Proces začínající instrukcí **P** n musí být ukončen instrukcí **E** n se stejným parametrem. Tato podmínka je formální a není na závadu, je-li v programu skok do jiného procesu bez návratu, i když tento postup není programátorsky příliš čistý.

Za normálních okolností se fáze konce (otočky) cyklu provede až po skončené aktivaci (po instrukci **E**) posledního z posloupnosti procesů, naplánovaných pro daný oběh cyklem. V případech, kdy chceme okamžitou reakci na danou situaci, můžeme použít instrukci **EOC**. Instrukce **EOC** zasahuje násilně do systému plánování a vykonávání procesů. Je tedy vhodné její použití omezit na havarijní stavy, pro zabezpečení rychlé odezvy na některé vstupy, apod. Je třeba si uvědomit, že veškeré změny hodnot Y zapsané v instrukcích po instrukci **EOC** již nebudou provedeny! Před instrukcí **EOC** by měl být zajištěn definovaný stav zápisníku pro nový cyklus.

Příklad

Přípustné řazení procesů

```
P 0
       :
       :
     JMD   skok
       :
       :
E 0
;
P 10
       :
       :
skok:
       :
       :
E 10       ;Je-li splněna podmínka JMD, je tento konec procesu platný
           ;i pro proces P0.
```

NOP Prázdná operace

Operandy

NOP	n	B D S M E

Funkce

NOP - žádná operace

Popis

Instrukce **NOP** neprovádí žádnou operaci. Z uživatelského hlediska nemá žádný význam. Obvykle ji generuje překladač vyššího jazyka pro odlišení začátku a konce programových modulů nebo k uložení parametrů těchto modulů.

BP Ladící bod

Operandy

BP	n	B D S M

n - číslo aktivovaného procesu P5n (0 až 7)

Funkce

BP - ladící bod

Popis

Instrukce **BP** je určena především pro fázi ladění uživatelského programu. Aktivuje obslužný proces podle hodnoty parametru. Parametr n smí nabývat jen hodnot 0 až 7 a udává číslo aktivovaného procesu P50 až P57, ve kterém lze na úrovni uživatelského programu zapsat ošetření situace, odpovídající umístění dané instrukce **BP** v uživatelském programu (např. odložení stavu zásobníku do zápisníku, upřesnění podmínky a definování hledaného stavu, výpis zprávy).

Instrukce **BP** n provede uložení aktivního zásobníku a předá řízení procesu P5n. Po ukončení tohoto procesu instrukcí **E**, **ED** nebo **EC** je aktivní zásobník obnoven a program pokračuje v provádění instrukce následující za instrukcí **BP** n. Jde tedy o zvláštní případ instrukce volání.

Instrukci **BP** nelze použít v rámci procesů P50 až P57.

Poznámka

Narozdíl od všech ostatních procesů, v případě vstupu do procesů P50 až P57 je zachován celý aktivní zásobník. Při ukončení těchto procesů je stav zásobníku a obsah příznakových registrů S0 a S1 obnoven na hodnoty, které zde byly při vstupu do procesu. Pokud použijeme v procesu P5n některou z instrukcí pro přepínání zásobníků (**NXT**, **PRV**, **CHG**, **CHGS**), bude sice po ukončení procesu obnoven stav zásobníku, ale aktivní zůstává ten zásobník, který byl aktivní při ukončení procesu P5n! Tímto způsobem tedy dojde k fyzické změně zásobníku, aniž se změnil jeho obsah. Tuto skutečnost lze využít k vytváření kopií zásobníku. Je však třeba věnovat těmto instrukcím zvýšenou pozornost.

SEQ Podmíněné přerušení procesu

Operandy

SEQ	Ln	B D S M

Funkce

SEQ - přerušení procesu podmíněné nulovostí vrcholu zásobníku, proces začne v příštím cyklu od návěští **L** n

Popis

Instrukce **SEQ** se zachová jako instrukce **E** (neslouží však jako zarážka procesu) v případě, že vrchol zásobníku A0 je 0 (logický součet OR všech 16 bitů A0 je log.0). Navíc způsobí, že proces příště začne od návěští Ln. Pokud podmínka není splněna, je instrukce ignorována a program pokračuje ve vykonávání další bezprostředně následující instrukce.

Instrukce **SEQ** umožňuje sekvenční programování v rámci jednoho procesu, kdy se vykonává vždy jen určitá část procesu a přechody mezi těmito částmi pomocí podmínek obstarává právě instrukce **SEQ**.

Upozornění

Instrukce **SEQ** je přípustná jen v procesech P0 až P40.

Příklad

Požadujeme, aby se provedla činnost 1, po nastavení signálu připojeného na X1.0 na log.1 aby se provedla činnost 2, po nastavení signálu připojeného na X1.1 na log.0 aby se provedla činnost 3 a po nastavení signálu připojeného na X1.2 na log.1 aby se provedla opět činnost 1 a dále dokola.

```
P 10
      :              ;činnost 1
navesti1
     LD   vstup1     ;podmínka 1
     SEQ  navesti1   ;dokud bude vstup1 = 0, proces P10 zde skončí
                     ;a příště začne na navesti1
      :              ;vstup1 = 1 - činnost 2
navesti2
     LDC  vstup2     ;podmínka 2
     SEQ  navesti2   ;dokud bude vstup2 = 1, proces P10 zde skončí
                     ;a příště začne na navesti2
      :              ;vstup2 = 0 - činnost 3
navesti3
     LD   vstup3     ;podmínka 3
     SEQ  navesti3   ;dokud bude vstup3 = 0, proces P10 zde skončí
                     ;a příště začne na navesti3
E 10                 ;vstup3 = 1 - konec procesu P10, příště začne
                     ;zase od začátku
```

8. TABULKOVÉ INSTRUKCE

LTB Čtení položky

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
LTB XYSDR							LIMIT	INDEX		A6	A5	A4	A3	A2	LIMIT	INDEX	VAL	
LTB T								INDEX		A6	A5	A4	A3	A2	LIMIT	INDEX	VAL	

LIMIT - hodnota meze tabulky (index poslední položky tabulky) (typ word, CPU řady M typ byte)
INDEX - index žádané položky (typ word, CPU řady M typ byte)
VAL - přečtený obsah (typ odpovídající typu operandu)

Operandy

		bit	byte	word
LTB	X Y S D R	B D S M	B D S M	B D S M
LTB	T	B D S M	B D S M	B D S M

Funkce

LTB - čtení položky z tabulky

Popis

Instrukce **LTB** je indexovanou obdobou instrukce **LD**. Nejdříve posune zásobník vpřed. Je-li zadaný index v rozsahu tabulky (není větší než její mez), je na vrchol zásobníku A0 předán obsah žádané položky a je nastaven příznak S1.0. Je-li požadována položka mimo rozsah tabulky (index je vyšší než její mez), je příznak S1.0 nulován.

Instrukce typu **bit** sejme obsah položky a souhlasně s ním nastaví všech 16 bitů vrcholu zásobníku A0.

Instrukce typu **byte** sejme obsah položky a beze změny jej uloží do dolního bytu vrcholu zásobníku A0, horní byte vynuluje.

Instrukce typu **word** sejme obsah položky a beze změny jej uloží na vrchol zásobníku A0. Do dolního bytu se ukládá byte s nižší adresou v tabulce v rámci položky.

Poznámka

Je-li operandem bitové pole na zápisníku, **musí** toto pole začínat na bitu 0 (pomocí direktivy *#reg aligned*)!

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	IS

S1.0 (IS) - 0 - žádost o položku mimo tabulku
1 - žádost o položku v tabulce

Příklady

Čtení položky formátu bit

```
#table bit  Tab = 0,1,0,1
#reg word INDEX
#reg bit  VAL
;
P 0
     LD   INDEX
     LTB  Tab        ;tabulka T
     WR   VAL
E 0

#def LIMIT  3
#reg aligned bit Tab[LIMIT+1]
#reg word INDEX
#reg bit  VAL
;
P 0
     LD   LIMIT
     LD   INDEX
     LTB  Tab        ;tabulka na zápisníku
     WR   VAL
E 0
```

Čtení položky formátu byte

```
#table byte Tab = 0,1,2,3
#reg word INDEX
#reg byte VAL
;
P 0
     LD   INDEX
     LTB  Tab        ;tabulka T
     WR   VAL
E 0

#def LIMIT  3
#reg byte Tab[LIMIT+1]
#reg word INDEX
#reg byte VAL
;
P 0
     LD   LIMIT
     LD   INDEX
     LTB  Tab        ;tabulka na zápisníku
     WR   VAL
E 0
```

Čtení položky formátu word

```
#table word Tab = 0,1,2,3
#reg word INDEX
#reg word VAL
;
P 0
     LD   INDEX
     LTB  Tab        ;tabulka T
     WR   VAL
E 0

#def LIMIT  3
#reg word Tab[LIMIT+1]
#reg word INDEX
#reg word VAL
;
P 0
     LD   LIMIT
     LD   INDEX
     LTB  Tab        ;tabulka na zápisníku
     WR   VAL
E 0
```

WTB Zápis položky

Instrukce	Vstupní parametry								Výsledek							
	zásobník								zásobník							
	A7	A6	A5	A4	A3	A2	A1	A0	A7	A6	A5	A4	A3	A2	A1	A0
WTB XYSR						LIMIT	INDEX	VAL						LIMIT	INDEX	VAL
WTB T							INDEX	VAL						LIMIT	INDEX	VAL

LIMIT - hodnota meze tabulky (index poslední položky tabulky) (typ word, CPU řady M typ byte)
INDEX - index žádané položky (typ word, CPU řady M typ byte)
VAL - zapisovaný obsah (typ odpovídající typu operandu)

Operandy

		bit	byte	word
WTB	X Y S R	B D S M	B D S M	B D S M
WTB	T	B D	B D	B D

Funkce

WTB - zápis položky do tabulky

Popis

Instrukce **WTB** je indexovanou obdobou instrukce **WR**. Nemění obsah zásobníku. Je-li zadaný index v rozsahu tabulky (není větší než její mez), je do určené položky předán obsah vrcholu zásobníku A0 a je nastaven příznak S1.0. Je-li požadován zápis do položky mimo rozsah tabulky (index je vyšší než její mez), je příznak S1.0 nulován.

Instrukce typu **bit** zapíše do položky hodnotu logického součtu (OR) všech 16 bitů vrcholu zásobníku A0.

Instrukce typu **byte** zapíše do položky dolní byte vrcholu zásobníku A0.

Instrukce typu **word** zapisuje do položky vrchol zásobníku A0. Dolní byte vrcholu zásobníku se ukládá do bytu s nižší adresou v tabulce v rámci položky.

Poznámka

Je-li operandem bitové pole na zápisníku, musí být toto pole začínat na bitu 0 (pomocí direktivy *#reg aligned*)!

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	IS

S1.0 (IS) - 0 - žádost o položku mimo tabulku
1 - žádost o položku v tabulce

Příklady

Zápis položky typu bit

```
#table bit  Tab = 0,1,0,1
#reg word INDEX
#reg bit  VAL
;
P 0
     LD   INDEX
     LD   VAL
     WTB  Tab        ;tabulka T
E 0

#def LIMIT  3
#reg aligned bit Tab[LIMIT+1]
#reg word INDEX
#reg bit  VAL
;
P 0
     LD   LIMIT
     LD   INDEX
     LD   VAL
     WTB  Tab        ;tabulka na zápisníku
E 0
```

Zápis položky typu byte

```
#table byte Tab = 0,1,2,3
#reg word INDEX
#reg byte VAL
;
P 0
     LD   INDEX
     LD   VAL
     WTB  Tab        ;tabulka T
E 0

#def LIMIT  3
#reg byte Tab[LIMIT+1]
#reg word INDEX
#reg byte VAL
;
P 0
     LD   LIMIT
     LD   INDEX
     LD   VAL
     WTB  Tab        ;tabulka na zápisníku
E 0
```

Zápis položky typu word

```
#table word Tab = 0,1,2,3
#reg word INDEX,VAL
;
P 0
     LD   INDEX
     LD   VAL
     WTB  Tab        ;tabulka T
E 0

#def LIMIT  3
#reg word Tab[LIMIT+1]
#reg word INDEX,VAL
;
P 0
     LD   LIMIT
     LD   INDEX
     LD   VAL
     WTB  Tab        ;tabulka na zápisníku
E 0
```

FTB Hledání položky

Instrukce	Vstupní parametry								Výsledek							
	zásobník								zásobník							
	A7	A6	A5	A4	A3	A2	A1	A0	A7	A6	A5	A4	A3	A2	A1	A0
FTB XYSDR							LIMIT	VAL							LIMIT	INDEX
FTB T								VAL							LIMIT	INDEX

LIMIT - hodnota meze tabulky (index poslední položky tabulky) (typ word, CPU řady M typ byte)
VAL - obsah, který má být v tabulce nalezen (typ odpovídající typu operandu)
INDEX - index nalezené položky (pokud není nalezena odpovídající položka, má index hodnotu LIMIT+1) (typ word, CPU řady M typ byte)

Operandy

		bit	byte	word
FTB	X Y S D R	B D S M	B D S M	B D S M
FTB	T	B D S M	B D S M	B D S M

Funkce

FTB - hledání položky v tabulce

Popis

Instrukce **FTB** postupně porovnává údaj na vrcholu zásobníku s obsahy položek tabulky, dokud nenalezne souhlasnou položku, nebo nevyčerpá celou tabulku. Pokud souhlasnou položku nalezne, zapíše její index na vrchol zásobníku A0 a nastaví příznak S1.0. V opačném případě je příznak S1.0 nulován a na vrcholu zásobníku A0 je hodnota rovná mezi zvýšené o 1. Obsahuje-li tabulka více souhlasných položek, funkce nalezne pouze první (s nejnižším indexem).

Instrukce typu **bit** porovnávají hodnotu logického součtu (OR) všech 16 bitů vrcholu zásobníku A0 s položkami tabulky. Bitovou instrukci **FTB** lze využít například pro testování bitového pole, kde má odlišnou hodnotu obvykle jen jeden bit (např. klávesnice).

Instrukce typu **byte** porovnávají obsah dolního bytu vrcholu zásobníku A0 s položkami tabulky.

Instrukce typu **word** porovnávají obsah vrcholu zásobníku A0 s položkami tabulky. Dolní byte vrcholu zásobníku se porovnává s bytem s nižší adresou v tabulce v rámci položky.

Poznámka

Je-li operandem bitové pole na zápisníku, musí toto pole začínat na bitu 0 (pomocí direktivy *#reg aligned*)!

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	IS

S1.0 (IS) - 0 - žádost o položku mimo tabulku
1 - žádost o položku v tabulce

Příklady

Hledání položky typu bit

```
#table bit  Tab = 1,1,0,1
#reg word INDEX
#reg bit  VAL
;
P 0
     LD   VAL
     FTB  Tab        ;tabulka T
     WR   INDEX
E 0

#def LIMIT  3
#reg aligned bit Tab[LIMIT+1]
#reg word INDEX
#reg bit  VAL
;
P 0
     LD   LIMIT
     LD   VAL
     FTB  Tab        ;tabulka na zápisníku
     WR   INDEX
E 0
```

FTM Hledání části položky

Instrukce	Vstupní parametry								Výsledek							
	zásobník								zásobník							
	A7	A6	A5	A4	A3	A2	A1	A0	A7	A6	A5	A4	A3	A2	A1	A0
FTM XYSDR							LIMIT	VAL							LIMIT	INDEX
FTM T								VAL							LIMIT	INDEX

LIMIT - hodnota meze tabulky (index poslední položky tabulky) (typ word, CPU řady M typ byte)
VAL - obsah, který má být v tabulce nalezen (typ odpovídající typu operandu)
INDEX - index nalezené položky (pokud není nalezena odpovídající položka, má index hodnotu LIMIT+1) (typ word, CPU řady M typ byte)

Operandy

		byte	word
FTM	X Y S D R	B D S M	B D S M
FTM	T	B D S M	B D S M

Funkce

FTM - hledání části položky v tabulce

Popis

Instrukce **FTM** je zobecněním instrukce **FTB**, kdy vyhodnocovaná tabulka má dvojnásobný formát. Každá položka obsahuje dvě části pod společným indexem. První část obsahuje hodnotu, druhá část obsahuje výběrovou masku.

	index n		index n+1		
...	položka n	maska n	položka n+1	maska n+1	...

Instrukce **FTM** postupně porovnává údaj na vrcholu zásobníku s položkami tabulky a výsledky porovnání maskuje příslušnými maskami tak, že jsou respektovány jen ty bity výsledku porovnání, kterým odpovídá jednička v bitech ve výběrové masce, dokud nenalezne souhlasnou položku, nebo nevyčerpá celou tabulku. Pokud souhlasnou položku nalezne, zapíše její index na vrchol zásobníku A0 a nastaví příznak S1.0. V opačném případě je příznak S1.0 nulován a na vrcholu zásobníku A0 je hodnota rovna mezi zvýšené o 1.

Funkci porovnávání lze napsat pomocí logických operátorů takto:

(VAL XOR položka) AND maska = výsledek

Je-li tento výsledek 0, jedná se o souhlasnou položku a její index je předán na vrcholu zásobníku. Obsahuje-li tabulka více souhlasných položek, funkce nalezne pouze první (s nejnižším indexem).

Instrukce typu **byte** porovnávají obsah dolního bytu vrcholu zásobníku A0 s položkami tabulky.

Instrukce typu **word** porovnávají obsah vrcholu zásobníku A0 s položkami tabulky. Dolní byte vrcholu zásobníku se porovnává s bytem s nižší adresou v tabulce v rámci položky.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	IS

S1.0 (IS) - 0 - žádost o položku mimo tabulku
1 - žádost o položku v tabulce

Příklady

Hledání položky typu byte

```
#table byte  Tab = 1,7,0,1
#reg long INDEX
#reg byte  VAL
;
P 0
     LD   VAL
     FTM  Tab        ;tabulka T
     WR   INDEX
E 0

#def LIMIT  3
#reg aligned byte Tab[LIMIT+1]
#reg long INDEX
#reg byte  VAL
;
P 0
     LD   LIMIT
     LD   VAL
     FTM  Tab        ;tabulka na zápisníku
     WR   INDEX
E 0
```

FTS Zařazení položky

Instrukce	Vstupní parametry								Výsledek							
	zásobník								zásobník							
	A7	A6	A5	A4	A3	A2	A1	A0	A7	A6	A5	A4	A3	A2	A1	A0
FTS XYSDR							LIMIT	VAL							LIMIT	INDEX
FTS T								VAL							LIMIT	INDEX

LIMIT - hodnota meze tabulky (index poslední položky tabulky) (typ word, CPU řady M typ byte)
VAL - obsah, který má být v tabulce nalezen (typ odpovídající typu operandu)
INDEX - index nalezené položky (pokud není nalezena odpovídající položka, má index hodnotu LIMIT+1) (typ word, CPU řady M typ byte)

Operandy

		byte	word
FTS	X Y S D R	B D S M	B D S M
FTS	T	B D S M	B D S M

Funkce

FTS - zařazení položky podle tabulky
FTSF - zařazení položky podle tabulky (plovoucí řádová čárka)
FTSS - zařazení položky se znaménkem podle tabulky

Popis

Instrukce **FTS** je zobecněním instrukce **FTB** a realizuje mnohoúrovňové porovnání nebo třídění. K tomu je třeba, aby položky v tabulce byly uspořádány vzestupně podle hodnot, protože reprezentují meze oddělující jednotlivé třídy, do nichž instrukce zařazuje obsah A0.

Instrukce **FTS** neposouvá zásobník. Postupně porovnává údaj v A0 s obsahy položek tabulky, dokud nenalezne položku větší nebo rovnou porovnávané hodnotě, nebo nevyčerpá celou tabulku. Pokud souhlasnou položku nalezne, zapíše její index na vrchol zásobníku A0 a nastaví příznak S1.0. V opačném případě je příznak S1.0 nulován a na vrcholu zásobníku A0 je hodnota rovná mezi zvýšené o 1.

Zařazování do tříd je následující (k odpovídá hodnotě LIMIT):

$0 \leq VAL \leq$ položka 0	třída 0
položka 0 $< VAL \leq$ položka 1	třída 1
položka 1 $< VAL \leq$ položka 2	třída 2
položka 2 $< VAL \leq$ položka 3	třída 3
:	:
položka k–1 $< VAL \leq$ položka k	třída k
položka k $< VAL \leq$ maximum	třída k+1

Instrukce typu **byte** porovnávají obsah dolního bytu vrcholu zásobníku A0 s položkami tabulky.

Instrukce typu **word** porovnávají obsah vrcholu zásobníku A0 s položkami tabulky. Dolní byte vrcholu zásobníku se porovnává s bytem s nižší adresou v tabulce v rámci položky.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	IS

S1.0 (IS) - 0 - žádost o položku mimo tabulku
1 - žádost o položku v tabulce

Příklady

Zařazení položky typu byte

```
#table byte  Tab = 1,4,8,15
#reg word INDEX
#reg byte  VAL
;
P 0
     LD   VAL
     FTS  Tab        ;tabulka T
     WR   INDEX
E 0

#def LIMIT  3
#reg aligned byte Tab[LIMIT+1]
#reg word INDEX
#reg byte  VAL
;
P 0
     LD   LIMIT
     LD   VAL
     FTS  Tab        ;tabulka na zápisníku
     WR   INDEX
E 0
```

9. BLOKOVÉ OPERACE

SRC	**Zdroj dat pro přesun**
MOV	**Přesun bloku dat**

Instrukce	Vstupní parametry								Výsledek							
	zásobník								zásobník							
	A7	A6	A5	A4	A3	A2	A1	A0	A7	A6	A5	A4	A3	A2	A1	A0
SRC								INDEX								INDEX
MOV							INDEX	LEN							INDEX	LEN

INDEX - index první položky ve vymezené zdrojové / cílové zóně (typ word, CPU řady M typ byte)
LEN - počet přesouvaných bytových položek (typ word, CPU řady M typ byte)

Operandy

		byte
SRC	X Y S D R	B D S M
SRC	T	B D S M
MOV	X Y S R	B D S M
MOV	T	B D S M

Funkce

SRC - specifikace zdrojové zóny pro přesun bloku dat
MOV - přesun bloku dat do cílové zóny

Popis

Instrukce **SRC** slouží jako příprava před instrukcí **MOV**. Uloží do vnitřní paměti systému údaje o počáteční adrese zdrojové zóny. Adresu první položky udává operand instrukce zvýšený o index uložený na vrcholu zásobníku. Pomocí indexu lze dynamicky počátek zóny měnit.

Instrukce **MOV** přesouvá do cílové zóny obsah zdrojové zóny zadané instrukcí **SRC**.

Počet přesouvaných bytových položek se načítá z vrcholu zásobníku A0. Maximální počet položek je omezen u centrálních jednotek řady M na 255. U ostatních centrálních jednotek je omezen pouze velikostí zápisníku, resp. tabulky.

Adresu první položky udává operand instrukce zvýšený o index uložený ve vrstvě A1.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	IS

S1.0 (IS) - 0 - adresa zdrojové zóny určené instrukcí **SRC** nebo cílové zóny určené instrukcí **MOV** je mimo rozsah tabulky T nebo zápisníku, přesun se neprovede
1 - adresa zdrojové i cílové zóny je v rozsahu tabulky T nebo zápisníku, přesun se provede

S34 = 20 ($14) zdrojový blok dat byl definován mimo rozsah
S34 = 21 ($15) cílový blok dat byl definován mimo rozsah

Poznámka

Specifikace zdrojové zóny zůstává uložena v paměti tak dlouho, dokud není přepsána novou instrukcí **SRC**. Lze ji tedy využít pro více instrukcí **MOV**.

Příklad

Přesun bloku dat

```
#def LEN  30
#reg word INDEX_SRC, INDEX_MOV
#reg byte Zdroj[LEN], Cil[LEN]
;
P 0
      LD    INDEX_SRC
      SRC   Zdroj
       :
       :
      LD    INDEX_MOV
      LD    LEN
      MOV   Cil
E 0
```

MTN	**Přesun tabulky do zápisníku**
MNT	**Naplnění tabulky ze zápisníku**

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
MTN							TAB	REG								TAB	LEN	
MNT							TAB	REG								TAB	LEN	

TAB - číslo přesouvané / plněné tabulky (typ word)
REG - index prvního registru R vymezené zóny v zápisníku (typ word)
LEN - počet přenesených bytů (typ word)

Operandy

		byte
MTN	bez operandu	B D
MNT	bez operandu	B D

Funkce

MTN - přesun tabulky do zápisníku
MNT - naplnění tabulky ze zápisníku

Popis

Instrukce **MTN** přesouvá do cílové zóny v zápisníku celý obsah vybrané tabulky T. Počet přesouvaných bytových položek je určen velikostí tabulky a instrukce jej po přesunu zveřejní na vrcholu zásobníku.

Instrukce **MNT** plní ze zdrojové zóny v zápisníku celý obsah vybrané tabulky T. Počet přesouvaných bytových položek je určen velikostí tabulky a instrukce jej po přesunu zveřejní na vrcholu zásobníku.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	IS

S1.0 (IS) - 0 - adresa zóny v zápisníku je mimo rozsah, přesun se neprovede
1 - adresa zóny v zápisníku je v rozsahu, přesun se provede

S34 = 20 ($14) zdrojový blok dat byl definován mimo rozsah
S34 = 21 ($15) cílový blok dat byl definován mimo rozsah

Příklady

Přesun z tabulky

```
#def MaxDelka  30
#table byte Tab = 0,1,2,3
#reg byte Cil[MaxDelka]
;
P 0
     LD   __indx (Tab)    ;TAB
     LD   __indx (Cil)    ;REG
     MTN
E 0
```

Přesun do tabulky

```
#def MaxDelka  30
#table byte Tab = 0,1,2,3
#reg byte Zdroj[MaxDelka]
;
P 0
      LD    __indx (Tab)    ;TAB
      LD    __indx (Zdroj)  ;REG
      MNT
E 0
```

FIL Plnění bloku

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
FIL							LEN	VAL								LEN	VAL	

LEN - délka plněné zóny (typ word, CPU řady M typ byte)
VAL - zapisovaná konstanta (typ word)

Operandy

		word
FIL	X Y S R	B D S M

Funkce

FIL - plnění zóny konstantou

Popis

Počet bytových položek se načítá z vrstvy A1 zásobníku. Adresu první položky udává operand instrukce, položky jsou plněny střídavě hodnotou z nejnižšího a z druhého nejnižšího bytu vrcholu zásobníku A0.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	IS

S1.0 (IS) - 0 - adresa plněné zóny je mimo rozsah zápisníku
1 - adresa plněné zóny je v rozsahu zápisníku

S34 = 21 ($15) cílový blok dat byl definován mimo rozsah

Příklad

Plnění zóny

```
#def LEN 30
#reg byte Cil[LEN]
#reg word VAL
;
P 0
     LD   LEN
     LD   VAL
     FIL  Cil
E 0
```

10. OPERACE SE STRUKTUROVANÝMI TABULKAMI

LDS Čtení položky ze strukturované tabulky

Instr.	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
LDS					INDEX	SIZE	TAB	REG						INDEX	SIZE	TAB	REG	

INDEX - číslo položky strukturované tabulky (typ word)
SIZE - velikost položky strukturované tabulky v bytech (typ byte)
TAB - číslo čtené tabulky (typ word)
REG - index prvního registru R vymezené cílové zóny (typ word)

Operandy

		byte
LDS	bez operandu	B D

Funkce

LDS - čtení položky ze strukturované tabulky T

Popis

Určená tabulka je strukturovaná na jednotlivé položky o velikosti určené parametrem SIZE. Instrukce **LDS** přesouvá do cílové zóny zápisníku jednu položku tabulky TAB danou parametrem INDEX.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	IS

S1.0 (IS) - 0 - žádaná položka je mimo rozsah tabulky T, nebo adresa cílové zóny je mimo rozsah zápisníku, přesun se neprovede
1 - parametry jsou v pořádku, přesun se provede

Příklad

Čtení položky ze strukturované tabulky

```
#def MaxDelka  30
#table byte Tab = 0,1,2,3,4,5,6,7,8,9
#reg byte Cil[MaxDelka]
#reg byte SIZE
#reg word INDEX
;
P 0
      LD    INDEX
      LD    SIZE
      LD    __indx (Tab)     ;TAB
      LD    __indx (Cil)     ;REG
      LDS
      JNS   skok
       :                     ;operace v pořádku
skok:
       :                     ;chybná operace
E 0
```

WRS Zápis položky do strukturované tabulky

Instr.	Vstupní parametry								Výsledek							
	zásobník								zásobník							
	A7	A6	A5	A4	A3	A2	A1	A0	A7	A6	A5	A4	A3	A2	A1	A0
WRS					INDEX	SIZE	TAB	REG					INDEX	SIZE	TAB	REG

INDEX - číslo položky strukturované tabulky (typ word)
SIZE - velikost položky strukturované tabulky v bytech (typ byte)
TAB - číslo cílové tabulky (typ word)
REG - index prvního registru R vymezené zdrojové zóny (typ word)

Operandy

		byte
WRS	bez operandu	B D

Funkce

WRS - zápis položky do strukturované tabulky T

Popis

Určená tabulka je strukturovaná na jednotlivé položky o velikosti určené parametrem SIZE. Instrukce **WRS** plní ze zdrojové zóny zápisníku jednu položku tabulky TAB danou parametrem INDEX.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	IS

S1.0 (IS) - 0 - žádaná položka je mimo rozsah tabulky T, nebo adresa cílové zóny je mimo rozsah zápisníku, přesun se neprovede
1 - parametry jsou v pořádku, přesun se provede

Příklad

Zápis položky do strukturované tabulky

```
#def MaxDelka  30
#table byte Tab = 0,1,2,3,4,5,6,7,8,9
#reg byte Zdroj[MaxDelka]
#reg byte SIZE
#reg word INDEX
;
P 0
      LD    INDEX
      LD    SIZE
      LD    __indx (Tab)    ;TAB
      LD    __indx (Zdroj)  ;REG
      WRS
      JNS   skok
       :                    ;operace v pořádku
skok:
       :                    ;chybná operace
E 0
```

FIS	**Plnění položky strukturované tabulky v zápisníku**
FIT	**Plnění položky strukturované tabulky**

Instr.	Vstupní parametry								Výsledek							
	zásobník								zásobník							
	A7	A6	A5	A4	A3	A2	A1	A0	A7	A6	A5	A4	A3	A2	A1	A0
FIS					INDEX	SIZE	REGT	VAL					INDEX	SIZE	REGT	VAL
FIT					INDEX	SIZE	TAB	VAL					INDEX	SIZE	TAB	VAL

INDEX - číslo položky strukturované tabulky (typ word)
SIZE - velikost položky strukturované tabulky v bytech (typ byte)
TAB - číslo čtené tabulky (typ word)
REGT - index prvního registru R čtené tabulky (typ word)
VAL - plněná hodnota (typ word)

Operandy

		byte
FIS	bez operandu	B D
FIT	bez operandu	B D

Funkce

FIS - plnění položky strukturované tabulky v zápisníku
FIT - plnění položky strukturované tabulky T

Popis

Určená část zápisníku je strukturovaná na jednotlivé položky o velikosti určené parametrem SIZE. Instrukce **FIS** naplní zadanou hodnotou VAL jednu položku zápisníku danou parametrem INDEX.

Určená tabulka je strukturovaná na jednotlivé položky o velikosti určené parametrem SIZE. Instrukce **FIT** naplní zadanou hodnotou VAL jednu položku tabulky T danou parametrem INDEX.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	IS

S1.0 (IS) - 0 - žádaná položka je mimo rozsah zápisníku, resp. tabulky T, přesun se neprovede
1 - parametry jsou v pořádku, přesun se provede

Příklady

Plnění položky strukturované tabulky

```
#def MaxDelka 30
#reg byte Pole[MaxDelka]
#reg byte SIZE
#reg byte VAL
#reg word INDEX
;
P 0
     LD   INDEX
     LD   SIZE
```

```
      LD    __indx (Pole)   ;REG
      LD    VAL
      FIS
      JNS   skok
       :                    ;operace v pořádku
skok:
       :                    ;chybná operace
E 0

#table byte Tab = 0,1,2,3,4,5,6,7,8,9
#reg byte SIZE
#reg byte VAL
#reg word INDEX
;
P 0
      LD    INDEX
      LD    SIZE
      LD    __indx (Tab)    ;TAB
      LD    VAL
      FIT
      JNS   skok
       :              ;operace v pořádku
skok:
       :              ;chybná operace
E 0
```

FNS	**Hledání položky strukturované tabulky v zápisníku**
FNT	**Hledání položky strukturované tabulky**

Instr.	Vstupní parametry								Výsledek							
	zásobník								zásobník							
	A7	A6	A5	A4	A3	A2	A1	A0	A7	A6	A5	A4	A3	A2	A1	A0
FNS				NUM	BYTE	SIZE	REGT	VAL				NUM	BYTE	SIZE	REGT	INDEX
FNT					BYTE	SIZE	TAB	VAL					BYTE	SIZE	TAB	INDEX

NUM - počet prohledávaných položek (formát word)
BYTE - index prohledávaného bytu v položce (formát byte)
SIZE - velikost položky strukturované tabulky v bytech (formát byte)
REGT - index prvního registru R tabulky v zápisníku (formát word)
TAB - číslo prohledávané tabulky (formát word)
VAL - hledaná hodnota (formát word)
INDEX - index nalezené položky (formát word)

Operandy

		byte
FNS	bez operandu	B D
FNT	bez operandu	B D

Funkce

FNS - hledání položky strukturované tabulky v zápisníku
FNT - hledání položky strukturované tabulky T

Popis

Určená část zápisníku je strukturovaná na jednotlivé položky o velikosti určené parametrem SIZE. Instrukce **FNS** porovnává zadanou hodnotu VAL s jedním bytem položky daným parametrem BYTE. Instrukce prohledává počet položek určený parametrem NUM. Index nalezené položky je zveřejněn na vrcholu zásobníku a je nastaven příznak S1.0. Pokud má více položek stejnou hodnotu prohledávaného bytu, je zveřejněna vždy položka s nejnižším indexem. Pokud položka není nalezena, hodnota zveřejněného indexu je o 1 vyšší než index poslední prohledávané položky. Protože se indexuje od 0, je tato hodnota rovna hodnotě parametru NUM.

Určená tabulka je strukturovaná na jednotlivé položky o velikosti určené parametrem SIZE. Instrukce **FNT** porovnává zadanou hodnotu VAL s jedním bytem položky daným parametrem BYTE. Instrukce prohledává všechny položky tabulky. Index nalezené položky je zveřejněn na vrcholu zásobníku a je nastaven příznak S1.0. Pokud má více položek stejnou hodnotu prohledávaného bytu, je zveřejněna vždy položka s nejnižším indexem. Pokud položka není nalezena, hodnota zveřejněného indexu je o 1 vyšší než index poslední prohledávané položky. Protože se indexuje od 0, je tato hodnota rovna počtu položek v tabulce.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	IS

S1.0 (IS) - 0 - žádaná položka nebyla nalezena
1 - žádaná položka byla nalezena

Příklady

Hledání položky strukturované tabulky

```
#def NUM  30
#reg byte Pole[NUM],SIZE,BYTE
#reg byte VAL
#reg word INDEX
;
P 0
      LD    NUM
      LD    BYTE
      LD    SIZE
      LD    __indx (Pole)   ;REG
      LD    VAL
      FNS
      JNS   skok
      WR    INDEX           ;položka byla nalezena
skok:
       :                    ;položka nebyla nalezena
E 0

#table byte Tab = 0,1,2,3,4,5,6,7,8,9
#reg byte SIZE,BYTE
#reg byte VAL
#reg word INDEX
;
P 0
      LD    BYTE
      LD    SIZE
      LD    __indx (Tab)    ;TAB
      LD    VAL
      FNT
      JNS   skok
      WR    INDEX           ;položka byla nalezena
skok:
       :                    ;položka nebyla nalezena
E 0
```

11. ARITMETICKÉ INSTRUKCE V POHYBLIVÉ ŘÁDOVÉ ČÁRCE

ADF	**Sčítání**
SUF	**Odčítání**

Instrukce	**Vstupní parametry**									**Výsledek**								
	zásobník								ope-rand	zásobník								ope-rand
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
ADF							a		b							$a+b$		b
ADF bez op.					a		b			b		A7	A6	A5	A4	$a+b$		
SUF							a		b							$a-b$		b
SUF bez op.					a		b			b		A7	A6	A5	A4	$a-b$		

Operandy

		float
ADF	X Y S D R	B D
ADF	#	B D
ADF	bez operandu	B D
SUF	X Y S D R	B D
SUF	#	B D
SUF	bez operandu	B D

Funkce

ADF - sčítání (float)
SUF - odčítání (float)

Popis

Instrukce **ADF** s operandem přičte k vrcholu zásobníku A01 obsah zadaného operandu. Instrukce **SUF** s operandem odečte od vrcholu zásobníku A01 obsah zadaného operandu. Výsledek je zapsán na vrchol zásobníku A01. Obsah ostatních vrstev zásobníku se nemění. Instrukce nenastavují žádné příznaky.

Instrukce **ADF** bez operandu sečte obsahy dvojvrstev A23 a A01. Instrukce **SUF** bez operandu odečte obsah dvojvrstvy A01 od obsahu dvojvrstvy A23. Po operaci je posunut zásobník o dvě úrovně zpět a na vrchol zásobníku A01 zapíše výsledek. Instrukce nenastavují žádné příznaky.

Příklady

Realizace výrazu $d = a + (b - c)$

```
#reg float va, vb, vc, vd
;
P 0
     LD   vb
     SUF  vc          ;(b - c)
     ADF  va          ;a + ( )
     WR   vd
E 0
```

MUF	**Násobení**
DIF	**Dělení**

Instrukce	**Vstupní parametry**									**Výsledek**								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
MUF							a		b							$a \cdot b$		b
MUF bez op.					a		b			b		A7	A6	A5	A4	$a \cdot b$		
DIF							a		b							a / b		b
DIF bez op.					a		b			b		A7	A6	A5	A4	a / b		

Operandy

		float
MUF	X Y S D R	B D
MUF	#	B D
MUF	bez operandu	B D
DIF	X Y S D R	B D
DIF	#	B D
DIF	bez operandu	B D

Funkce

MUF - násobení (float)
DIF - dělení (float)

Popis

Instrukce **MUF** s operandem vynásobí obsah vrcholu zásobníku A01 obsahem zadaného operandu. Výsledek je zapsán na vrchol zásobníku A01. Obsah ostatních vrstev zásobníku se nemění. Instrukce nenastavuje žádné příznaky.

Instrukce **MUF** bez operandu vynásobí obsahy vrstev A23 a A01. Pak posune zásobník o dvě úrovně zpět a na nový vrchol zásobníku A01 zapíše výsledek. Instrukce nenastavuje žádné příznaky.

Instrukce **DIF** s operandem vydělí obsah vrcholu zásobníku A01 obsahem zadaného operandu. Výsledek je zapsán na vrchol zásobníku A01. Obsah ostatních vrstev zásobníku se nemění.

Instrukce **DIF** bez operandu vydělí obsah dvojvrstvy A23 obsahem dvojvrstvy A01. Pak posune zásobník o dvě úrovně zpět a na nový vrchol zásobníku A01 zapíše výsledek.

Pokud dojde k dělení nulou, nastaví se bit S0.0 na log.1 a do registru S34 se zapíše chyba 16. Vrchol zásobníku obsahuje samé jedničky (neplatné číslo podle konvence formátu float).

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	-	-	-	-	-	-	ZR

S0.0 (ZR) - dělení nulou
1 - došlo k dělení nulou, výsledek je neplatný

S34 = 16 ($10) chyba dělení nulou

Příklady

Realizace výrazu $d = a + (b \cdot c)$

```
#reg float va, vb, vc, vd
;
P 0
     LD   vb
     MUF  vc          ;(b . c)
     ADF  va          ;a + ( )
     WR   vd
E 0
```

Realizace výrazu $d = a + \frac{b}{c}$

```
#reg float va, vb, vc, vd
;
P 0
     LD   vb
     DIF  vc          ;(b / c)
     ADF  va          ;a + ( )
     WR   vd
E 0
```

CMF Porovnání

Instrukce	Vstupní parametry									Výsledek								
	zásobník								ope-	zásobník								ope-
	A7	A6	A5	A4	A3	A2	A1	A0	rand	A7	A6	A5	A4	A3	A2	A1	A0	rand
CMF							*a*		*b*							*a*		*b*
CMF bez op.					*a*		*b*							*a*		*b*		

Operandy

		float
CMF	X Y S D R	B D
CMF	#	B D
CMF	bez operandu	B D

Funkce

CMF - porovnání hodnot a nastavení příznaků výsledku (float)

Popis

Instrukce **CMF** s operandem porovná obsah vrcholu zásobníku s operandem a nastaví příznaky v S0. Obsah zásobníku se nemění.

Instrukce **CMF** bez operandu porovná obsah vrstvy A23 s obsahem vrstvy A01 a nastaví příznaky v S0. Obsah zásobníku se nemění.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S0	-	-	-	-	-	≤	CO	ZR

S0.0 (ZR) - porovnání na shodu
0 - platí $a \neq b$
1 - platí $a = b$

S0.1 (CO) - výstupní přenos
0 - platí $a \geq b$
1 - platí $a < b$

S0.2 (≤) - logický součet S0.0 OR S0.1
0 - platí $a > b$
1 - platí $a \leq b$

CEI	**Zaokrouhlení nahoru**
FLO	**Zaokrouhlení dolů**
ABS	**Absolutní hodnota**

Instrukce	**Vstupní parametry**									**Výsledek**								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
CEI							a									$a \nearrow$		
FLO							a									$a \searrow$		
ABS							a									$\lvert a \rvert$		

Operandy

		float
CEI	bez operandu	B D
FLO	bez operandu	B D
ABS	bez operandu	B D

Funkce

CEI - zaokrouhlení čísla v pohyblivé řádové čárce na nejbližší vyšší celé číslo (float)
FLO - zaokrouhlení čísla v pohyblivé řádové čárce na nejbližší nižší celé číslo (float)
ABS - výpočet absolutní hodnoty čísla (float)

Popis

Instrukce **CEI** provede zaokrouhlení čísla na vrcholu zásobníku na nejbližší vyšší celé číslo a toto číslo uloží na vrchol zásobníku. Obsah ostatních vrstev zásobníku se nemění.

Instrukce **FLO** provede zaokrouhlení čísla na vrcholu zásobníku na nejbližší nižší celé číslo a toto číslo uloží na vrchol zásobníku. Obsah ostatních vrstev zásobníku se nemění.

Instrukce **ABS** provede vynulování nejvyššího bitu čísla na vrcholu zásobníku, který nese znaménko. Tento bit je nejvyšší ve vrstvě A1 zásobníku. Obsah ostatních vrstev zásobníku se nemění.

LOG	**Dekadický logaritmus**
LN	**Přirozený logaritmus**
EXP	**Exponenciální funkce**
POW	**Obecná mocnina**
SQR	**Druhá odmocnina**
HYP	**Euklidovská vzdálenost**

Instrukce	**Vstupní parametry**									**Výsledek**								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
LOG							a									$\log_{10} a$		
LN							a									$\ln a$		
EXP							a									e^{a}		
POW					a		b			b		A7	A6	A5	A4	a^{b}		
SQR							a									$\sqrt{a}$		
HYP					a		b			b		A7	A6	A5	A4	$\sqrt{a^2+b^2}$		

Operandy

		float
LOG	bez operandu	B D
LN	bez operandu	B D
EXP	bez operandu	B D
POW	bez operandu	B D
SQR	bez operandu	B D
HYP	bez operandu	B D

Funkce

LOG - výpočet dekadického logaritmu (float)
LN - výpočet přirozeného logaritmu (float)
EXP - výpočet exponenciální funkce (float)
POW - výpočet obecné mocniny (float)
SQR - výpočet druhé odmocniny (float)
HYP - výpočet Euklidovské vzdálenosti (float)

Popis

Instrukce **LOG** provede výpočet dekadického logaritmu a instrukce **LN** výpočet přirozeného logaritmu obsahu vrcholu zásobníku. Obsah vrcholu zásobníku musí být větší než 0. Výsledek uloží na vrchol zásobníku. Obsah ostatních vrstev zásobníku se nemění.

Instrukce **EXP** provede výpočet exponenciální funkce. Mocnitel Eulerova čísla je očekáván na vrcholu zásobníku. Výsledek je uložen na vrchol zásobníku. Obsah ostatních vrstev zásobníku se nemění. Instrukce nenastavuje žádné příznaky.

Instrukce **POW** provede výpočet obecné mocniny. Mocnitel b je očekáván na vrcholu zásobníku A01, mocněnec a ve dvojvrstvě A23. Zásobník je posunut o dvě úrovně zpět a výsledek je uložen na vrchol zásobníku.

Čísla předávaná instrukci **POW** nesmí být obě současně nulová. Jestliže je umocňované číslo záporné, potom mocnitel může mít pouze celočíselnou hodnotu.

Instrukce **SQR** provede výpočet druhé odmocniny obsahu vrcholu zásobníku. Odmocňované číslo nesmí být záporné. Výsledek je uložen na vrchol zásobníku. Obsah ostatních vrstev zásobníku se nemění.

Instrukce **HYP** provede výpočet Euklidovské vzdálenosti. Parametry jsou instrukcí **HYP** očekávány v dvojvrstvách A01 a A23. Zásobník je posunut o dvě úrovně zpět a výsledek je uložen na vrchol zásobníku. Instrukce nenastavuje žádné příznaky.

Upozornění: Vstupní parametry instrukce **HYP** musí být takové, aby výraz $a^2 + b^2$ nepřekročil maximální rozsah formátu float.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	S

S1.0 (S) - 1 - parametry jsou v pořádku, výsledek je platný (nastavují instrukce **LOG**, **LN**, **POW**, **SQR**)
0 - chybné parametry, výsledek je neplatný

SIN	**Sinus**
COS	**Cosinus**
TAN	**Tangens**
ASN	**Arc sinus**
ACS	**Arc cosinus**
ATN	**Arc tangens**

Instrukce	**Vstupní parametry**									**Výsledek**								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
SIN							a									$\sin a$		
COS							a									$\cos a$		
TAN							a									$\tan a$		
ASN							a									$\arcsin a$		
ACS							a									$\arccos a$		
ATN							a									$\arctan a$		

Operandy

		float
SIN	bez operandu	B D
COS	bez operandu	B D
TAN	bez operandu	B D
ASN	bez operandu	B D
ACS	bez operandu	B D
ATN	bez operandu	B D

Funkce

SIN - sinus
COS - cosinus
TAN - tangens
ASN - opačná funkce k sinu
ACS - opačná funkce ke cosinu
ATN - opačná funkce k tangens

Popis

Instrukce **SIN** provede sinus obsahu vrcholu zásobníku. Parametr je očekáván v radiánech v rozsahu <–65 536; +65 536>. Výsledek je uložen na vrchol zásobníku. Obsah ostatních vrstev zásobníku se nemění.

Instrukce **COS** provede cosinus obsahu vrcholu zásobníku. Parametr je očekáván v radiánech v rozsahu <–65 536; +65 536>. Výsledek je uložen na vrchol zásobníku. Obsah ostatních vrstev zásobníku se nemění.

Instrukce **TAN** provede tangens obsahu vrcholu zásobníku. Parametr je očekáván v radiánech v rozsahu $<-\frac{\pi}{2};\ +\frac{\pi}{2}>$. Výsledek je uložen na vrchol zásobníku. Obsah ostatních vrstev zásobníku se nemění.

Instrukce **ASN** provede arc sinus obsahu vrcholu zásobníku. Parametr je očekáván v rozsahu <–1; +1>. Výsledek v rozsahu $<-\frac{\pi}{2};\ +\frac{\pi}{2}>$ je uložen na vrchol zásobníku. Obsah ostatních vrstev zásobníku se nemění.

Instrukce **ACS** provede arc cosinus obsahu vrcholu zásobníku. Parametr je očekáván v rozsahu <–1; +1>. Výsledek v rozsahu $<-\frac{\pi}{2};\ +\frac{\pi}{2}>$ je uložen na vrchol zásobníku. Obsah ostatních vrstev zásobníku se nemění.

Instrukce **ATN** provede arc tangens obsahu vrcholu zásobníku. Výsledek v rozsahu $<-\frac{\pi}{2};\ +\frac{\pi}{2}>$ je uložen na vrchol zásobníku. Obsah ostatních vrstev zásobníku se nemění. Příznak v registru S1 není nastavován, výsledek je vždy platný.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	S

S1.0 (S) - 1 - parametry jsou v pořádku, výsledek je platný (nenastavuje instrukce **ATN**)
0 - chybné parametry, výsledek je neplatný

UWF	**Převod hodnoty word bez znaménka na float**
IWF	**Převod hodnoty word se znaménkem na float**
ULF	**Převod hodnoty long bez znaménka na float**
ILF	**Převod hodnoty long se znaménkem na float**

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
UWF								NUW		A6	A5	A4	A3	A2	A1	NF		
IWF								NIW		A6	A5	A4	A3	A2	A1	NF		
ULF							NUL									NF		
ILF							NIL									NF		

NUW - hodnota typu word bez znaménka
NIW - hodnota typu word se znaménkem
NUL - hodnota typu long bez znaménka
NIL - hodnota typu long se znaménkem
NF - hodnota převedená na typ float

Operandy

		word	long
UWF	bez operandu	B D	
IWF	bez operandu	B D	
ULF	bez operandu		B D
ILF	bez operandu		B D

Funkce

UWF - převod hodnoty typu word bez znaménka na typ float
IWF - převod hodnoty typu word se znaménkem na typ float
ULF - převod hodnoty typu long bez znaménka na typ float
ILF - převod hodnoty typu long se znaménkem na typ float

Popis

Instrukce **UWF** zpracuje vrchol zásobníku A0 jako číslo typu word v rozsahu <0; 65 535> a převede jej na typ float. Zásobník je posunut o jednu úroveň vpřed a výsledek je uložen na vrchol zásobníku A01.

Instrukce **IWF** zpracuje vrchol zásobníku A0 jako číslo typu word v rozsahu <–32 768; +32 767> a převede jej na typ float. Zásobník je posunut o jednu úroveň vpřed a výsledek je uložen na vrchol zásobníku A01.

Instrukce **ULF** zpracuje vrchol zásobníku A01 jako číslo typu long v rozsahu <0; 4 294 967 295> a převede jej na typ float. Výsledek je uložen na vrchol zásobníku A01. Obsah ostatních vrstev zásobníku se nemění.

Instrukce **ILF** zpracuje vrchol zásobníku A01 jako číslo typu long v rozsahu <–2 147 483 648; +2 147 483 647> a převede jej na typ float. Výsledek je uložen na vrchol zásobníku A01. Obsah ostatních vrstev zásobníku se nemění.

UFW	**Převod hodnoty float na word bez znaménka**
IFW	**Převod hodnoty float na word se znaménkem**
UFL	**Převod hodnoty float na long bez znaménka**
IFL	**Převod hodnoty float na long se znaménkem**

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
UFW							NF			-	A7	A6	A5	A4	A3	A2	NUW	
IFW							NF			-	A7	A6	A5	A4	A3	A2	NIW	
UFL							NF									NUL		
IFL							NF									NIL		

NF - hodnota typu float
NUW - hodnota převedená na typ word bez znaménka
NIW - hodnota převedená na typ word se znaménkem
NUL - hodnota převedená na typ long bez znaménka
NIL - hodnota převedená na typ long se znaménkem

Operandy

		word	long
UFW	bez operandu	B D	
IFW	bez operandu	B D	
UFL	bez operandu		B D
IFL	bez operandu		B D

Funkce

UFW - převod hodnoty typu float na typ word bez znaménka
IFW - převod hodnoty typu float na typ word se znaménkem
UFL - převod hodnoty typu float na typ long bez znaménka
IFL - převod hodnoty typu float na typ long se znaménkem

Popis

Instrukce **UFW** zpracuje vrchol zásobníku A01 jako číslo typu float a převede jej na typ word v rozsahu <0; 65 535>. Zásobník je posunut o jednu úroveň zpět a výsledek je uložen na vrchol zásobníku A0.

Instrukce **IFW** zpracuje vrchol zásobníku A01 jako číslo typu float a převede jej na typ word v rozsahu <–32 768; +32 767>. Zásobník je posunut o jednu úroveň zpět a výsledek je uložen na vrchol zásobníku A0.

Instrukce **UFL** zpracuje vrchol zásobníku A01 jako číslo typu float a převede jej na typ long v rozsahu <0; 4 294 967 295>. Výsledek je uložen na vrchol zásobníku A01. Obsah ostatních vrstev zásobníku se nemění.

Instrukce **IFL** zpracuje vrchol zásobníku A01 jako číslo typu float a převede jej na typ long v rozsahu <–2 147 483 648; +2 147 483 647>. Výsledek je uložen na vrchol zásobníku A01. Obsah ostatních vrstev zásobníku se nemění.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	S

S1.0 (S) - 1 - výsledek je platný
0 - překročen rozsah formátu, výsledek je neplatný

12. INSTRUKCE REGULÁTORU PID

CNV	Konverze a zpracování dat z analogových vstupů

Instr.	Vstupní parametry								Výsledek							
	zásobník								zásobník							
	A7	A6	A5	A4	A3	A2	A1	A0	A7	A6	A5	A4	A3	A2	A1	A0
CNV			INDF	TAU	INDM	FCE	AVAL	MODE			INDF	TAU	INDM	FCE	AVAL	VAL

INDF - index registru stavové proměnné filtru - volitelné, viz dále
TAU - časová konstanta filtru t - volitelné, viz dále
INDM - index registru proměnné pro měřítkování - volitelné, viz dále
FCE - aktivované funkce
AVAL - měřená analogová hodnota převzatá z analogového vstupu
MODE - číslo jednotky × 100 + typ konverze
VAL - výsledek konverze

Operandy

CNV	bez operandu	B D

Funkce

CNV - převod měřených analogových hodnot na normalizované hodnoty a diagnostika okrajových stavů měření

Popis

Instrukce **CNV** je určena především pro převody hodnot získaných z běžných analogových vstupů. Analogové jednotky, které poskytují v registrech normalizované hodnoty přímo, nevyžadují tuto konverzi. Instrukce normalizuje (tj. provádí měřítkování a případně linearizaci) hodnoty z analogových vstupů a provádí základní diagnostiku měřených hodnot.

Instrukce také provádí kontrolu platnosti měřených hodnot. Při podtečení rozsahu, resp. přetečení rozsahu v záporných hodnotách, je předávána hodnota VAL = $–7FFF (tj. $8001). Při přetečení rozsahu v kladných hodnotách je předávána hodnota VAL = $7FFF.

Tab.12.1 Rozsahy normalizovaných hodnot

Typ vstupu	Analogový rozsah	Vnitřní reprezentace
Unifikovaný rozsah unipolární	0 ÷ 10 V, 0 ÷ 20 mA, 4 ÷ 20 mA	0 ÷ 10000 (0 ÷ 100,00%)
	Diagnostika chyby polarity analogového signálu a přetečení převodníku	
Unifikovaný rozsah bipolární	–10 ÷ +10 V, –20 ÷ +20 mA	–10000 ÷ +10000 (–100,00 ÷ +100,00%)
	Diagnostika přetečení převodníku	
Měření odporu*	podle nastavení jednotek	v desetinách Ω nebo jednotkách Ω
	Diagnostika chyby polarity a přetečení převodníku	
Měření teplot**	podle nastavení jednotek	v desetinách °C
	Diagnostika zkratu a přerušení čidla	

* Měření odporu je určeno pro snímání odporových snímačů 100 Ω nebo 1000 Ω.

** Měření teplot je určeno pro snímání odporových snímačů Pt a Ni, součástí přepočtu hodnot je i linearizace průběhu.

Instrukce **CNV** dále obsahuje následující funkce:

- normalizace analogových vstupů do rozsahů podle tabulky tab.12.1
- normalizace analogových vstupů se zápisem rozsahu měření
- převod unifikovaného rozsahu na jiný (inženýrské jednotky)
- měřítkování lineární interpolací
- filtrace 1. řádu
- druhá odmocnina

Jednotlivé funkce se liší nároky na počet použitých registrů zápisníku a počtem předávaných parametrů v zásobníku (viz následující popisy jednotlivých funkcí). Funkce se aktivují pomocí řídících bitů hodnoty FCE ve vrstvě A2 zásobníku. Funkce se mohou navzájem kombinovat.

Pozor! Pokud je vyhodnocena neplatná naměřená hodnota, aktivované funkce podle parametru FCE se neprovádí!

	.7	.6	.5	.4	.3	.2	.1	.0
FCE	-	-	-	-	SQ	FI	F1	F0

FCE.1,.0 - 00 - normalizace
01 - normalizace se zápisem měřítka
10 - převod vstupu na inženýrské jednotky
11 - lineární interpolace mezi dvěma body

FCE.2 (FI) - filtrace hodnot lineárním filtrem 1. řádu
0 - vypnuta
1 - zapnuta

FCE.3 (SQ) - druhá odmocnina
0 - vypnuta
1 - zapnuta

Podrobnosti k jednotlivým funkcím jsou uvedeny v následujícím textu.

Instrukce **CNV** neposouvá zásobník a na jeho vrchol umístí výsledek konverze.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	S

S1.0 (S) - 1 - instrukce se provedla
0 - datová struktura je mimo zápisník, instrukce se neprovede

S34 = 20 ($14) překročen rozsah zápisníku

Normalizace hodnot z analogových vstupů

Normalizace hodnot se provádí při nastavení bitů FCE.0 a FCE.1 na log.0 (vrstva A2 zásobníku). Instrukce převede naměřenou hodnotu z analogového vstupu do normalizovaného rozsahu.

Vrstvy A3, A4 a A5 zásobníku nejsou využity. Pokud není aktivována funkce filtrace, která je vyžaduje, není třeba je zadávat.

Rozsahy normalizovaných hodnot jsou závislé na typu PLC. Konkrétní hodnoty jsou uvedeny v tabulkách 12.2 až 12.8.

IT-04 NS950

Tab.12.2 Tabulka typů konverze pro jednotku IT-04

MODE	Analogový vstup	Rozsah měření	Výstup VAL
401	0 ÷ 10 V 0 ÷ 20 mA	0 ÷ 10 V 0 ÷ 20 mA	0 ÷ 10000
402	–10 ÷ +10 V –20 ÷ +20 mA	–10 ÷ +10 V –20 ÷ +20 mA	–10000 ÷ +10000
410	4 ÷ 20 mA	–20 ÷ +20 mA	0 ÷ 10000
420	0 ÷ 256 Ω	0 ÷ 0,256 V	0 ÷ 2560 (desetiny Ω)
421	Pt100 (W_{100} = 1,385)	0 ÷ 0,256 V	–1000 ÷ +4260 (desetiny °C)
430	0 ÷ 1024 Ω	0 ÷ 1,024 V	0 ÷ 1024 (jednotky Ω)
431	Pt500 (W_{100} = 1,385)	0 ÷ 1,024 V	–1000 ÷ +2680 (desetiny °C)
432	Ni500 (W_{100} = 1,618)	0 ÷ 1,024 V	–500 ÷ +1520 (desetiny °C)

IT-06 NS950

Unifikované rozsahy uvedené v tab.12.1 poskytuje i jednotka IT-06, která navíc zajišťuje i měření termočlánků. Pokud chceme použít měřené hodnoty poskytované jednotkou IT-06, do parametru MODE zapíšeme hodnotu 0. Potom jsou použity vstupní hodnoty přímo pro další funkce. Předpokládá se, že kódy $–7FFF a $7FFF jsou chyby.

IT-11, IT-12 NS950

Tab.12.3 Tabulka typů konverze pro jednotku IT-12 a piggyback IT-11

MODE	Analogový vstup	Rozsah měření	Výstup VAL
1101, 1201	0 ÷ 10 V 0 ÷ 20 mA	0 ÷ 10 V 0 ÷ 20 mA	0 ÷ 10000
1102, 1202	–10 ÷ +10 V –20 ÷ +20 mA	–10 ÷ +10 V –20 ÷ +20 mA	–10000 ÷ +10000
1110, 1210	4 ÷ 20 mA	–20 ÷ +20 mA	0 ÷ 10000
1120, 1220	0 ÷ 500 Ω	0 ÷ 0,5 V	0 ÷ 5000 (desetiny Ω)
1121, 1221	Pt100 (W_{100} = 1,385, do 390 Ω)	0 ÷ 0,5 V	–1000 ÷ +8500 (desetiny °C)
1130, 1230	0 ÷ 2000 Ω	0 ÷ 2 V	0 ÷ 2000 (jednotky Ω)
1131, 1231	Pt1000 (W_{100} = 1,385)	0 ÷ 2 V	–1000 ÷ +2600 (desetiny °C)
1132, 1232	Ni1000 (W_{100} = 1,618)	0 ÷ 2 V	–500 ÷ +1520 (desetiny °C)

IT-15 NS950

Tab.12.4 Tabulka typů konverze pro jednotku IT-15

MODE	Analogový vstup	Rozsah měření	Výstup VAL
1501	0 ÷ 20 mA	0 ÷ 20 mA	0 ÷ 10000
1502	–20 ÷ +20 mA	–20 ÷ +20 mA	–10000 ÷ +10000
1510	4 ÷ 20 mA	–20 ÷ +20 mA	0 ÷ 10000
1520	0 ÷ 157 Ω	0 ÷ 0,157 V	0 ÷ 1570 (desetiny Ω)
1521	Pt100 (W_{100} = 1,385)	0 ÷ 0,157 V	–1000 ÷ +1470 (desetiny °C)
1525	0 ÷ 320 Ω	0 ÷ 0,32 V	0 ÷ 3200 (desetiny Ω)
1526	Pt100 (W_{100} = 1,385)	0 ÷ 0,32 V	–1000 ÷ +6200 (desetiny °C)
1530	0 ÷ 2000 Ω	0 ÷ 2 V	0 ÷ 2000 (jednotky Ω)
1531	Pt1000 (W_{100} = 1,385)	0 ÷ 2 V	–1000 ÷ +2600 (desetiny °C)
1532	Ni1000 (W_{100} = 1,618)	0 ÷ 2 V	–500 ÷ +1520 (desetiny °C)

TR050

Tab.12.5 Tabulka typů konverze pro analogové vstupy TR051, TR052, TR053, TR054

MODE	Analogový vstup	Typ vstupu	Výstup VAL
2001	0 ÷ 20 mA	proudový	0 ÷ 10000
2010	4 ÷ 20 mA	proudový	0 ÷ 10000
2025	0 ÷ 1000 Ω	pasivní	0 ÷ 1000 (jednotky Ω)
2032	Ni1000 (W_{100} = 1,618)	pasivní	–600 ÷ +1350 (desetiny °C)
2033	Ni1000 (W_{100} = 1,500)	pasivní	–600 ÷ +1610 (desetiny °C)

TR200, TR300

Tab.12.6 Tabulka typů konverze pro analogové vstupy TR201, TR202, TR203, TR204, TR301, TR302, TR303, TR304, TR321, TR322

MODE	Analogový vstup	Typ vstupu	Výstup VAL
2001	0 ÷ 20 mA	proudový	0 ÷ 10000
2010	4 ÷ 20 mA	proudový	0 ÷ 10000
2030	0 ÷ 1870 Ω	pasivní	0 ÷ 1870 (jednotky Ω)
2031	Pt1000 (W_{100} = 1,385)	pasivní	–1000 ÷ +2315 (desetiny °C)
2032	Ni1000 (W_{100} = 1,618)	pasivní	–500 ÷ +1352 (desetiny °C)
2033	Ni1000 (W_{100} = 1,500)	pasivní	–500 ÷ +1600 (desetiny °C)

TC400

Tab.12.7 Tabulka typů konverze pro analogové vstupy TC402

MODE	Analogový vstup	Typ vstupu	Výstup VAL
4001	0 ÷ 20 mA	proudový	0 ÷ 10000
	0 ÷ 2 V	napěťový	0 ÷ 10000
4010	4 ÷ 20 mA	proudový	0 ÷ 10000

TC500, TC600

Tab.12.8 Tabulka typů konverze pro analogové vstupy TC505, TC506, TC515, TC516, TC605, TC606, TC625, TC626

MODE	Analogový vstup	Typ vstupu	Výstup VAL
501	0 ÷ 20 mA	proudový	0 ÷ 10000
	0 ÷ 10 V	napěťový	0 ÷ 10000
	0 ÷ 2 V	napěťový	0 ÷ 10000
510	4 ÷ 20 mA	proudový	0 ÷ 10000

TC634

Unifikované rozsahy uvedené v tab.12.1 poskytuje i modul TC634, která navíc zajišťuje i měření termočlánků. Pokud chceme použít měřené hodnoty poskytované modulem TC634, do parametru MODE zapíšeme hodnotu 0. Potom jsou použity vstupní hodnoty přímo pro další funkce. Předpokládá se, že kódy $–7FFF a $7FFF jsou chyby.

Ostatní zdroje

Pokud chceme použít měřené hodnoty získané jinak, do parametru MODE zapíšeme hodnotu 0. Potom jsou použity vstupní hodnoty přímo pro další funkce. Předpokládá se, že kódy $–7FFF a $7FFF jsou chyby.

Příklad

Chceme měřit teplotu v rozsahu 10 ÷ 100 °C. K měření použijeme snímač Pt100 připojený na kanál 0 analogové jednotky IT-15 v PLC TECOMAT NS950. V zápisníku PLC chceme mít hodnotu teploty v desetinách °C.
Výsledkem této konverze je údaj v desetinách °C v proměnné *teplota*.

```
;Zvolen typ vstupu pro připojení Pt100 v rozsahu 0 až 157 Ω.
;
#reg word adata[8],teplota
;
P 0
     LD   0          ;FCE  - normalizace
     LD   adata      ;AVAL - načtení kanálu 0
     LD   1521       ;MODE - IT-15, Pt100, rozsah -1000 až +1470
     CNV
     WR   teplota    ;VAL  - výsledná teplota
E 0
```

Normalizace hodnot z analogových vstupů se zápisem rozsahu

Normalizace hodnot se zápisem rozsahu se provádí při nastavení bitů FCE.0 na log.1 a FCE.1 na log.0 (vrstva A2 zásobníku). Instrukce převede naměřenou hodnotu z analogového vstupu do normalizovaného rozsahu. Do proměnných datové struktury, která je daná indexem počátečního registru INDM (vrstva A3), uloží minimální a maximální hodnotu rozsahu.

Datová struktura:
MinY - minimální měřitelná hodnota (typ word)
MaxY - maximální měřitelná hodnota (typ word)

Vrstvy A4 a A5 zásobníku nejsou využity. Pokud není aktivována funkce filtrace, která je vyžaduje, není třeba je zadávat.

Rozsahy normalizovaných hodnot jsou závislé na typu PLC. Konkrétní hodnoty jsou uvedeny v tabulkách 12.2 až 12.8.

Poznámka

Instrukce **CNV** vyžaduje, aby datová struktura nezačínala blíže než 8 bytů od konce zápisníku.

Instrukce **PID** používá na začátku své datové struktury stejné proměnné. Je tedy výhodné použít při spojení obou instrukcí překrytí těchto struktur, protože instrukce **CNV** v této funkci dosadí měřené rozsahy pro **PID** automaticky.

Příklad

Chceme měřit teplotu v rozsahu –50 až 200 °C. K měření použijeme snímač Pt100 připojený na kanál 0 analogové jednotky IT-15 v PLC TECOMAT NS950. V zápisníku PLC chceme mít teplotu v desetinách °C a maximální měřící rozsah pro instrukci PID.
Výsledkem této konverze je údaj v desetinách °C v proměnné *teplota* a měřící rozsah v proměnných *MinY* a *MaxY*.

```
;Zvolen typ vstupu pro připojení Pt100 v rozsahu 0 až 320 Ω.
;
#reg word adata[8],teplota
#reg word MinY,       ;minimální měřená hodnota
          MaxY        ;maximální měřená hodnota
;
P 0
     LD   __indx (MinY)   ;INDM - index reg., kde začíná dat. struktura
     LD   1               ;FCE  - zápis měřícího rozsahu
     LD   adata           ;AVAL - načtení kanálu 0
     LD   1526            ;MODE - IT-15, Pt100, rozsah -1000 až +6200
     CNV                  ;MinY = -1000, MaxY = +6200
     WR   teplota         ;VAL  - výsledná teplota
E 0
```

Převod unifikovaného rozsahu na jiný (inženýrské jednotky)

Převod unifikovaného rozsahu na jiný se provádí při nastavení bitů FCE.0 na log.0 a FCE.1 na log.1 (vrstva A2 zásobníku). Instrukce **CNV** nejdříve převede naměřenou hodnotu z analogového vstupu do normalizovaného rozsahu. Potom podle zadaných hodnot v datové struktuře, která je daná indexem počátečního registru INDM (vrstva A3), tuto převedenou hodnotu v rozsahu 0 ÷ 10000 (případně –10000 ÷ +10000) převede do nového rozsahu.

Datová struktura:
MinY - minimální měřitelná hodnota (typ word)
MaxY - maximální měřitelná hodnota (typ word)

Interval 0 ÷ 10000 je tedy lineárně transformován na interval MinY a MaxY. Pro bipolární rozsah –10000 ÷ +10000 je převod symetrický, v tomto případě je MinY = 0.

Vrstvy A4 a A5 zásobníku nejsou využity. Pokud není aktivována funkce filtrace, která je vyžaduje, není třeba je zadávat.

Rozsahy normalizovaných hodnot jsou závislé na typu PLC. Konkrétní hodnoty jsou uvedeny v tabulkách 12.2 až 12.8.

Poznámka

Instrukce **CNV** vyžaduje, aby datová struktura nezačínala blíže než 8 bytů od konce zápisníku.

Instrukce převádí rozsah měřené hodnoty 0 ÷ 10000 (případně –10000 ÷ +10000) do nového rozsahu.

V případě měření záporné teploty se jedná o bipolární rozsah (MinY je brána jako nulová, na jejím obsahu nezáleží), výsledná hodnota je převedena na interval –MaxY ÷ +MaxY.

Pokud nechceme, aby teplota 0°C neodpovídala hodnotě 0, ale byla posunutá, musíme zaručit pouze kladný rozsah měřené teploty (potom je rozsah unipolární a výsledná hodnota je převedena na interval MinY ÷ MaxY) anebo použijeme měřítkování lineární interpolací.

Příklad

Kanálem 0 analogové jednotky IT-15 v PLC TECOMAT NS950 snímáme proud v rozmezí 0 ÷ 10 mA. Pro další výpočty potřebujeme mít rozsah měřeného proudu s přesností na desetiny mA (0 ÷ 100, tj. 0 ÷ 10,0 mA).
Protože jednotka IT-15 pracuje v rozsahu 0 ÷ 20 mA, nastavíme parametry tak, aby výsledek instrukce CNV pro 20 mA byl 200. Tímto je nastaven požadovaný rozsah v desetinách mA (0 mA odpovídá 0, 10 mA odpovídá 100).

```
#reg word adata[8],proud
#reg word MinY,       ;minimální měřená hodnota
          MaxY        ;maximální měřená hodnota
#def upravenyRozsah  200
;
P 63
     LD   upravenyRozsah  ;proměnná MinY zůstane nulová
     WR   MaxY
E 63
;
P 0
     LD   __indx (MinY)   ;INDM - index reg., kde začíná dat. struktura
     LD   2               ;FCE  - inženýrské jednotky
     LD   adata           ;AVAL - hodnota z analogové jednotky
     LD   1501            ;MODE - IT-15, rozsah 0 ÷ 10000
     CNV
     WR   proud           ;VAL  - výsl. proud 0 ÷ 100, tj. 0 ÷ 10,0 mA
E 0
```

Měřítkování lineární interpolací

Měřítkování lineární interpolací se provádí při nastavení bitů FCE.0 a FCE.1 na log.1 (vrstva A2 zásobníku). Instrukce **CNV** nejdříve převede naměřenou hodnotu ze vstupu analogové jednotky do normalizovaného rozsahu. Potom pro tuto převedenou hodnotu najde její funkční hodnotu na přímce určené dvěma body. Souřadnice těchto dvou bodů jsou uvedeny v datové struktuře instrukce, která je daná indexem počátečního registru INDM (vrstva A3).

Tato funkce je vhodná např. pro lineární kalibraci měřícího řetězce, změnu rozsahu odporového vysílače, generování žádané hodnoty, definované pomocí lineárních úseků daných tabulkou.

Datová struktura:
MinY - 1. souřadnice 1. bodu přímky (např. měřená hodnota skutečná) (typ word)
MaxY - 1. souřadnice 2. bodu přímky (např. měřená hodnota skutečná) (typ word)
MinW - 2. souřadnice 1. bodu přímky (např. hodnota žádaná) (typ word)
MaxW- 2. souřadnice 2. bodu přímky (např. hodnota žádaná) (typ word)

Vrstvy A4 a A5 zásobníku nejsou využity. Pokud není aktivována funkce filtrace, která je vyžaduje, není třeba je zadávat.

Rozsahy normalizovaných hodnot jsou závislé na typu PLC. Konkrétní hodnoty jsou uvedeny v tabulkách 12.2 až 12.8.

Poznámka

Instrukce **CNV** vyžaduje, aby datová struktura nezačínala blíže než 8 bytů od konce zápisníku.

Příklad

Chceme korigovat měření odporového vysílače s rozsahem 20 ÷ 952 Ω na rozsah 0 ÷ 10 000. Odporový vysílač je připojený na kanál 0 analogové jednotky IT-15 v PLC TECOMAT NS950.

```
;Zvolen typ vstupu pro připojení odporového vysílače v rozsahu 0 ÷ 2000Ω
;
#reg word adata[8],odpor
#reg word MinY,       ;1. souřadnice 1. bodu přímky (hodnota skutečná)
          MaxY,       ;1. souřadnice 2. bodu přímky (hodnota skutečná)
          MinW,       ;2. souřadnice 1. bodu přímky (hodnota žádaná)
          MaxW        ;2. souřadnice 2. bodu přímky (hodnota žádaná)
;
P 63
     LD   20          ;skutečná hodnota
     WR   MinY
     LD   952
     WR   MaxY
     LD   0           ;žádaná hodnota
     WR   MinW
     LD   10000
     WR   MaxW        ;rovnice této přímky je y = (10000(x-20))/932
E 63
;
P 0
     LD   __indx (MinY)   ;INDM - index reg., kde začíná dat. struktura
     LD   3               ;FCE  - lineární interpolace
     LD   adata           ;AVAL - načtení kanálu 0
     LD   1530            ;MODE - odporový rozsah 0 ÷ 2000 Ω
     CNV
     WR   odpor           ;VAL  - výsledná hodnota
E 0
```

Filtrace 1. řádu

Tuto funkci je možno kombinovat se všemi předchozími funkcemi. Výsledná hodnota po předchozích funkcích je filtrována číslicovým filtrem 1. řádu (provádí průměrování). Vzorkovací frekvence je dána délkou cyklu automatu.

Filtr je dán tímto vztahem:

$$y_t = \frac{y_{t-1} \cdot \tau + x}{\tau + 1}$$

x - převedená hodnota analogového vstupu
y_t - výstup CNV (hodnota VAL)
y_{t-1} - minulý výstup CNV (hodnota VAL)
t - časová konstanta filtru 1. řádu

Hodnota konstanty τ je zadávána v násobcích doby cyklu PLC v parametru TAU (vrstva A4). Pokud je např. doba cyklu PLC 100 ms a TAU = 10, pak výsledná časová konstanta τ je 1 s (mění se s délkou doby cyklu PLC).

Je-li TAU = 0, je dosazována měřená hodnota do stavové proměnné filtru (inicializace filtru).

Vícenásobné použití instrukce **CNV** pro filtraci je podmíněno samostatnou deklarací stavové proměnné filtru pro každou instrukci **CNV** (index počátečního registru proměnné INDF je předáván ve vrstvě A5). Nad jednou stavovou proměnnou nesmějí pracovat dvě nebo více instrukcí **CNV** (kromě vlastní inicializace).

Poznámka

Instrukce **CNV** vyžaduje, aby deklarovaná stavová proměnná nezačínala blíže než 4 byty od konce zápisníku.

Příklad

Měříme teplotu pomocí snímače Pt1000 připojeného na kanál 0 analogové jednotky IT-15 v PLC TECOMAT NS950. Signál je nutné filtrovat filtrem cca 0,2 s.
Výsledkem je v proměnné *teplota* filtrovaná hodnota.

```
#reg word adata[8],teplota
#reg byte AuxD[4]     ;pomocná stavová proměnná
;
P 63
;inicializace filtru
      LD    __indx (AuxD)    ;INDF - index stavové proměnné filtru
      LD    0                ;TAU  - tau=0, přepis do stavové proměnné
      LD    0                ;INDM - nepoužito
      LD    4                ;FCE  - pouze filtrace
      LD    adata            ;AVAL - načtení kanálu 0
      LD    1531             ;MODE - Pt1000, rozsah -1000 ÷ +2600
      CNV
E 63
;
P 0
      LD    __indx (AuxD)    ;INDF - index stavové proměnné filtru
      LD    20               ;TAU  - tau = 20x doba cyklu PLC [ms]
      LD    0                ;INDM - nepoužito
      LD    4                ;FCE  - pouze filtrace
      LD    adata            ;AVAL - načtení kanálu 0
      LD    1531             ;MODE - Pt1000, rozsah -1000 ÷ +2600
      CNV
      WR    teplota          ;VAL  - výsledná hodnota
E 0
```

Druhá odmocnina

Tuto funkci je možno kombinovat se všemi předchozími funkcemi. Výsledná hodnota po předchozích funkcích je odmocněna.

Vrstvy A3, A4 a A5 zásobníku nejsou využity. Pokud je nevyžaduje některá z předchozích funkcí, není třeba je zadávat.

Příklad

Proveďme normalizaci hodnoty a její odmocnění.

```
#reg word adata[8],odmocnina
;
P 0
     LD   8              ;FCE  - normalizace a odmocnina
     LD   adata          ;AVAL - načtení kanálu 0
     LD   1521           ;MODE - Pt100, rozsah -1000 ÷ +1470
     CNV
     WR   odmocnina      ;VAL  - výsledná hodnota
E 0
```

PID PID regulátor

Instrukce	Vstupní parametry								Výsledek							
	zásobník								zásobník							
	A7	A6	A5	A4	A3	A2	A1	A0	A7	A6	A5	A4	A3	A2	A1	A0
PID						INPUT3	INPUT2	INDEX						INPUT3	INPUT2	DET

INPUT3- y_3 - poloha servoventilu - volitelné, viz dále
INPUT2- y_2 - poměrová regulace - volitelné, viz dále
INDEX - index registru, kde začíná datová struktura regulátoru
DET - detekce chyby a akčního zásahu (typ byte)

Operandy

PID	bez operandu	C

Funkce

PID - PID regulátor

Popis

Pomocí instrukce **PID** lze řídit soustavy, u kterých je doba přechodového děje aspoň o řád delší, než je vzorkování regulátoru (např. při vzorkování 1 s, lze řídit soustavu s přechodovým dějem trvajícím řádově desítky sekund). Vzorkování regulátoru je nutné stanovit s ohledem na dobu cyklu PLC tak, aby byla zaručena určitá přesnost vzorkování. Je vhodné, aby vzorkování regulátoru bylo stanoveno o řád výše, než je doba cyklu PLC (např. doba cyklu 30 ms dovoluje stanovit vzorkování regulátoru na 300 ms).

Instrukce **PID** umožňuje začlenění do přerušovacího procesu P41, který je zařazován pravidelně každých 10 ms. To umožňuje nastavit vzorkování regulátoru na 10 ms a řídit tak soustavy s krátkou dobou přechodového děje. Doba výkonu přerušovacího procesu nesmí překročit 5 ms!

Základní výhodou regulátoru realizovaného instrukcí **PID** je začlenění všech jeho proměnných do systému PLC. Tím je uživateli dána možnost definovat pomocí instrukcí PLC libovolné podmínky pro alarmy, ovládání akčních orgánů až po nastavování regulátoru v závislosti na stavu celé technologie. Měření regulovaných veličin je zajištěno analogovými jednotkami PLC.

K měřítkování nebo filtraci spojitých hodnot je možné použít instrukci **CNV**. Ta provádí také základní normalizaci unifikovaných, odporových a teplotních rozsahů a provádí základní diagnostiku měření analogových jednotek.

Poznámka: V textu se často používá slovo regulátor jako synonymum pro řídící algoritmus. Dále se používá z praxe vžitý název PID regulátor místo přesnější zkratky PSD pro číslicovou verzi klasického spojitého algoritmu.

Instrukce **PID** zajišťuje ve volitelných násobcích 10 ms výpočet hodnoty akčního zásahu podle algoritmu PID, přesněji PPID. Ovládání algoritmu je zajištěno pomocí struktury proměnných, která je definována na registrech PLC. PID pracuje v zásadě podle diskrétní verze této rovnice:

$$u(t) = K\left[e(t) + \frac{1}{T_I}\int_0^t e(\tau)d\tau + T_D\frac{de(t)}{dt} \right]$$

Řídící algoritmus zajišťuje následující funkce:

1. **Beznárazové přepínání ručního a automatického režimu**, které je prováděno na základě odhadu stavu řízené soustavy. **Parametry regulátoru je možné měnit i v automatickém režimu.**
2. Přepínání druhého pásma proporcionality podle znaménka odchylky. (Určeno pro rychlejší potlačení překmitu regulované soustavy.)
3. Nastavení rozsahu zóny působení I a D složky v % rozsahu měřené veličiny.
4. Nastavení pásma necitlivosti regulátoru, tj. povolení nových změn zásahů až od dané úrovně odchylky v % rozsahu měřené veličiny.
5. Přírůstkové řízení polohových ventilů i bez nutnosti odměřování jejich polohy. Je-li poloha ventilu měřená, je prováděna korekce akčního zásahu podle skutečné polohy ventilu.
6. Realizaci poměrové regulace, filtrace nebo lineární interpolace žádané hodnoty (rampa).
7. Zadávání rozsahů akčního zásahu. (To umožňuje realizovat např. rozsah akčního zásahu od 0 do 100%, nebo od –100% do +100%.) Pro akční zásah je možné zadávat i omezení jeho přírůstku.

Parametry regulačního algoritmu se zadávají do rezervované datové oblasti v zóně registrů. V procesu P63 je vhodné dosadit všechny žádané parametry regulátoru (implicitně jsou nulové!). V průběhu regulace je nutné dosazovat do příslušné proměnné hodnotu regulované veličiny případně pomocnou hodnotu polohy akčního orgánu při přírůstkovém řízení. Po výpočtu stačí přepsat hodnotu z proměnné *ConOut* do analogové výstupní jednotky. Při řízení zapnuto / vypnuto se přenesou bity z proměnné *Status* na výstupy binární jednotky.

Poznámka: V průběhu regulace je možné měnit i parametry regulátoru dosazením nových hodnot do patřičných proměnných. Řídící algoritmus zaručuje správnou činnost i pro nestacionární regulátor (s parametry proměnnými v čase).

Instrukce **PID** používá datovou strukturu velikosti 72 bytů, ve které má uložené všechny své proměnné. Každý regulátor musí mít vyhrazenou svoji výlučnou strukturu!

Seznam proměnných, které rezervují patřičné místo v zóně registrů je proveden následujícím způsobem:

```
#struct _PID
     word MinY,       ;minimální měřená hodnota
     word MaxY,       ;maximální měřená hodnota
     word Input1,     ;měřená veličina (y, regulovaná)
     word gW,         ;žádaná hodnota (w), cílová
     word ConW,       ;žádaná hodnota současná
     word tiW,        ;časová konstanta filtru w nebo časový interval
                      ;rampy v násobcích výstupního cyklu
     word Dev,        ;odchylka [%]
     word Output,     ;přímý akční zásah (u) žádaný algoritmem nebo
                      ;manuálně [%]
     word LastOut,    ;minulý akční zásah, tj. o 1 krok zpožděný [%]
     word CurOut,     ;výstup skutečně žádaný [%]
     word ConOut,     ;výstup regulátorem realizovaný
     word DefOut,     ;implicitní hodnota výstupu při chybě měření [%]
     word MinU,       ;minimální povolený akční zásah [%]
     word MaxU,       ;maximální povolený akční zásah [%]
     word dMaxU,      ;maximální povolený přírůstek akčního zásahu [%]
     word OutCycle,   ;délka výstupního cyklu (perioda vzorkování)
     word PBnd,       ;pásmo proporcionality [%]
```

```
word RelCool,    ;pomocné pásmo proporcionality [%]
word Ti,         ;integrační konstanta [s]
word Td,         ;derivační konstanta [s]
word EGap,       ;symetrické pásmo necitlivosti [%]
word DGap,       ;symetrické pásmo odchylky, ve kterém působí
                 ;derivační složka [%]
word IGap,       ;symetrické pásmo odchylky, ve kterém působí
                 ;integrační složka [%]
word Control,    ;řídící slovo
byte Status,     ;status
byte[23] AuxD    ;pomocné proměnné - zakázán zápis!
```

Parametry datové struktury jsou blíže popsány v následujícím textu.

Instrukce **PID** vrací na vrcholu zásobníku výsledek detekce chyb a okrajových stavů.

	.7	.6	.5	.4	.3	.2	.1	.0
DET	EY3	EY2	EY1	UMX	UMN	ER2	ER1	ER0

DET.2,.1,.0 - chyby parametrů
- 000 - parametry v pořádku
- 001 - chybně zadaný čas výstupního cyklu *OutCycle* (implicitní hodnota 1)
- 010 - nepřípustná hodnota omezení akčního zásahu (implicitní hodnota 0 ÷ 10000)
- 011 - nepřípustná hodnota přírůstku akčního zásahu (implicitní hodnota 10000)
- 100 - *MinY* ≥ *MaxY* (dolní a horní mez vstupní veličiny) (implicitní hodnota 0 ÷ 10000)
- 101 - pásmo proporcionality *PBnd* je nulové (implicitní hodnota 1000)
- 110 - druhé pásmo proporcionality *RelCool* je nulové, (implicitní hodnota 1000)

Je-li chybně zadán některý z parametrů *OutCycle*, *PBnd*, *RelCool*, instrukce **PID** nastaví implicitní hodnotu chybného parametru. Chyba je indikována pouze v cyklu, kdy nastala a kdy byla instrukcí opravena. V dalším cyklu již tato skutečnost indikována není, tj. na dolních 3 bitech je log.0.

DET.3 (UMN)- detekce minimálního akčního zásahu
- 1 - akční zásah je menší než *MinU*

DET.4 (UMX)- detekce maximálního akčního zásahu
- 1 - akční zásah je větší než *MaxU*

DET.5 (EY1) - detekce chyby měření y_1 (*Input1*)
- 1 - y_1 mimo interval <*MinY*, *MaxY*>

DET.6 (EY2) - detekce chyby měření y_2 (*Input2*)
- 1 - y_2 mimo interval <*MinY*, *MaxY*>

DET.7 (EY3) - detekce chyby měření y_3 (*Input3*)
- 1 - y_3 mimo interval <*MinY*, *MaxY*>

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	S

S1.0 (S) - 1 - instrukce se provedla
- 0 - datová struktura je mimo zápisník, instrukce se neprovede

S34 = 20 ($14) překročen rozsah zápisníku

Popis jednotlivých parametrů datové struktury

MinY - Minimální měřená hodnota. Používá se pro normalizaci odchylky.

MaxY - Maximální měřená hodnota. Používá se pro normalizaci odchylky.

Input1 (y1) - Měřená (regulovaná) veličina.

gW (w) - Žádaná hodnota, ležící v intervalu měřené veličiny <*MinY*, *MaxY*>.

tiW - Časová konstanta pro filtr 1. řádu nebo lineární interpolaci žádané hodnoty v násobcích *OutCycle*.

ConW - Žádaná hodnota současná.

Dev (e) - Odchylka skutečné hodnoty od žádané [%].

Output - Výstup žádaný algoritmem nebo manuálně. Akční zásah může ležet maximálně v rozsahu –10000 až +10000 (tj. –100,00% až +100,00%). Je tedy normován tak, že pro zesíleni 1 (pásmo proporcionality 100%) a odchylku 100,00% je zásah 100,00%. Rozsah je vždy omezen v rozsahu <*MinU*, *MaxU*>.

LastOut - Minulý akční zásah, tj. o 1 krok zpožděný [%] nebo poloha ventilu (viz kaskádní řízení).

CurOut - Výstup skutečně žádaný v daném kroku [%] nebo přírůstek akčního zásahu.

ConOut - Výstup regulátorem realizovaný [%] nebo skutečná hodnota realizovaná výstupní jednotkou nebo časově proporcionálním řízením on / off v absolutní hodnotě.

DefOut - Implicitní hodnota výstupu při chybě měření.

MinU - Minimální povolený akční zásah [%]. Akční zásah přímý nemůže být menší než tato hodnota.

MaxU - Maximální povolený akční zásah [%]. Akční zásah přímý nemůže být větší než tato hodnota.

dMaxU - Maximální povolený přírůstek akčního zásahu [%]. Nový akční zásah se nemůže v absolutní hodnotě lišit o více než *dMaxU* od minulé hodnoty.

OutCycle - Délka výstupního cyklu, perioda vzorkování [setiny s]. Určuje periodu, po kterou se nemění akční zásah, respektive periodu opakovacího kmitočtu pro časově proporcionální řízení. Nejmenší hodnota je 1, tj. 10 ms, a může být nastavena až na 65535, tj. přes 10 minut.

PBnd - Pásmo proporcionality. Nastavuje se v rozsahu 1 až 30000 (0,1 až 3000,0%). Určuje zesílení vztahem

$$K = \frac{1000}{PBnd}$$

RelCool - Pomocné pásmo proporcionality pro zápornou odchylku. Nastavuje se v rozsahu 1 až 30000 (0,1 až 3000,0%). Zesílení je pak určeno vztahem

$$K = \frac{1000}{PBnd} \cdot \frac{1000}{RelCool}$$

Z toho plyne, že pro *RelCool* = 1000 (100,0%) je tato složka bez vlivu.

Ti - Integrační konstanta [desetiny s]. Nastavuje se v rozsahu 0 až 30000 (0 až 3000,0 s). Pro nulovou hodnotu je integrační složka vypnutá.

Td - Derivační konstanta [desetiny s]. Nastavuje se v rozsahu 0 až 30000 (0 až 3000,0 s).

Egap - Symetrické pásmo necitlivosti. Rozsah je od 0 do 10000 (0 až 100,00%). Je-li odchylka menší než *EGap*, tj. leží v pásmu necitlivosti, zůstává akční zásah neměnný.

Dgap - Symetrické pásmo odchylky, ve kterém působí derivační složka. Rozsah je od 0 do 10000 (0 až 100,00%). To znamená, že derivační složka působí stále pro *DGAP* = 10000.

Igap - Symetrické pásmo odchylky, ve kterém působí integrační složka. Rozsah je od 0 do 10000 (0 až 100,00%). To znamená, že integrační složka působí stále pro *IGAP* = 10000.

Control - Řídící slovo slouží k nastavení činnosti regulátoru. Regulátor se může nacházet v režimu automatickém, ručním nebo havarijním. Může pracovat jako regulátor s přímým nebo přírůstkovým algoritmem. Je-li jako akční orgán použitý servoventil, je možné použít pro korekci přírůstku akčního zásahu naměřenou hodnotu jeho polohy, tj. jedná se o kaskádní řízení. Za předpokladu delšího výstupního cyklu je možné realizovat časově proporcionální řízení výstupu on / off. Rozlišení je dáno dobou cyklu automatu. Např. je-li doba cyklu automatu 100 ms a výstupní cyklus 10 s, rozlišení je 1%.

.15	.14	.13	.12	.11	.10	.9	.8	.7	.6	.5	.4	.3	.2	.1	.0
FU2	FU1	FU0	-	-	P41	RIO	RF	HR	AM	IP	BU	KC	A12	AO	RC

RC - 1 - žádost o studený start regulátoru (instrukce sama nuluje bit)
AO - 1 - posun nuly výstupu regulátoru pro rozsah 4 ÷ 20 mA
A12 - 1 - výstup na 12 bitový převodník D/A
KC - 1 - kaskádní řízení
BU - 0 - unifikovaný výstup
1 - binární výstup (časově proporcionální, řízení on / off)
IP - 0 - přímé řízení
1 - přírůstkové řízení
AM - 0 - ruční režim
1 - automatický režim
HR - 1 - režim spolehlivějšího měření, využívá dvě měřené hodnoty
RF - 0 - úprava žádané hodnoty filtrem 1. řádu
1 - úprava žádané hodnoty lineární interpolací
RIO - 1 - poměrová regulace
P41 - 1 - instrukce **PID** se volá v procesu P41, tj. v rastru 10 ms (pouze pro CPU řady B a C)

Poznámka:
I v tomto případě je možné použít nastavení periody v proměnné *OutCycle*. Praktický význam však má jen pro řízení typu on / off.
Např. je-li *OutCycle* = 100, je perioda 1 s a rozlišení šířky výstupního pulzu je 10 ms, tj. 1%. Při použití např. výstupu 230 V AC, je možné takto realizovat výstup typu cyklického řízení, tj. s rozlišením jedné periody fázového napětí.

FU2-FU0 - filtrace krátkých akčních zásahů

Obecně platí, že pokud *CurOut* < 32 ∗ FU, zásah se neprovede a obsah *CurOut* je vynulován.
0 - všechny akční zásahy povoleny
1 - potlačeny akční zásahy menší než 32 (tj. 0,32%)
2 - potlačeny akční zásahy menší než 64 (tj. 0,64%)
3 - potlačeny akční zásahy menší než 96 (tj. 0,96%)
4 - potlačeny akční zásahy menší než 128 (tj. 1,28%)
5 - potlačeny akční zásahy menší než 160 (tj. 1,6%)
6 - potlačeny akční zásahy menší než 192 (tj. 1,92%)
7 - potlačeny akční zásahy menší než 224 (tj. 2,24%)

Status - Slouží zejména pro přenos hodnoty bitů pro on / off řízení, tedy pokud je akční zásah řešen jako časově proporcionální řízení (šířka pulsů). Dále obsahuje chybové bity měření.

.7	.6	.5	.4	.3	.2	.1	.0
-	EY3	EY2	EY1	DR	U–	UC	UH

UH - výstup pro kladný akční zásah, tj. topení
UC - výstup pro záporný akční zásah, tj. chlazení
U– - signalizace akčního zásahu
0 - kladný akční zásah
1 - záporný akční zásah
DR - detekce průběhu lineární interpolace žádané hodnoty
1 - interpolace aktivní
EY1 - detekce chyby měření y_1 (*Input1*)
1 - y_1 mimo interval <*MinY*, *MaxY*>
EY2 - detekce chyby měření y_2 (*Input2*)
1 - y_2 mimo interval <*MinY*, *MaxY*>
EY3 - detekce chyby měření y_3 (*Input3*)
1 - y_3 mimo interval <*MinY*, *MaxY*>

AuxD - Pomocné proměnné regulátoru. Zápis do této zóny je zakázán!!

Rozsahy a formáty měřených veličin

Do registru *Input1* (y_1) se zadává měřená (regulovaná) hodnota z analogového vstupu. Rozsahy měřené veličiny *Input1* se zadávají do registrů *MinY* a *MaxY*.

Ve vrstvě A1 zásobníku je hodnota *Input2* (y_2) použitá pro poměrovou regulaci nebo pro spolehlivější měření. V prvém případě musí být v registru *Control* nastaven bit RIO na log.1, ve druhém případě bit HR na log.1 (viz Ovládání žádané hodnoty). Pokud se tyto režimy nepoužívají, proměnná nemusí být zadávána.

Ve vrstvě A2 zásobníku je hodnota *Input3* (y_3) využita pro měření polohy ventilu. V tom případě jde o kaskádní řízení a v registru Control je třeba nastavit bit KC na log.1 (viz Kaskádní řízení). Pokud se tento režim nevyužívá, proměnná nemusí být zadávána.

Pro převody z analogových jednotek je možné použít instrukci **CNV** (při chybě měření instrukce vrací kód $–7FFF, resp. $8001, chyba minimální hodnoty, nebo $7FFF, chyba maximální hodnoty). Při větším zarušení vstupního signálu je možné použít filtr 1. řádu, který je součástí instrukce **CNV**.

Ovládání žádané hodnoty a výpočet odchylky

Řídící bity: *Control* - RF, RIO, RC
Diagnostické bity: *Status* - DR
Registry: *gW, ConW, tiW*

Žádaná hodnota se zadává do registru *gW*. Regulátor však bere pro výpočet odchylky žádanou hodnotu z registru *ConW*!

Filtrace (RF = log.0)

Pokud má bit RF hodnotu log.0, je aktivován filtr 1. řádu žádané hodnoty. Registr *tiW* udává časovou konstantu pro tento filtr.

Je-li *tiW* = 0, je po zadání nové hodnoty do registru *gW* dosazena tato hodnota i do registru *ConW*, tedy *ConW* = *gW*.

Je-li *tiW* > 1, pak po změně *gW* je v *ConW* hodnota filtrovaná s časovou konstantou *tiW*. Např. pro *tiW* = 5, *OutCyclus* = 10 je časová konstanta w = 5 s.

Je-li po restartu PLC hodnota bitu RC = log.1, je v prvním cyklu *ConW* = *Input1*. Stejná hodnota je dosazena do stavové proměnné filtru. Takto se dá dosáhnout najetí na žádanou hodnotu s minimálním překývnutím.

Rampa (RF = log.1)

Pokud má bit RF hodnotu log.1, je aktivována lineární interpolace žádané hodnoty. Registr *tiW* udává dobu interpolace žádané hodnoty.

Je-li *tiW* = 0, je možné bez vzájemného vlivu měnit hodnoty registrů *ConW* a *gW*.

Po dosazení požadovaného času do *tiW* je prováděna lineární interpolace směrem od hodnoty v *ConW* do *gW*. Z tabulky žádaných hodnot realizované instrukcemi PLC je možné realizovat např. libovolné teplotní cykly. Výběr nové hodnoty tabulky je možné synchronizovat pomocí bitu DR.

Je-li po restartu PLC hodnota bitu RC = log.1, je do stavové proměnné dosazena hodnota registru *ConW*, ale *ConW* se nemění.

Výpočet odchylky

Výpočet odchylky se provádí podle hodnoty bitu RIO. Vnitřně se odchylka normuje do rozsahu –10000 ÷ +10000 (–100,00% ÷ +100,00%) podle vztahu:

Vlečná regulace (RIO = log.0):

$$e = 10000 \cdot \frac{ConW - y_1}{MaxY - MinY}$$

Poměrová regulace (RIO = log.1):

$$e = 10000 \cdot \frac{\frac{ConW}{100} \cdot y_2 - y_1}{MaxY - MinY}$$

V tomto případě se *ConW* zadává v rozsahu 0 až 10000. Pro stejný poměr y_1 a y_2 je *ConW* = 100. Je-li vyregulováno, tj. e = 0, je y1/y2 rovno žádanému poměru *ConW*/100.

Režim spolehlivějšího měření

Je-li hodnota bitu HR = log.1, používá regulátor dva vstupy měření *Input1* (y_1) a *Input2* (y_2) následovně v závislosti na výskytu chyby měření:

obě měření v pořádku - pro výpočet odchylky se použije průměr z y_1, y_2

v pořádku jen jedno měření - pro výpočet odchylky se použije měření, které je v pořádku. Regulátor nepřechází do havarijního stavu! Indikace chyby příslušného měření je v registru *Status* a na vrcholu zásobníku A0 po provedení instrukce PID.

chyba obou měření - havarijní stav (viz Havarijní režim)

Diagnostické bity měření jsou trvale v činnosti.

Pozor! V proměnné *Input1* v tomto režimu je po vykonání instrukce uložena regulovaná hodnota použitá pro regulaci, tj. buďto průměr y_1, y_2, nebo platná hodnota z y_1, y_2, nebo chybový kód ($±7FFF).

Režimy regulátoru

Řídící bity:	*Control*	- AM
Diagnostické bity:	*Status*	- EY1, EY2, EY3
	A0	- EY1, EY2, EY3

Regulátor může pracovat v automatickém, ručním nebo havarijním režimu.

Ruční režim (AM = log.0)

Přechod do **ručního** režimu (AM = log.0) z automatického spočívá v pozastavení výpočtu akčního zásahu regulátoru. Regulátor stále zobrazuje změny odchylky a diagnostikuje chyby měření. Po zadání nového akčního zásahu do registru *Output* se tento okamžitě začíná provádět s vlivem omezení přírůstku akčního zásahu (rychlosti)! Do *Output* se zadává vždy přímá hodnota výstupu.

Automatický režim (AM = log.1)

Přechod do **automatického** režimu (AM = log.1) z ručního je beznárazový a liší se podle toho, je-li přítomna integrační složka v regulátoru.

- Jde-li o typ regulátoru PD, tak poslední ručně zadaná hodnota akčního zásahu je hodnota offsetu, který se přičítá ke složkám PD regulátoru. Tato vlastnost se dá použít např. při regulaci soustav s posunutou nulou, kde poruchy mají charakter "šumu" s nulovou střední hodnotou a není vhodné zadávat I složku. Pro astatické soustavy musí tedy být poslední zadaná hodnota v manuálním režimu 0 (pro přímý algoritmus).
- Jde-li o typ regulátoru s I složkou je počáteční podmínka regulátoru určena na základě odhadu ustáleného stavu regulované soustavy.

V blízkosti ustáleného stavu se po přepnutí akční zásah prakticky nemění. Mimo ustálený stav se integrační složka nuluje. Po přepnutí AM = log.1 se okamžitě provede 1. krok regulace.

Havarijní režim

Dojde-li k chybě měření (první výskyt chyby), je hodnota parametru *DefOut* dosazena do proměnné *Output* a regulátor se přepne do ručního režimu. Při trvalém chybovém stavu tato hodnota zůstává stejná, regulátor se ovládá v ručním režimu.

Je-li hodnota *DefOut* větší než 10000, tak při chybě zůstává v *Output* poslední hodnota akčního zásahu. Po přechodu do ručního režimu je akční zásah řízen hodnotou *Output*. Chyba měření se indikuje po ukončení instrukce na vrcholu zásobníku A0 v bitech detekce chyby měření EY1 a EY2. V proměnné Status je uchován příznak výskytu chyby měření v bitech EY1 nebo EY2. Bity se nastaví při chybě měření a shodí se pouze po studeném startu regulátoru (nastavením bitu RC na log.1). Obsluha bitů EY2 je samozřejmě aktivní jen v případě použití pomocného vstupu *Input2*.

Kaskádní řízení (KC = log.1)

Input3 (y_3 - předávaná ve vrstvě A2 zásobníku) slouží jako třetí měřená veličina pro měření polohy ventilu. V tomto případě je podle této hodnoty korigován akční zásah hlavní smyčky, je-li nastaven řídící bit KC (kaskádní řízení) v řídícím slově Control. Měřená hodnota odporového vysílače měřená analogovou jednotkou musí přímo vyjadřovat otevření ventilu v desetinách promile., tj. mít rozsah 0 až 10000. To lze snadno zajistit pomocí instrukce **CNV**. Poloha ventilu je v tomto případě dostupná v proměnné *LastOut*.

Je-li měřený vstup y_3 chybný, regulátor do havarijního stavu nepřechází. Pouze vynuluje bit KC v řídícím slově a pokračuje v řízení bez odměřování polohy ventilu. Zároveň je nastaven chybový bit EY3 v registru *Status*, resp. na vrcholu zásobníku A0.

Algoritmy regulátoru

Řídící bit: *Control* - IP

Regulátor pracuje jako přímý (polohový) nebo přírůstkový.

Přímý algoritmus

Přímý algoritmus (IP = log.0) je klasický algoritmus doplněný o jednoduchá nelineární pásma, která mohou zabránit nežádoucí reakci regulátoru v netypických situacích, např. po zapnutí regulovaného obvodu.

Pásmo necitlivosti je vhodné nastavovat podle odhadu hodnoty rozptylu měření. (Může významně ovlivnit ustálené stavy.)

V proměnné *CurOut* je vracen výstup skutečně žádaný.

Přírůstkový algoritmus

Přírůstkový algoritmus (IP = log.1) vrací v proměnné *CurOut* přírůstek nového akčního zásahu. V proměnné *Output* je stále udržována přímá hodnota akčního zásahu. Přírůstkový algoritmus je určen pro řízení astatických soustav (zejména s polohovým ventilem). I při ručním řízení se stále zadává přímá hodnota akčního zásahu, to znamená, že regulátor sleduje akční zásah až za přenosem s nulovým pólem, který je uvažován jako součást regulátoru.

Např. je-li řízen polohový ventil, i bez odporového snímače jeho polohy je stále k dispozici odhad výstupu ventilu. Jedná se v tomto případě skutečně jen o odhad a při ručním řízení je nutné vždy provést kalibraci výstupní hodnoty podle skutečné polohy ventilu (viz dále).

Přechody mezi oběma typy řídících algoritmů by se měly provádět v ručním režimu.

Výstupy regulátoru

Řídící bity:	*Control*	- BU, KC, A12, AO
Diagnostické bity:	*Status*	- U–, UH, UC
	A0	- UMX, UMN

Výstup může být unifikovaný, spojitý, realizovaný pomocí analogové výstupní jednotky nebo binární, časově proporcionální on / off, realizovaný pomocí binární výstupní jednotky.

Spojitý výstup regulátoru (bit BU = log.0) bez ohledu na režim a algoritmus vrací v proměnné *ConOut* **absolutní hodnotu** akčního zásahu. Je tedy možné podle bitu U–, indikujícího záporné znaménko akčního zásahu ovládat dva analogové výstupy, apod.

Parametr *OutCycle* zde má význam periody vzorkování. Výstupní hodnota je vždy omezena v rozsahu 0 ÷ 10000. V případě nastavení bitu A12 = log.1 v proměnné *Control* je výstupní hodnota normována na rozsah 4095 tj. 12 bitů. Je-li nastaven bit AO = log.1, je provedena transformace rozsahu 0 ÷ 10000 na rozsah 2000 ÷ 10000. Výsledek je v obou případech uložen do proměnné *ConOut*. Tak je možné přímo ovládat unifikované výstupy analogových jednotek.

Binární výstup on / off (bit BU = log.1), časově proporcionální, se používá pro přímé ovládání akčního orgánu. Zde *OutCycle* je hodnota výstupního cyklu, tj. perioda opakovacího kmitočtu. Při přírůstkovém řízení je měřena doba skutečného sepnutí v jednom výstupním cyklu a po jejím uplynutí je automaticky korigována hodnota změny akčního zásahu.

Pro větší rozlišení při přírůstkovém řízení je možné použít omezení přírůstku akčního zásahu. Tím je totiž možné dosáhnout toho, že za čas výstupního cyklu se vždy zrealizuje maximálně tato hodnota změny.

Je-li použito kaskádní řízení servoventilu, je proměnnými *dMaxU* a *OutCycle* určena rychlost posuvu polohového servoventilu.

Např. *dMaxU* = 1000 tj. 10% a *OutCycle* = 1000 tj. 10,0 s.

V tomto případě je zadána rychlost ventilu 1% za sekundu, tj. doba přeběhu ventilu je 100 s. Je-li střední doba cyklu PLC 100 ms je takto realizované rozlišení 0,1%.

Postup kalibrace servoventilu bez odporového vysílače polohy

Proměnná *Control* = $10

Regulátor přímý s rozsahem –10000 ÷ +10000, *dMaxU* = 10000. V ručním režimu nastavíme polohu ventilu spínáním pomocí ručně zadaných hodnot –10000 nebo +10000.

Proměnná *Control* = 0

Do proměnné *Output* zapíšeme hodnotu polohy ventilu s přesností na setinu procenta a nastavíme *MinU* = 0. Tento krok je nutný z důvodu potlačení nežádoucích řídících pulzů do ventilu.

Proměnná *Control* = $30

Je nastaven přírůstkový regulátor s rozsahem 0 ÷ +10000. Nyní je možné ručně zadávat do *Output* žádané polohy ventilu.

Změny jsou dostupné až po uplynutí *OutCycle*! Z tohoto důvodu je je třeba nastavit do *OutCycle* malou hodnotu.

Příklad propojení instrukcí CNV a PID

Regulujeme teplotu měřenou odporovým teploměrem Ni1000 připojeným k jednotce IT-12 NS950. Pro regulaci je nutná filtrace vstupu. Použitý akční orgán je servoventil s odměřováním polohy, odporový vysílač (OV) 0 až 200 Ω. Údaj OV je také filtrován. Regulátor je nastaven jako kaskádní, přírůstkový s binárním řízením (ventil je zapojen do kaskády).

```
#program Kaskada
;
#def vystup0 %X1.0
#def vystup1 %X1.1
;
;Data pro CNV, měření polohy odporového vysílače
#reg float AuxDR;stavová proměnná filtru
#reg word  MinR,;Minimální hodnota odporu naměřena
           MaxR,;Maximální hodnota odporu naměřena
           MinV,;Minimální hodnota otevření ventilu v desetinách promile
           MaxV ;Maximální hodnota otevření ventilu v desetinách promile
;
;Data pro CNV, měření teploty
#reg float AuxDT;stavová proměnná filtru
;
;Data pro PID
#reg word  MinY,
           MaxY,
           InPut1,
           gW,
           ConW,
           tiW,
           Dev,
           Output,
```

```
          LastOut,
          CurOut,
          ConOut,
          DefOut,
          MinU,
          MaxU,
          dMaxU,
          OutCycle,
          PBnd,
          RelCool,
          Ti,
          Td,
          EGap,
          DGap,
          IGap,
          Control
#reg byte  Status
#reg byte  AuxD[23]
;
;Inicializace regulace teploty:
P 63
;inicializace měření odporového vysílače
     LD   10
     WR   MinR
     LD   2200
     WR   MaxR
     LD   0
     WR   MinV
     LD   10000
     WR   MaxV
;inicializace stavové proměnné filtru odporového vysílače
     LD   __indx (AuxDR)
     LD   0          ;tau = 0!
     LD   __indx (MinR)
     LD   %111       ;filtrace + měřítkování
     LD   word adata+2
     LD   1220       ;IT-12, odpor 0 az 500 Ohm
     CNV
;inicializace stavové proměnné filtru měřené teploty
     LD   __indx (AuxDT)
     LD   0          ;tau = 0!
     LD   __indx (MinY)
     LD   %101       ;filtrace + přepis do struktury PID
     LD   word adata
     LD   1232       ;IT-12, Ni1000
     CNV
;inicializace PID
;(Hodnoty MinY=-500 a MaxY=1520 jsou zadány voláním CNV pro měření
;teploty.)
     LD   0          ;rozsah výstupních hodnot
     WR   MinU
     LD   10000
     WR   MaxU
     ;
     LD   1000       ;Definice rychlosti přeběhu ventilu
     WR   dMaxU      ;10,00% (maximální povolený
     LD   1000       ;přírůstek polohy ventilu) za 10 s,
     WR   OutCycle   ;tj. doba přeběhu ventilu je 100 s.
```

```
      ;
      LD    11000       ;DefOut>10000, tj. po havárii
      WR    DefOut      ;zůstává hodnota výstupu nezměněna.
      ;                 ;Regulátor se přepne do ručního režimu.
      ;
      LD    500         ;žádaná teplota 50,0°C (na desetiny)
      WR    gW
      ;
;Nastavení parametrů regulátoru:
      LD    1000        ;Pásmo proporcionality 100,0%, tj. zesílení 1
      WR    Pbnd
      LD    1000        ;Druhé pásmo proporcionality 100,0%
      WR    RelCool
      LD    0
      WR    Ti          ;Bez integrace.
      LD    10
      WR    Td          ;Derivační složka 1,0 s
      LD    10
      WR    EGap        ;Pásmo necitlivosti 0,1%
      LD    10000
      WR    DGap        ;Derivace povolena v celém rozsahu
      LD    0           ;Bez integrace
      WR    IGap
      ;
      LD    %01111001   ;Řídící slovo: přírůstkové řízení ventilu v kaskádě
      WR    Control
E 63
;
P 0
;měření teploty
      LD    __indx (AuxDT)
      LD    150         ;tau = 150
      LD    __indx (MinY)
      LD    %101        ;filtrace + přepis do struktury PID
      LD    word adata
      LD    1232        ;IT-12, Ni1000
      CNV
      WR    Input1      ;zápis do struktury PID
      ;
;měření odporového vysílače
      LD    __indx (AuxDR)
      LD    60          ;tau = 60
      LD    __indx (MinR)
      LD    %111        ;filtrace + měřítkování
      LD    word adata+2
      LD    1220        ;IT-12, odpor 0 až 500 Ohm
      CNV
      ;                 ;vrací v [A0] měřenou polohu, pro PID Input3=[A2]
      LD    0           ;nevyužitá hodnota Input2=[A1]
      LD    __indx (MinY)   ;index datové struktury PID=[A0]
      PID
      LD    Status.0
      WR    vystup0     ;více otevři ventil
      LD    Status.1
      WR    vystup1     ;méně otevři ventil
E 0
```

13. OPERACE SE ZNAKY ASCII

BAS Převod z binárního formátu do ASCII

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
BAS								VAL									ASCII	

VAL - čtyřciferné číslo v binárním formátu nebo v BCD (typ word)
ASCII - čtveřice ASCII znaků

Operandy

		word
BAS	bez operandu	B D

Funkce

BAS - převod čísla word v binárním formátu na 4 ASCII znaky

Popis

Instrukce **BAS** zpracovává vrchol zásobníku A0 jako čtyřciferné číslo, které převede na čtyři ASCII znaky. Instrukce posune zásobník o jednu úroveň vpřed a uloží ASCII znaky na vrstvy A0 a A1 tak, že nejvyšší číslice je uložena v dolním bytu A0, nejnižší v horním bytu A1, tedy opačně než je tomu v binárním formátu. Tento formát umožňuje zápis čtveřice ASCII znaků naráz instrukcí **WR** RLn do zápisníku, odkud se tento řetězec bude například zobrazovat na displej.

Poznámka

Instrukce **BAS** pracuje s číslicemi hexadecimálního formátu 0 až F. Pokud chceme zobrazit číslo dekadicky, musíme jej nejprve převést do BCD formátu instrukcí **BCD** nebo **BCL**.

Příklad

Převod binárního čísla do BCD a na ASCII

```
#reg word DesetL      ;nižší 4 číslice BCD (4. až 1.)
#reg word Binar
#reg byte ASCII[5]
;
P 0
      LD    Binar
      BCD
      WR    DesetL
      LD    %S0
      ROL   12
      AND   $0007
      BAS
      WR    ASCII           ;nejvyšší 5. číslice
      LD    DesetL
      BAS
      WR    long ASCII+1    ;4. až 1. číslice
E 0
```

ASB Převod z ASCII do binárního formátu

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
ASB								ASCII									VAL	

ASCII - čtveřice ASCII znaků
VAL - čtyřciferné číslo v binárním formátu nebo v BCD (typ word)

Operandy

		word
ASB	bez operandu	B D

Funkce

ASB - převod čísla z ASCII znaků do binárního formátu

Popis

Instrukce **ASB** zpracovává vrstvy A0 a A1 zásobníku jako čtyři ASCII znaky, nejvyšší číslice je uložena v dolním bytu A0, nejnižší v horním bytu A1, které převede na binární číslo. Instrukce posune zásobník o jednu úroveň vzad a uloží binární číslo na vrchol zásobníku A0.

Poznámka

Instrukce **ASB** pracuje s číslicemi hexadecimálního formátu 0 až F. Pokud bylo v ASCII znacích uloženo dekadické číslo, získáme převodem instrukcí **ASB** číslo v BCD formátu. Chceme-li toto číslo dále zpracovávat, musíme jej nejprve převést do dvojkové soustavy instrukcí **BIN** nebo **BIL**.

Příklad

Převod dekadického čísla v BCD kódu v ASCII znacích na binární číslo

```
#reg word Binar
#reg byte ASCII[4]         ;4 číslice
;
P 0
      LD    long ASCII
      ASB
      BIN
      WR    Binar
E 0
```

STF Převod ASCII řetězce na float

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
STF							REG	LEN								VAL		

REG - index registru R, ve kterém je uložen první znak řetězce
LEN - délka řetězce znaků (počet zaplněných registrů R)
VAL - převedená číselná hodnota

Operandy

		float
STF	bez operandu	B D

Funkce

STF - převod řetězce ASCII znaků na hodnotu typu float

Popis

Instrukce **STF** očekává ve vrstvě A1 zásobníku číslo registru R, ve kterém začíná ASCII řetězec délky zadané ve vrstvě A0 zásobníku, a převede jej na typ float. Výsledek je uložen na vrchol zásobníku A01. Ostatní vrstvy zásobníku se nemění.

Instrukce připouští ASCII řetězec ve tvarech podle konvence jazyka C, např.:

1.15 - číslo 1,15
–45 - číslo –45
1.5e3 - číslo 1,5 x 10^3, neboli 1500
2.48e–4 - číslo 2,48 x 10^{-4}, neboli 0,000248

V jazyce C je každý ASCII řetězec povinně zakončen hodnotou 0 (hodnota $00, nikoli ASCII kód číslice 0). Pokud má zpracovávaný ASCII řetězec délku, která přesně odpovídá parametru LEN, není třeba tuto koncovou 0 zadávat. Naopak v případech, kdy se zpracovávají ASCII řetězce různých délek a parametr LEN udává pouze délku maximální, je připojení 0 na konec řetězce žádoucí.

Instrukce **STF** uznává jako konec řetězce kromě hodnoty 0 i ASCII znak mezery (hodnota $20). Přípustné ASCII znaky v řetězci jsou ‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, ‘+’, ‘–’, ‘.’, ‘e’, ‘E’.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	S

S1.0 (S) - 1 - výsledek je platný
0 - chyba v řetězci, výsledek je neplatný

Příklad

Převod řetězce na číslo typu float

```
#def LEN 10
#reg byte Zona[LEN]
#reg float VAL
;
P 0
      LD    __indx (Zona)   ;REG
      LD    LEN
      STF
      WR    VAL
E 0
```

FST Převod float na ASCII řetězec

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
FST					VAL		REG	LEN		REG	LEN	A7	A6	A5	A4	VAL		

VAL - převáděná číselná hodnota
REG - index registru R, ve kterém je uložen první znak řetězce
LEN - délka řetězce znaků (počet zaplněných registrů R)

Operandy

		float
FST	bez operandu	B D

Funkce

FST - převod hodnoty ve formátu float na řetězec ASCII znaků

Popis

Instrukce **FST** očekává ve vrstvě A1 zásobníku číslo registru R, ve kterém začíná pole délky zadané ve vrstvě A0 zásobníku. Ve dvojvrstvě A23 očekává hodnotu typu float. Tato hodnota je převedena na řetězec ASCII znaků, který je uložen do pole registrů R, jehož parametry udávají vrstvy A1 a A0. Zásobník se posune o dvě úrovně zpět, čímž se na vrchol zásobníku opět vrátí převáděná hodnota a lze ji tedy použít k dalšímu zpracování. Instrukce tak umožňují zobrazování i různých mezivýsledků.

Instrukce vytváří ASCII řetězec ve tvarech podle konvence jazyka C, např.:

1.15 - číslo 1,15
–45 - číslo –45
1.5e+03 - číslo 1,5 x 10^3, neboli 1500
2.48e–04 - číslo 2,48 x 10^{-4}, neboli 0,000248

Pokud je výsledný řetězec kratší, než udává parametr LEN, je doplněn ASCII kódy mezery ($20). Pokud je parametrem LEN zadána tak malá délka řetězce, že výsledné číslo nelze zobrazit (nevejde se exponent), je pole registrů určené k zápisu řetězce vyplněno ASCII kódy písmene X ($58).

Příklad

Zobrazení mezivýsledků

```
#def LEN 16          ;délka řádku displeje
#reg byte Radek1[LEN], Radek2[LEN]
#reg float va, vb, vc
;
P 0
     LD   va
     MUF  vb              ;VAL = a.b
     LD   __indx (Radek1) ;REG
     LD   LEN
     FST
     DIF  vc              ;VAL = (a.b)/c
     LD   __indx (Radek2) ;REG
     LD   LEN
     FST
E 0
```

14. SYSTÉMOVÉ INSTRUKCE

HPE	**Zapnutí okamžitého přístupu do zápisníku pro komunikace**
HPD	**Vypnutí okamžitého přístupu do zápisníku pro komunikace**

Operandy

HPE	bez operandu	B D
HPD	bez operandu	B D

Funkce

HPE - zapnutí okamžitého přístupu do zápisníku pro komunikace
HPD - vypnutí okamžitého přístupu do zápisníku pro komunikace

Časové poměry výměny dat

Vizualizační programy používají pro výměnu dat s PLC datové služby režimu PC - READN, READND, WRITEN, WANDRN, WANDRND, READB, READBD a WRITEB (viz příručka Sériová komunikace programovatelných automatů TECOMAT a regulátorů TECOREG TXV 001 06.01 - kap.8.5.3.).

Standardně tyto komunikace mají tzv. nízkou prioritu (low priority), což znamená, že po přijetí služby PLC počká na dokončení smyčky uživatelského programu a v otočce provede požadovanou akci (zápis nebo čtení dat). Pak zahájí vysílání odpovědi.

Výhodou tohoto přístupu je zaručená časová konzistence dat, tj. že všechna přečtená data pocházejí ze stejného časového okamžiku. Dále je zde zaručen princip neměnnosti obsahu zápisníku během smyčky uživatelského programu jiným způsobem než činností uživatelského programu samého.

Nevýhodou je zpoždění odpovědi PLC až o dobu cyklu, což při komunikaci s více systémy může znamenat znatelný pokles propustnosti linky.

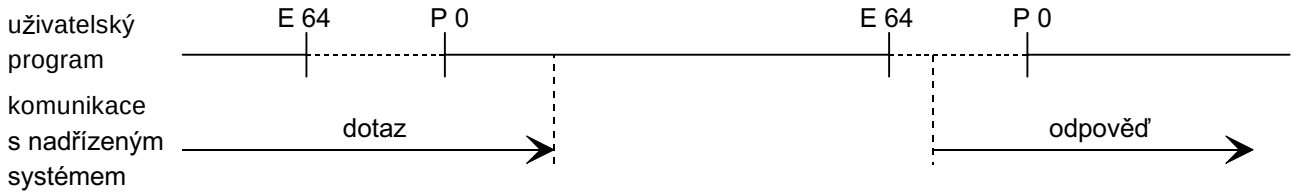
Příklad

Požadujeme zápis 10 bytů do PLC a čtení 10 bytů z PLC.

Máme-li sériový kanál PLC nastaven v režimu PC, rychlost 19,2 kBd, prodleva odpovědi 0, a doba cyklu PLC je 100 ms, pak skutečná prodleva odpovědi je v rozmezí 0,57 ms až 100 ms. Vlastní komunikace trvá 13,11 ms dotaz (23 bytů včetně rámce a klidového stavu před zahájením vysílání) a 10,83 ms odpověď (19 bytů včetně rámce).

Výsledný komunikační cyklus je tedy součet trvání přenosu dotazu, prodlevy odpovědi a přenosu odpovědi. Výsledná hodnota se pohybuje mezi 24,51 ms a 123,94 ms.

Pokud máme na jedné lince připojeno 10 takovýchto PLC, pak je výsledný komunikační cyklus desetinásobný, tedy 245,1 ms až 1239,4 ms. Data ve vizualizaci se tak budou obnovovat zhruba jedenkrát za sekundu, což je v mnoha případech nedostatečné.



Obr.13.1 Klasická komunikace s nízkou prioritou

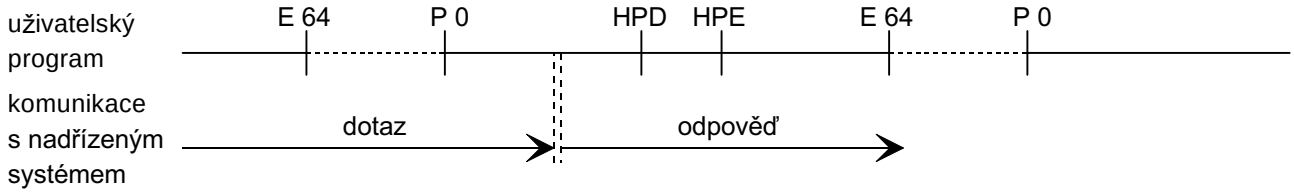
Použití instrukcí HPE, HPD

Uvážíme-li, že zpracování dat, která zapisujeme a čteme po sériové lince, zpravidla zabírá v uživatelském programu jeho nepatrnou část, můžeme tento problém řešit pomocí systémových instrukcí **HPE** a **HPD**. Instrukce **HPE** (High priority enable) nastaví vysokou prioritu (high priority) těchto komunikací, což má za následek, že bezprostředně po vyhodnocení dotazu PLC připraví odpověď a odvysílá ji bez ohledu na to, kterou část programu právě vykonává.

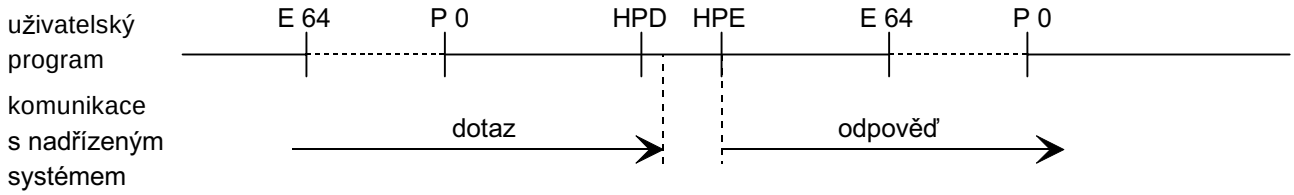
Komunikace s vysokou prioritou však způsobuje problémy, protože nezaručuje časovou konzistenci čtených dat a data zapisuje a čte kdykoliv během výkonu uživatelského programu i uprostřed instrukce!

K ošetření možných hazardních stavů lze použít instrukci **HPD** (High priority disable), která nastaví nízkou prioritu a zaručí, že od tohoto okamžiku nedojde ke změně dat v zápisníku vlivem sériové komunikace. Pak mohou následovat instrukce, které zpracují data v příslušné oblasti, případně provedou jejich aktualizaci, a nakonec pomocí instrukce **HPE** opět nastavíme vysokou prioritu. Pokud byl přijat nějaký dotaz v době, kdy byla nastavena nízká priorita, je v okamžiku změny priority na vysokou připravena odpověď a odvysílána. Maximální prodleva doby odpovědi je tak zkrácena na časový úsek mezi instrukcemi **HPD** a **HPE**.

Data pro vizualizaci z výše uvedeného příkladu by se při použití instrukcí **HPD**, **HPE** obnovovala cca čtyřikrát za sekundu.



Obr.13.2 Komunikace s vysokou prioritou



Obr.13.3 Komunikace s přepínanou prioritou

Doporučujeme vyhradit pro výměnu dat zvláštní komunikační zónu v zápisníku, které bude uživatelský program zpracovávat a aktualizovat jen na jednom místě. Pokud je třeba do komunikační zóny přistupovat víckrát, je lepší vytvořit tzv. stínovou komunikační zónu, se kterou bude pracovat uživatelský program a která se jednou za cyklus sesouhlasí se skutečnou komunikační zónou.

Příklad

```
#def delkadovnitr 10
#def delkaven 10
#reg byte stindovnitr[delkadovnitr],stinven[delkaven]
#reg byte dovnitr[delkadovnitr],ven[delkaven]
;
P 0
     :
     :               ;uživatelský program pracuje výlučně
     :               ;se zónami stindovnitr a stinven
     :
```

```
      HPD             ;nízká priorita
      LD    0
      SRC   dovnitr
      LD    0
      LD    delkadovnitr
      MOV   stindovnitr     ;kopie přijímací zóny
      LD    0
      SRC   stinven
      LD    0
      LD    delkaven
      MOV   ven         ;kopie vysílací zóny
      HPE               ;vysoká priorita
      :
      :                 ;uživatelský program pracuje výlučně
      :                 ;se zónami stindovnitr a stinven
      :
E 0
;
P 63
      :
      HPE               ;zapnutí vysoké priority
      :
E 63
```

RDT	**Čtení současného času z RTC**
WRT	**Nastavení času do RTC**

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
RDT								REG									REG	
WRT								REG									REG	

REG - index prvního registru R časové zóny, ve které je uložen časový údaj (viz dále)

Operandy

RDT	bez operandu	B D
WRT	bez operandu	B D

Funkce

RDT - čtení současného času přímo z obvodu reálného času (RTC) centrální jednotky
WRT - nastavení času v obvodu reálného času (RTC) centrální jednotky

Popis

Instrukce **RDT** slouží k vytváření přesných časových značek k určité události (obvykle zpracovávané v přerušovacím procesu). Zatímco v registrech S5 až S12 je čas aktualizován vždy v otočce cyklu a během zpracovávání uživatelského programu je tedy neměnný, instrukce **RDT** čte aktuální čas přímo z obvodu reálného času centrální jednotky a provádí synchronizační korekci (viz upozornění).

Časová zóna v zápisníku má následující strukturu:

index registru	časový údaj	rozsah
REG	rok	0 - 99
REG+1	měsíc	1 - 12
REG+2	den	1 - 28 / 29 / 30 / 31 (podle měsíce a roku)
REG+3	hodina	0 - 23
REG+4	minuta	0 - 59
REG+5	sekunda	0 - 59
REG+6	den v týdnu	1 - 7
REG+7, +8	milisekunda	0 - 999 (v pořadí dolní byte, horní byte)

Všechny časové údaje jsou ukládány v binárním kódu.

Upozornění: Časový údaj čtený instrukcí **RDT** je synchronizován, to znamená, že okamžikem zápisu času do RTC buď instrukcí **WRT** nebo komunikační službou po sériové lince je vynulován údaj milisekund. Naproti tomu v registrech S5 až S12 je čas průběžný, to znamená, že milisekundy se nenulují. Tento čas je oproti časové značce čtené instrukcí **RDT** posunut o 0 až 999 ms. Proto není vhodné používat souběžně oba časové údaje.

Instrukce **WRT** slouží k přenastavení obvodu reálného času centrální jednotky. To má význam zejména pro změnu času z letního na zimní a naopak, případně pro synchronizaci času s vnějším signálem.

Časová zóna v zápisníku má následující strukturu:

index registru	časový údaj	rozsah
REG	rok	0 - 99
REG+1	měsíc	1 - 12
REG+2	den	1 - 28 / 29 / 30 / 31 (podle měsíce a roku)
REG+3	hodina	0 - 23
REG+4	minuta	0 - 59
REG+5	sekunda	0 - 59
REG+6	den v týdnu	1 - 7

Všechny časové údaje jsou ukládány v binárním kódu.

Příklad

```
#struct cas
      byte rok, byte mesic, byte den, byte hod, byte min,
      byte sek, byte denvt, word milisek
#reg cas znacka
;
P 0
      :
E 0
;
P 42
      :                ;přerušení od periferie
      LD   __indx (znacka)
      RDT              ;zápis přesné časové značky do proměnné znacka
      :
E 42
```

RDB	**Čtení z DataBoxu**
WDB	**Zápis do DataBoxu**
IDB	**Identifikace DataBoxu**

Instrukce	Vstupní parametry									Výsledek								
	zásobník									zásobník								
	A7	A6	A5	A4	A3	A2	A1	A0		A7	A6	A5	A4	A3	A2	A1	A0	
RDB								REG									LEN	
WDB								REG									LEN	
IDB										A6	A5	A4	A3	A2	A1	A0	SIZE	

REG - index prvního registru R parametrické zóny (viz dále)
LEN - počet přenesených bytů
SIZE - velikost DataBoxu v KB

Operandy

RDB	bez operandu	B D
WDB	bez operandu	B D
IDB	bez operandu	B D

Funkce

RDB - čtení bloku dat z přídavné paměti DataBox
WDB - zápis bloku dat do přídavné paměti DataBox
IDB - zjištění velikosti DataBoxu

Popis

Pro zjištění velikosti osazeného DataBoxu je určena instrukce **IDB**. Tato instrukce nevyžaduje žádné vstupní parametry. Po vykonání zvýší uživatelský zásobník o jednu úroveň a na vrchol zásobníku zapíše zjištěnou velikost DataBoxu v KB, tzn. např. hodnotu 256. Pokud není DataBox nalezen, vrací instrukce hodnotu 0.

Před voláním instrukcí **RDB** a **WDB** je nutno nastavit několik parametrů. Tyto parametry jsou umístěny v registrech R, musí být uloženy těsně za sebou a jejich pořadí je nutné dodržet. Číslo registru, ve kterém je umístěn první parametr, se předává na zásobníku při volání instrukce **RDB** a **WDB** (viz dále). Parametry jsou seřazeny v následujícím pořadí:

Název parametru	Typ	Význam
adrDB	long	adresa v paměti DataBox
indR	word	index počátečního registru v zápisníku
len	byte	počet přenášených bytů

Instrukce **RDB** a **WDB** nemění úroveň uživatelského zásobníku. Na vrcholu zásobníku vrací počet skutečně přenesených dat. Zároveň nastavují obsah systémového registru S1.0 na log.1, pokud je výsledek je platný.

Je-li S1.0 = log.0, nedojde k přenosu žádných dat a zároveň je do registru S34 zapsána chyba 14 nebo 15 (zdrojový nebo cílový blok dat byl definován mimo rozsah).

Podle velikosti osazené paměti DataBox je pro použití dostupný následující adresový prostor:

Velikost paměti DataBox	Dostupný adresový prostor
128 KB (CPU řady D a S)	0 - $1FFFB
128 KB (CPU řady B)	0 - $1FFFF
512 KB (CPU řady D a S)	0 - $7FFEF
1,5 MB (CPU řady B)	0 - $19FFFF

Při pokusu o čtení nebo zápis mimo tento dostupný prostor je nastaven S1.0 = log.0 a současně je nastaven i příslušný chybový kód v S34.

Příznaky

	.7	.6	.5	.4	.3	.2	.1	.0
S1	-	-	-	-	-	-	-	IS

S1.0 (IS) - 0 - adresa zdrojové zóny v DataBoxu (**RDB**), resp. v zápisníku (**WDB**) nebo cílové zóny v zápisníku (**RDB**), resp. v DataBoxu (**WDB**), je mimo rozsah, přesun se neprovede
1 - adresa zdrojové i cílové zóny je v rozsahu DataBoxu nebo zápisníku, přesun se provede

S34 = 20 ($14) zdrojový blok dat byl definován mimo rozsah
S34 = 21 ($15) cílový blok dat byl definován mimo rozsah

Příklad

```
#struct parDB              ;jméno struktury
      long adrDB,          ;adresa v DataBoxu
      word indR,           ;index počátečního registru v zápisníku
      byte len             ;počet přenášených bytů
#reg parDB parusi
#def lenDat 56
#reg byte blokDat[lenDat]
#reg bit  DataBoxOK        ;příznak DataBox v pořádku
;
P 63
      :
      LD    32             ;požadovaná velikost DataBoxu pro aplikaci
      IDB                  ;identifikace velikosti DataBoxu
      GT
      NEG                  ;DataBox aspoň požadované velikosti?
      WR    DataBoxOK      ;nastavit příznak
      :
E 63
;
P 0
      :
      LD    DataBoxOK      ;DataBox v pořádku ?
      JMC   konecDBX       ;ne
      LDL   $FC00          ;adresa v DataBoxu (long !!!)
      WR    parusi~adrDB
      LD    __indx (blokDat);do kterého reg. se přenesou data z DataBoxu
      WR    parusi~indR
      LD    lenDat         ;počet přenášených bytů
      WR    parusi~len
      LD    __indx (parusi) ;číslo registru, kde leží parametry
      RDB                  ;čtení bloku dat z DataBoxu do zápisníku
konecDBX:                  ;blok o délce 56 bytů se přečte z adresy $FC00
      :                    ;a uloží se do pole blokDat
E 0
```

REI Reinicializace periferních modulů

Operandy

REI	bez operandu	B D

Funkce

REI - reinicializace periferních modulů

Popis

Instrukce **REI** předá PLC požadavek na opětovnou inicializaci periferních modulů, která se provede v následující otočce cyklu. Obsah zásobníku zůstává zachován beze změny. Pokud potřebujeme za chodu změnit např. měřicí rozsah analogového modulu, zapíšeme do její inicializační tabulky nové parametry a zavoláme instrukci **REI**.

Pozor! Je třeba si uvědomit, že budou inicializovány všechny periferní moduly. Zkontrolujte si v příslušné dokumentaci chování použitých modulů při restartu. Některé provádějí vnitřní reset. U periferních modulů, které používají pro výměnu dat s centrální jednotkou tzv. alternační bity, je nutné před reinicializací **vynulovat všechny řídicí byty**! Moduly očekávají po restartu v těchto bytech nulové hodnoty jako výchozí. Týká se to systémových sériových kanálů v režimu **uni**, komunikačních jednotek SC-11, CD-xx, UP-xx a polohovacích jednotek GT-40, GT-41.

PŘEHLED INSTRUKCÍ

PŘEHLED INSTRUKCÍ S PŘÍPUSTNÝMI OPERANDY

Přehled použitých symbolů operandů:

Z - zápisník X, Y, S, D, R
T - tabulky
A - bez operandu (pracuje pouze na uživatelském zásobníku)
n - číselný parametr
- konstanta
Ln - návěští číslo n

Instrukce pro čtení a zápis dat

Mnemo kód	Typ operandu bit	byte	word	long	float	Význam instrukce	Popis na str.
LD	Z	Z U	Z # U	Z	Z	Čtení přímých dat	8
LDL				#	#	Čtení přímých dat	8
LDC	Z	Z	Z #	Z		Čtení negovaných dat	8
WR	Z	Z U	Z U	Z	Z	Zápis přímých dat	11
WRC	Z	Z	Z	Z		Zápis negovaných dat	11
WRA		Z	Z	Z		Zápis přímých dat s alternací	14
PUT	Z	Z	Z	Z	Z	Podmíněný zápis dat	16

Logické instrukce

Mnemo kód	Typ operandu bit	byte	word	long	Význam instrukce	Popis na str.
AND	Z	Z	Z # A		AND s přímým operandem	18
ANL				# A	AND s přímým operandem	18
ANC	Z	Z	Z		AND s negovaným operandem	18
OR	Z	Z	Z # A		OR s přímým operandem	21
ORL				# A	OR s přímým operandem	21
ORC	Z	Z	Z		OR s negovaným operandem	21
XOR	Z	Z	Z # A		XOR s přímým operandem	24
XOL				# A	XOR s přímým operandem	24
XOC	Z	Z	Z		XOR s negovaným operandem	24
NEG			A		Negace vrcholu uživatelského zásobníku	27
NGL				A	Negace vrcholu uživatelského zásobníku	27
SET	Z	Z	Z		Podmíněné nastavení	28
RES	Z	Z	Z		Podmíněné nulování	28
LET	Z	Z	Z		Impulz od náběžné hrany	30
BET	Z	Z	Z		Impulz od libovolné hrany	30
FLG			A		Logické AND všech bitů a příčné funkce bytů A0 v S1	32
STK				A	Sklopení logických hodnot úrovní zásobníku do A0	34
ROL n			A		Rotace hodnoty vlevo n-krát	35
ROR n			A		Rotace hodnoty vpravo n-krát	35
SWP			A		Záměna dolního a horního bytu A0	37
SWL				A	Záměna vrstev A0 a A1	37

Čítače, posuvné registry, časovače, krokový řadič

Mnemo kód	Typ operandu word	Význam instrukce	Popis na str.
CTU	R	Dopředný čítač	38
CTD	R	Zpětný čítač	38
CNT	R	Obousměrný čítač	38
SFL	R	Posuvný registr vlevo	44
SFR	R	Posuvný registr vpravo	44
TON	R*	Časovač (zpožděný přítah)	46
TOF	R*	Časovač (zpožděný odpad)	46
RTO	R*	Integrující časovač, měřič času	50
IMP	R*	Časovač - generátor impulzu zadané délky	53
STE	R	Krokový řadič (stepper)	55

* Každý z časovačů může být programován s jednotkou inkrementu: .0 - 10ms; .1 - 100ms; .2 - 1s; .3 - 10s

Aritmetické instrukce

Mnemo kód	Typ operandu byte	word	long	Význam instrukce	Popis na str.
ADD		Z # A		Sčítání s přenosem	57
ADX	Z	Z	Z	Sčítání	57
ADL			# A	Sčítání	57
SUB		Z # A		Odčítání s přenosem	59
SUX	Z	Z	Z	Odčítání	59
SUL			# A	Odčítání	59
MUL	Z # A			Násobení (byte x byte = word)	61
MUD		Z # A		Násobení (word x word = long)	61
DIV	Z # A			Dělení se zbytkem (byte / byte = byte)	62
DID		Z # A		Dělení se zbytkem (long / word = long)	62
INR	Z	Z A	Z	Inkrementace (+ 1)	64
DCR	Z	Z A	Z	Dekrementace (– 1)	64
EQ		Z # A		Porovnání (rovnost)	66
LT		Z # A		Porovnání (menší než)	66
GT		Z # A		Porovnání (větší než)	66
CMP	Z	Z # A	Z	Porovnání	68
CML			Z # A	Porovnání	68
BIN		A		Převod čísla z BCD formátu do binárního	69
BIL			A	Převod čísla z BCD formátu do binárního	69
BCD		A		Převod čísla z binárního formátu do BCD	69
BCL			A	Převod čísla z binárního formátu do BCD	69

Operace s uživatelskými zásobníky

Mnemo kód	Operand	Význam instrukce	Popis na str.
POP	n	Posun (rotace) uživatelského zásobníku zpět o n úrovní	71
NXT		Aktivace následujícího uživatelského zásobníku v řadě	72
PRV		Aktivace předcházejícího uživatelského zásobníku v řadě	72
CHG	n	Aktivace zvoleného uživatelského zásobníku (n je 0 až 7)	72
CHGS	n	Aktivace zvoleného uživatelského zásobníku (n je 0 až 7)	72
LAC	n	Načtení hodnoty z vrcholu zvoleného uživatelského zásobníku (n je 0 až 7)	73
WAC	n	Zápis hodnoty na vrchol zvoleného uživatelského zásobníku (n je 0 až 7)	73

Instrukce skoků a volání

Mnemo kód	Operand	Význam instrukce	Popis na str.
JMP	Ln	Nepodmíněný skok	74
JMD	Ln	Skok podmíněný nenulovostí výsledku	74
JMC	Ln	Skok podmíněný nulovostí výsledku	74
JMI	A	Skok na nepřímý cíl	74
JZ	Ln	Skok podmíněný nenulovostí příznaku rovnosti ZR	75
JNZ	Ln	Skok podmíněný nulovostí příznaku rovnosti ZR	75
JC	Ln	Skok podmíněný nenulovostí příznaku přenosu CO	75
JNC	Ln	Skok podmíněný nulovostí příznaku přenosu CO	75
JS	Ln	Skok podmíněný nenulovostí příznaku S1.0	75
JNS	Ln	Skok podmíněný nulovostí příznaku S1.0	75
CAL	Ln	Nepodmíněné volání podprogramu	77
CAD	Ln	Volání podprogramu podmíněné nenulovostí výsledku	77
CAC	Ln	Volání podprogramu podmíněné nulovostí výsledku	77
CAI	A	Volání podprogramu nepřímého cíle	77
RET		Nepodmíněný návrat z podprogramu	78
RED		Návrat z podprogramu podmíněný nenulovostí výsledku	78
REC		Návrat z podprogramu podmíněný nulovostí výsledku	78
L	n	Návěští n (cíl skoků a volání)	79

Organizační instrukce

Mnemo kód	Operand	Význam instrukce	Popis na str.
P	n	Začátek procesu	80
E	n	Nepodmíněný konec procesu	80
ED		Konec procesu při nenulovosti výsledku	80
EC		Konec procesu při nulovosti výsledku	80
EOC		Konec cyklu	80
NOP	n	Prázdná operace	82
BP	n	Ladicí bod	83
SEQ	Ln	Podmíněné přerušení procesu	84

Tabulkové instrukce

Mnemo kód	Typ operandu			Význam instrukce	Popis na str.
	bit	byte	word		
LTB	Z T	Z T	Z T	Čtení položky z tabulky	85
WTB	Z T	Z T	Z T	Zápis položky do tabulky	88
FTB	Z T	Z T	Z T	Hledání položky v tabulce	91
FTM		Z T	Z T	Hledání části položky v tabulce	93
FTS		Z T	Z T	Zařazení položky podle tabulky	95

Blokové operace

Mnemo kód	Operand	Význam instrukce	Popis na str.
SRC	Z T	Specifikace zdroje dat pro přesun	97
MOV	Z T	Přesun bloku dat	97
MTN	A	Přesun tabulky do zápisníku	99
MNT	A	Naplnění tabulky ze zápisníku	99
FIL	Z	Naplnění bloku konstantou	101

Operace se strukturovanými tabulkami

Mnemo kód	Operand	Význam instrukce	Popis na str.
LDS	A	Čtení položky ze strukturované tabulky T	102
WRS	A	Zápis položky do strukturované tabulky T	103
FIS	A	Plnění položky strukturované tabulky v zápisníku	104
FIT	A	Plnění položky strukturované tabulky T	104
FNS	A	Hledání položky strukturované tabulky v zápisníku	106
FNT	A	Hledání položky strukturované tabulky T	106

Aritmetické instrukce v plovoucí řádové čárce

Mnemo kód	Typ operandu float	Význam instrukce	Popis na str.
ADF	Z # A	Sčítání	108
SUF	Z # A	Odčítání	108
MUF	Z # A	Násobení	109
DIF	Z # A	Dělení	109
CMF	Z # A	Porovnání	111
CEI	A	Zaokrouhlení nahoru	112
FLO	A	Zaokrouhlení dolů	112
ABS	A	Absolutní hodnota	112
LOG	A	Dekadický logaritmus	113
LN	A	Přirozený logaritmus	113
EXP	A	Exponenciální funkce	113
POW	A	Obecná mocnina	113
SQR	A	Druhá odmocnina	113
HYP	A	Euklidovská vzdálenost	113
SIN	A	Sinus	115
ASN	A	Arc sinus	115
COS	A	Cosinus	115
ACS	A	Arc cosinus	115
TAN	A	Tangens	115

Aritmetické instrukce v plovoucí řádové čárce

Mnemo kód	Typ operandu float	Význam instrukce	Popis na str.
ATN	A	Arc tangens	115
UWF	A	Převod word bez znaménka na float	117
IWF	A	Převod word se znaménkem na float	117
ULF	A	Převod long bez znaménka na float	117
ILF	A	Převod long se znaménkem na float	117
UFW	A	Převod float na word bez znaménka	118
IFW	A	Převod float na word se znaménkem	118
UFL	A	Převod float na long bez znaménka	118
IFL	A	Převod float na long se znaménkem	118

Instrukce regulátoru PID

Mnemo kód	Operand	Význam instrukce	Popis na str.
CNV	A	Konverze a zpracování dat z analogových jednotek	119
PID	A	PID regulátor	129

Operace se znaky ASCII

Mnemo kód	Typ operandu word	Typ operandu float	Význam instrukce	Popis na str.
BAS	A		Převod čísla z binárního formátu na ASCII	141
ASB	A		Převod čísla z ASCII do binárního formátu	142
STF		A	Převod ASCII řetězce na float	143
FST		A	Převod float na ASCII řetězec	145

Systémové instrukce

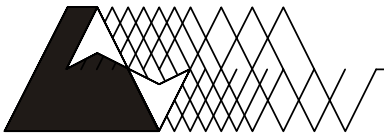
Mnemo kód	Ekvivalent	Význam instrukce	Popis na str.
HPE	SYS 1	Zapnutí okamžitého přístupu do zápisníku pro komunikace	146
HPD	SYS 2	Vypnutí okamžitého přístupu do zápisníku pro komunikace	146
RDT	SYS 3	Čtení současného času z RTC	149
WRT	SYS 4	Nastavení času do RTC	149
RDB	SYS 5	Čtení z DataBoxu	151
WDB	SYS 6	Zápis do DataBoxu	151
IDB	SYS 7	Identifikace DataBoxu	151
REI	SYS 8	Reinicializace periferních modulů	153

ABECEDNÍ SEZNAM INSTRUKCÍ

Mnemo kód	Význam instrukce	Popis na str.
ABS	Absolutní hodnota	112
ACS	Arc cosinus	115
ADD	Sčítání s přenosem	57
ADF	Sčítání v plovoucí řádové čárce	108
ADL	Sčítání	57
ADX	Sčítání	57
ANC	AND s negovaným operandem	18
AND	AND s přímým operandem	18
ANL	AND s přímým operandem	18
ASB	Převod čísla z ASCII do binárního formátu	142
ASN	Arc sinus	115
ATN	Arc tangens	115
BAS	Převod čísla z binárního formátu na ASCII	141
BCD	Převod čísla z binárního formátu do BCD	69
BCL	Převod čísla z binárního formátu do BCD	69
BET	Impulz od libovolné hrany	30
BIL	Převod čísla z BCD do binárního formátu	69
BIN	Převod čísla z BCD do binárního formátu	69
BP	Ladicí bod	83
CAC	Volání podprogramu podmíněné nulovostí výsledku	77
CAD	Volání podprogramu podmíněné nenulovostí výsledku	77
CAI	Volání podprogramu nepřímého cíle	77
CAL	Nepodmíněné volání podprogramu	77
CEI	Zaokrouhlení nahoru	112
CHG	Aktivace zvoleného uživatelského zásobníku	72
CHGS	Aktivace zvoleného uživatelského zásobníku se zálohováním S0 a S1	72
CMF	Porovnání v plovoucí řádové čárce	111
CML	Porovnání	68
CMP	Porovnání	68
CNT	Obousměrný čítač	38
CNV	Konverze dat z analogových jednotek	119
COS	Cosinus	115
CTD	Zpětný čítač	38
CTU	Dopředný čítač	38
DCR	Dekrementace (– 1)	64
DID	Dělení se zbytkem (long / word = long)	62
DIF	Dělení v plovoucí řádové čárce	109
DIV	Dělení se zbytkem (byte / byte = byte)	62
E	Nepodmíněný konec procesu	80
EC	Konec procesu při nulovosti výsledku	80
ED	Konec procesu při nenulovosti výsledku	80
EOC	Konec cyklu	80
EQ	Porovnání (rovnost)	66
EXP	Exponenciální funkce	113
FIL	Naplnění bloku konstantou	101
FIS	Plnění položky strukturované tabulky v zápisníku	104
FIT	Plnění položky strukturované tabulky T	104
FLG	Logické AND všech bitů a příčné funkce bytů A0 v S1	32
FLO	Zaokrouhlení dolů	112
FNS	Hledání položky strukturované tabulky v zápisníku	106
FNT	Hledání položky strukturované tabulky T	106
FST	Převod float na ASCII řetězec	145
FTB	Hledání položky v tabulce	91
FTM	Hledání části položky v tabulce	93
FTS	Zařazení položky podle tabulky	95
GT	Porovnání (větší než)	66
HPD	Vypnutí okamžitého přístupu do zápisníku pro komunikace	146
HPE	Zapnutí okamžitého přístupu do zápisníku pro komunikace	146
HYP	Euklidovská vzdálenost	113

Mnemo kód	Význam instrukce	Popis na str.
IDB	Identifikace DataBoxu	151
IFL	Převod float na long se znaménkem	118
IFW	Převod float na word se znaménkem	118
ILF	Převod long se znaménkem na float	117
IMP	Časovač - generátor impulzu zadané délky	53
INR	Inkrementace (+ 1)	64
IWF	Převod word se znaménkem na float	117
JC	Skok podmíněný nenulovostí příznaku přenosu CO	75
JMC	Skok podmíněný nulovostí výsledku	74
JMD	Skok podmíněný nenulovostí výsledku	74
JMI	Skok na nepřímý cíl	74
JMP	Nepodmíněný skok	74
JNC	Skok podmíněný nulovostí příznaku přenosu CO	75
JNS	Skok podmíněný nulovostí příznaku S1.0	75
JNZ	Skok podmíněný nulovostí příznaku rovnosti ZR	75
JS	Skok podmíněný nenulovostí příznaku S1.0	75
JZ	Skok podmíněný nenulovostí příznaku rovnosti ZR	75
L	Návěští n (cíl skoků a volání)	79
LAC	Načtení hodnoty z vrcholu zvoleného uživatelského zásobníku	73
LD	Čtení přímých dat	8
LDC	Čtení negovaných dat	8
LDL	Čtení přímých dat	8
LDS	Čtení položky ze strukturované tabulky T	102
LET	Impulz od náběžné hrany	30
LN	Přirozený logaritmus	113
LOG	Dekadický logaritmus	113
LT	Porovnání (menší než)	66
LTB	Čtení položky z tabulky	85
MNT	Naplnění tabulky ze zápisníku	99
MOV	Přesun bloku dat	97
MTN	Přesun tabulky do zápisníku	99
MUD	Násobení (word x word = long)	61
MUF	Násobení v plovoucí řádové čárce	109
MUL	Násobení (byte x byte = word)	61
NEG	Negace vrcholu zásobníku	27
NGL	Negace vrcholu zásobníku	27
NOP	Prázdná operace	82
NXT	Aktivace následujícího uživatelského zásobníku v řadě	72
OR	OR s přímým operandem	21
ORC	OR s negovaným operandem	21
ORL	OR s přímým operandem	21
P	Začátek procesu	80
PID	PID regulátor	129
POP	Posun (rotace) uživatelského zásobníku zpět o n úrovní	71
POW	Obecná mocnina	113
PRV	Aktivace předcházejícího uživatelského zásobníku v řadě	72
PUT	Podmíněný zápis dat	16
RDB	Čtení z DataBoxu	151
RDT	Čtení současného času z RTC	149
REC	Návrat z podprogramu podmíněný nulovostí výsledku	78
RED	Návrat z podprogramu podmíněný nenulovostí výsledku	78
REI	Reinicializace periferních modulů	153
RES	Podmíněné nulování	28
RET	Nepodmíněný návrat z podprogramu	78
ROL	Rotace hodnoty vlevo n-krát	35
ROR	Rotace hodnoty vpravo n-krát	35
RTO	Integrující časovač, měřič času	50

Mnemo kód	Význam instrukce	Popis na str.
SEQ	Podmíněné přerušení procesu	84
SET	Podmíněné nastavení	28
SFL	Posuvný registr vlevo	44
SFR	Posuvný registr vpravo	44
SIN	Sinus	115
SQR	Druhá odmocnina	113
SRC	Specifikace zdroje dat pro přesun	97
STE	Krokový řadič (stepper)	55
STF	Převod ASCII řetězce na float	143
STK	Sklopení logických hodnot 8 úrovní zásobníku do A0	34
SUB	Odčítání s přenosem	59
SUF	Odčítání v plovoucí řádové čárce	108
SUL	Odčítání	59
SUX	Odčítání	59
SWL	Záměna vrstev A0 a A1	37
SWP	Záměna dolního a horního bytu A0	37
TAN	Tangens	115
TOF	Časovač (zpožděný odpad)	46
TON	Časovač (zpožděný přítah)	46
UFL	Převod float na long bez znaménka	118
UFW	Převod float na word bez znaménka	118
ULF	Převod long bez znaménka na float	117
UNLK	Obnovení systémových registrů před ukončením podprogramu	84
UWF	Převod word bez znaménka na float	117
WAC	Zápis hodnoty na vrchol zvoleného uživatelského zásobníku	73
WDB	Zápis do DataBoxu	151
WR	Zápis přímých dat	11
WRA	Zápis přímých dat s alternací	14
WRC	Zápis negovaných dat	11
WRS	Zápis položky do strukturované tabulky T	103
WRT	Nastavení času do RTC	149
WTB	Zápis položky do tabulky	88
XOC	XOR s negovaným operandem	24
XOL	XOR s přímým operandem	24
XOR	XOR s přímým operandem	24



teco

Objednávky a informace:
Teco a. s. Havlíčkova 260, 280 58 Kolín 4, tel. 321 737 611, fax 321 737 633

TXV 001 05.01