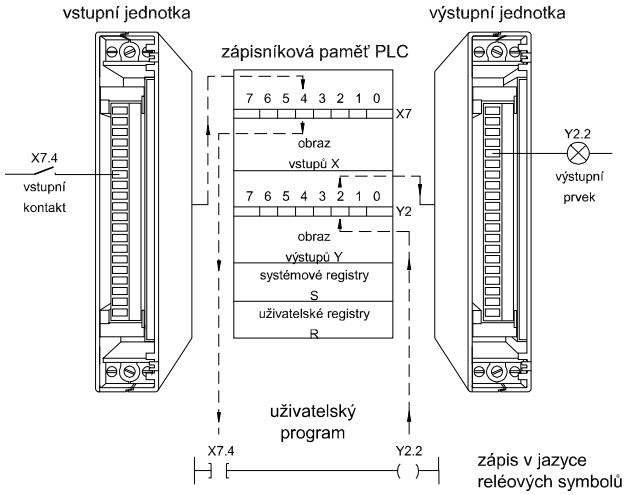
PROGRAMOVATELNÉ AUTOMATY TECOMAT®

Příručka programátora



Ing. Luděk Kohout

I. OBECNÉ PRINCIPY PROGRAMOVATELNÝCH AUTOMATŮ

1. ÚVOD

Jsou obory lidské činnosti, ve kterých vývoj probíhá pomalu, obory, které kladou malé nároky na inovaci techniky i znalostí člověka. Na druhé straně existují oblasti, které se rozvíjí extrémně rychle, kde několik let znamená již celou generaci ve vývoji technických prostředků. Automatizační technika rychlostí inovačních cyklů patří právě do této skupiny. Není to tak dlouho, kdy dominantní postavení v řídících obvodech měly klasické spojité, či nespojité regulátory, často řešené jednoúčelově. Rychle klesající cena mikroelektronických obvodů umožnila přechod od této koncepce k číslicovým řídícím obvodům.

V tomto studijním textu se budeme zabývat pouze jednou skupinou automatizačních prostředků - programovatelnými automaty. Byly vyvinuty a poprvé aplikovány koncem šedesátých let v USA. Zde také vzniklo jejich označení PLC (Programable Logical Controler). Původně byly určeny k programovému řešení jednoduchých logických obvodů, v současnosti je jejich použití mnohem širší. Umožňují provádět kromě základních logických funkcí i matematické operace, přesuny bloků dat, zpracovávat spojité signály, signály ze speciálních zařízení (např. CCD kamera, impulsní snímače polohy, selsyny, atd.). Často jsou součástí většího řídícího celku, tzv. distribuovaného řídicího systému, jehož jednotlivé součásti jsou propojeny soustavou sítí.

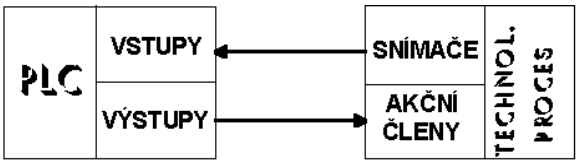
Použití PLC je velmi široké, od jednoduchých zařízení realizujících logické funkce např. při řízení soustavy dopravníků, přes aplikace ve sklářském , či tabákovém průmyslu až po PLC zabudovaných jako subsystém v CNC systémech pro řízení obráběcích strojů.

Funkce PLC je určena programem, který je uložen v operační paměti systému. Základním cílem při vzniku PLC bylo vytvoření "přátelského" programovacího prostředí, které by umožnilo vytvářet uživatelské programy i technikům neprogramátorům. Vzniklo několik základních skupin programovacích "jazyků". Jedním z nejjednodušších je jazyk vycházející ze symbolů liniových schémat (Ladder Diagram), jazyk blokových schémat, používající normované značky hradel AND, OR, klopných obvodů R-S atd., sekvenční grafický jazyk GRAFCET nebo SFC, strukturovaný text - jazyk blízký Pascalu a jazyk logických instrukcí.

V tomto textu se zaměříme především na PLC tuzemské firmy TECO a.s., která vyrábí kompaktní PLC TC 400, TC 500, TC 600 a modulové PLC NS 950 a TC 700. V rozsahu této publikace budou uživateli dány základy, které mu umožní orientovat se v dané problematice, vytvořit, odladit a spustit jednoduché programy.

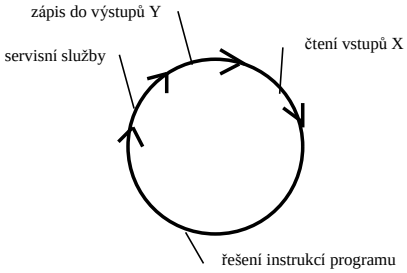
2. SPOJENÍ PROGRAMOVATELNÉHO AUTOMATU S ŘÍZENÝM PROCESEM

Z obr. 1 vidíme, že řízená technologie je spojena s PLC prostřednictvím senzorů (snímačů) a akčních členů (zařízení provádějící zásahy do regulované soustavy - ventily, servomotory, stykače atd.)
Programovatelný automat načítá prostřednictvím svých vstupů signály ze snímačů a ovládacích prvků programově je zpracovává a svými výstupy pomocí akčních členů zasahuje zpětně do regulované soustavy.



Obr. 1 Spojení programovatelného automatu s technologickým procesem

Programovatelný automat pracuje v uzavřeném cyklu. U většiny řídících systémů probíhají postupně tyto činnosti.

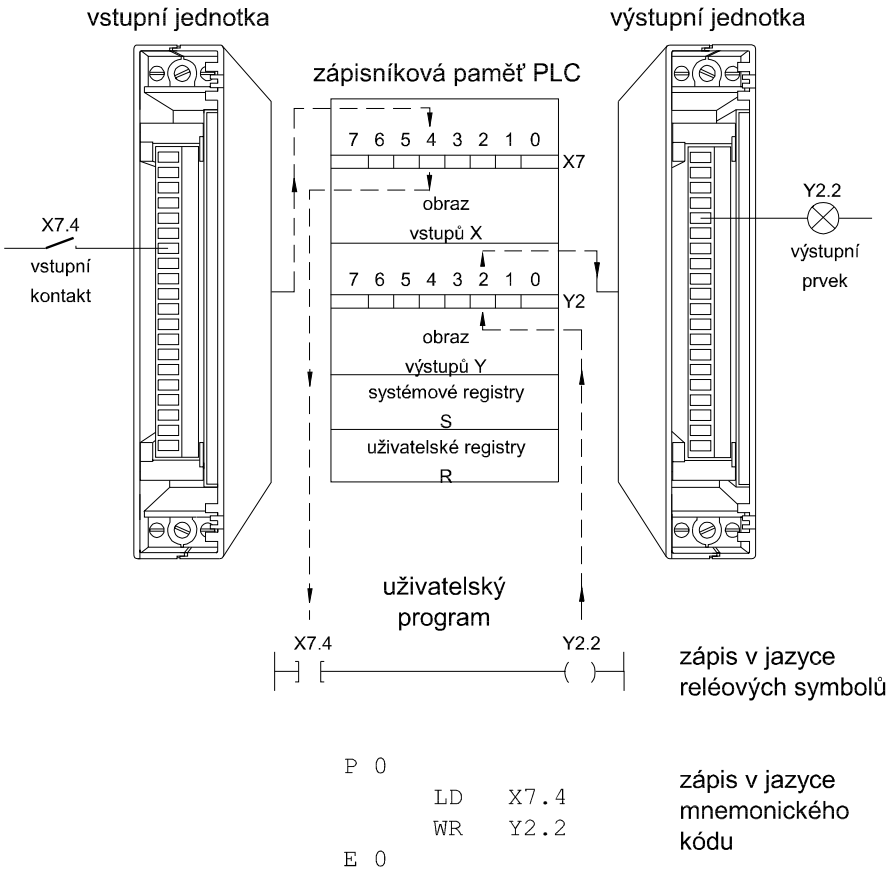


- načtení signálů ze vstupních jednotek do vstupních registrů automatu
- zpracování programu
- zápis obsahu výstupních registrů do výstupních jednotek
- servisní služby automatu (příprava CPU k řešení dalšího cyklu)

Obr. 2 Programový cyklus PLC

Řídící algoritmus programovatelného automatu je zapsán jako posloupnost instrukcí v paměti uživatelského programu. Centrální jednotka postupně čte z této paměti jednotlivé instrukce, provádí příslušné operace s daty v

zápisníkové paměti a zásobníku, případně provádí přechody v posloupnosti instrukcí, je-li instrukce ze skupiny organizačních instrukcí.
Jsou-li provedeny všechny instrukce požadovaného algoritmu, provádí centrální jednotka aktualizaci výstupních proměnných do výstupních periferních jednotek a aktualizuje stavy ze vstupních periferních jednotek do zápisníkové paměti. Tento děj se stále opakuje a nazýváme jej cyklem programu (Obr. 2).
Jednorázová aktualizace stavů vstupních proměnných během celého cyklu programu odstraňuje možnosti vzniku hazardních stavů při řešení algoritmu řízení (během výpočtu nemůže dojít ke změně vstupních proměnných).
Vazba mezi vnitřními proměnnými PLC, vstupy, výstupy a programem je znázorněna na obr. 3.



Obr. 3 Schéma zpracování signálu programovatelným automatem

3. METODY PROGRAMOVÁNÍ PLC

Jednou ze základních vlastností programovatelných automatů je jejich snadné programování. Technik z příslušného oboru (strojírenství, potravinářství, tepelná technika atd.) by měl zvládnout základy programování PLC daného typu během několika týdnů. Tato vlastnost společně s vysokou spolehlivostí a nízkou cenou (ve srovnání s řídícími počítači klasického typu) předurčila obrovské rozšíření průmyslových automatů prakticky do všech průmyslových oblastí.

V oblasti PLC není bohužel ujednocen univerzální programovací jazyk jako je tomu u personálních počítačů, ale každý výrobce má svůj způsob programování. Tyto způsoby programování jsou si však natolik podobné, že přechod na další typ automatu je velmi snadný. Programovatelné automaty se programují v podstatě následujícími způsoby:

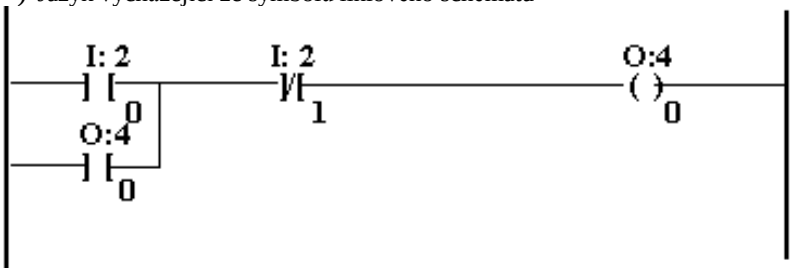
- Jazyk vycházející z logických instrukcí
- Jazyk vycházející ze symbolů liniového schématu - tzv. Ladder diagram
- Jazyk vycházející ze symbolů blokového schématu (hradla AND, OR, klopné obvody R-S atd.)
- Jazyk sekvenčních blokových schémat GRAPHCET, SFC.

Demonstrační příklad
Napište řídící algoritmus pro ovládání motoru pomocí tlačítek "START" a "STOP".

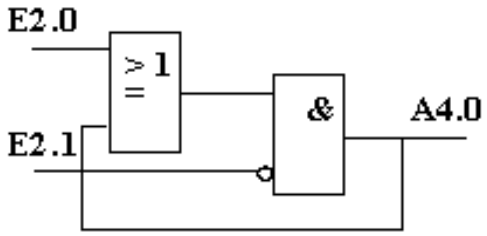
1) Jazyk logických instrukcí (TECOMAT)

```
LD X0.0     ; START
OR Y0.0     ; Přídržný "kontakt"
ANC X0.1    ; STOP
WR Y0.0     ; MOTOR
```

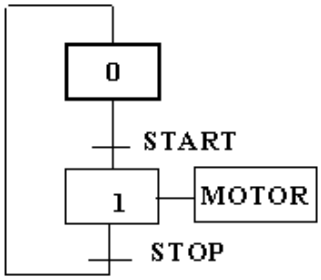
2) Jazyk vycházející ze symbolů liniového schématu



3) Jazyk vycházející ze symbolů blokového schématu



4) Jazyk sekvenčních blokových schémat SFC.



II. PROGRAMOVÁNÍ PLC TECOMAT

Dříve, než přikročíme k vlastnímu programování, je třeba vysvětlit způsob adresování proměnných uživatelského programu, princip zásobníku a akumulátoru, rozdělení programu do tzv. procesů a jejich aktivaci atd.

4. OPERANDY PLC TECOMAT

Operandem rozumíme proměnnou, se kterou pracuje daná instrukce. Podle významu můžeme rozlišit čtyři typy operandů:

- adresový operand
- přímý operand
- cíl přechodu
- parametr instrukce

4.1 ADRESOVÝ OPERAND

Adresový operand má význam adresy místa, odkud se čte informace nebo místa, kam se ukládá výsledek operace. Můžeme používat operandy z oblasti zápisníkové paměti (X,Y,S,R,), přímé vstupy a výstupy a operandy, které jsou součástí uživatelského programu (D,T).

4.1.1 Zápisníková paměť

Zápisníkem nebo též zápisníkovou pamětí rozumíme úsek paměťového prostoru PLC, který je přístupný jak pro čtení, tak i pro zápis uživatelských dat. Instrukce PLC umožňují přístup na libovolnou část zápisníku. Tato paměť je předem rozdělena do několika částí s vyhrazeným významem.

Struktura zápisníkové paměti:

(platí pro CPM-1D)

- X obrazy vstupů (vstupní registry) X0 .....X127
- Y obrazy výstupů (výstupní registry) Y0 .....Y127
- S systémové registry S0 .....S63
- R uživatelské registry R0 .....R8191

Přístup systémového programu k zápisníkové paměti se uskutečňuje výhradně ve fázi otočky cyklu uživatelského programu. To se týká nejenom snímání fyzických vstupů do oblasti X a nastavování hodnot z oblasti Y na fyzické výstupy, ale i změn hodnot systémových proměnných S. To znamená, že po dobu cyklu uživatelského programu jsou údaje zápisníku zmrazeny a aktualizují se až po nejbližší otočce cyklu.

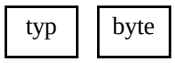
Podle formátu lze rozdělit operandy na:

- bitové - nesou pouze dvouhodnotovou informaci
- slabikové - nesou informaci v šířce osmi bitů
- slovní - nesou informaci v šířce šestnácti bitů (dvou slabik)
- long - nesou informaci v šířce 32 bitů (čtyř slabik)
- float - reálné číslo v rozsahu $\pm 1{,}175494 \times 10^{-38}$ až $\pm 3{,}402823 \times 10^{38}$ (čtyři slabiky)

Bitový operand

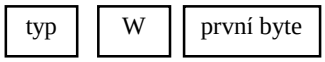
Příklad instrukce s bitovým operandem: LD X0.5
instrukce načte do akumulátoru bit číslo 5 nultého vstupního registru

Slabikový operand



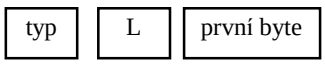
Příklad instrukce se slabikovým operandem: WR Y0
instrukce zapíše 8 bitů dolní části akumulátoru do výstupů Y0.0 až Y0.7

Slovní operand



Příklad instrukce se slovním operandem: LD SW0
instrukce načte dvojici systémových registrů S0, S1 do akumulátoru

Operand long



Příklad instrukce se slovním operandem: LD RL10
instrukce načte čtveřici registrů R10, R11, R12, R13 do zásobníku

Obrazy vstupů X.

Před každým začátkem cyklu programu zajišťuje centrální jednotka aktualizaci této oblasti zápisníkové paměti ze vstupních periferních jednotek na základě deklarační tabulky zadané v uživatelském programu, která popisuje přiřazení mezi obrazy vstupů X a fyzickými adresami jednotlivých jednotek (v programu xPRO pomocí direktivy #unit). V případě nedostatku paměti v oblasti X, lze bez omezení použít k témuž účelu oblast uživatelských registrů R.

Obrazy výstupů Y.

Po každém ukončení cyklu programu zajišťuje centrální jednotka přesun výsledků z této oblasti zápisníkové paměti do výstupů periferních jednotek na základě deklarační tabulky zadané v uživatelském programu, která popisuje přiřazení mezi obrazy výstupů Y a fyzickými adresami jednotlivých jednotek (v programu xPRO pomocí direktivy #unit). V případě nedostatku paměti v oblasti Y, lze bez omezení použít k témuž účelu oblast uživatelských registrů R.

Systémové registry S,

Tato oblast zápisníkové paměti je vyhrazena pro specifické použití systémovým programem automatu a nedoporučuje se ji používat pro jiný účel. Některé bity a byty jsou pravidelně v otočce cyklu nastavovány systémovým programem a jsou vhodné pouze pro čtení. Některé bity naopak modifikují svým nastavením chování systémového programu.
Přehled a popis systémových registrů je uveden v příručce *Instrukce s systémové služby* (str. 6, 7).
Nejpoužívanějšími S registry jsou aritmetické a logické příznaky, registry obsahující reálný čas a datum, časové jednotky a masky uživatelských procesů.

S0 : aritmetické příznaky
Registr S0 ovlivňují některé aritmetické instrukce, instrukce čítačů a časovačů. Ostatní instrukce jej nemění. Příznakové bity jsou aktivní v úrovni "1".

Struktura S0:

S0.7	S0.6	S0.5	S0.4	S0.3	S0.2	S0.1	S0.0
0	D5.2	D5.1	D5.0	CI	≤	<(CO)	=(ZR)

Význam jednotlivých bitů S0:

S0.0 - rovnost obou operandů nebo nulovost výsledku

S0.1 - první operand < druhý operand resp. A0 < operand

- výstupní přenos CO (při operaci došlo k přetečení akumulátoru)

S0.2 - prvý operand je menší nebo roven druhému operandu

- logický součet S0.0 a S0.1

S0.3 - vstupní přenos

S0.4 až S0.6 - po instrukci BCD obsahují nejvyšší číslici výsledku

S0.4 - překročení max. rozsahu časovače (kdykoli během jeho současné aktivace)

S0.5 - překročení max. rozsahu časovače právě v tomto cyklu

S0.7 - trvale nulován

S1 : Logické příznaky
Po instrukci *FLG* se nastavuje podle obsahu A0

Provedení FLG
- nastaví se příznaky S1
- úrovně zásobníků se zvýší o 1 a na nový vrchol se zapíše AND všech bitů původního A0

Struktura S1:

S1.7	S1.6	S1.5	S1.4	S1.3	S1.2	S1.1	S1.0
ORH	ORL	ANH	ANL,N4	N3	N2	N1	N0

S1.0 až S1.4 N(N4 Až N0) -dvojkové číslo, které udává počet jedničkových bitů A0

S1.4 : ANL AND dolní části A0

S1.5 : ANH AND horní části A0

S1.6 : ORL OR dolní části A0

S1.7 : ORH OR horní části A0

S2 - příznaky stavu systému

S2.7	S2.6	S2.5	S2.4	S2.3	S2.2	S2.1	S2.0
MEZ	KOM	ON	RST	HOT	RUN	MS	SP

S2.0	(SP)	- stav služebního bitu SP - externí spouštění PLC
S2.1	(MS)	- stav služebního bitu SP - externí blokování výstupů
S2.2	(RUN)	1- režim RUN
		0- režim HALT
S2.3	(HOT)	1 - první průchod cyklem po teplém restartu
S2.4	(RST)	1 - první průchod cyklem po studeném restartu
S2.5	(ON)	1 - aktivní výstupy
		0 - zablokované výstupy
S2.6	(NRS)	1 - první průchod cyklem bez restartu
S2.7	(MEZ)	1 - překročena první mez doby cyklu

Registr S2 je určen pouze pro indikaci - nepřepisovat!

S5 - S12 : Systémový čas a datum

S5 - čítač 10 ms (0 až 99)
S6 - čítač 1s (0 až 59)
S7 - čítač 1 min (0 až 59)
S8 - čítač 1 hod (0 až 23)
S9 - čítač dny v týdnu (1 až 7)
S10 - čítač dny v měsíci (1 až poslední den v měsíci)
S11 - čítač měsíce (1 až 12)
S12 - čítač roky (0 až 99)

S13 : Časové jednotky

Na jednotlivých bitech registru S13 je obdélníkový periodický signál se střídou 1 : 1 a periodou uvedenou v tabulce:

S13.7	S13.6	S13.5	S13.4	S13.3	S13.2	S13.1	S13.0
1den	1hod	10min	1min	10s	1s	500ms	100ms

S14 - S19 : Čítače časových jednotek

S14,15 čítač v 100 ms
S16,17 čítač v 1 s
S18,19 čítač v 10 s
Rozsahy čítačů jsou 0 až 65 535 časových jednotek.

Uživatelské registry R.

Paměťová oblast určená pro proměnné uživatelského programu, pro realizaci čítačů, posuvných registrů, časovačů, atd. Studený restart nuluje všechny registry R, teplý restart remanentní část registrů R uchovává.

4.1.2 Přímé vstupy a výstupy - operand U

Existují případy, kdy je z časových důvodů nutné načíst z jednotky okamžitý stav, či neprodleně zapsat do jednotky hodnotu. K tomu pak slouží fyzické adresy označované operandem U. Operand U tedy poskytuje alternativu k oblastem X a Y v zápisníku, která umožňuje přímý styk s periferními jednotkami v daném okamžiku bez čekání na otočku cyklu.
Jeho využití je vhodné pouze pro zabezpečení časově kritických reakcí. *Nadbytečné využívání má za následek zpomalení výkonu programu*, protože přímý přístup k periferním jednotkám je časově náročnější než operace se zápisníkovou pamětí.

4.1.3 Datový a tabulkový operand

Programovatelné automaty Tecomat mohou používat i proměnné, které jsou součástí uživatelského programu.

Data - operand D

Data D mají význam konstant uživatelského programu. Jsou součástí uživatelského programu a jsou pro něj dostupná pouze pro čtení. Mohou se zadávat a měnit pouze v rámci editace uživatelského programu.
Výhodně mohou být použity jako parametry, které modifikují uživatelský program.

Tabulky - operand T

Tabulky T jsou stejně jako data D součástí uživatelského programu a i jejich použití může být obdobné. Na rozdíl od dat D, která jsou dostupná většinou instrukcí pro čtení, jsou tabulky T dostupné pouze zvláštními instrukcemi, které se odvolávají na adresový prostor T (tabulkové instrukce, instrukce blokových přenosů).
Data D mohou být nestrukturovaná nebo mohou mít libovolně složitou strukturu, lze přímo přečíst jejich libovolné místo. Naproti tomu mají tabulky T vždy předepsanou strukturu - je to vždy řada hodnot stejného formátu (bit, byte nebo word) s přídavným údajem o délce této řady.
Každé položce (hodnotě z této řady) je přiřazeno pořadové číslo - index.

4.2 PŘÍMÝ OPERAND

Některé instrukce mohou pracovat s čísly, která jsou zanesena přímo v instrukci. Přímý operand zapisujeme ve zvolené číselné soustavě. Můžeme používat soustavy s libovolným základem.

Forma zápisu:
 # základ soustavy # číslo ve zvolené soustavě
pro běžně používané soustavy platí pro označení základu výjimky
desítková soustava - bez vyznačení základu (implicitně)
dvojková soustava - znak %
šestnáctková soustava - znak $
Příklad: LD 45 operand v desítkové soustavě
LD # 2# 1011 operand ve dvojkové soustavě
nebo
LD %1011
LD #16# AF operand v šestnáctkové soustavě
nebo
LD $AF
LD # 60# 15.28.35 (15 hod.,28 min.,35 sec.=55715 sec.)

4.3 CÍL PŘECHODU

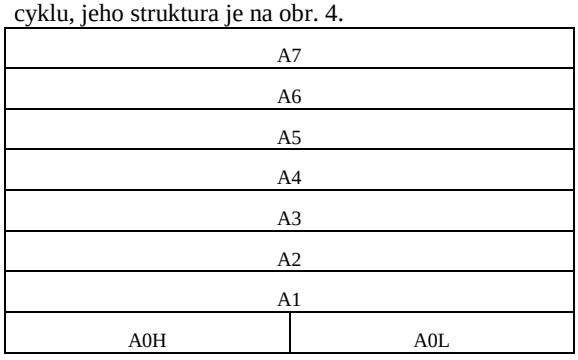
U instrukcí skoků a volání podprogramů je operandem návěští.
příklad: JMP L1 nepodmíněný skok na návěští L1

4.4 PARAMETR INSTRUKCE

Některé instrukce vyžadují zadání číselného parametru. Používá se u instrukcí návěští, začátku a konce procesu, rotace akumulátoru, atd.

5. ZÁSOBNÍKOVÁ PAMĚŤ

Každá instrukce pracuje s vyjádřenými či nevyjádřenými operandy. Vždy alespoň jeden je uložen v zásobníku. Zásobník je tvořen oblastí paměti typu RAM, má osm 16 bitových úrovní označených A0 až A7, přičemž úroveň **A0 má funkci akumulátoru.** Vrstvy A1, až A7 obsahují postupně sled předchozích hodnot akumulátoru. Posun zásobníku způsobují instrukce LD, LDC (načti) a některé složitější funkce. Zásobník A se nuluje v každé otočce cyklu, jeho struktura je na obr. 4.



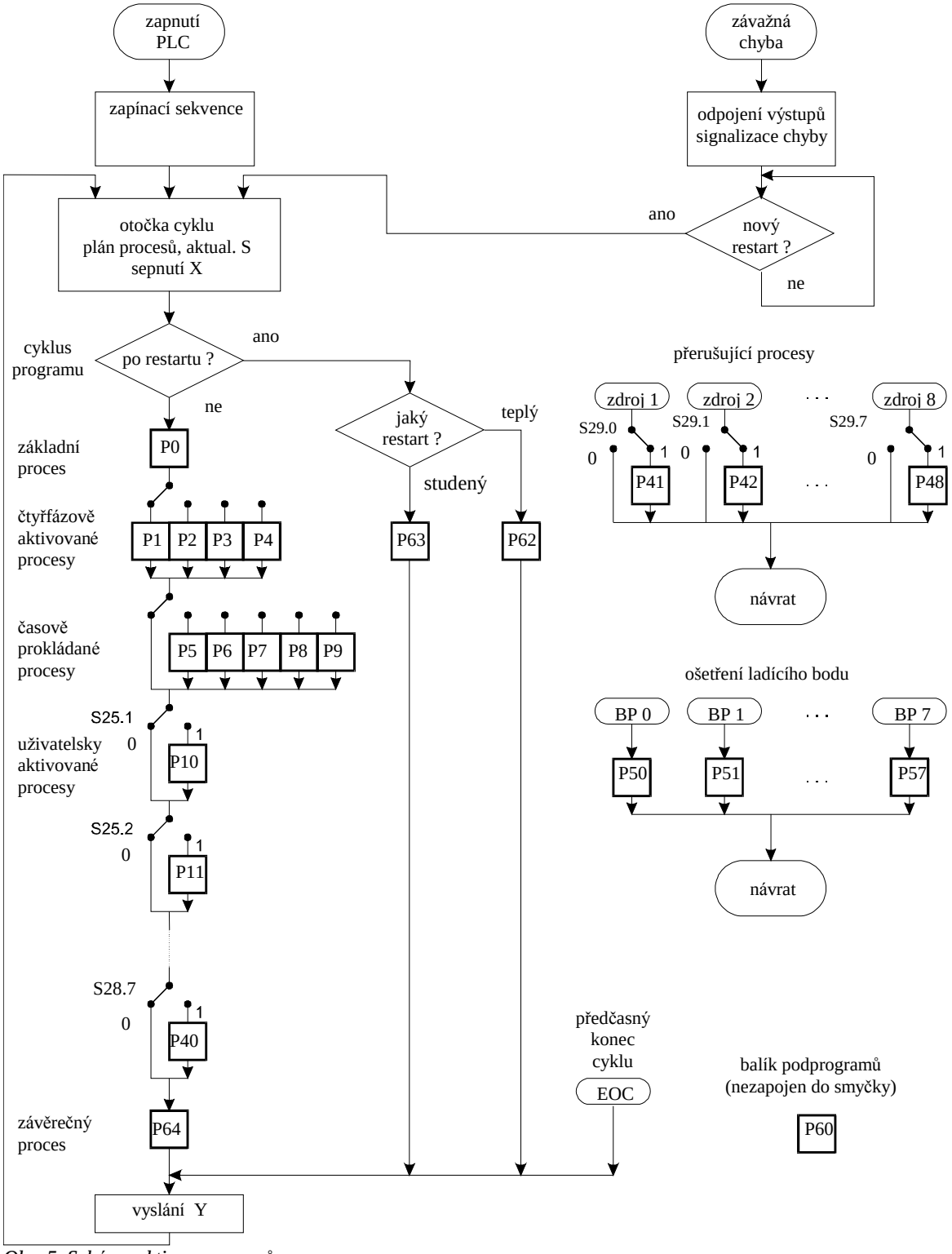
Obr. 4 Struktura zásobníku

Některé instrukce umožňují rozlišit spodní a dolní byte dané vrstvy zásobníku. Označují se písmenem H (High) a L (Low). Takovýchto zásobníků máme k dispozici celkem osm (A až H). Můžeme je aktivovat instrukcemi NXT, PRV, CHG.

6. UŽIVATELSKÉ PROCESY

Uživatelský program je složen z uživatelských procesů. Může jich být až 65, prakticky se jich však využívá podstatně méně. Značí se P0 až P64. Uživatel tedy může rozložit svůj program do několika procesů, a tím jej učinit přehlednějším, časově oddělit jednotlivé akce, atd. Procesy jsou aktivovány podle předem definovaných pravidel. Jak zjistíme později, je možné aktivovat procesy P10 až P40 nastavením tzv. masky procesů vytvořené v oblasti systémových registrů. Schéma aktivace procesů je znázorněno na obr. 5.

Uživatel však není nucen všechny procesy využívat. Zvláště pro jednoduché programy obvykle vystačíme s jednosmyčkovým řízením (celý program je v procesu P0).



Obr. 5 Schéma aktivace procesů

7. INSTRUKČNÍ SOUBOR NS - 950 (TC 400, TC 500, TC 600)

Instrukční soubor Tecomat je velice široký. Umožňuje uživateli řešit rozsáhlý soubor úloh v nejrůznějších aplikacích. Podle použité centrální jednotky hovoříme o třech typech instrukčních souborů:

- instrukční soubor redukovaný - CPU řady E
- instrukční soubor standardní - CPU řady A, S, M
- instrukční soubor rozšířený - CPU řady D, B

Instrukce lze rozdělit do několika základních skupin:

- čtení a zápis dat
- logické instrukce
- operace se zásobníky
- aritmetické instrukce
- organizační instrukce
- funkční bloky (čítače, časovače, posuvné registry, sekvenční řadič)
- instrukce s tabulkami
- přesuny bloků dat

Není v možnostech tohoto textu vysvětlit podrobně celý instrukční soubor. Popíšeme si nejdůležitější instrukce nutné k sestavení základních uživatelských programů.

7.1 STANDARDNÍ INSTRUKČNÍ SOUBOR

V úvodu kapitoly 7 byl uveden velice stručný přehled instrukcí programovacího jazyka TECOMAT. Na následujících stránkách podrobně probereme většinu z dříve uvedených instrukcí, uvedeme si jednoduché příklady názorně ilustrující funkci těchto instrukcí. Obtížnost příkladů se bude postupně zvyšovat. U každé instrukce bude uvedena její symbolická zkratka a anglický název, budou vypsány přípustné operandy (X, Y, S, R, D, #, U, T) a jejich šířka (bit, slabika, slovo).
U každé skupiny instrukcí budou také naznačeny možnosti rozšířeného instrukčního souboru centrál typu D, B.

7.1.1 Čtení a zápis dat

Instrukce LD, LDC

- **LD** (Load) čti přímá data
- **LDC** (Load Complement) čti negovaná data

PROSTOR OPERANDU: X,Y,S,R,D,#
ŠÍŘKA OPERANDU : bit
slabika
slovo

ZÁSOBNÍK : **zvyšuje se, vrchol se plní snímanými daty !**

PROVEDENÍ: **bitová instrukce LD** sejme hodnotu adresovaného bitu a zaplní jím všech 16 bitů A0

slabiková instrukce LD zapíše obsah adresované slabiky do spodní poloviny akumulátoru A0L. Horní část A0 (A0H) se nuluje.

slovní instrukce LD zapíše obsah adresovaného slova do A0.

Příklad:	LDC R0.0	non R0.0	→	na všechny bity A0.
	LD X0	X0 (8 bitů)	→	A0L
	LD SW0	S0,S1	→	A0 (S0 → A0L, S1 → A0H)

Příklad:
Předpokládejme, že spínač, připojený na vstup X0.0 je sepnutý, v registru R10je číslo 100H a je 1. července. Po instrukcích
```
LD X0.0
LD R10
LD SW 10
```
dostaneme následující obsah zásobníku:

A0 = 07 01H	S10=1 (1. den v měsíci)	S11=7 (červenec)
A1 = 00 40H	R10 = 40H	
A2 = FFFFH	vstup X0.0 je sepnutý	

Rozšířený instrukční soubor:

Instrukce LD a LDC podporují operand *long* a *float*
Instrukce LDL slouží ke čtení konstanty typu *long*

Instrukce WR, WRC

- **WR** (Write) piš přímá data
- **WRC** (Write Complement) piš negovaná data

PROSTOR OPERANDU: X,Y,S,R

ŠÍŘKA OPERANDU : bit
slabika
slovo

ZÁSOBNÍK : **nemění se**

PROVEDENÍ: **bitová instrukce WR** provede OR všech bitů A0 a výsledek uloží do adresovaného bitu.

slabiková instrukce WR zapíše obsah A0L do adresované slabiky.

slovní instrukce WR zapíše obsah A0 do adresovaného slova.

Příklad:

```
LD  #2# 1001 1111 0111 0101        do A0 zapíšeme 9F75H
        WR      Y0.0                        Y0.0 = 1
        WRC      Y0.0                       Y0.0 = 0
        WR      R1                          R1 = 75H
        WRC     R1                          R1 = 1000 1010 = 8AH
        WR      RW4                         R4 = 75H, R5 = 9FH
        WRC     RW4                         R4 = 8AH, R5 = 60H
```

Instrukce PUT

podmíněný zápis

PROSTOR OPERANDU: X,Y,S,R

ŠÍŘKA OPERANDU : bit
slabika
slovo

ZÁSOBNÍK : **nemění se**

PROVEDENÍ: Instrukce probíhá obdobně jako WR, její vykonání je podmíněno hodnotou systémového bitu **S1.0**.

S1.0 = 1 ..... instrukce je vykonána

S1.0 = 0 ..... instrukce není vykonána

Rozšířený instrukční soubor:

Instrukce WR a PUT podporují operand *long* a *float*
Instrukce WRC podporuje operand *long*
Instrukce WRA slouží k zápisu dat s alternací nejvyššího bitu

7.2 LOGICKÉ INSTRUKCE

Logické instrukce tvoří základ instrukčního souboru PLC. U systému TECOMAT budeme pracovat se třemi základními logickými funkcemi:

- OR Logický součet
- AND Logický součin
- XOR Exclusive OR (různoznačnost)

Funkce NEGACE je realizována pomocí instrukcí LDC, WRC nebo instrukcí s negovaným operandem, případně samostatnou instrukcí NEG (bude popsána v další kapitole). Pro zopakování uvádím pravdivostní tabulky základních logických funkcí pro dvě vstupní proměnné A,B a výstupní proměnnou Z.

A	B	A OR B	A AND B	A XOR B
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

Logické instrukce se provádějí mezi dvěma operandy, z nichž jeden je vždy uložen v A0 a druhý je buď vyjádřen za kódem instrukce nebo je uložen v A1. Logické instrukce se dělí na:

- Logické instrukce s vyjádřeným operandem
- Logické instrukce s nevyjádřeným operandem
- Paměťové instrukce
- Generování impulzu od náběžné hrany

Logické instrukce s vyjádřeným operandem

AND	- logický součin s přímým operandem
ANC	- logický součin s negovaným operandem
OR	- logický součet s přímým operandem
ORC	- logický součet s negovaným operandem
XOR	- Exclusive Or s přímým operandem
XOC	- Exclusive Or s negovaným operandem

PROSTOR OPERANDU: X,Y,S,R,U,D, # v šíři slova

ŠÍŘKA OPERANDU : bit
slabika

ZÁSOBNÍK : A0 se přepisuje výsledkem operace

PROVEDENÍ INSTRUKCE:

Logická funkce je provedena mezi obsahem A0 a obsahem operandu. Výsledek se ukládá do A0. U instrukcí s "C" na konci (ANC, ORC, XOC) se provede příslušná operace mezi obsahem A0 a negovaným obsahem operandu.

Bitová instrukce - načte adresovaný bit nebo jeho negaci a provede logickou operaci se všemi bity A0, výsledek uloží do A0.

Slabiková instrukce -provede logickou funkci mezi slabikou operandu a A0L, výsledek uloží do A0L. Horní část akumulátoru A0H se nemění.

Příklady:

1. Realizujte výraz: $Z = A \cdot B \cdot \overline{C}$, kde A, B, C jsou spínače připojené na vstupy X0.0 až X0.2, Z je spotřebič připojený na výstup Y0.0.

```
LD      X0.0
AND     X0.1
ANC     X0.2
WR      Y0.0
```

2. Zobrazte na výstupech Y0.0 až Y0.7 stavy stejnolehlých vstupů X0.0 až X0.7, není - li sepnuto tlačítko připojené ke vstupu X1.0. V opačném případě výstupy nulujte.

Řešíme 8 logických rovnic:
Y0.0 = X0.0 . X1.0'
Y0.1 = X0.1 . X1.0'
.....
Y0.7 = X0.7 . X1.0'

```
LD      X0
ANC     X1.0
WR      Y0
```

Logické instrukce s nevyjádřeným operandem

AND	- logický součin
OR	- logický součet
XOR	- Exclusive OR
NEG	- negace vrcholu zásobníku

Provádějí danou logickou funkci mezi A0 a A1 zásobníku. Jsou určeny k řešení výrazů se závorkami a k realizaci logických výrazů se 16 bity.

PROVEDENÍ:

Napřed musíme do úrovní A0 a A1 zapsat oba operandy. Operace AND, OR, XOR sníží zásobník o jednu úroveň, přičemž vysouvaný obsah A0 je odložen do pomocného registru, pak se provede daná operace mezi obsahem A0 a pomocného registru a výsledek se uloží do A0.

Úroveň zásobníku se snižuje o 1 !!

Příklady:

1) Realizujte výraz: Y1.2 = (X0.0 + X0.1).(X0.2 + X0.3)

```
LD X0.0          ; A0 ← X0.0
OR X0.1          ; A0 ← (X0.0 + X0.1)
LD X0.2          ; A0 ← X0.2              A1 ← (X0.0 + X0.1)
OR X0.3          ; A0 ← (X0.2 + X0.3)     A1 ← (X0.0 + X0.1)
AND              ; A0 ← (X0.2 + X0.3).(X0.0 + X0.1)
WR Y1.2
```

2) Zapište spodní 4 bity registru R10 do spodní části výstupního registru Y0. Zbylé bity vynulujte.

```
LD      R10
AND     15 nebo AND % 0000 1111
WR      Y0
```

3) Zadání z předchozího příkladu realizujte pro zápis z registrového páru RW0.

```
LD      RW0
AND     15
WR      Y0
```

Pozn. pro NS-940 je nutné psát:

```
LD      RW0
LD      15
AND
WR      Y0
```

Paměťové funkce

Slouží k podmíněnému nastavení nebo nulování adresovaného bitu nebo slabiky.

- **SET** (set) - podmíněné nastavení operandu
- **RES** (reset) - podmíněné nulování operandu

PROSTOR OPERANDU: X,Y, S, R (Prakticky má význam pouze použití Y, R)

ŠÍŘKA OPERANDU : bit
slabika

ZÁSOBNÍK : nemění se

PROVEDENÍ: **Bitová instrukce SET, resp. RES** - nastaví, resp. vynuluje adresovaný bit, je - li obsah A0 nenulový. Jinak obsah nemění.

Slabiková instrukce SET, resp. RES - nastaví, resp. vynuluje adresovanou slabiku podle masky tvořené obsahem A0L.

Příklady:

1) Realizujte ovládání stykače pomocí tlačítek START a STOP.
START..... X0.0
STOP X0.1
STYKAČ..... Y0.0

```
LD      X0.0
SET     Y0.0
LD      X0.1
RES     Y0.0
```

2) Podle polohy "0" na vstupech X0 nulujte stejnolehlé bity registru R2. Ostatní bity R2 neměňte.

```
LDC X0
RES R2
```

Generování impulsu od náběžné hrany

Často se v praktických aplikacích setkáme s požadavkem vyhodnotit okamžik, kdy se mění určitá proměnná z úrovně "0" do úrovně "1". Mluvíme o tzv. náběžné, v opačném případě o spádové hraně signálu. K tomuto účelu nám slouží instrukce LET.

LET (Lead Edge Triggering) - spouštěcí impuls od náběžné hrany

PROSTOR OPERANDU: X,Y,S,R

ŠÍŘKA OPERANDU : bit

slabika

ZÁSOBNÍK : přepisuje se obsah A0

PROVEDENÍ :

Pracuje obdobně jako instrukce WR, navíc porovná ve dvou po sobě jdoucích programových cyklech původní a nově zapsaný obsah adresovaného operandu. Pokud dochází ke změně z "0" do "1" (náběžná hrana), pak nastaví na příslušném místě A0 "1" , v opačném případě "0".

Pro správnou činnost instrukce LET je nutné, aby zápis do proměnné adresované instrukcí LET prováděla pouze jediná instrukce!

Příklady:

1) Vynulujte slovo RW0 při náběžné hraně signálu na vstupu X0.0

```
LD X0.0
LET R5.0        ;stavovou proměnnou volíme libovolně v prostoru, kde
RES R0          ;"nehospodaří" žádná další instrukce
RES R1          ;podmíněné nulování slova RW0 musíme provést postupně
```

2) Vynulujte slovo RW0 při spádové hraně signálu na vstupu X0.0

```
LDC X0.0
LET R5.0
RES R0
RES R1
```

Rozšířený instrukční soubor:

Logické instrukce s operandem a paměťové instrukce a LET podporují *slovní* operand
Instrukce ANL, ORL a XOL a NGL podporují operand typu *long*
Instrukce BET slouží ke generování impulzu od libovolné hrany

7.3 KOMPARAČNÍ INSTRUKCE (INSTRUKCE POROVNÁNÍ)

V instrukčním souboru TECOMAT existuje skupina instrukcí umožňující otestovat relaci mezi dvěma čísly. Jsou to instrukce:

- EQ - rovnost
- LT - menší než
- GT - větší než

Vzhledem k tomu, že dané instrukce nastavují aritmetické příznaky S0, je možné s výhodou použít k otestování dalších relací mezi dvěma čísly registr příznaků S0.

PROSTOR OPERANDU : X,Y,S,R,P,D,#,bez operandu

ŠÍŘKA OPERANDU : slovo

ÚROVEŇ ZÁSOBNÍKU : nemění se

PŘÍZNAKY : nastavuje S0

PROVEDENÍ:

[A0] EQ,LT,GT [operand] + CI → A0 ;porovnání s operandem

[A1] EQ,LT,GT [A0] + CI → A0 ;porovnání bez operandu

Instrukce provedou porovnání obsahu A0 s obsahem operandu nebo obsahu úrovní A1 a A0 zásobníku. Podle pravdivostní hodnoty porovnání nastaví do A0 samé 0 nebo samé 1 (Když je relace pravdivá - A0 = 65 535. Pořadí operandů při porovnání je zřejmé z výše uvedeného schématu.

Příklady:

1) Sepněte výstup Y0.0 platí - li: RW0 < XW0.

```
LD RW0          LD RW0
LT XW0   nebo   LD XW0
```

```
WR Y0.0          LT
                 WR Y0.0
```

Rozšířený instrukční soubor:
Instrukce CMP nastavuje aritmetické příznaky a podporuje operandu typu ***byte, word long***
Instrukce CML slouží k porovnání s konstantou typu ***long***

7.4 OPERACE SE ZÁSOBNÍKY

STK (Stack) - sklopení zásobníku: pro každou úroveň A0 až A7 se provede OR všech 16 bitů a výsledky jednotlivých úrovní se sklopí do úrovně A0
Nový obsah A0L:

OR 7	OR 6	OR 5	OR 4	OR 3	OR 2	OR 1	OR 0

Pozn. OR 0 znamená logický součet všech 16 bitů vrstvy A0, OR 1 znamená logický součet všech 16 bitů vrstvy A1, atd.

SWP (Swap) - záměna A0L a A0H

POP (Pop) - snížení úrovně zásobníku
Operandem instrukce POP je číselný parametr 1 - 7 udávající o kolik úrovní se sníží zásobník.

FLG (Flags)
Instrukce FLG (bez operandu) nastavuje registr logických příznaků S1 (viz kap. 3.2.). Úroveň zásobníku se zvyšuje o jednu a nový vrchol A0 se nastavuje na AND všech 16 bitů původního A0.

Příklad:

Na vstupy X0.0 až X0.7 je připojeno osm spínačů. Sepněte výstup Y0.0, je - li sepnuta většina z nich.

```
LD X0                ;načtení stavu X0.0 až X0.7 do A0
FLG                  ;aktivace registru S1
LD S1                ;načtení S1 do A0
AND 2# 1111          ;odmaskování nadbytečných bitů
GT  4                ;otestování, zda číslo v A0 je větší než 4
WR Y0.0
```

CHG m aktivace vybraného zásobníku m
NXT aktivace následujícího zásobníku v řadě
PRV aktivace předcházejícího zásobníku v řadě
Instrukce CHG aktivuje vybraný zásobník určený parametrem m, který představuje znaky A, B, C, až H. Instrukce umožňuje i současné ukládání a vybírání stavu systémových registrů S0 a S1. Parametr m pak představuje znaky AS, BS, CS, až HS.

Rozšířený instrukční soubor:

Instrukce SWL zamění vrstvy A0 a A1 zásobníku
Instrukce LAC m načte hodnotu z akumulátoru zvoleného zásobníku
Instrukce WAC m zapíše hodnotu do akumulátoru zvoleného zásobníku

7.5 INSTRUKCE FUNKČNÍCH BLOKŮ

Vytvářejí ucelené funkce sekvenčních logických členů - čítače, posuvné registry, časovače a sekvenční řadič. Jejich funkci lze znázornit blokovými schématy jaká jsou používána v pevné logice.
Funkce čítačů, posuvných registrů, časovačů a sekvenčního řadiče jsou realizovány vždy nad registrovými páry RW0 až RW254. Většinou je nutné zajistit, aby v průběhu cyklu byl určitý registrový pár použit pouze jednou. Nevhodné definování funkčního bloku může vést k nežádoucí inicializaci daného registrového páru (vynulování obsahu slova a nastavení dynamických řídících proměnných).

Vstupní proměnné

Instrukce funkčních bloků očekávají vstupní proměnné v jednotlivých vrstvách zásobníku. Je-li vstupní proměnná dvouhodnotová, její úroveň je rovna logickému součtu všech bitů příslušné vrstvy.

Výstupní proměnné

Instrukce funkčních bloků předávají výstupní proměnné v jednotlivých vrstvách zásobníku.

7.5.1 ČÍTAČE

- **CTU** (Count Up) čítej nahoru
- **CTD** (Count Down) čítej dolů
- **CNT** (Count) čítej oběma směry

Formát instrukcí:

CTU RW číslo
CTD RW číslo
CNT RW číslo

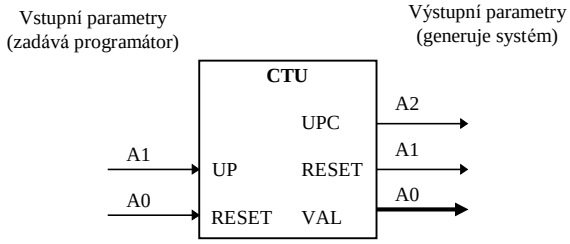
Pozn. V jednom cyklu uživatelského programu nesmí pracovat nad jedním slovem více instrukcí CTU, nebo CTD, případně kombinace CTU (CTD) a CNT s různými řídícími proměnnými.

Instrukce CTU

Vstupní parametry:
A1: řídící proměnná (UP)
A0: nulovací proměnná (RESET)

Výstupní parametry:
A2: přenos do vyšší kaskády (UPC)
A1: nulovací proměnná (RESET)
A0: stav čítače (VAL)

Pro snazší pochopení uvádím odpovídající schematickou značku v blokovém schématu.



Funkce: Pokud se hodnota řídící proměnné UP změní proti předchozí aktivaci z 0 na 1 (náběžná hrana), pak se obsah slova čítače zvýší o 1.
Nebyla-li vyhodnocena náběžná hrana, zůstává stav zachován.
Proměnná RESET = 1 nuluje obsah čítače.

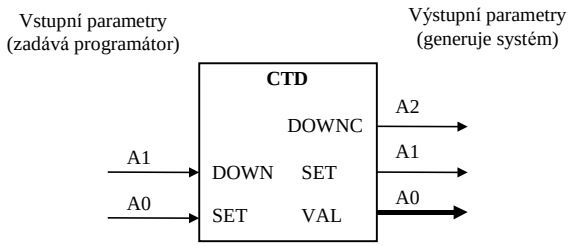
Příznaky S0

S0.0 (ZR)	nulovost čítače
S0.1 (CO)	výstupní přenos čítače
S0.2 (ZR + CO)	nulovost nebo přenos
S0.3 až S0.7 = 0	

Instrukce CTD

Vstupní parametry:
A1: řídící proměnná (DOWN)
A0: nastavovací proměnná (SET)

Výstupní parametry:
A2: přenos do vyšší kaskády (DOWNC)
A1: nastavovací proměnná (SET)
A0: stav čítače (VAL)



Funkce: Pokud se hodnota řídící proměnné DOWN změní proti předchozí aktivaci z 0 na 1 (náběžná hrana), pak se obsah slova čítače sníží o 1.
Nebyla-li vyhodnocena náběžná hrana, zůstává stav zachován.
Proměnná SET = 1 nastavuje obsah čítače.

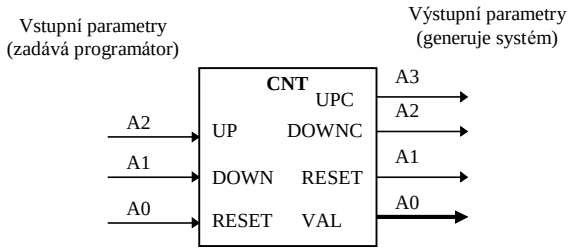
Instrukce CNT

Vstupní parametry:
A2: řídící proměnná (UP)

Výstupní parametry:
A3: přenos do vyšší kaskády (UPC)

A1: řídící proměnná (DOWN)
A0: nulovací proměnná (RESET)

A2: přenos do vyšší kaskády (DOWNC)
A1: nulovací proměnná (RESET)
A0: stav čítače (VAL)



Funkce: Při náběžné hraně signálu UP se stav čítače inkrementuje, při náběžné hraně signálu DOWN se stav čítače dekrementuje.Vyskytnou-li se současně, stav čítače se nemění.
Nebyla-li vyhodnocena náběžná hrana, zůstává stav zachován.
Proměnná RESET = 1 nuluje obsah čítače.

Příklady:
1) Počítejte výrobky na pásu linky. Přítomnost předmětu je snímána kapacitním čidlem s binárním výstupem připojeným na vstup X0.0 programovatelného automatu. Čítač nulujte tlačítkem B připojeným na vstup X0.1.

```
#program  linka, v1.0
#def      snimac          x0.0
#def      nuluj           x0.1
#reg      word            citac
p 0
          ld      snimac
          ld      nuluj
          ctu     citac
e 0
```

2) Čítač z příkladu 1 realizujte jako dekadický (modulo 10).

```
#program  linka, v2.0
#def      snimac          x0.0
#def      nuluj           x0.1
#def      modulo  10
#reg      word            citac
p 0
          ld      snimac
          ld      nuluj
          ctu     citac
          gt      modulo-1
          res     low citac
          res     high citac
e 0
```

3) Navrhněte čítač, který bude zpracovávat impulsy na vstupu *snímač* (X0.0). Jeho činnost bude modifikována vstupem *volba* (X0.1). Stav čítače zobrazte binárně na *výstupu* (YW0).
X0.1 = 1 .... čítač nahoru
X0.1 = 0 .... čítač dolů
Vstup X0.2 je nulovací.

```
#program  linka, v3.0
#def      snimac          x0.0
#def      volba           x0.1
#def      nuluj           x0.2
#reg      word            citac
p 0
          ld      snimac
          and     volba
          ld      snimac
          anc     volba
          ld      nuluj
          cnt     citac
          wr      vystupy
e 0
```

Program bude správně fungovat, nebudeme-li přepínat volbu v okamžiku, je sepnut vstup snímač. Chceme-li ošetřit i tento stav, musíme zajistit, aby čítač nereagoval na změny vstupu volba.

```
p 0
          ld      volba           ;test náběžné hrany signálu volba
          let     nabezna
```

```
        ldc     volba           ;test spádové hrany signálu volba
        let     spadova
        or
        set     hrana_volba     ;pomocný bit hrana_volba se nastaví při změně volba
        ld      signal
        let     hrana_signal
        res     hrana_volba     ;pomocný bit hrana_volba se nuluje při náběžné hraně
        ld      snimac          ;signal
        and     volba
        anc     hrana_volba
        ld      snimac
        anc     volba
        anc     hrana_volba
        ld      nuluj
        cnt     citac
        wr      vystupy
e 0
```

7.5.2 POSUVNÉ REGISTRY

Pod pojmem posuvný registr rozumíme oblast paměti, která kromě paměťové funkce dokáže posouvat informaci od jednoho bitu ke druhému doprava nebo doleva. Princip posuvného registru vlevo se sériovým vstupem je zřejmý z obr. 6.

V instrukčním souboru TECOMAT existují následující instrukce posuvných registrů:

- **SFL** (Shift to Left) - posun vlevo
- **SFR** (Shift to Right) - posun vpravo

Formát instrukcí:

SFL RW číslo
SFR RW číslo

Pozn. Instrukce SFL a SFR mohou pracovat nad stejným objektem, ale musí mít různé řídící proměnné.

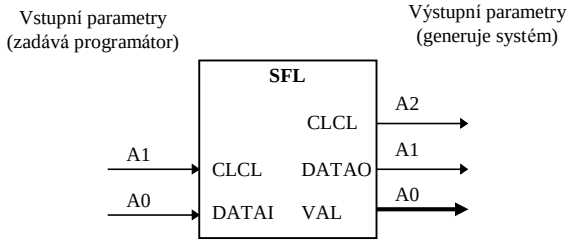
Instrukce SFL

Obr. 6 Princip posuvného registru

Vstupní parametry:
A1: řídící proměnná (CLCL)
A0: sériový vstup (DATAI)

Výstupní parametry:
A2: řídící proměnná (CLCL)
A1: hodnota vysouvaného bitu (DATA0)
A0: stav posuvného registru (VAL)

Odpovídající schematická značka:



Funkce: Při náběžné hraně CLCL se celý obsah slova posune o jedno místo vlevo a na pozici nejnižšího bitu se nasune obsah proměnné DATAI. Nejvyšší bit se vysune jako DATA0 do A1.

Instrukce SFR

Pracuje podobně jako SFL ale v opačném směru.

7.5.3 ČASOVAČE

Umožňují doplnit funkci navrženého obvodu o časové zpoždění sepnutí výstupu (obdoba relé se zpožděným přítahem), časové zpoždění rozpojení výstupu (obdoba relé se zpožděným odpadem) a generování impulsu zvolené šířky. Máme k dispozici čtyři typy časovačů:

- **TON** (Timer On) - časování od sepnutí vstupu - posunutá náběžná hrana
- **TOF** (Timer Off) - časování od rozepnutí vstupu - posunutá spádová hrana
- **RTO** (Retentive Timer Object) - integrující časovač
- **IMP** (Impuls) - generátor impulsů od náběžné hrany

Formát instrukcí

TON RW číslo . kód	kód =	0 .... 10 ms	Pozn. není-li zadán kód,
TOF RW číslo . kód		1 .... 100 ms	pracuje čítač s
RTO RW číslo . kód		2 .... 1 s	jednotkou 10 ms (kód 0)
IMP RW číslo . kód		3 .... 10 s	

Základní pravidla
Nad jedním objektem smí pracovat pouze jediná instrukce časovače!
Časoměrné proměnné jsou aktualizovány pouze v otočce cyklu, v jednom oběhu programu se čas nemění.
Časovač musí být obsluhován v každém cyklu (jinak se čas zastaví).

Příznaky S0

S0.0 (ZR)	předvolba = obsah časovače (předvolba právě dosažena)
S0.1 (CO)	předvolba < obsah časovače (předvolba již překročena)
S0.2 (CO + ZR)	předvolba ≤ obsahu časovače
S0.4	překročení max. rozsahu časovače
S0.5	překročení max. rozsahu časovače právě v tomto cyklu

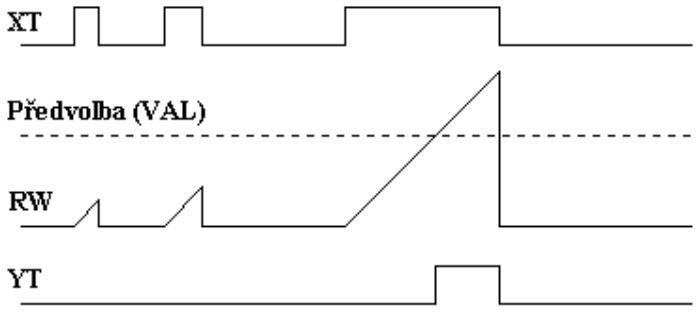
Instrukce TON

Vstupní parametry:	*Výstupní parametry:*
A1: řídící proměnná (XT)	A1: řídící proměnná (XT)
A0: číselná hodnota předvolby (VAL)	A0: výstupní proměnná (YT)

Funkce: Časovač TON je aktivní je-li řídící proměnná XT v úrovni "1". Od okamžiku náběžné hrany XT začíná časovat - inkrementuje stav čítače každou časovou jednotku a výsledek porovnává s předvolbou. Není-li dosaženo předvolby, je výstup YT nulový, je-li předvolba dosažena nebo překročena je YT = 1.

Nulová hodnota řídící proměnné XT uvádí časovač do pasivního režimu, ve kterém:
je nulován obsah časovače
jsou nulovány příznaky S0

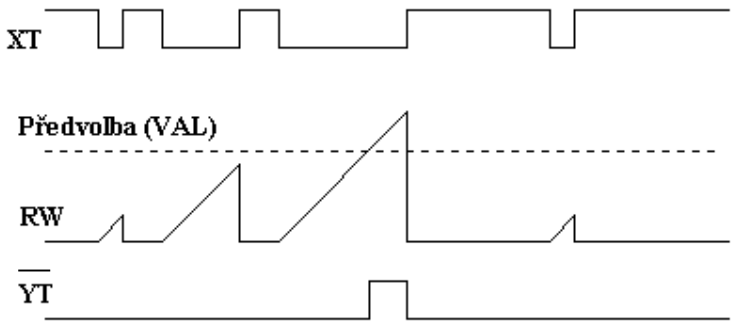
Činnost časovače TON je ilustrována časovými diagramy na obr. 7.



Obr. 7 Princip činnosti časovače TON

Instrukce TOF

Vstupní a výstupní parametry jsou stejné jako u instrukce TON.
Funkce: Časovač TOF je aktivní, je-li řídící proměnná XT v úrovni "0". Od okamžiku spádové hrany XT začíná časovat.
Jedničková hodnota řídící proměnné uvádí časovač do pasivního režimu. Na rozdíl od funkce TON posouvá TOF spádovou hranu XT o čas daný předvolbou násobenou kódem.
Činnost časovače TOF je ilustrována časovými diagramy na obr. 8.

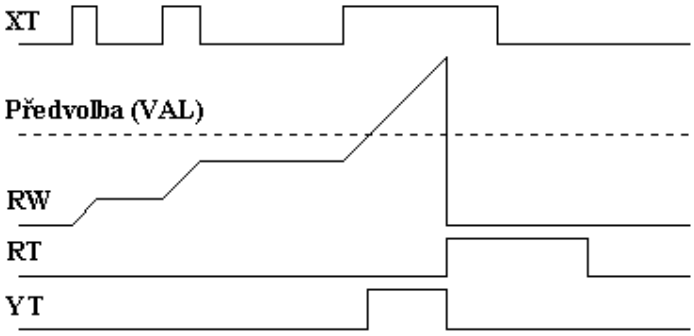


Instrukce RTO

Realizuje funkci integrujícího časovače, který je obdobou instrukce TON s tím rozdílem, že nulová hodnota XT nenuluje údaj časovače ani příznaky, pouze se na tuto dobu zastaví časování. Pro nulování časovače je zavedena nová proměnná RT.

Vstupní parametry:	**Výstupní parametry:**
A2: řídící proměnná (XT)	A2: přetečení přes max. rozsah (YC)
A1: nulovací proměnná (RT)	A1: nulovací proměnná (RT)
A0: číselná hodnota předvolby (VAL)	A0: výstupní proměnná (YT)

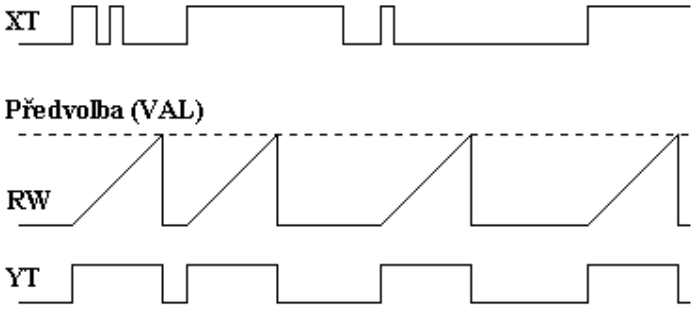
aktivní stav: RT = 0 XT = 1
čekací stav: RT = 0 XT = 0
pasivní stav: RT = 1 XT = X (na stavu proměnné XT nezáleží)



Obr. 8 Princip činnosti časovače RTO

Instrukce IMP

Vytváří impuls délky rovné hodnotě předvolby odvozený od náběžné hrany řídící proměnné. Vstupní a výstupní parametry stejné jako u TON a TOF. Do aktivního stavu se časovač dostává náběžnou hranou řídící proměnné XT. V aktivním stavu je až do okamžiku dosažení předvolby. Potom se vrací do pasivního stavu. Další náběžnou hranou XT je znovu aktivován. Během aktivního stavu nemá průběh XT vliv na činnost časovače.



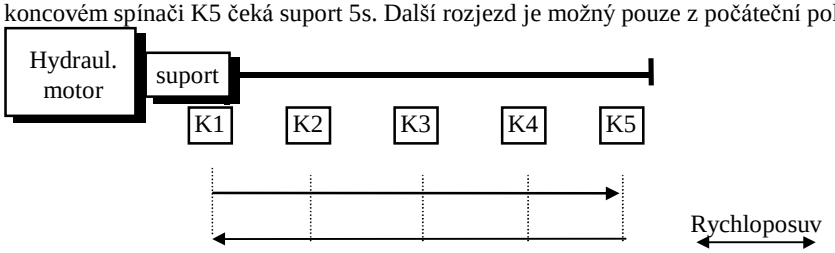
Příklady:

1) Realizujte časovou funkci, která potlačí jedničkové impulsy na vstupu *Signal* (X0.0) kratší než 250 ms, delší impulsy předá na *výstup* Y0.0 se zpožděním 250 ms.

```
#program filtr
#def    signal          x0.0
#def    vystup          y0.0
#def    zpozdeni  25
#reg    word            casovac
p 0
```

```
        ld      signal
        ld      zpozdeni
        ton     casovac.0
        wr      vystup
e 0
```

2) Navrhněte logický obvod pro řízení hydraulické posuvové jednotky v režimu dle obrázku. Po současném stisku tlačítek START1 a START2 se vykoná jeden pracovní cyklus, který může být kdykoli zastaven tlačítkem STOP. Na koncovém spínači K5 čeká suport 5s. Další rozjezd je možný pouze z počáteční polohy (K1).



Hydraulický motor je řízen elektromagnetickými ventily EM1, EM2, EM3.

EM1	elektromagnetický ventil - pohyb vpřed
EM2	elektromagnetický ventil - pohyb vzad
EM3	elektromagnetický ventil-řízení rychlosti (EM1=1 prac.posuv)

```
#program suport
#include  konfigurace.mdl
#def            start1          x0.0
#def            start2          x0.1
#def            k1              x0.2
#def            k5              x0.3
#def            em1             y0.0
#def            em2             y0.1
#def            cekani          5
#def            sec             2
#reg            word            casovac
;********************************** deklarace proměnných a konstant
p 0
        ld      start1          ; obsluha em1
        and     start2
        and     k1
        set     em1
        ld      k5
        res     em1
;**********************************
        ld      k5              ; obsluha em2
        ld      cekani
        ton     casovac.sec
        set     em2
        ld      k1
        res     em2
e 0
```

3) Signalizujte pomocí výstupu Y0.7, že celková doba, po kterou byl v chodu motorek připojený na výstup Y0.0 přesáhla 1 hodinu.

```
#program nabrousit_vrtak
#include  konfigurace.mdl
#def            motor           y0.0
#def            signalizacey0.7
#def            zivotnost   3600
#def            sec             2
#def            nuluj           x0.0
#reg            word            casovac
;**********************************  deklarace proměnných a konstant
p 0
        ld      motor
        ld      nuluj
        ld      zivotnost
        rto     casovac.sec
        wr      signalizace
e 0
```

7.6 ORGANIZAČNÍ A ŘÍDÍCÍ INSTRUKCE

Organizační a řídící instrukce řídí algoritmus vykonávání uživatelského programu, definují začátky a konce procesů, ladící body, realizují skoky v programu, volání podprogramů atd.

Instrukce P n, E n

Označují místo v programu, na kterém začíná, případně končí příslušný proces.
Parametr *n* nabývá hodnot pouze v rozsahu čísel přípustných procesů (pro centrální jednotky řady E je to pouze 0).
Proces začínající instrukcí P n musí být ukončen párovou instrukcí E n.

Instrukce ED, EC, EOC

Instrukce ED	konec procesu při nenulovém obsahu akumulátoru
Instrukce EC	konec procesu při nulovém obsahu akumulátoru
Instrukce EOC	nepodmíněný konec cyklu

Instrukce BP

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 L (Label)

L n
Instrukce označuje podprogramy nebo cílové místo skoků. Neprovádí žádnou činnost, chová se jako prázdná instrukce. V programu *nesmí být více instrukcí L* se stejným operandem.
V prostředí xPRO se návěští většinou definuje symbolickým jménem (řetězec znaků zakončený dvojtečkou nebo direktivou #Label).

Instrukce skoků v programu

- **JMP** L n (Jump) - nepodmíněný skok
- **JMD** L n (Jump if Direct Condition) - Skok podmíněný nenulovým obsahem akumulátoru.
  Skok se provede, není -li A0 = 0, v ostatních případech program pokračuje další instrukcí.
- **JMC** L n (Jump if Complement Condition) - Skok podmíněný nulovým obsahem akumulátoru.
  Skok se provede, je -li A0 = 0, v ostatních případech program pokračuje další instrukcí.
- **JMI** (Jump Indirect) Nepodmíněný skok na návěští uložené v akumulátoru.

Instrukce skoků nemění obsah ani úroveň zásobníku.

Instrukce volání podprogramů

Všechny podprogramy jsou uloženy v procesu **P 60**, který není zapojen v programové smyčce. Každý podprogram začíná návěštím, které slouží jako parametr při jeho volání a končí některou z návratových instrukcí.
Z aktuálního podprogramu je možné volat další podprogram (počet vnoření je max. 8).

- **CAL** L n (Call) - nepodmíněné volání podprogramu
- **CAD** L n (Call if Direct Condition) - volání podprogramu podmíněné nenulovým obsahem akumulátoru
- **CAC** L n (Call if Complement Condition) - volání podprogramu podmíněné nulovým obsahem akumulátoru
- **CAI** (Call Indirect) - nepodmíněné volání podprogramu, jehož L je uloženo v akumulátoru

Návratové instrukce

- **RET** (Return) - nepodmíněný návrat z podprogramu
- **RED** (Return if Direct Condition) - návrat podmíněný nenulovým obsahem akumulátoru
- **REC** (Return if Complement Condition) - návrat podmíněný nulovým obsahem akumulátoru

Příklady:
1) Ve všedních dnech bude řízený objekt obsluhován podle algoritmu, zapsaného v sekci_A uživatelského programu, jinak podle algoritmu zapsaného v sekci_B.

```
#program        skok
#include  konfigur.mdl
#def            dny_v_tydnu     s9

p 0
        ld      dny_v_tydnu
```

Výhodnější je použít podprogramy:

```
p 0
        ld      dny_v_tydnu
        gt      5
        cad     sekce_B
        cac     sekce_A
e 0
p 60
```

```
            gt          5
            jmd         sekce_B
;sekce_A
;blok programu sekce_A
            .
            .
            .
            jmp         konec
sekce_B:
;blok programu sekce_B
            .
            .
            .
konec:
e 0
```

2) Výrobní linka může pracovat ve čtyřech různých režimech, jejichž volbu provádí obsluha spínači A, B.

B	A	REŽIM
0	0	režim_a
0	1	režim_b
1	0	režim_c
1	1	režim_d

```
p 0
            ld          b                   ; příprava návěští do akumulátoru
            ld          a
            stk
            and         %11
            cai
e 0

p 60
L 0         ; program řešící režim_a
.
.
ret
L 1         ; program řešící režim_b
.
.
ret
L2          ; program řešící režim_c
.
.
ret
L 3          ;program řešící režim_d
.
.
ret
e 60
```

Rozšířený instrukční soubor:
Instrukce SEQ přerušení procesu podmíněné nulovostí vrcholu zásobníku, proces začne v příštím cyklu od návěští L n
Instrukce JZ, JNZ, JC, JNC, JS, JNS provádí skoky podmíněné hodnotami příznakových bitů

7.7 INSTRUKCE NAD TABULKAMI

Tabulka je úsek dat v uživatelské paměti RAM strukturovaný jako řada bitových, bytových nebo slovních položek.

Položky tabulek mají nejčastěji význam pravdivostních hodnot logických funkcí, čísel návěští nebo libovolných číselných hodnot (posloupnosti časových údajů, žádaných hodnot řídící veličiny atd.).

Tabulkami lze jednoduše realizovat složité kombinační i sekvenční funkce i časové procesory. Instrukce s tabulkami dávají uživateli do rukou silný nástroj umožňující řešit úlohy mnohem efektivněji, než klasickým způsobem.
Tabulky je možné vytvořit nejen v uživatelské paměti jako součást programu, nýbrž i v oblasti zápisníkové paměti.
Formáty tabulek

- bitové : řada bitových hodnot v max. délce 256 bitů
- slabikové: řada bytových hodnot v max. délce 256 slabik
- slovní : řada slovních hodnot v max. délce 128 slov

Index tabulky
Jednotlivé položky tabulky jsou číslovány počínaje nulou. Pořadové číslo položky se nazývá index tabulky.

Mez tabulky
Index poslední položky se nazývá mez tabulky. Délka tabulka je tedy rovna MEZ + 1.

Příklad definování bitové tabulky a její uložení v paměti:
definice tabulky logik_bit v xPRO:

```
#table    bit       logik_bit = 1,0,0,1,0[5],1,1,0,1
```

uložení tabulky logik_bit v paměti:

Index =	7	6	5	4	3	2	1	0	
	0	0	0	0	1	0	0	1	slabika paměti č. 0
	0	0	0	1	0	1	1	0	slabika paměti č. 1
Index =	15	14	13	12	11	10	9	8	

Příklad definování slabikové tabulky a její uložení v paměti:
definice tabulky cisla_byte v xPRO:

```
#table    byte      cisla_byte = 22,45,240,10,100
```

uložení tabulky cisla_byte v paměti:

Index=0	22	slabika paměti č. 0
Index=1	45	slabika paměti č. 1
Index=2	240	slabika paměti č. 2
Index=3	10	slabika paměti č. 3
Index=4	100	slabika paměti č. 4

Příklad definování slovní tabulky a její uložení v paměti:
definice tabulky data_word v xPRO:

```
#table    word      data_word = $56A7,$250F,$B9AA,$FF
```

uložení tabulky data_word v paměti:

Index=0	A7	slabika paměti č. 0
	56	č. 1
Index=1	0F	č. 2
	25	č. 3
Index=2	AA	č. 4
	B9	č. 5
Index=3	0	č. 6
	FF	č. 7

Nad tabulkami T lze provádět tyto operace:
- výběr položky k zadanému indexu (LTB)

- zápis položky podle zadaného indexu (WTB)
- nalezení indexu k zadané hodnotě položky (FTB)
- nalezení indexu k vybrané části položky (FTM)
- nalezení indexu třídy - zařazení zadané hodnoty do jedné z tříd (skupin), které jsou určeny tabulkou mezí v uspořádané řadě hodnot (FTS)
- blokové přenosy dat mezi tabulkou zápisníkem (SRC, MOV, MTN, MNT)
- sekvenční čtení z tabulky (LMS)
- sekvenční zápis do tabulky (WMS)
- práce se strukturovanými tabulkami (LDS, WRS, FIT, FNT)

Instrukce LTB

(Load from table) - čtení z tabulky

LTB nad tabulkou T

FORMÁT TABULKY: bit, byte, word
LTB jméno tabulky

V AKUMULÁTORU OČEKÁVÁ:
A0 - index položky

PO VYKONÁNÍ:
A2 - mez tabulky
A1 - index žádané položky
A0 - obsah položky

ÚROVEŇ ZÁSOBNÍKU: zvyšuje se o jednotku

PŘÍZNAKY:
Je-li index ≤ mez (je v rozsahu tabulky) => **S1.0 = 1**
Je-li index > mez (index je přepočítán "modulo mez+1") => **S1.0 = 0**

LTB nad zápisníkovou pamětí
Funkce je obdobná, vstupními parametry jsou: A0 .... INDEX
A1 .... LIMIT (mez tabulky)
V zásobníku je třeba vyhradit pro tabulku direktivou #reg LIMIT + 1 položek.

Příklady:
1) Řešte logickou funkci Z(A,B,C), která je zadána pravdivostní tabulkou:

Index	A	B	C	Z
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Vytvoříme bitovou tabulku, do které podle rostoucího indexu zapíšeme pravdivostní hodnoty Z:

```
#table  bit     majorita=0,0,0,1,0,1,1,1
p 0
        ld      a               ;načtení vstupů
        ld      b
        ld      c
        stk
        and     %111            ;maska
        ltb     majorita        ;čtení z tabulky
        wr      z               ;zápis na výstup
e 0
```

Z daného programu vyplývá, že řešení logických obvodů pomocí tabulek je velmi jednoduché. Při klasickém způsobu řešení bychom museli napřed provést syntézu logického obvodu, získat algebraické rovnice a tyto pak naprogramovat.
Řešení pomocí instrukcí s tabulkami je navíc naprosto univerzální - program je pro všechny logické obvody stejný, řešení se liší pouze v obsahu tabulky.

2. Realizujte převodník binárního kódu na bitech A,B,C (A je LSB) na kód 1 z 8.
Logický obvod bude mít 3 vstupy A,B,C a 8 výstupů Z_0, Z_1, ... Z_7 , ze kterých bude aktivní (v úrovni 1) pouze jeden podle kombinace vstupních proměnných.

Přiřazení proměnných na svorky automatu:

C	B	A	Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7
X0.2	X0.1	X0.0	Y0.0	Y0.1	Y0.2	Y0.3	Y0.4	Y0.5	Y0.6	Y0.7

Kombinace C,B,A určuje adresu Index v tabulce, kde bude v šířce slabiky zapsaný požadovaný výstupní stav.

```
#table    byte      dekoder=%1,%10,%100,%1000,%10000,%100000,%1000000,%10000000
p 0
          ld        c                 ;načtení vstupů
          ld        b
          ld        a
          stk
          and       %111              ;maska
          ltb       dekoder  ;čtení z tabulky
          wr        y0                ;zápis na výstupy
e 0
```

Instrukce FTB

(Find in table) - hledej v tabulce

FTB nad tabulkou T

FORMÁT TABULKY: byte, word

FTB jméno tabulky

V AKUMULÁTORU OČEKÁVÁ:
A0 - obsah, který má být v tabulce nalezen

PO VYKONÁNÍ:
A1 mez tabulky
A0 a) hodnota indexu (A0L=index, A0H=0), pokud zadaný obsah leží v tabulce
b) mez+1, pokud zadaný obsah není v tabulce

ÚROVEŇ ZÁSOBNÍKU: nemění se

PŘÍZNAKY:
je-li zadaná položka v tabulce **S1.0=1**
není-li zadaná položka v tabulce **S1.0=0**

Instrukce FTB postupně porovnává údaj zadaný v akumulátoru s obsahy položek tabulky (počínaje položkou 0). Obsahuje-li tabulka více položek shodných s A0, zastaví se hledání po nalezení první shody. Po nalezení položky v tabulce instrukce předá její index do A0.

Variantou instrukce FTB je:

- **FTM** - hledání části položka (zadává se maska)
- **FTS** - zařazení položky (rozděluje data do tříd, jejichž meze jsou definovány v tabulce

FTB nad zápisníkovou pamětí
Funkce je obdobná, vstupními parametry jsou: A0 .... obsah, který má být v tabulce nalezen
A1 .... LIMIT (mez tabulky)
V zásobníku je třeba vyhradit pro tabulku direktivou #reg LIMIT + 1 položek.

Příklady:
1. Testujte obsah registru R10. Pokud nabývá některou z hodnot: 3, 15, 30, 65, 137, 240, pak R7 neměňte, jinak jej vynulujte.

Uvedené hodnoty zapíšeme do slabikové tabulky Tab_reset.

```
#program            reset
#table    byte      Tab_reset = 3, 15, 30, 65, 137, 240
P 0
          LD R10              ;načtení obsahu R10
          FTB Tab_reset       ;porovnání obsahu R10 s tabulkou Tab_reset
```

```
        LDC S1.0        ;nenajde-li, pak A0 = 65 535
        RES R7          ;podmíněné nulování R7
E 0
```

2. Proveďte větvení programu podle hodnoty registru R7.

a) Přiřazení návěští:

R7 = 2	L 10
R7 = 10	L 11
R7 = 40	L 12
jinak	L 13

Do slabikové tabulky zapíšeme hodnoty podle nichž větvíme program:

```
#program        navesti1
#table  byte    vetve = 2,10, 40
p 0
        ld      r7      ;načtení obsahu r7
        ftb     vetve   ;porovnání s obsahem tabulky
        add     #10     ;k nalezenému indexu přičteme 10
        jmi             ;skok na návěští s adresou uloženou v a0l
        .........
```

Čtenář může namítnout, že seřazením návěští jsem si usnadnil práci. Proto uvádím variantu b.

b) Přiřazení návěští:

R7 = 2	L 20
R7 = 20	L 8
R7 = 35	L 25
R7 = 100	L 8
jinak	L50

Připravíme tabulky VETVE (obsahy R7) a NAVESTI (návěští). Při sestavení tabulek dáme pozor na to, aby rozhodovací hodnota a odpovídající návěští měly stejný index.

```
#program        navesti1
#table  byte    vetve = 2, 20, 35, 100
#table  byte    navesti = 20, 8, 25, 8, 50
p 0
        ld r7           ;načtení obsahu r7
        ftb vetve       ;porovnání s obsahem tabulky VETVE
        ltb navesti     ;čtení návěští z NAVESTI podle nalezeného indexu
        jmi             ;skok na návěští uložené v A0L
        ........
```

3. Navrhněte časový procesor pro výstup Y0.0 podle pravidel:

Y0.0=1 denně	od 8^{00} do 8^{45}
	od 8^{50} do 9^{35}
	od 9^{45} do 10^{30}
	od 10^{50} do 11^{35}
	od 11^{40} do 12^{25}
	od 14^{10} do 14^{55}
	od 15^{05} do 15^{50}
Využijeme registry	S7 - minuty
	S8 - hodiny

Připravíme 2 slovní tabulky:

```
; sepni (spínací časy)
#table  word  sepni =          $800,$832,$92D,$A32, $B28,  $E0A,  $F05
;                       (8 00) (8 50) (9 45) (10 50) (11 40) (14 10) (15 05)
; vypni (vypínací časy)
#table  word  vypni  = $82D,$923,$A1D,$B13,$C19,$E37,$F32
p 0
        ld sw7
        ftb sepni  ; zapínací časy
        ld s1.0
        set y0.0   ; když našel, tak zapne
        ld sw7
        ftb vypni  ; vypínací časy
        ld s1.0
        res y0.0   ; když našel, tak vypne
e 0
```

Instrukce LMS

sekvenční čtení z tabulky

OPERAND: tabulka T
FORMÁT TABULKY: word
LMS jméno tabulky

Vstupní parametry:
Operand - tabulka T, ze které se čte (formát word)
RWt - index čtené položky, číslo registru je totožné s číslem tabulky
(formát word, CPU řady M formát byte)

Výstupní parametry:
A0 VAL - přečtený obsah (formát word)
A1 až A7 - původní vrstvy A0 až A6

Úroveň zásobníku: zvyšuje se o jednotku

Příznaky:
Index je v rozsahu tabulky => **S1.0 = 1**
Index podtekl pod 0, je nastaven na horní mez tabulky => **S1.0 = 0**

Funkce:
Instrukce LMS je sekvenční obdobou instrukce LD. Pomocí této instrukce se přečte z tabulky položka a připraví se ke čtení další položka o nižším indexu. K uchovávání indexu položky připravené ke čtení se používá registr typu word (u centrálních jednotek řady M typu byte) stejného čísla jako adresovaná tabulka. Tak lze postupně číst položky typu word z tabulky tehdy, kdy je třeba, bez nutnosti starat se o index (musíme samozřejmě rezervovat příslušný registr RW).

Instrukce WMS

sekvenční zápis do tabulky

OPERAND: tabulka T
FORMÁT TABULKY: word

WMS jméno tabulky

Vstupní parametry:
A0 - zapisovaný obsah (word)
Operand - tabulka T, do které se zapisuje (formát word)
RWt - index zapisované položky, číslo registru je totožné s číslem tabulky
(formát word, CPU řady M formát byte)

Úroveň zásobníku: nemění se

Příznaky:
Index je v rozsahu tabulky => **S1.0 = 1**
index přetekl přes horní mez tabulky, je nastaven na 0 => **S1.0 = 0**

Funkce:
Instrukce WMS je sekvenční obdobou instrukce WR. Pomocí této instrukce se zvýší index a položka se zapíše do tabulky. K uchovávání indexu položky připravené ke zápisu se používá registr typu word (u centrálních jednotek řady M typu byte) stejného čísla jako adresovaná tabulka.

Rozšířený instrukční soubor:
U centrálních jednotek typu D, B je tabulkový prostor omezen pouze kapacitou paměti
Instrukce WTB provádí zápis do zvolené položky tabulky
Instrukce se strukturovanými tabulkami

7.8 ARITMETICKÉ INSTRUKCE

Umožňují jednoduché matematické operace s přirozenými čísly. Operandy většiny instrukcí mají šířku 16 bitů (slovo). Vhodným použitím těchto instrukcí můžeme vyřešit daný algoritmus mnohdy efektivněji než při použití samotných logických instrukcí. V tomto textu si vysvětlíme *instrukci ADD* - sčítání a uvedeme přehled dalších aritmetických operací

Instrukce ADD - sčítání s přenosem

PROSTOR OPERANDU: X,Y, S, R, D, #, bez operandu (A0, A1)

ŠÍŘKA OPERANDU : word

ZÁSOBNÍK : A0 se přepisuje výsledkem operace
úroveň se u bezoperandové instrukce snižuje

FUNKCE:

(operandová instrukce) *Vstupní parametry:*

A0 - proměnná A
Operand - proměnná B

Výstupní parametry:

A0 - výsledek = A + B + CI

(bez operandová instrukce) *Vstupní parametry:*

A0 - proměnná B
A1 - proměnná A

Výstupní parametry:

A0 - výsledek = A + B + CI
A1 až A6 - původní vrstvy A2 až A7

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.
Bezoperandové instrukce jsou určeny k realizaci závorkovaných výrazů a pracují výhradně se zásobníkem. 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.

Další aritmetické instrukce

- **SUB** - odčítání s přenosem
- **MUL** - násobení
- **DIV** - celočíselné dělení
- **INR** - inkrementace akumulátoru
- **DCR** - dekrementace akumulátoru
- **ROL** - rotace akumulátoru vlevo
- **BIN** - převod desítkového čísla v kódu BCD do binárního formátu
- **BCD** - převod čísla v binárním formátu do kódu BCD

Rozšířený instrukční soubor:
Instrukce ADX, ADL provádí sčítání ve formátu *long*
Instrukce SUX, SUL provádí odčítání ve formátu *long*
Instrukce MUD, DID násobení, případně dělení s operandy ve formátu *word*
Instrukce INR a DCR se zápisníkem
Instrukce ROR - rotace akumulátoru doprava
Instrukce BIL, BCL - převody mezi binárním a BCD kódem ve formátu *long*
Instrukce BAS, ASB - převody mezi binárním kódem a formátem ASCII

Centrálních jednotky typu D, B obsahují širokou sadu instrukcí v plovoucí desetinné čárce

Obsah