Prvky mikroprocesorového systému řady 3000

ING. TOMÁŠ GUCKÝ

Článek informuje potenciální uživatele o připravovaných mikroprocesorových obvodech TESLA řady MH3000. Definuje základní pojmy a popisuje funkce jednotlivých prvků. Navazující článek bude věnován metodice aplikací s uvedenými obvody.

Základní pojmy

Ústřední jednotkou soudobých elektronických systémů na zpracování informace je zpravidla procesor, doplněný příslušnými vstupy, výstupy a paměťmi. Procesorem rozumíme tu část systému, která provádí hlavní výpočetní operace s daty podle zadaného programu.

Zavedením programovatelných číslicových obvodů velké integrace se tato hierarchie pojmů posouvá z oblasti systémů do oblasti dílčích jednotek systémů. Jako mikroprocesor označujeme procesor sestávající z jednoho nebo několika pouzder obvodů LSI. Jeho prioritní aplikací je mikropočítač, kterým v současných systémech řešíme dílčí úlohy zpracování dat v jednotkách realizovaných dříve jednoúčelovou logikou.

Zatímco unipolární polovodičové technologie umožňují integrovat standardně 8 bitový procesor v jednom pouzdře, tradiční bipolární technologie tuto možnost neposkytují. Nevýhodu nižšího stupně integrace však vyvažují vyšší operační rychlostí a přizpůsobitelností aplikací.

Bipolární mikroprocesory sestávají zpravidla z mikroprogramovaného řadiče a z operační jednotky, složené z obvodově totožných řezů. Mikroprogramování umožňuje aplikátorovi volbu optimálního souboru instrukcí, přizpůsobeného řešenému problému. Řezová skruktura dovoluje volit šířku toku dat procesorem podle potřeby aplikace pouhým zařazením zvoleného počtu obvodů. Na *obr. 1* je naznačena struktura takového mikroprocesoru a jeho vztah ke spolupracujícím jednotkám v nadřazeném systémovém celku. V *tabulce* 1 jsou definovány některé pojmy, dále v článku používané.

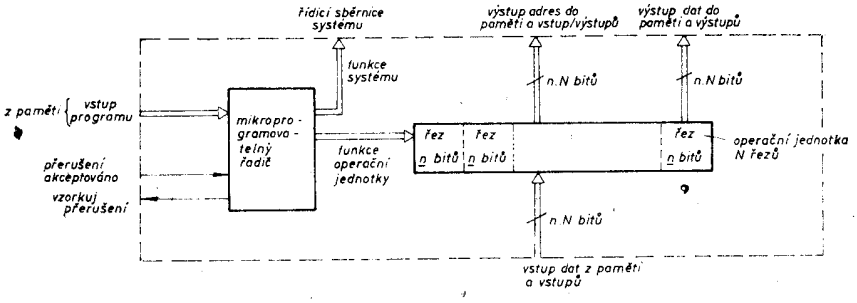
Architektura mikroprocesorového systému řady 3000

Z možností použití technologie STTL a ze snahy po minimalizaci počtu typů vyplynula potřeba realizace následujících obvodů:

— MH3001 — mikrořadič, obvod řízení průběhu mikroprogramů,
— MH3002 — dvoubitový řez operační jednotky,
— MH3003 — rychlý přenosový obvod; realizuje aritmetický přenos 16 bitového součtu,
— MH3205 — dekodér 1 z 8 pro rychlé přepínání pamětí, vstupů, výstupů,
— MH3212 — osmibitová vstupní/výstupní brána,
—MH3214 — řadič priority přerušení,
— MH3216 (26) — čtyřbitový neinvertující (invertující) budič sběrnice.

Hlavními prvky systému jsou obvody 3001 a 3002. Ostatní vytvářejí doplňkové funkce, umožňující styk procesoru s okolím nebo zlepšení některých jeho vlastností. Základní uspořádání procesoru z obvodů řady 3000 je naznačeno na *obr. 2*. Musíme rozlišovat jeho funkce na systémové úrovni a na vnitřní úrovni, kterou budeme označovat *mikro*.

Procesor vykonává systémové operace podle instrukcí programu, které čte z paměti ve formě strojového kódu na vstupní sběrnici *M*. Provádí operace s daty vyžádanými ze vstupních sběrnic *M*, *I*, nebo z vnitřních registrů operační jednotky. Výsledek operace umisťuje do cílového registru operační jednotky nebo na výstupní sběrnici *A* jako vypočtenou adresu nebo na sběrnici *D* jako vypočtená data. V příslušných fázích provedení



Obr. 1. Základní struktura typického bipolárního mikroprocesoru

Tab. 1. Význam základních pojmů

Pojem	Význam	Poznámka
Řadič	jednotka, která řídí chod (mikro) počítače nebo jeho části podle instrukcí programu	
Mikroprogramován[1]	způsob návrhu práce řadiče	
Mikrořadič (mikroprogramová jednotka)	část mikroprogramovaného řadiče, která řídí jeho chod podle instrukcí mikroprogramu	Microprogramm Control Unit
Řídicí paměť (paměť mikroprogramů)	část mikroprogr. řadiče, určená k trvalému uchování mikroprogramů	Control Memory
Fáze	jeden ze stavů řadiče během provádění instrukce	
Mikrooperace	činnost řadiče (nebo spolupracující jednotky) během jedné fáze	
Mikroinstrukce	kódovaný předpis pro provádění mikrooperace	
Operační jednotka	část (mikro) počítače, která provádí operace	
Aritmetická jednotka	část oper. jednotky, která provádí operace aritmetické, logické a posuvy	Arithmetic-Logic Unit
Pole registrů	část oper. jednotky pro přechodné pamatování operandů a výsledků operací a pro provádění operací přesunů; některé registry mohou mít spec. funkce	Register Array
Střádač	registr, v němž se vytvoří výsledek operace aritmetické jednotky	Accumulator
Operace	vytvoření výsledků z jednoho nebo více údajů (operandů) podle určených pravidel	
Strojová instrukce (stroj. kód) [1]	(kódovaný) předpis provedení jedné operace (mikro) počítače; obvykle sestává z oper. znaku a cílového výrazu	Computer Instruction (Code)
Operační znak (operač. část instr.) [1]	posloupnost symbolů, určující typ operace prováděné s operandy	Operation Code (Operation Part)
Cílový výraz (adresová část instrukce) [1]	posloupnost symbolů, jímž se jako hodnota přiřazuje adresa (např. operandů skoku umístění výsledku ap.)	Designational Expression (Adress Part)
Přerušení	dočasné potlačení prováděné posloupnosti (mikro) instrukcí následované plněním jiné nezávislé posloupnosti nebo návratem k dříve potlačené	Interrupt
Návěstí operační jednotky	jednobitový signál charakterizující nebo ovlivňující výsledek operace	Flag; u obv. 3001 FI-vstupní FO-výstupní
Indikátorový registr	jednobitový registr nastavitelný do definovaného stavu, který může záviset na výsledcích předchozích operací	u obv. 3001 klop. obvody C (pamatuje přenos); z (pamatuje nulový výsledek)

Poznámka: [1]) Pojem uvedený v závorce označuje kódovaný popis předchozího pojmu.

instrukce generuje řídící signály na sběrnici *R*. Tyto signály řídí vybavení, zápis a čtení paměti nebo vstupních/ /výstupních bran systému.

Funkce prvků procesoru ve fázích provádění jedné instrukce systémového programu budeme chápat jako mikrofunkce, vyjadřující činnost a vzájemný vztah operační jednotky, řadiče a systému. Tyto mikrofunkce popisujeme tzv. mikroinstrukcemi. Konečná sekvence mikroinstrukcí — mikroprogram — popisuje provedení jedné instrukce systému.

Mikroprogram je zpravidla započat určením adresy počáteční mikroinstrukce. Tu určuje řadič dekódováním tzv. primární části operačního znaku systémové instrukce, přivedené na sběrnici *PX*. Současně se tzv. sekundární část operačního znaku ukládá do vnitřního registru programu (*PR*) obvodu 3001, odkud může být v dalších fázích provádění instrukce dekódována.

Vypočtenou startovací adresou mikroprogramu (registr *MA*, sběrnice *MA*) je adresováno paměťové místo v tzv. řídicí paměti *CM*, nazývané též paměť mikroprogramů. Na výstupu řídicí paměti se na několika oddělených sběrnicích objeví mikrooperační kódy právě čtené mikroinstrukce. Stav jednotlivých sběrnic, odpovídajících jednotlivým polím mikroinstrukce, určí mikrooperace, které mají být v probíhající fázi provedeny. Například stav sběrnice *F* určuje mikrooperaci všech paralelních řezů 3002 operační jednotky. Stav sběrnice *K* definuje konstantu pro zvolenou mikrooperaci. Stav sběrnice *FC* určuje způsob převzetí a nastavení návěští mezi operační jednotkou a řadičem. Stav sběrnice *AC* určuje způsob výpočtu adresy následující mikroinstrukce. Stav řídicí sběrnice *R* určuje další mikrofunkce procesoru ve vztahu k řízení systému, podle volby aplikátora.

Další průběh mikroprogramu závisí obecně na obsahu příslušné systémové instrukce, kterou realizuje. Zpravidla v úvodní fázi čte cílový výraz ze sběrnice *M* a dokončuje dekódování operačního znaku z registru *PR*, v prováděcí fázi čte operandy, provádí operaci, umístí výsledek, změní obsah programového čítače apod.

V každém taktu hodin mikrořadiče je čtena a provedena jedna mikroinstrukce. Ukončení mikroprogramu je hlášeno obvodem 3001 na výstupu *ISE* za účelem případné obsluhy požadavku přerušení běžícího programu.

Délka paralelně zpracovávaného slova dat (tj. šířka vstupních/výstupních sběrnic *M*, *I*, *K*, *A*, *D*) je volitelná uživatelem v násobku 2× počet pouzder 3002. Aritmetický přenos mezi jednotlivými řezy se šíří postupně. To může mít omezující vliv na rychlost provedení aritmetických operací u procesorů s dlouhým slovem. Přidáním rychlého přenosového obvodu 3003 se závislost na délce slova vylučuje. Dosažitelná rychlost operační jednotky i řadiče je řádu 100 ns a bude podrobně diskutována v aplikačním popisu. Tokový a časový diagram funkcí procesoru je naznačen na *obr. 3*. Úvodní popis nezahrnuje pochopitelně všechny vlastnosti použitých prvků, ale dává jen počáteční představu o jejich součinnosti. Podrobnější charakteristika prvků následuje.

Mikořadič MH3001

Obvod má dvě základní funkce:

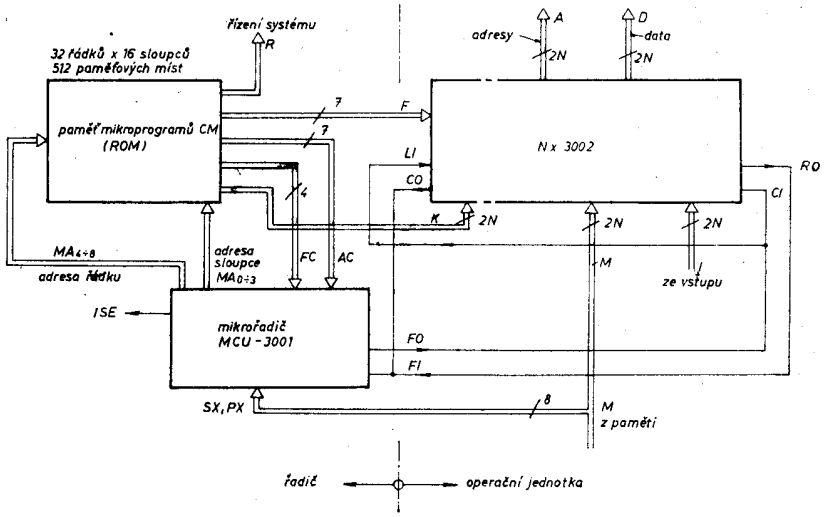
— výpočet adresy příští mikroinstrukce,
— převzetí návěští operační jednotky do indikátorových registrů a generování výstupních návěští.

Tabulky 2, 3 definují obě zmíněné funkce ve vztahu ke stavu řídicí sběrnice *AC* a *FC*, který určuje typ prováděné mikro operace obvodu 3001.

Tabulka 4 popisuje význam jednotlivých vývodů poudra obvodu, definuje vztah elektrického a logického signálu a ekvivalentní zátěž nebo zatížitelnost.

Funkční schéma — *obr. 4.* — sestává z:

— registru adresy a výstupních budičů



Obr. 2. Základní zapojení prvků řady 3000

Tab. 2. Funkce pro nastavení adresy příští mikroinstrukce MH 3001

druh mikroinstrukce	popis mikrofunkce	symbol	LD vst.	stav vstupů $AC_{0\div6}$ v taktu (t)							adresa příštího řádku (t+1)					odr. příštího sloupce (t+1)			
				6	5	4	3	2	1	0	MA_8	MA_7	MA_6	MA_5	MA_4	MA_3	MA_2	MA_1	MA_0
nepodmíněný skok	skok ve stávajícím sloupci	JCC	0	0	0	d_4	d_3	d_2	d_1	d_0	d_4	d_3	d_2	d_1	d_0	m_3	m_2	m_1	m_0
	skok na nultý řádek	JZR	0	0	1	0	d_3	d_2	d_1	d_0	0	0	0	0	0	d_3	d_2	d_1	d_0
	skok ve stávajícím řádku	JCR	0	0	1	1	d_3	d_2	d_1	d_0	m_8	m_7	m_6	m_5	m_4	d_3	d_2	d_1	d_0
	skok ve sloupci/vybavení PR výstupů (asyncho)	JCE	0	1	1	1	0	d_2	d_1	d_0	m_8	m_7	d_2	d_1	d_0	m_3	m_2	m_1	m_0
podmíněné skoky - dvouhodnotová větvení podle F,C,Z	skok s testem střadače F	JFL	0	1	0	0	d_3	d_2	d_1	d_0	m_8	d_3	d_2	d_1	d_0	m_3	0	1	f
	skok s testem indikátoru C	JCF	0	1	0	1	0	d_2	d_1	d_0	m_8	m_7	d_2	d_1	d_0	m_3	0	1	c
	skok s testem indikátoru Z	JZF	0	1	0	1	1	d_2	d_1	d_0	m_8	m_7	d_2	d_1	d_0	m_3	0	1	z
vícehodnotová větvení podle PX, SX, PR	skok s testem registru PR (16 hodnot)	JPR	0	1	1	0	0	d_2	d_1	d_0	m_8	m_7	d_2	d_1	d_0	p_3	p_2	p_1	p_0
	skok s testem levých bitů PR (4 hodnoty)	JLL	0	1	1	0	1	d_2	d_1	d_0	m_8	m_7	d_2	d_1	d_0	0	1	p_3	p_2
	skok s testem pravých bitů PR (4 hodnoty)	JRL	0	1	1	1	1	1	d_1	d_0	m_8	m_7	1	d_1	d_0	1	1	p_1	p_0
	skok s testem sběrnice PX (16 hodnot)	JPX	0	1	1	1	1	0	d_1	d_0	m_8	m_7	m_6	d_1	d_0	x_7	x_6	x_5	x_4
	přímé plnění X→MA (256 hodnot)	LD	1	libovolný stav							0	x_3	x_2	x_1	x_0	x_7	x_6	x_5	x_4

Legenda: d_0 až d_4 ..... stav příslušných vstupů AC v taktu t, určující částečně adresu MA v taktu t+1
m_0 až m_8 ..... stav adresového registru MA v taktu t
p_0 až p_3 ..... stav registru PR v taktu t
f, c, z ..... stav střadače F a registrů C, Z v taktu t před příchodem týlové hrany CLK
x_0 až x_3, x_4 až x_7 .. stav sběrnice SX_0 až SX_3, PX_4 až PX_7 v taktu t

platné stavy f, c, z
takt t — takt t+1

Tab. 3. Funkce pro nastavení návěstí MH3001

Popis funkce	Symbol	Typ funkce	Stav vstupů v taktu t				Stav registrů v taktu t + 1		
			FC_3	FC_2	FC_1	FC_0	C_{t+1}	Z_{t+1}	
Nastavení indikátorů *C*, *Z* na *f*	*SCZ*	vstupní	—	—	0	0	*f*	*f*	
Nastavení indikátoru *Z* na *f*	*STZ*	vstupní	—	—	0	1	*c*	*f*	
Nastavení indikátoru *C* na *f*	*STC*	vstupní	—	—	1	0	*f*	*z*	
Indikátory *C*, *Z* beze změny	*HCZ*	vstupní	—	—	1	1	*c*	*z*	
Naplň *FO* logickou 0	*FFO*	výstupní	0	0	—	—	$F_0 = 0$		asychronně v taktu t
Naplň *FO* obsahem indikátoru *C*	*FFC*	výstupní	0	1	—	—	$F_0 = C$		
Naplň *FO* obsahem indikátoru *Z*	*FFZ*	výstupní	1	0	—	—	$F_0 = Z$		
Naplň *FO* logickou 1	*FF1*	výstupní	1	1	—	—	$F_0 = 1$		

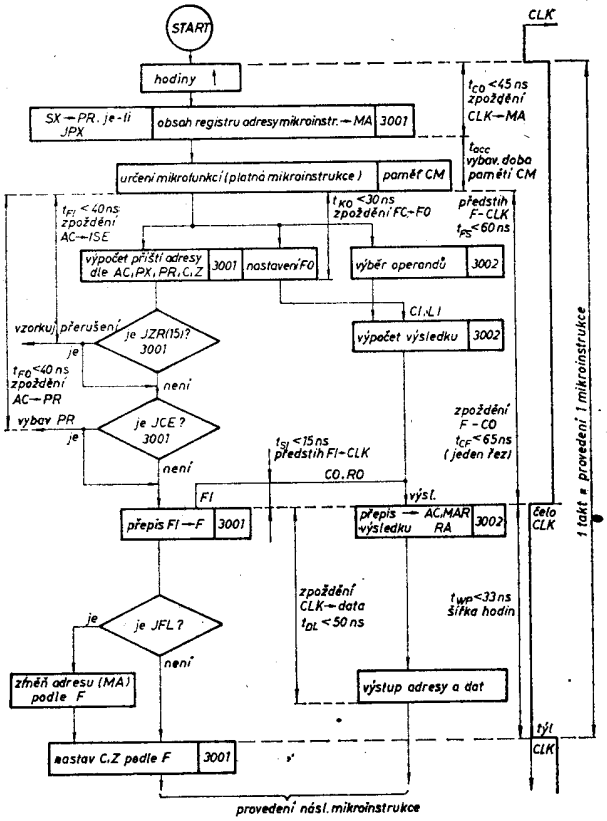
f .....obsah střadače *F* v taktu t (je dán stavem vstupu *FI* v době čela *CLK*)
c, *z* ..obsah registrů *C*, *Z* v taktu t

sběrnice MA,
— logiky výpočtu příští adresy,
— registru programu PR,
— návěstní logiky se střadačem F, indikátorovými registry C, Z a výstupním budičem návěstí FO.

Funkce obvodu počíná přepisem vypočtené adresy příští mikroinstrukce do registru MA s týlovou hranou hodin CLK. Po ustálení výstupů řídicí paměti se objeví na sběrnicích AC, FC platný kód, určující mikrofunkci pro tento takt. S čelní hranou hodin je převzato návěští operační jednotky ze vstupu FI do střádače F a může ještě proběhnout větvení mikroprogramu — nové určení adresy příští mikroinstrukce — jestliže byl zvolen skok podle návěští — JFL. S týlovou hranou hodin se opakuje přepis adresy a současně je přepisován obsah střádače F do klopných obvodů C a nebo Z. Větvení podle nového obsahu C, Z je možné v následujícím taktu.

Nastavení adresy MA je možné také přímým přepisem obsahu sběrnice PX, SX s týlovou hranou hodin, je-li vstup LD ve stavu H. Výstup registru PR je zpřístupněn jen v případě funkce JCE, a současného stavu H na vstupu EN, jinak je ve stavu H (otevřené kolektory).

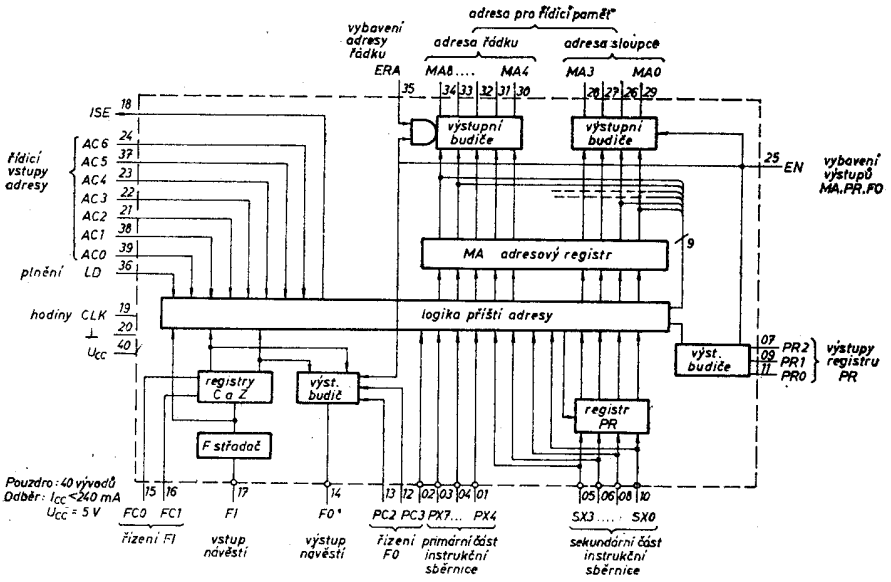
Výstup vzorkování přerušení ISE je ve stavu H jen v případě funkce JZR (15) — skoku na nultý řádek, 15. sloupec paměti CM. Vybavovací vstup ERA ve stavu L uvádí výstupy $MA4$ až $MA8$ adresy řádku do 3. stavu. Protože dosažení adresy příští mikroinstrukce je možné pouze podmíněným či nepodmíněným skokem, zavádí se pro lepší orientaci v mikroprogramu fiktivní organizace řídicí paměti CM jako matice s 32 řádky a 16 sloupci. Každá ze skokových funkcí uvedených v *tab. 2* umožňuje definovat v poli AC jen část adresy řádku nebo sloupce. Pro ilustraci je na *obr. 5* znázorněn skok z momentální na příští adresu pro některé stavy řídicí sběrnice AC. *Obr. 6* pak uvádí některé významné dynamické parametry obvodu 3001.



Obr. 3. Tokový a časový diagram funkcí procesoru 3000

Tab. 4. Význam vývodů MH3001

Vývod	Číslo vývodu pouzdra	Vstup proudy I_{IH}/I_{IL} [μA]	Výstup. proudy $I_{OH}/I_{OL}/I_Z$ [mA]	Logický stav / Elektrický stav	Význam
$SX_0 \div SX_3$	10, 08 06, 05	40/250	—	0 = H 1 = L	vstup sekundární části operačního znaku
$PX_4 \div PX_7$	01, 04, 03, 02		—	0 = H 1 = L	vstup primární části operačního znaku
$AC_0 \div AC$	39, 38, 21, 22,		—	0 = L 1 = H	vstupy řízení příští adresy
$FC_0 \div FC_3$	15, 16, 13, 12		—	0 = L 1 = H	vstupy řízení návěstí
FI	17		—	0 = H 1 = L	vstup návěstí
LD	36		—	0 = L 1 = H	vstup řízení přímého plnění MA ze sběrnice PX, SX
EN	25	80/500	—	0 = L 1 = H	vybavení výstupů MA, PR, FO pro EN = 1
ERA	35	40/250	—	0 = L 1 = H	vybavení výstupů adresy řádky $MA_4 \div MA$ pro ERA = 1
$MA_0 \div MA_3$	29, 26, 27, 28	—	třístavový —1/10/0,1	0 = L 1 = H EN = L → Z	výstup adresy sloupce 3. stav pro EN = L
$MA_4 \div M_8$	30, 31, 32, 33, 34	—	třístavový —1/10/0,1	0 = L 1 = Z_4 $EN \vee ERA$ = ⇒ Z	Adresy sloupce, 3. stav pro $EN \vee ERA$ = L.
$PR_0 \div PR_3$	11, 09, 07	—	otevřený kolektor —/10/0,1	0 = L 1 = Z	výstup střádače PR. Asynchronně vybaven při EN = H a funkci JCE
FO	14	—	třístavový —1/10/0,1	0 = H 1 = L EN = L → Z	výstup návěští 3. stav pro EN = L
ISE	18	—	dvoustavový —1/10/—	0 = H 1 = H	hlášení vzorkuj přerušení, asynchr. ISE = H jen při funkci JZR (15)
CLK	19	120/750	—	0 = L 1 = H	vstup hodin

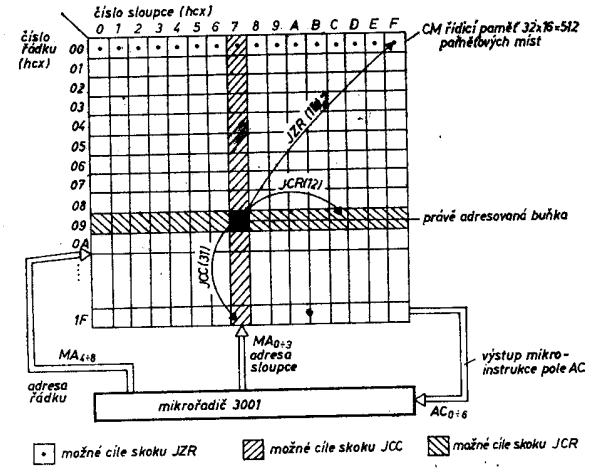


Obr. 4. Funkční schéma obvodu MH3001

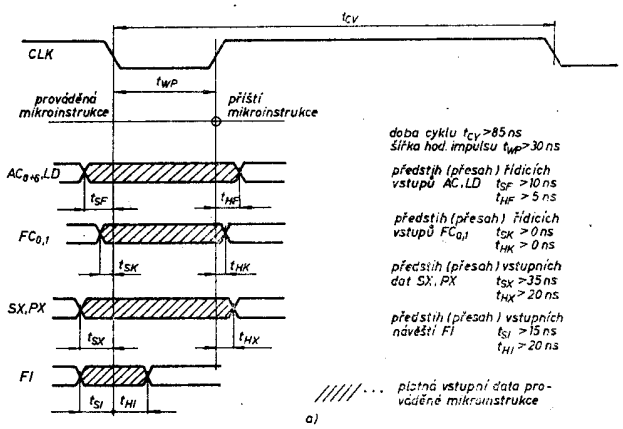
Řez operační jednotky 3002

Obvod realizuje aritmetické, logické a registrové mikrooperace s dvoubitovými slovy dat. Mikrooperace je zvolena stavem sběrnice *F* a ovlivněna stavem vstupní sběrnice *K* a vstupů *LI*, *CI*. Operandy jsou vybírány vstupními multiplexery aritmetické jednotky obvodu z pole vnitřních registrů nebo ze vstupních sběrnic — viz *obr. 7*. Jako cíl umístění výsledku operace je volen některý z registrů pole, nebo výstupní adresový registr *MAR* nebo střádač *AC*. Možnosti volby typu operace, zdrojových a cílových registrů (sběrnic) ukazuje *tab. 7*. *Tab. 6* pak definuje mikrooperace ve vztahu ke stavu řídicí sběrnice *F* a sběrnice *K*. *Tab. 5* popisuje význam jednotlivých vývodů pouzdra 3002.

Funkce obvodu počíná po týlové hraně hodin *CLK*. Po ustálení vstupů *F*, *K*, *M*, *I*, *LI*, *CI* jsou zvoleny zdroje operandů a typ operace. Výstupní přenosy *CO*, *RO* se ustálí s předstihem před čelem hodinového impulsu, s nímž se přepisuje výsledek operace do cílového registru. Význam aritmetického přenosu *CO* se ve stavu *L* na vstupu *CLK* změní. Přesah *CO* přes čelní hranu hodin však stačí pro zápis do střádače *F* při spolu-



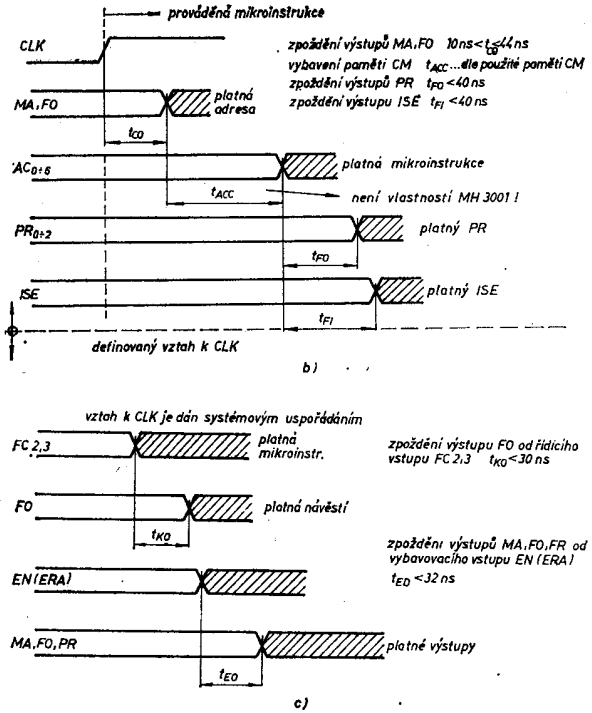
Obr. 5. Příklad možností skoků na příští adresu (JZR, JCC, JCR)



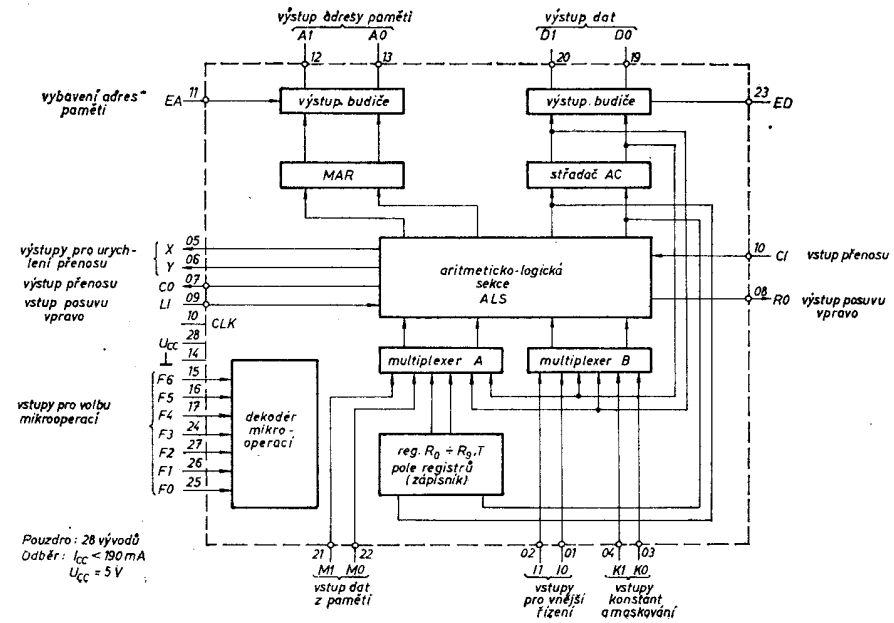
Obr. 6a. Předstihy a přesahy vstupních signálů vůči hodinovému impulsu MH3001

Tab. 5. Význam vývodů MH3002 (obsah jednotlivých sloupců jako v tab. 4)

$I_{0,1}$	01, 02	60/1500	—	0 = H 1 = L	vstup dat pro vnější zařízení
$K_{0,1}$	03, 04	40/ 250	—	0 = H 1 = L	vstup konstant a maskování
$M_{0,1}$	22, 21	60/1500	—	0 = H 1 = L	vstup dat z paměti
$F_0 \div F_4$	25, 26, 27, 24, 17, 16, 15	40/ 250	—	0 = L 1 = H	vstupy řízení mikrooperací
CI, *LI*	10, 09	CI 180/4000 LI 60/1500	—	0 = H 1 = L	vstup aritmetického přenosu vstup pro posuv vpravo
EA, *ED*	11, 23	40/ 250	—	0 = H 1 = L	vybavení adresy a dat na sběrnice *A*, *D* při *EA* = *ED* = 1 = L
D_0, D_1	19, 20	—	třístavové 1/10/0,1	0 = H 1 = L *ED* = H → *Z*	výstup dat 3. stav při *ED* = 0 = H
A_0, A_1	13, 12	—		*EA* = H → *Z*	výstup adres 3. stav při *EA* = 0 = H
CO, *RO*	0,7 08	—		0 = H 1 = L vzáj. vybav. *Z*	výstup aritmet. přenosu a posuvu vpravo. Jsou vzájemně vybavovány. *RO* vybaven jen při *SRA*
X, *Y*	05, 06	—	dvoustavové 1/10	0 = L 1 = H	výstupy pro zprac. rychlého přenosu obvodem 3003
CLK	18	40/ 250	—	0 = L 1 = H	vstup hodin



Obr. 6b, c. Zpoždění výstupů obvodu MH3001



Obr. 7. Funkční schéma obvodu MH3002

Tab. 6. Definice, kódování a symbolické označení mikrooperací MH 3002

SKUPINA REGISTRŮ (určena stavem F0,F1,F2,F3)					I			II			III		
operandy A,B cíl výsledku C					A: Rn	B: AC∧K nebo K	C: Rn AC nebo Rn	A: M	B: AC∧K nebo K	C: AT	A: AT	B: I∧K nebo K	C: AT
typ funkce je určen stavem F4,F5,F6			skupina funkcí	popis funkcí skupiny									
F6	F5	F4											
0	0	0	0	aritm. součet nebo posun vlevo nebo posun vpravo nebo přesun	$R_n+(AC\wedge K)+CI\to R_n, AC$; $a\to CO$	K=0: $R_n+CI\to R_n, AC$ ILR; K=1: $R_n+CI+AC\to R_n, AC$ ALR		$M+(AC\wedge K)+CI\to AT$; $a\to CO$	K=0: $M+CI\to AT$ ACM; K=1: $M+AC+CI\to AT$ AMA		$AT_L\wedge(\overline{I_L\wedge K_L})\to R0$; $[AT_L\wedge(I_L\wedge K_L)]\vee[AT_H\vee(I_H\wedge K_H)]\to AT_L$; $LI\vee[(I_H\wedge K_H)\wedge AT_H]\to AT_H$ 2)	K=0: posuv vpravo SRA	
0	0	1	1	výpočet adresy a plnění MAR nebo doplněk AT	$K\vee R_n\to MAR$; $R_n+K+CI\to R_n$; $a\to CO$ 3)	K=0: $R_n\to MAR$; $R_n+CI\to R_n$ LMI; K=1: $\wedge\to MAR$; $R_n+1+CI\to R_n$ DSM		$K\vee M\to MAR$; $M+K+CI\to AT$; $a\to CO$ 3)	K=0: $M\to MAR$; $M+CI\to AT$ LMM; K=1: $1\to MAR$; $M+1+CI\to AT$ LDM		$(\overline{AT}\vee K)+(AT\wedge K)+CI\to AT$; $a\to CO$	K=0: doplněk $\overline{AT}+CI\to AT$ CIA; K=1: dekrement $AT-1+CI\to AT$ DCA 1)	
0	1	0	2	nastavení, nulování, přesun, decrement	$(AC\wedge K)-1+CI\to R_n$; $a\to CO$ 1)	K=0: $CI-1\to R_n$ CSR; K=1: $AC-1+CL\to R_n$ SDR		$(AC\wedge K)-1+CI\to AT$; $a\to CO$	K=0: $CI-1\to AT$ CSA; K=1: $AC-1+CI\to AT$ SDA		$(I\wedge K)-1+CI\to AT$; $a\to CO$ 1)	K=0: ≡CSA; K=1: $I-1+CI\to AT$ LDI	
0	1	1	3	aritmetický součet, inkrement	$R_n+(AC\wedge K)+CI\to R_n$; $a\to CO$	K=0: $R_n+CI\to R_n$ INR; K=1: $R_n+AC+CI\to R_n$ ADR		$M+(AC\wedge K)+CI\to AT$; $a\to CO$	K=0: ≡ACM; K=1: ≡AMA		$AT+(I\wedge K)+CI\to AT$; $a\to CO$	K=0: $AT+CI\to AT$ INA; K=1: $AT+I+CI\to AT$ AIA	
1	0	0	4	logický součin, test na nulový výsledek	$R_n\wedge(AC\wedge K)\to R_n$; $I\to CO$	K=0: $0\to R_n$, $CL\to CO$ CLR; K=1: $R_n\wedge AC\to R_n$ test ANR		$M\wedge(AC\wedge K)\to AT$; $I\to CO$	K=0: $0\to AT$, $CL\to CO$ CLA; K=1: $M\wedge AC\to AT$ test ANM 4)		$AT\wedge(I\wedge K)\to AT$; $I\to CO$	K=0: ≡CLA; K=1: $I\wedge AT\to AT$ test ANI	
1	0	1	5	maskování, test na nulový výsledek	$K\wedge R_n\to R_n$; $I\to CO$	K=0: ≡CLR; $R_n\to R_n$, test TZR 4)		$K\wedge M\to AT$; $I\to CO$	K=0: ≡CLA; $M\to AT$, test TM 4)		$K\wedge AT\to AT$; $I\to CO$	K=0: ≡CLA; $AT\to AT$, test TZA 4)	
1	1	0	6	logický součet, testnulý operand B	$R_n\vee(AC\wedge K)\to R_n$; $CIW(AC\wedge K)\to CO$	K=0: $R_n\to R_n$, $CI\to CO$ NOP; K=1: $R_n\vee AC\to R_n$ test ORR 4)		$M\vee(AC\wedge K)\to AT$; $CIW(AC\wedge K)\to CO$	K=0: $M\to AT$, $CI\to CO$ LMF; K=1: $M\vee AC\to AT$ test DRM 4)		$AT\vee(I\wedge K)\to AT$; $CIW(I\vee K)\to CO$	K=0: ≡NOP; K=1: $AT\vee I\to AT$ test ORI 4)	
1	1	1	7	negace, ekvivalence, test nuly součinu A.B	$R_n\equiv(AC\wedge K)\to R_n$; $CIW(R_n\wedge AC\wedge K)\to CO$	K=0: $R_n\to R_n$, $CI\to CO$ CMR; K=1: $R_n\equiv AC\to R_n$ test $R_n\wedge AC$ XNR 4)		$M\equiv(AC\wedge K)\to AT$; $CIW(M\wedge AC\wedge K)\to CO$	K=0: $M\to AT$, $CI\to CO$ LCM; K=1: $M\equiv AC\to AT$ test $M\wedge AC$ XNM 4)		$AT\equiv(I\wedge K)\to AT$; $CIW(AT\wedge I\wedge K)\to CO$	K=0: $\overline{AT}\to AT$, $CI\to CO$ CMA; K=1: $AT\equiv I\to AT$ test $AC\wedge I$ XNI 4)	

symbolický název mikrooperace

Ostatní symboly dle tab. 7.

K = 0 .... znamená samé nuly na sběrnici K

K = 1 .... znamená samé jedničky na sběrnici K

1.) Symbol -1 znamená přičtení 11 (dvojk. doplněk)

2.) Při posuvu vpravo CO ve 3. stavu, RO vybaven. Při ostatních mikroop. RO ve 3. stavu, CO vybaven.

3.) Při plnění MAR probíhá operace v registru

4.) Test: je-li výsledek operace nula 1→CO

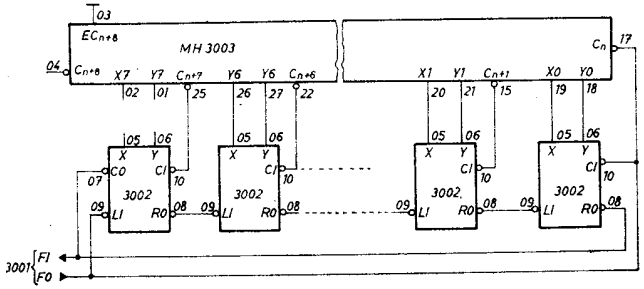
práci s obvodem 3001. Třístavové provedení výstupu přenosu *CO* a výstupu přenosu vpravo *RO* spolu se vzájemným blokováním vyvolaným volbou funkce pro posuv vpravo umožňuje jejich pří-

Tab. 7. Typy mikrooperací obvodu MH3002

Mikrooperace jsou typu $AopB \rightarrow C$ nebo $opA \rightarrow C$; $a \rightarrow CO$ nebo $l \rightarrow CO$ nebo přes vpravo $\rightarrow RO$		
Zdroje operandu A	Rn; M; AT	jsou přípustné jen některé kombinace naznačených zdrojů operandů a cílů výsledků viz *tab. 56*
Zdroje operandu B	$AC \wedge K$; $I \wedge K$; K	
Umístění (cíl) výsledku C	Rn; AC; Rn; AT	

Typ mikrooperace	Symbol	Popis	
Přesun dat	*MOV*	$A \rightarrow C$, $a \rightarrow CO$	
Přesun a výpočet adresy	*MAR*	$KVA \rightarrow MAR$; $A + K + CI \rightarrow C$; $a \rightarrow CO$	
Aritm. součet	*ADD*	$A + B + CI \rightarrow C$; $a \rightarrow CO$	
Inkrement	*INC*	$A + 1 \rightarrow C$;	$a \rightarrow CO$
Dekrement	*DEC*	$A - 1 \rightarrow C$;	$a \rightarrow CO$
Logický součin	*AND*	$A \wedge B \rightarrow C$;	$l \rightarrow CO$
Logický součet	*IOR*	$A \vee B \rightarrow C$;	$l \rightarrow CO$
Ekvivalence (exclusive NOR)	*XNR*	$A \equiv B \rightarrow C$;	$CIW\ (A \vee B) \rightarrow CO$
Negace (z doplněk)	*NOT (NEG)*	$\overline{A} \rightarrow C$;	$(\overline{A} + 1) \rightarrow C$; $a \rightarrow CO$)
Maskování (test)	*MSK (TST)*	$K \wedge A \rightarrow C$	(je-li $C = 0\ 1 \rightarrow CO$)
Nulování	*CLR*	$0 \rightarrow C$	
Nastavení	*SET*	$1 \rightarrow C$	
Posuv vpravo o 1 bit	*SHR*	$LI \rightarrow A_H$; A vpravo posunutý $\rightarrow C$; $A_L \rightarrow R$	

I, K, M	vstupní sběrnice nebo data na nich
Rn	registr zápisníku (nebo jeho obsah). Zahrnuje i registry *AC, T*
AC	střadač (nebo jeho obsah)
AT	registr *AC* nebo *T* (nebo obsah zvoleného z nich)
MAR	registr adresy (nebo jeho obsah)
CI, LI	data na přenosových vstupech
CO, RO	výstupy přenosu
a	aritmetický přenos
$l = CIwC$	logický přenos
index *L (H)*	označení bitu nejnižšího (nejvyššího) řádu zprac. slova dat
+ (—)	sčítání (odčítání) v aritmetice dvojkového doplňku
$\vee$ (1)	logický součet (součin)
$\equiv$	ekvivalence
$\overline{X}$	negace X
$A \rightarrow C$	přesun obsahu A do C
$CIwC = CI \vee (C_L \vee C_H)$	logický součet CI s logickým součtem bitů výsledku C



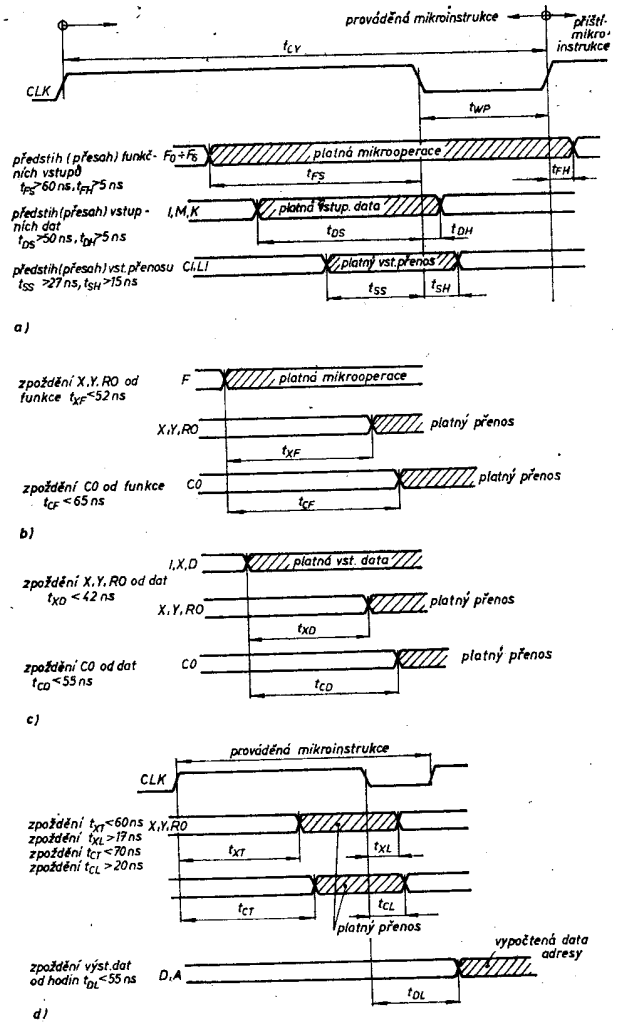
Obr. 8. Možný způsob spojení obvodů 3002 a 3003 v 16-bitovém procesoru; rychlý přenos přes 7 řezů, postupní přes poslední řez

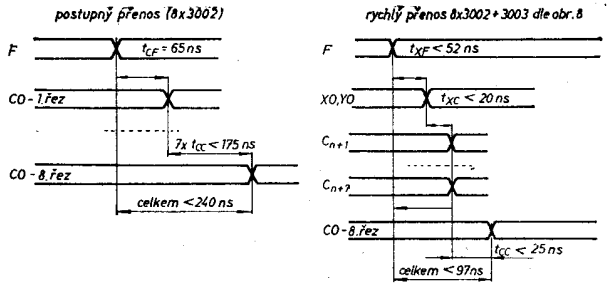
Obr. 9a. Předstihy a přesahy vstupního signálu 3002 vůči hodinám

Obr. 9b. Zpoždění přenosů od funkčních vstupů 3002

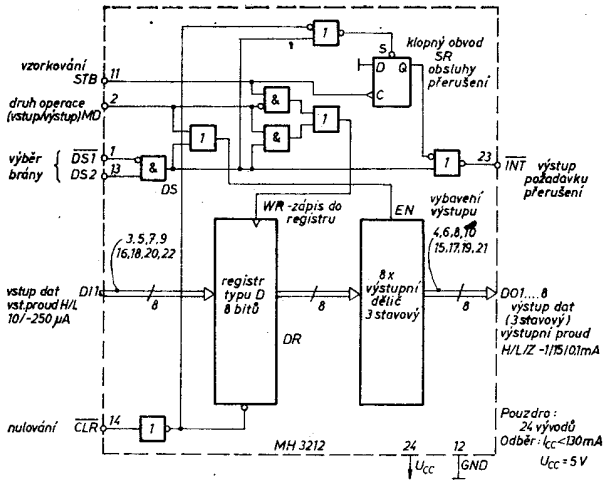
Obr. 9c. Zpoždění přenosů od vstupů dat 3002

Obr. 9d. Zpoždění přenosů a výstupních dat od hodin 3002

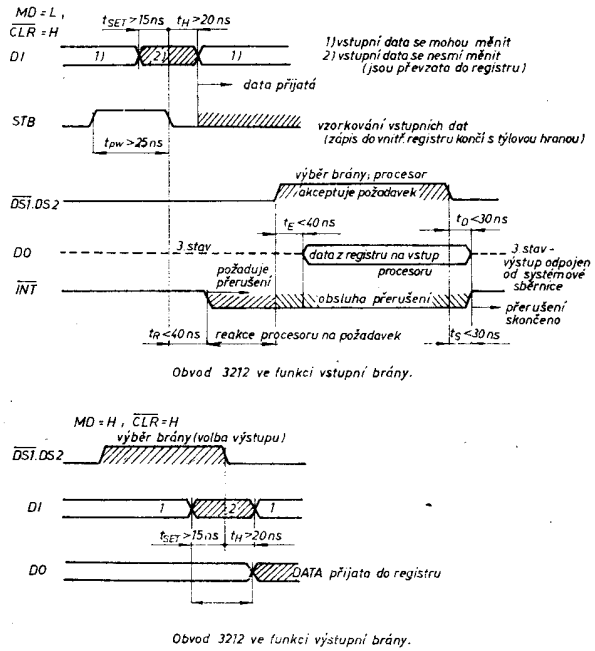




Obr. 9e. Šíření přenosu v 16-bitovém procesoru z obvodů 3002, 3003



Obr. 10a. Funkční schéma 8-bitového univerzálního vstup/výstupního registru MH3212

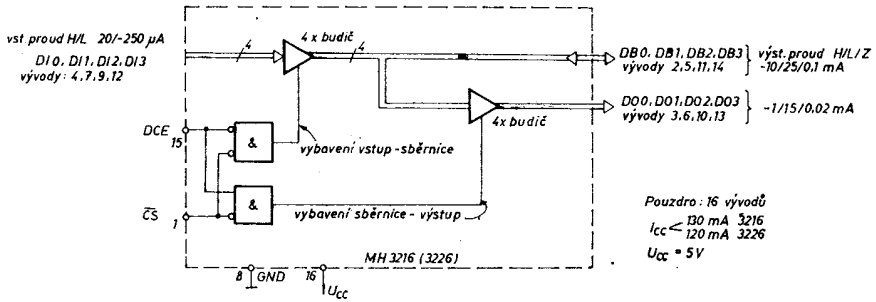


Obr. 10b. Časové charakteristiky obvodu MH3212 ve funkci vstupu a výstupu systému

Tab. 8. Funkce obvodu MH 3212

Použití	Vstupy: Vzorkování *STB*	Vstupy: Výběr *DS* = $\overline{DS1}.DS2$	Vstupy: Volba operace *MD*	Výstup dat *DO*	Funkce
Systémový vstup	L	L	L	Z (3. stav)	ignoruje *DI*, výstup odpojen
	H	L	L	Z (3. stav)	zapisuje *DI* → *DR*, výstup odpojen
	L	L	L	*DR*	ignoruje *DI*, pamatovaná data na výstup *DR* → *DO*
	H	H	L	DI	zapisuje data a připojuje výstup *DI* → *DR* → *DO*
Systémový vstup	L	L	H	*DR*	ignoruje *DI*, pamatovaná data na výstup *DR* → *DO*
	H	L	H	*DR*	
	L	H	H	*DI*	zapisuje data a připojuje výstup *DI* → *DR* → *DO*
	H	H	H	*DI*	

Tabulka platí pro $\overline{CLR}$ = H. $\overline{CLR}$ = L nuluje registr a nastavuje klopný obvod *SR*.
Klopný obvod obsluhy přerušení *SR* — nastaven pro *DS* = H nebo $\overline{CLR}$ = L
— nulován změnou H — L na vstupu *STB*.
Výstup přerušení $\overline{INT}$ = *L* — 3212 požaduje přerušení, je-li *DS* = H nebo klopný obvod *SR* vynulován.

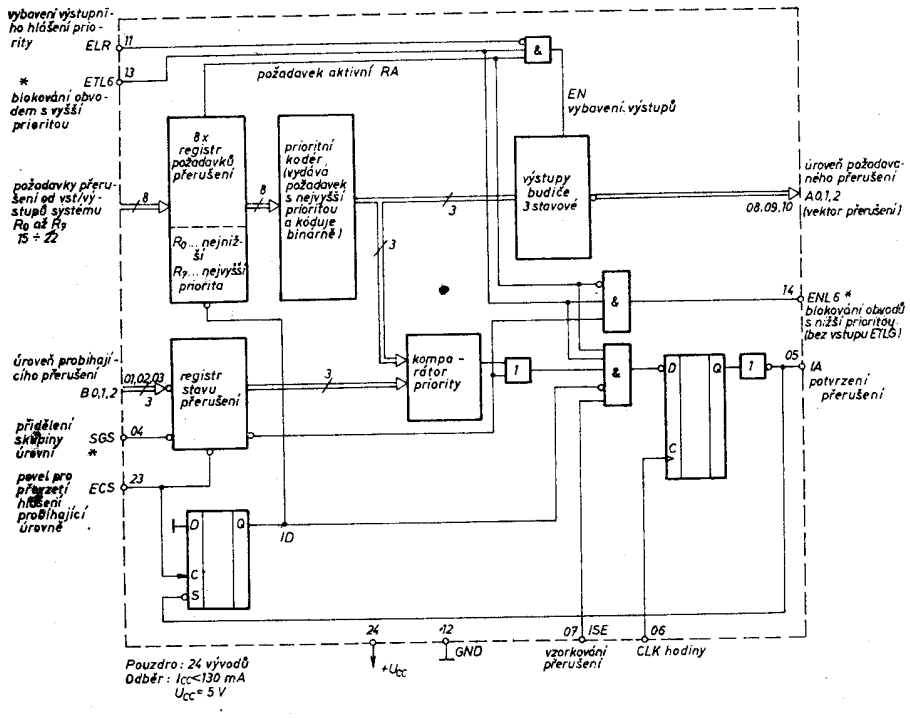


Obr. 11. Funkční schéma obousměrného čtyřnásobného budiče sběrnice MH3216 (neinvertující) nebo MH3226 (invertující)

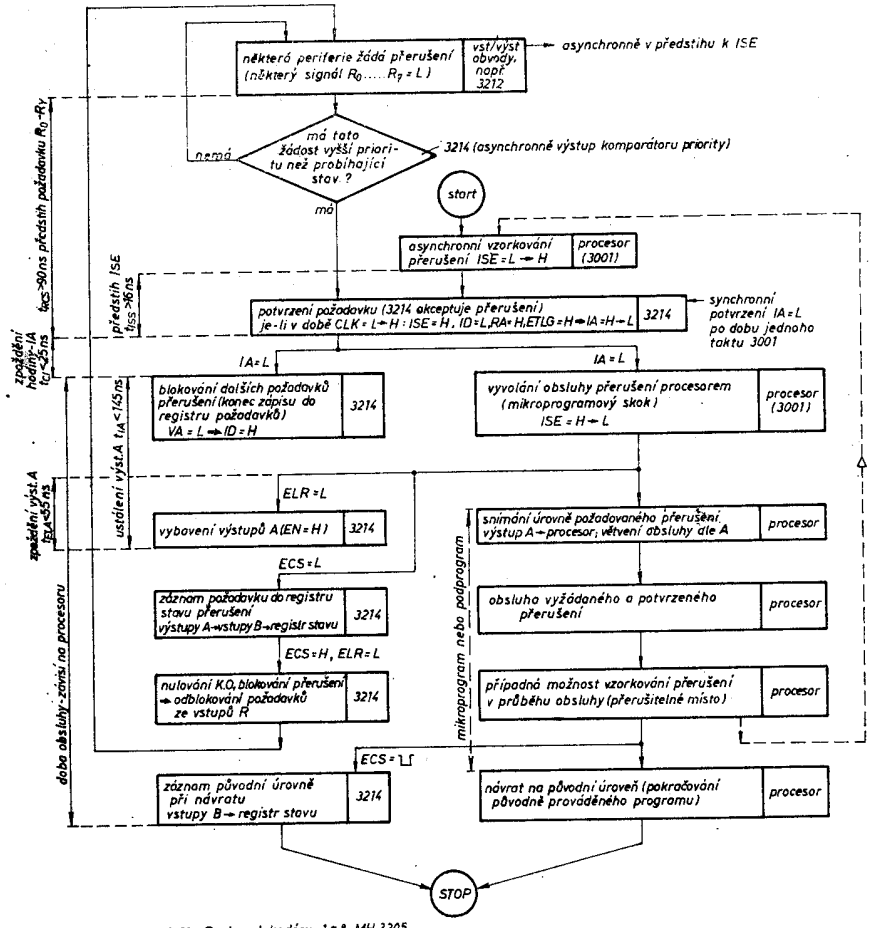
Tab. 9. Funkce obvodu MH3216 (3226).

Funkce	Vstupy: *DCE*	Vstupy: $\overline{CS}$	Výstupy 3216: *DB*	Výstupy 3216: *DO*	Výstupy 3226: *DB*	Výstupy 3226: *DO*
Přenos vstup → sběrnice	L	L	DI	Z	$\overline{DI}$	Z
Přenos sběrnice I výstup	H	L	Z	DB	Z	$\overline{DB}$
Žádný přenos	L	H	Z	Z	Z	Z
Žádný přenos	H	H	Z	Z	Z	Z

Z — výstup ve stavu vysoké impedance

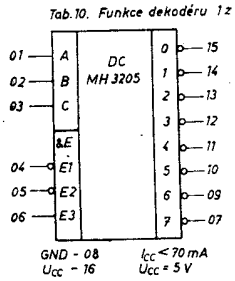


Obr. 12a. Funkční schéma obvodu MH3214; * signál pro kaskádní řazení více obvodů 3214 v systémech s více než osmi úrovněmi přerušení. Pro systém s 8 nebo méně úrovněmi je SGS = L a ETLG = H



Obr. 12b. Zjednodušený časový a tokový diagram funkcí obvodu MH3214 (bez funkcí kaskády)

Tab. 10. Funkce dekodéru 1 z 8 MH 3205



adresa			výstup								vybavení		
C	B	A	0	1	2	3	4	5	6	7	E1	E2	E3
L	L	L	L	H	H	H	H	H	H	H	L	L	H
L	L	H	H	L	H	H	H	H	H	H	vybaven pro E1 E2 $\overline{E3}$ = L		
L	H	L	H	H	L	H	H	H	H	H			
L	H	H	H	H	H	L	H	H	H	H			
H	L	L	H	H	H	H	L	H	H	H			
H	L	H	H	H	H	H	H	L	H	H			
H	H	L	H	H	H	H	H	H	L	H			
H	H	H	H	H	H	H	H	H	H	L	L	L	H
X	X	X	H	H	H	H	H	H	H	H	E1 E2 $\overline{E3}$ = H		

X .... stav H nebo L

mé spojení. Shodně jsou ošetřeny vstupy přenosu *CI* a *LI*.

Pro spojení s obvodem rychlého přenosu realizuje obvod 3002 funkce na výstupech *X*, *Y*. Způsob spojení obvodů 3002 a 3003 v 16 bitovém procesoru je naznačen na *obr. 8*. Na *obr. 9* jsou nakresleny časové vztahy vstupů a výstupů obvodu 3002 a naznačeno možné ovlivnění zpoždění aritmetického přenosu připojením obvodu 3003.

Doplňkové obvody řady 3000

Tyto obvody nemají charakter procesorových prvků. Uvedeme zde proto jen jejich základní charakteristiky v *tabulkách 8, 9, 10* a funkční schemata na *obr. 10, 11, 12*.

Závěr

Uvedená stručná charakteristika prvků řady TESLA MH3000, funkčně ekvivalentních obvodům Intel 3000, nepostihuje všechny jejich elektrické parametry a řadu důležitých funkčních vlastností nechává stranou. Ke zdárnému zvládnutí aplikací uvedených obvodů je pochopitelně potřeba důkladná znalost prvků, metodiky aplikace a praktická zkušenost s jejich použitím. Navíc pak ve sféře mikroprogramovatelných procesorů musí návrhář zvládnout programování procesoru alespoň na úrovni mikroprogramů. Struktura procesoru a použitých mikroprogramů spolu úzce souvisí a řešení problému nelze rozdělit na návrh mikroprogramů a návrh technického vybavení procesoru.

LITERATURA

[1] Kolektiv: Sborník semináře „Mikroprocesorový bipolární systém Intel 3000". DT ČVTS Ostrava, PČVTS TESLA Rožnov, listopad 1977,

[2] Kolektiv: Mikroprocesorový bipolární systém Intel 3000. DT ČVTS Ostrava, březen 1978

POČÍTAČ PRO DOMÁCNOST

Japonská firma Hitachi vystavovala v květnu 1978 v Tokiu počítač pro domácnost, který chce v brzké době uvést na trh. Počítač velikosti myčky na nádobí má vestavěný mikroprocesor, paměti a klávesnici, vše ergonomicky přizpůsobeno potřebám ženy v domácnosti, která si tak může bez potřeby zvláštních předchozích znalostí organizovat práci po celý den.

Počítač řídí např. celý klimatizační systém domu (topení, větrání a chlazení), kontroluje vydání pro domácnost a sestavuje jídelníčky pro 50 dní (s přídavným magnetofonem až pro 200 dní). Navíc je plánováno: zabezpečení oken, ochrana proti požáru a vloupání, jakož i krmení domácích zvířat.

-hák-

Funkschau 1978 č. 22, str. 1101

Aplikace mikroprocesorových prvků řady 3000

ING. TOMÁŠ GUCKÝ

Článek uvádí základní možnosti aplikací a metod návrhu s obvody řady TESLA MH3000, volně navazuje na popis jednotlivých prvků řady uvedený v [1]. Zabývá se postupem návrhu zapojení procesoru, jeho časováním a návrhem mikroprogramů.

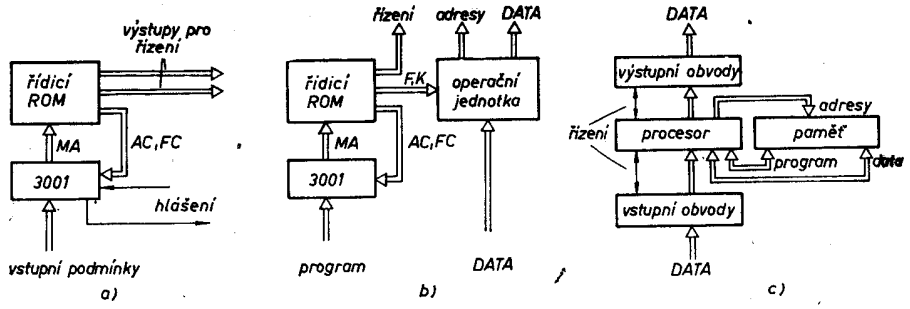
Obor použití obvodů řady 3000

Základní aplikací řady 3000 jsou rychlé mikroprogramovatelné řadiče, procesory resp. mikropočítače *(obr. 1)*. Výsledný stroj, který navrhujeme, může podle nároků aplikace obsahovat jednu nebo dvě úrovně programování a řízení. V nejjednodušším případě je možno použít jediný obvod 3001 jako mikroprogramový sekvenční automat, který generuje posloupnosti skupin logických signálů, čtených z řídicí paměti v závislosti na posloupnosti vstupních signálů a vnitřních stavů. Takové uspořádání je použitelné jako náhrada automatu sestaveného z klasických logických obvodů SSI, MSI. Jeho výhodou je i snadná realizace změn funkcí pouhou výměnou obsahu řídicí paměti. Pokud má navrhovaný stroj provádět operace s vícebitovými slovy, neobejde se bez operační jednotky. Procesor pak bude obsahovat pro $2N$-bitové slovo nejméně N obvodů 3002, řídicí paměť a mikrořadič 3001. Jeho výhodou je možnost volby délky slova zpracovávaných dat a možnost volby souboru operací podle nároku aplikace. Druh aplikace procesoru ovlivňuje také výslednou konfiguraci spolupracujících jednotek. Spolu s pamětí pro program a data, vstupními a výstupními obvody můžeme vytvořit mikropočítač „na míru" dané aplikaci. Je nezbytné, aby vedle mikroprogramů uživatel vytvořil také aplikační programy a vedle mikroprogramového řízení procesoru navrhl řízení jednotek mikropočítače a jeho vnějších zařízení na systémové úrovni. Tato dvojí úroveň návrhu sice komplikuje práci, ale na druhé straně optimalizuje výsledek. Z obvodů řady 3000 je tak možno sestavovat procesory a číslicová zařízení pro sběr dat, řízení periferních zařízení počítačů, číslicovou regulaci, řízení procesů. Je možno sestavit procesor napodobující funkčně procesor jiný apod.

Aplikační postupy

Při vývoji aplikace s obvody série 3000 uvážíme nejprve požadavky na typ vstupních a výstupních zařízení, posoudíme objem zpracovávaných dat a požadovaný způsob zpracování. Podle získaných poznatků volíme strukturu navrhovaného mikropočítače a specifikujeme požadavky na:

— rozsah a uspořádání sběrnic,
— rozsah paměti programu,
— rozsah paměti dat,
— organizaci vstupů a výstupů,
— rozsah a obsah souboru instrukcí,
— rychlost práce procesoru.

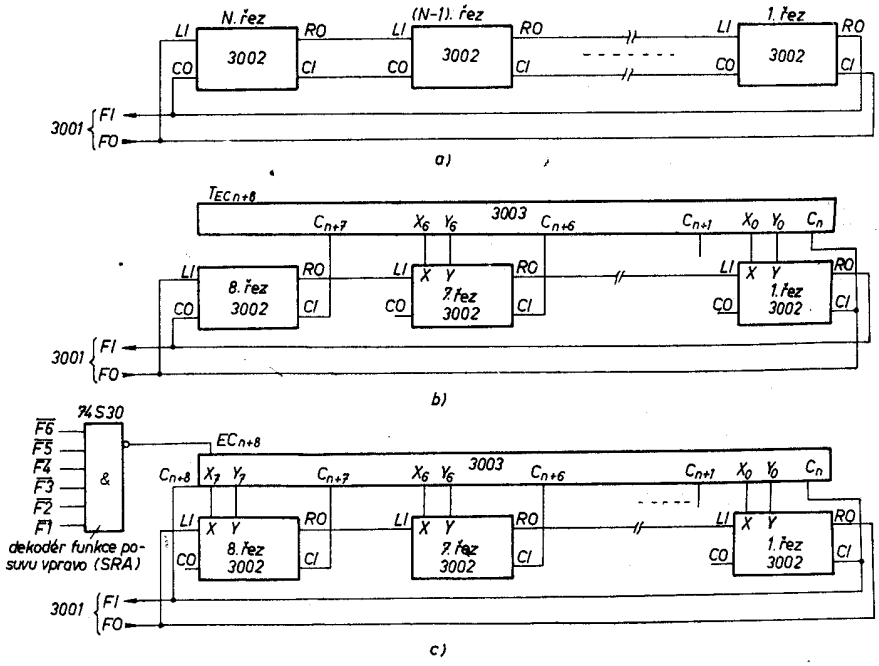


Obr. 1. Základní aplikace řady 3000; a) řídicí jednotka, b) mikroprocesor, c) mikropočítač

Tuto úvodní rozvahu musíme provést pečlivě a s přihlédnutím ke specifickým vlastnostem prvků řady 3000. Týká se to hlavně operačních rychlostí prvků, uspořádání sběrnic a souboru požadovaných operací. Je třeba podrobně rozpracovat algoritmus řešené úlohy, uvážit potřebný soubor instrukcí, napsat příklady aplikačních programů a soubor pokud možno podle nich optimalizovat. Vyhneme se tak omylům ve specifikaci, které mohou později způsobit rozsáhlé změny zapojení prvků a změny mikroprogramů. Návrh celého mikropočítače dále pokračuje detailním návrhem technického vybavení a jeho aplikačních programů, realizací vzorku a jeho ověřením. Klíčová je přitom problematika návrhu procesoru, na kterou se dále soustředíme.

Specifikace stroje na makroúrovni obsahuje již specifikaci jeho procesoru, kterou musíme dále rozpracovat. Především musíme v této etapě návrhu posoudit ekonomickou vhodnost řešení s obvody řady 3000. O použitelnosti pomocných obvodů — 3212, 3214, 3216, 3226, 3205 — není sporu. Jsou koncipovány obecně pro využití v paměťových systémech, obvodech vstup/výstup (dále v/v) a pomocných funkcích. Použití vlastních procesních prvků 3001, 3002, 3003 bude výhodné tam, kde jsou požadovány operace, realizovatelné krátkými mikroprogramy. Pak je plně využitelná jejich rychlost spolu s možností optimalizace procesoru podle účelu použití.

Vstupní specifikace procesoru musí obsahovat:
— formát zpracovávaných dat, z délky slova odvodíme počet použitých řezů operační jednotky 3002;
— formát instrukcí, podle něj stanovíme způsob dekódování instrukcí, způsob připojení sběrnic *PX*, *SX* obvodu 3001;
— určení funkčních registrů stroje (nejméně to bude programový čítač, střádač, stavový registr, pracovní registry);



Obr. 2. Varianty zapojení operační jednotky; a) postupný přenos $2N$-bitovou operační jednotkou, b) zrychlený přenos přes 16-bitovou operační jednotku (mimo poslední řez), c) kompletní zrychlený přenos 16-bitovou operační jednotkou. Vstupy *F1*, *F2*, *F3* hradla 74S30 mají být bez negace

— určení rozsahu a způsobů adresace paměti a zařízení v/v;
— definice souboru instrukcí;
— popis spouštěcích operací (iniciace);
— určení způsobů ovládání přerušení a styku s pamětí a obvody v/v, z toho plynoucí organizace řídicích a pomocných obvodů;
— požadavky na operační rychlost.

Další postup návrhu procesoru spočívá v:
— návrhu zapojení,
— návrhu časování,
— návrhu mikroprogramů pro zadaný soubor instrukcí.

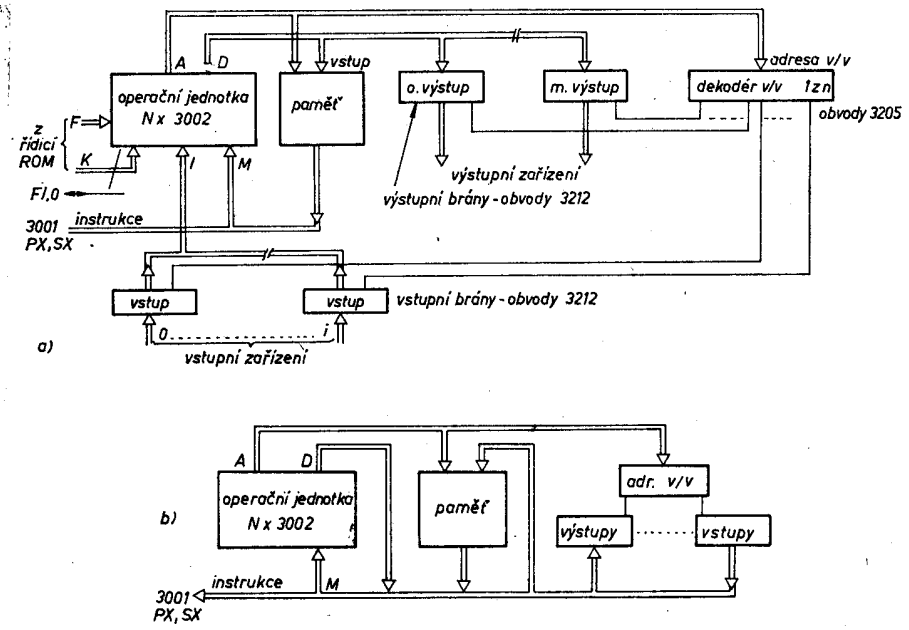
Jednotlivé oblasti návrhu se vzájemně prolínají a ovlivňují. Pojednáme o nich odděleně pouze z důvodu větší přehlednosti výkladu.

Zapojení operační jednotky

Funkce operační jednotky a řezu 3002 byly popsány v [1]. Jsou možná tři základní zapojení s rozdílnou operační rychlostí *(obr. 2)*:
— zapojení s postupným aritmetickým přenosem, zprostředkovaným mezi jednotlivými řezy 3002 signály *CI*, *CO*;
— zapojení se zrychleným přenosem s výjimkou posledního řezu;
— zapojení s úplným zrychleným přenosem, s přídavnou logikou. Přídavný obvod dekóduje mikrooperační kód pro posuv vpravo, při němž musí být blokován výstup přenosu operační jednotky, spojený současně s výstupem *RO* na návěstní vstup obvodu 3001.

Další možnosti variací zapojení operační jednotky poskytují vstupní sběrnice *K*, *I* *(obr. 3)*. V nejjednodušším případě je stav sběrnice *K* definován jedním bitem mikroinstrukce ($K = 0$, $K = 1$). Tímto způsobem je modifikován operační kód 3002 a zvětší se počet využitelných mikrooperací. Úplné využití sběrnice *K* dovoluje vkládat z mikroprogramu libovolnou konstantu, zamaskovat libovolné bity operandu a pracovat se skupinami bitů nebo s jednotlivými bity. To však vyžaduje samostatné určení každého bitu *K* v mikroinstrukci a zbytečně zvětšuje rozsah řídicí paměti. Účelné je definovat v poli *K* mikroinstrukce jen nezbytné konstanty a masky a ty vybírat dekodérem nebo vhodným připojením bitů pole *K* mikroinstrukce ke vstupům *K* jednotlivých řezů. Sběrnice *I* je určena pro vstup dat ze vstupních zařízení. Pokud k tomuto účelu použijeme společnou sběrnici s pamětí *(M)*, zůstane sběrnice *I* volná. Je možno ji využít pro připojení vnější logiky nebo zápisníkové paměti a rozšířit schopnosti operační jednotky.

Uspořádání styku s pamětí a obvody v/v prostřednictvím sběrnic *A*, *D*, *M*, *I* operační jednotky je také na vůli uživatele. Na *obr. 4* jsou naznačeny nejběžnější způsoby organizace sběrnic.



Obr. 4. Příklady způsobu organizace sběrnic; a) připojení operační jednotky systémem jednosměrných sběrnic, b) spojení jednotek na obousměrné sběrnici dat

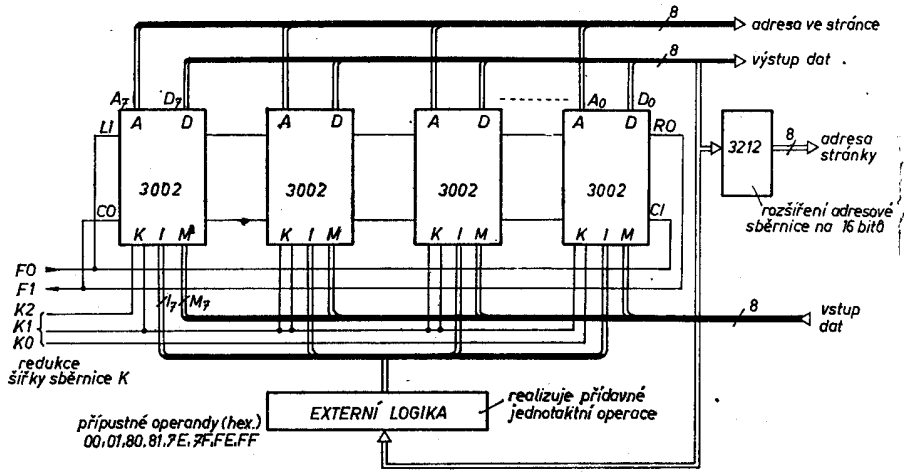
Zapojení řadiče

Základní funkce řadiče i jeho řídícího prvku 3001 byly popsány v [1]. Řízení procesoru pomocí mikroprogramového řadiče s obvodem 3001 je možno koncipovat dvěma způsoby:
— řízení s postupným výběrem a provedením mikroinstrukce,
— řízení s překrytým provedením mikroinstrukce a výběrem příští mikroinstrukce (pipelinning).

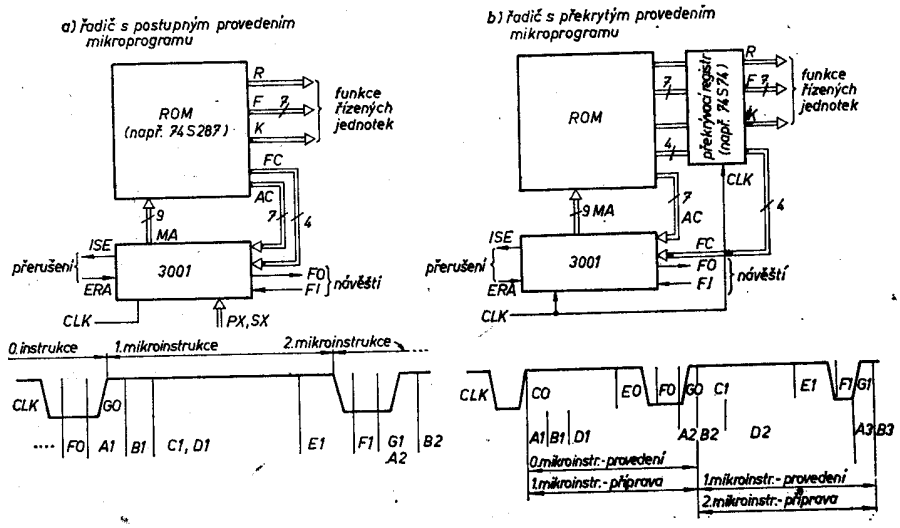
Práce řadiče sestává z několika činností, jejichž časový vztah je zvoleným zapojením ovlivněn. Tyto činnosti jsou:
A — adresování buňky řídicí paměti;
B — výběr mikroinstrukce z řídicí paměti;
C — provedení mikrooperací podle vybrané mikroinstrukce, nastavení výstupního návěstí *FO*;
D — výpočet adresy příští mikroinstrukce;
E — převzetí vstupního návěstí *FI*;
F — korekce adresy příští mikroinstrukce podle převzatého návěstí *FI*;
G — zápis vypočtené adresy do registru *MAR*; zápis do indikátorových registrů *C*, *Z*.

Na *obr. 5* je naznačen princip řadiče s postupnou i překrývanou funkcí a časový vztah uvedených činností v obou variantách zapojení. Překrývání je realizováno zařazením registrů typu D mezi výstup řídicí paměti a řízené obvody procesoru. To umožňuje současné provádění mikroinstrukce a snímání příští mikroinstrukce, důsledkem je zrychlení práce procesoru. Pokud však má v mikroprogramu probíhat větvení podle návěstí *FI* nebo indikátorů *C*, *Z*, musí řadič s překrýváním toto větvení zařadit až do následujících kroků mikroprogramu — viz tabulka v *obr. 5*.

K základním funkcím obvodu, jako je dekódování instrukce, nastavování návěstí, řízení průběhu mikroprogramu, generování signálu vzorkování přerušení, můžeme vnějšími obvody přidat další. Některé možnosti jsou naznačeny na *obr. 6*. Tzv. mapovací paměť ROM na vstupech *PX*, *SX* umožňuje překódování instrukce v případě, kdy je strojový kód zadán, ale řadič vyžaduje kód jiný. Adresu příští mikroinstrukce je možno plnit přímo ze sběrnice *PX*, *SX* napájené z několika zdrojů adresy, přes multiplexer. Je možno připojit zásobníkovou paměť pro adresy návratu z podprogramů v mikroprogramech. Pokud nestačí rozsah 512 paměťových míst řídicí paměti, který je přímo adresovatelný obvodem 3001, je vhodné rozšířit adresový registr *MAR* přidáním vnějšího registru. Jeho obsahem jsou adresovány „stránky" řídicí paměti, výběr

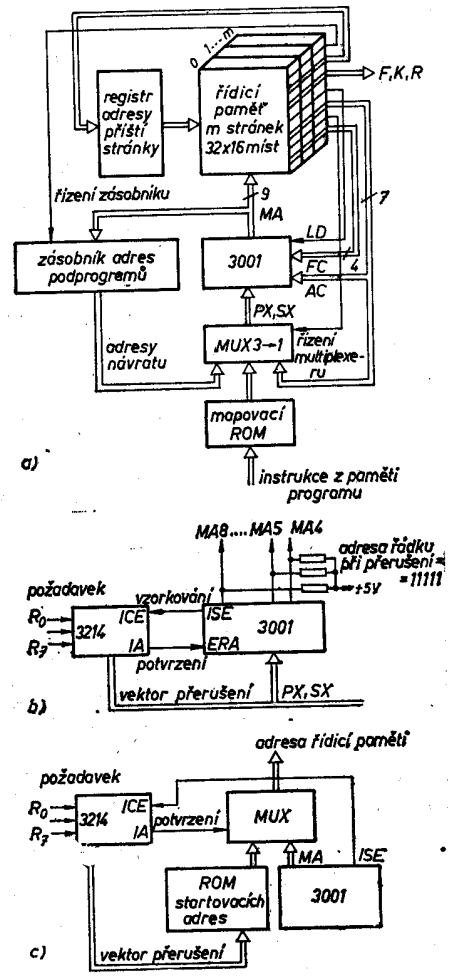


Obr. 3. Rozšíření možností 8-bitové operační jednotky. Přípustné operandy jsou vztaženy ke sběrnici K



průběh mikroprogramu 3001 + 3002	mikrooperace 3001	zpoždění mikrooperace 3001 vůči 3002	
		bez překrytí [taktů]	s překrytím [taktů]
nepodmíněný skok	JCC, JZR, JCE, JCR	0	0
vícecestné větvení	JPX, JPR, JLL, JRL	0	0
větvení podle F	JFL	0	1
větvení podle C, Z	JCF, JZF	1 a více	2 a více

Obr. 5. Princip práce řadiče s překrytým nebo postupným provedením mikroprogramu



Obr. 6. Způsoby realizace přídavných funkcí řadiče; a) rozšíření funkcí 3001, b) přerušení se startem na řádku 31, c) přerušení s libovolnou startovací adresou (vstup ICE má být označen ISE)

stránky příští mikroinstrukce pak bude určen speciálním polem mikroinstrukce. Přerušení práce procesoru provádí obvod 3214 signálem *IA* (akceptuje přerušení jako odezvu na signál *ISE* — vzorkování přerušení). Nastavení startovací adresy mikroprogramu obsluhy přerušení lze provést několika způsoby, naznačenými též na *obr. 6*.

Časování procesoru

Dosažitelné časové parametry procesoru závisí nejen na parametrech použitých součástek, ale hlavně na konfiguraci jejich zapojení. Proto musíme provést detailní rozbor všech kritických cest signálů v procesoru. Zvažujeme:

— potřebné předstihy a přesahy dat v operační jednotce,
— zpoždění přenosů operační jednotky,
— časovou vazbu návěstí operační jednotky a řadiče,
— vztah operační jednotky k paměti a obvodům v/v,
— zpoždění řídicích cest (řadiče a obvodů přerušení),
— synchronizaci signálů rozhraní.

Ukážeme dále na typickém zapojení 16-bitového procesoru *(obr. 7)* metodiku časového rozboru signálových cest. Všechny obvody zapojení jsou synchronizovány systémovými hodinami. Budeme uvažovat varianty zapojení řadiče z *obr. 5* z hlediska postupného nebo překrývaného řízení. Označení dynamických parametrů prvků v souhlase s [1], [2].

Zpoždění aritmetických cest

Cyklus vyvolání a provedení aritmetické funkce probíhá v následujících krocích:

1. krok — čtení mikrooperačního kódu — ustálení sběrnice *F*, *K*. Při použití překrývacího registru je to jen zpoždění hodiny — výstup ($t_{pLR} = 9$ ns pro MH74S74). Bez překrývacího registru je to zpoždění hodiny — adresa *MA* ($t_{CO} = 44$ ns) a doba výběru mikroinstrukce z řídicí paměti ($t_{ACC} = 50$ ns pro MH74188, nebo $t_{ACC} = 65$ ns pro MH74S287); celkem tedy 94 nebo 109 ns.

2. krok — dekódování mikrooperace, výběr operandů, provedení operace. Zde musíme brát v úvahu konfiguraci operační jednotky. S postupným přenosem je výstup *CO* 1. řezu generován se zpožděním $t_{CF} = 65$ ns a šíření přenosu ze vstupů *CI* na výstupy *CO* následujících řezů trvá $(N - 1).t_{CC} = 7.25$ ns = 175 ns. Při použití obvodů zrychleného přenosu je zpoždění funkce — výstup *X*, *Y*, *RO* $t_{XF} = 52$ ns a zpoždění obvodu 3003 $t_{XC} = 20$ ns.

3. krok — pamatování výsledku operace ve střádači a specifikovaném registru s čelem hodinového impulsu. Sem musíme zahrnout předstih přenosu $t_{SS} = 27$ ns pro 3002 a předstih přenosu na vstupu *FI* 3001 $t_{SI} = 15$ ns.

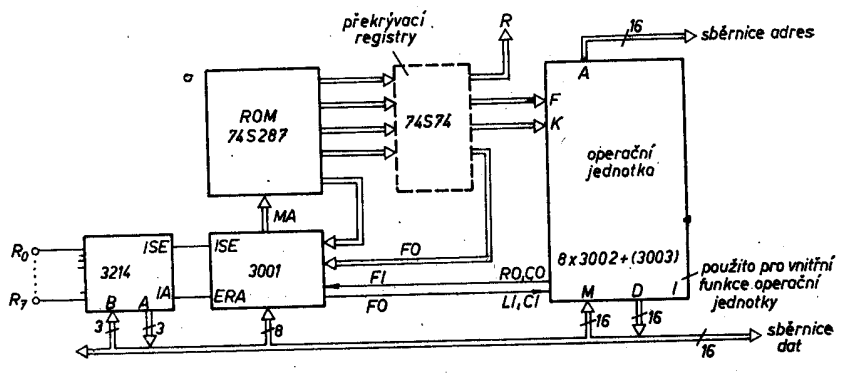
4. krok — operace je skončena a jednotka čeká na týlovou hranu hodin, zahajující následující mikroinstrukci. Musíme zajistit minimální šířku hodinového impulsu $t_{WP} = 33$ ns.

Na základě uvedeného postupu můžeme rozebrat časování cyklu možných konfigurací procesoru. Je evidentní, že nejpomalejší bude varianta bez překrývání a s použitím postupného přenosu (předpokládáme použití pamětí MH 74S287).

Doba cyklu

$$t_{CYKL} = t_{CC} + t_{ACC} + t_{CF} + (N - 1).t_{CC} + t_{SI} + t_{WP} = (44 + 65 + 65 + 175 + 15 + 33)\ \text{ns} = 397\ \text{ns}.$$

Nejrychlejší konfigurace bude opatřena



Obr. 7. Typické uspořádání 16-bitového procesoru

překrývacím registrem a úplným zrychleným přenosem (registr MH74S74, paměť MH74S287)

$$t_{CYKL} = t_{pLR} + t_{XF} + t_{XC} + t_{SS} + t_{WP} = (9 + 52 + 20 + 27 + 33)\ \text{ns} = 141\ \text{ns}.$$

Časování cyklu aritmetické operace může být jiné, pokud jako operandu využíváme dat ze sběrnice *M* nebo *I*. V takovém případě je obvykle styk s pamětí realizován v prodlouženém cyklu hodin (hodiny 3002 ve stavu H) a operační jednotka čeká na platná data ze sběrnice. Po ustálení dat musíme zajistit dostatečný předstih dat před přepisem výsledku operace čelní hranou hodin. Pro variantu s postupným přenosem je to $t_{setup} = 245$ ns, s rychlým přenosem bez posledního řezu $t_{setup} = 102$ ns a s úplným rychlým přenosem $t_{setup} = 89$ ns.

Zpoždění řídicích cest

Funkce 3001 probíhá v následujících krocích:

1. krok — přepis adresy následující mikroinstrukce do registru *MA*. Výstup *MA* se ustálí se zpožděním $t_{CC} = 44$ ns.
2. krok — čtení mikroinstrukce z adresované buňky řídicí paměti. Výstup mikroinstrukce platí se zpožděním ($t_{ACC} = 65$ ns pro MH74S287).
3. krok — generování adresy následující mikroinstrukce na základě stavu sběrnic *AC*, který musí být ustálen s předstihem $t_{SF} = 10$ ns.
4. krok — převzetí návěstí ze vstupu *FI* na sestupnou hranu hodin (předstih $t_{SI} = 15$ ns, přesah $t_{HI} = 20$ ns) a dokončení operace. 3001 čeká na týlovou hranu hodin, minimální šířka hodin $t_{WP} = 33$ ns, dána obvody 3002. Doba cyklu z hlediska řízení bude tedy minimálně

$$t_{CYKL} = t_{CO} + t_{ACC} + t_{SF} + t_{WP} = (44 + 65 + 10 + 33)\ \text{ns} = 152\ \text{ns}.$$

Horší bude situace v cyklu, realizujícím funkci přerušení. Jeho průběh, realizovaný standardním způsobem, je následující:

1. krok — vybavení výstupu *ISE* při skoku *JZR 15*. Zpoždění signálu t_{FI} za povelovými vstupy *AC* je 40 ns. Pro zpracování obvodem 3214 musí mít předstih $t_{ISS} = 16$ ns před náběžnou hranou hodin. Bude tedy

$$t_{CYKL} = t_{CO} + t_{ACC} + t_{FI} + t_{SS} = 165\ \text{ns}.$$

2. krok — pokud obvod 3214 potvrdí přerušení, generuje se zpožděním $t_{CI} = 25$ ns na výstup *IA*, připojený na vstup *ERA* obvodu 3001. Vybavení startovací adresy mikroprogramu obsluhy přerušení je zpožděno o dobu reakce na signál *ERA* $t_{ED} = 32$ ns a při použití pasivního mechanismu dosažení adresy řádku 11111 (pomocí zatěžovacích odporů) ještě o nabíjení (dobu t_R) adresové sběrnice *MA* ze zatěžovacích odporů. (Je-li $R_Z = 1$ kΩ $C_Z = 100$ pF, je úroveň H = 2 V dosažena za asi 100 ns.) Doba cyklu z hlediska nastavení startovací adresy přerušení bude

$$t_{CYKL} = t_{CI} + t_{EO} + t_R + t_{ACC} + t_{SF} + t_{WP} \doteq (25 + 32 + 100 + 65 + 10 + 33)\ \text{ns} \doteq 265\ \text{ns}.$$

Tab. 1. Příklad volby formátu mikroinstrukce

21 — 19	18 — 12	11 — 8	7	6 — 0	Mikroinstrukční slovo
2 — 0	6 — 0	3 — 0		6 — 0	Označení a šířka polí
R (3 bity)	*AC* (7 bitů)	*FC* (4bity)	*K* 1 bit	*F* (7 bitů)	
Řízení paměti a v/v	řízení adresy příští mikroinstrukce obvodu 3001	nastavení návěští obvodu 3001	*F* — standardní mikrooperace obvodů 3002 *K* = 0 . . . všechny bity sběrnice *K* = 0 *K* = 1 . . . všechny bity sběrnice *K* = 1		Význam polí

Prodloužení cyklu vlivem přerušení lze eliminovat aplikací multiplexeru pro zavedení řádkové adresy (vyloučíme dobu t_R), jak je naznačeno na *obr. 6c*. Jiná možnost je ve vzorkování speciálním bitem mikroinstrukce namísto výstupem *ISE*, ve spojení s modifikací adresy v nejnižším bitu sběrnice *AC* výstupem *IA*.

Celkové posouzení časových vztahů

Rozborem dílčích funkcí procesoru z hlediska časování jsme stanovili požadavky na dobu cyklu funkcí. Poté musíme posoudit komplexně časové vazby a zvolit vhodný způsob ovládání generátoru hodin (řízení tzv. časovacím polem mikroinstrukce). Dále uvážíme vliv zbývajících částí systému a časování ve vztahu k asynchronním hlášením. Asynchronní vstupy systému je nutno synchronizovat systémovými hodinami.

Po provedení návrhu vypracujeme definitivní časové diagramy funkcí a kontrolujeme předstihy a přesahy podle specifikací použitých součástek. Při synchronizaci prvků oddělenými hodinovými signály musíme uvážit vzájemnou časovou vazbu hodin a její přípustné tolerance.

Návrh mikroprogramů

Jestliže je stanovena konfigurace prvků procesoru a je stanoven soubor instrukcí, zbývá nám návrh mikroprogramů. K tomu musí být matematicky přesně definován soubor požadovaných instrukcí, programátor musí dokonale rozumět konfiguraci svého procesoru, znát funkce všech jeho jednotek a jejich vztah k formátu mikroinstrukce.

Postup návrhu mikroprogramů má následující etapy:

— návrh tokových diagramů mikroprogramů a jejich vyjádření v symbolech mikrooperací — vyplývá z požadovaných funkcí procesoru,
— obsazení adres řídicí paměti realizovanými mikroprogramy a vyjádření obsahu řídicí paměti ve formě tabulek programování použitých typů pamětí PROM.

Jak již bylo řečeno v předchozím výkladu, mikroprogram popisuje provedení instrukce procesoru sekvencí dílčích operací popsaných mikroinstrukcemi. V jednom taktu určuje adresu paměťového místa v řídicí paměti, vybírá z něj mikroinstrukční slovo, provádí jím zvolené mikrooperace a připravuje další průběh mikroprogramu. Každému bitu mikroinstrukčního slova odpovídá binární signál na výstupu řídicí paměti. Skupiny řídicích linek připojených na tyto výstupy se logicky dají roztřídit na tzv. pole, jejichž stav určuje funkci ovládané jednotky. Mikroinstrukce procesoru postaveného z obvodů řady 3000 sestává ze standardních polí *F* (7 bitů), *AC* (7 bitů), *FC* (4 bity) a z polí, jejichž rozsah a obsah určuje aplikátor. Pole *K* má rozsah podle počtu potřebných masek a konstant. Pole řízení *R* má rozsah a obsah podle potřebných systémových řídicích signálů — pro styk s pamětí a v/v.

Symbolický zápis mikroprogramů

Pro zpřehlednění programátorské práce je nezbytné vyjadřovat mikroinstrukce symbolicky. Obvyklá symbolika standardních mikrooperací obvodů 3001, 3002 je uvedena v [1]. Navíc si uživatel musí stanovit symboliku pro přídavné řídicí pole mikroinstrukce a symboliku instrukcí. Takto si uživatel vytvoří svůj symbolický jazyk — mikroassembler, s nímž bude dále při realizaci mikroprogramů pracovat. Pro přehlednost je účelné v symbolickém zápisu mikroprogramu vypisovat jen mnemonický popis právě aktivních polí.

Pro náš účel zaveďme např. formát mikroinstrukce podle *tab. 1*. Význam funkcí řídicího pole definujme v *tab. 2*.

Tab. 2. Příklad specifikace řídicích funkcí procesoru

Kód			Symbol	Funkce
R_2	R_1	R_0		
0	0	0	*RM*	čti z paměti
0	0	1	*WM*	piš do paměti
0	1	0	*RI*	čti data ze vstupního zařízení
0	1	1	*WO*	piš data do výstupního zařízení
1	0	0	*NOP*	žádná operace

Pro další popis jsme tím specifikovali formát mikroinstručního slova. Symbolické vyjádření mikrooperací ilustruje následující příklad.

Příklad 1

Zadání: Popište symbolicky a zakódujte operaci: „Sečti obsah registru *R5* s obsahem střadače *AC*, výsledek přesuň zpět do registru *R5*, aritmetický přenos výsledku přenes do indikátorového registru *C*, skoč na následující mikroinstrukci, umístěnou v nultém řádku, 10. sloupci".

Řešení: podle tabulek funkcí 3001, 3002 z [1] a *tab. 1, 2*:

Popis

R5 + AC + CI → R5 . . . . . ADR (R5)

$CI = 0; CO \rightarrow C$ ..... *FFO STC*
skok (00, 10) ..... *JZR (10)*
Paměť a v/v neaktivuj *NOP*

Symbolický popis dané operace tedy zní: *ADR (R5)*, $K = 1$, *FFO, STC, NOP, JZR* (10);

Kód
$F \equiv 0110101; K \equiv 1$
$FC \equiv 0010$
$AC \equiv 0101010$
$R \equiv 100$

Aby byl popis kompletní, nestačí definovat symboly mikrooperací. Pro účely umístění v paměti mikroprogramu musíme každou mikroinstrukci označit symbolickou adresou — v našem příkladu nazveme symbolicky uvažovanou operaci *ADD (R5)*. Toto označení pak slouží v našem mikroassembleru (jazyce symbolických adres mikroinstrukcí) jednak jako název operace, jednak jako symbolická adresa mikroinstrukce, kterou tuto operaci popisujeme. Pro zachování přehlednosti zápisu musíme zavést oddělovací znaménka, např.:

„:“ — oddělovač symbolické adresy
„,“ — oddělovač symbolických popisů polí
„;“ — označení konce zápisu jedné mikroinstrukce.

Výsledný zápis bude:
symbolicky — *ADD (R5) : ADR (R5)*, $K = 1$, *FFO, STC, NOP, JZR (10)*;

odpovídající mikroinstrukce strojově:
100 0101010 0010 1 0110101

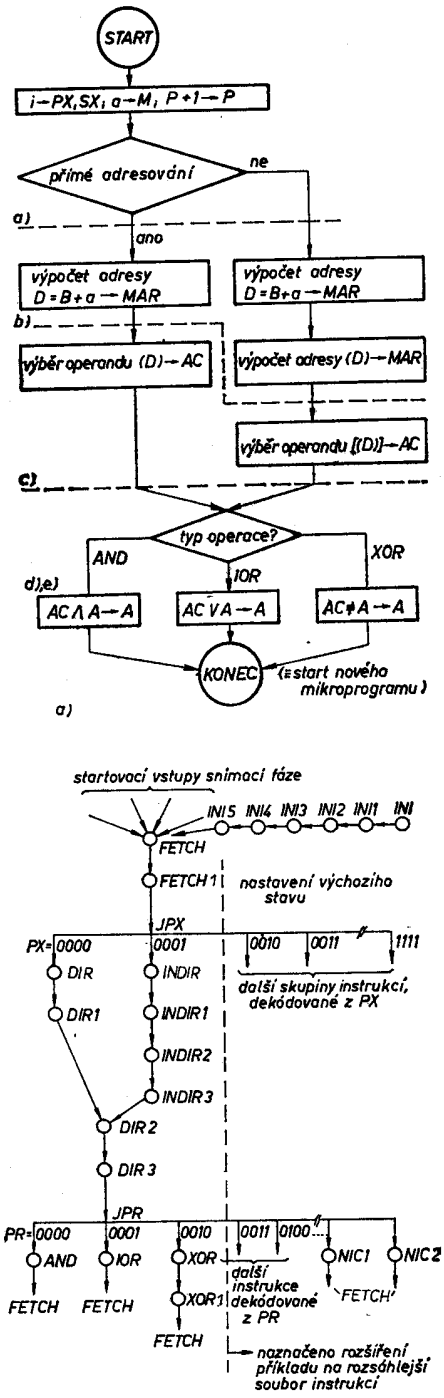
Grafický popis mikroprogramů

Pro každou požadovanou instrukci výsledného stroje navrhujeme mikroprogram. Ten popisujeme sledem symbolicky vyjádřených mikroinstrukcí. Protože části mikroprogramů mohou splývat, je vhodné, pro zachování přehledu v rozsáhlém souboru instrukcí, použít jednoduchého grafického znázornění průběhu mikroprogramů jejich vývojovým diagramem a znázornění souvislosti více mikroprogramů orientovaným grafem. Na následujících příkladech ukážeme užitečnost tohoto popisu.

Příklad 2

Zadání: Navrhněte mikroprogram logických operací součinu, součtu a nonekvivalence, popsané vztahy a symboly:
ANDD: $(D) \wedge (A) \rightarrow A, (P) + 1 \rightarrow P$;
ANDI: $|(D)| \wedge (A) \rightarrow A, (P) + 1 \rightarrow P$;
IORD: $(D) \vee (A) \rightarrow A, (P) + 1 \rightarrow P$;
IORI: $|(D)| \vee (A) \rightarrow A, (P) + 1 \rightarrow P$;
XORD: $(D) \not\equiv (A) \rightarrow A, (P) + 1 \rightarrow P$;
XORI: $|(D)| \not\equiv (A) \rightarrow A, (P) + 1 \rightarrow P$;

kde P je programový čítač procesoru umístěný do registru *RO* operační jednotky;
A je střádač procesoru, umístěný do registru *R1* operační jednotky;
B je registr báze adresy dat, umístěný do registru *R2* operační jednotky;
D je paměťové místo spolupracující paměti, jehož adresa je $(B) + a$ při přímém nebo $(B) + a$) při nepřímém adresování;
(X) je obsah registru X nebo paměťového místa s adresou X;



Obr. 8. Grafická interpretace příkladu 2; a) vývojový diagram, b) orientovaný graf

$|(X)|$ je obsah paměťového místa s adresou (X).

Struktura procesoru podle *obr. 7*, formát instrukce

15	8 7	0
	i	a

kde a je adresová část strojového kódu instrukce;
i je operační část strojového kódu instrukce.

Řešení: příprava a provedení operací zahrnuje následující kroky:
a) snímání strojového kódu instrukce, inkrementace programového čítače;
b) výpočet adresy paměťového místa D;
c) výběr operandu z paměťového místa D;
d) provedení operace (D) op (A);
e) umístění výsledku (D) op $(A) \rightarrow A$.

Grafický popis mikroprogramu vývojovým diagramem viz *obr. 8a*. Každému kroku zde odpovídá funkční blok, který se může v konkrétních případech popsat jednou i více mikroinstrukcemi. Je zřejmé, že pro řešenou skupinu mikroprogramů budou mikroprogramy splývat v přípravných krocích a v kroku umístění výsledku. Orientovaný graf skupiny instrukcí je znázorněn na *obr. 8b*.

Dále popíšeme mikroprogramy v jednotlivých fázích vývojového diagramu.

a) *FETCH*: *LMI(RO)*, $K = 0$, *FF1, HCZ, NOP, JMP(FETCH 1)*; Mikroinstrukce *FETCH* provádí přesun $(P) \rightarrow MAR$, adresování paměti pro čtení instrukce, inkrement programového čítače $(P) + 1 \rightarrow P$, skok na následující mikroinstrukci na symbolické adrese *FETCH 1*, obecný symbol *JMP* bude v navazujících etapách návrhu nahražen konkrétní mikrooperací obvodu 3001 — např. *JZR, JCC, JCR, JCE*).
FETCH 1: *LTM(T)*, K = *OOFF, FFO, HCZ, RM, JPX* (*DIR, INDIR*, ...)
Mikroinstrukce *FETCH 1* provádí přesun adresové části a instrukce z paměti do registru T, $a = (M) \wedge K \rightarrow T$; (nutno zavést masku K = OOFF hexadecimálně, což zpětně ovlivní zapojení procesoru a formát mikroinstrukce). Dekóduje primární část operačního znaku rozvětvením mikroprogramu na následující mikroinstrukce *DIR, INDIR* ... (další cíle větvení zde zatím nedefinovány; celkový počet větví 16).

b) *DIR*: *ILR(R2)*, $K = 0$, *FFO, HCZ, NOP, JMP (DIR 1)*; Tato mikroinstrukce připravuje výpočet adresy operandu přesunem obsahu bázového registru do registru *AC*. $(B) \rightarrow AC$. Skok na *DIR 1* bude definován podrobně v dalších fázích návrhu.
DIR 1: *ALR(T)*, $K = 1$, *FFO, HCZ, NOP, JMP (DIR 2)*;
Výpočet adresy paměťového místa D $(AC) + (T) \rightarrow AC, T$.

c) *DIR 2*: *LMI(AC)*, $K = 0$, *FFO, HCZ, NOP, JMP (DIR 3)* .Přesun vypočtené adresy do adresového registru, adresování paměti pro čtení operandu $(AC) \rightarrow MAR$.
DIR 3: *LTM(AC)*, $K = 1$, *FFO, HCZ, RM, JPK* (*AND, IOR, XOR*, ...)
Přesun obsahu paměťového místa do *AC*. $(M) \rightarrow AC$. Pro přímé adresování má význam operandu. 16-cestné větvení mikroprogramu podle sekundární části operačního znaku v registru *PR* obvodu 3001.

Fáze b), c) pro nepřímé adresování operandu jsou až po *DIR 3* totožné. Výsledný obsah *AC* má však význam adresy operandu a je nutno pokračovat adresováním paměti a výběrem operandu mikroinstrukcemi obsahově totožnými s *DIR 2, DIR 3*. Průběh nepřímého adresování bude:
INDIR — obsahově shodné s *DIR*
INDIR 1 — obsahově shodné s *DIR 1*
INDIR 2 — obsahově shodné s *DIR 2*
INDIR 3 — obsahově shodné s *DIR 3*,

místo *JPR* nepodmíněný skok *JMP*
INDIR 4 — obsahově shodný s *DIR 2*
INDIR 5 — obsahově shodný s *DIR 3*.

Oba mikroprogramy je možno sdružit ve fázi c) — *INDIR 4, 5* s *DIR 2, 3* — viz *obr. 8b*. Fáze provedení operace d) a umístění výsledku e) pak pro oba způsoby adresování splývá také. To je výhodné z hlediska úspory paměťových míst v řídící paměti.

d), e) Provedení operací *AND, IOR, XOR* a přesun výsledku do střadače *A* (≡ *R1*).
(Zkrácený zápis — jen aktivní pole)
AND: *ANR(R1), JMP (FETCH)*;
provádí $(A) \wedge (AC) \rightarrow A$
IOR: *ORR(R1), JMP (FETCH)*;
provádí $(A) \vee (AC) \rightarrow A$
XOR: *XNR(R1), JMP (XOR 1)*;
provádí $(A) \equiv (AC) \rightarrow A$, skok na *XOR 1*
XOR 1: *CMR(R1), JMP (FETCH)*;
provádí $(\overline{A}) \rightarrow A$.

Závěrečná mikroinstrukce pro každou požadovanou logickou operaci procesoru zároveň provádí skok na snímací fázi nové instrukce. Je-li požadováno vzorkování přerušení, je vhodné skok *JMP (FETCH)* definovat jako *JZR (15)* pro vybavení výstupu *ISE* obvodu 3001. Mikroinstrukce *FETCH* bude umístěna v nultém řádku, 15. sloupci řídicí paměti. Standardní mechanismus přerušení pomocí obvodu 3214 vyvolá (je-li přerušení akceptováno cestou *IA* → *ERA*) adresu řádku 31, kam musíme umístit startovací adresy mikroprogramů obsluhy přerušení.

Sestavení mikroprogramů daného instrukčního souboru

Z předchozího příkladu vyplývají obecné závěry pro vytváření mikroprogramů. Požadovaný instrukční soubor rozdělíme do skupin instrukcí s podobným průběhem mikroprogramů. V úvodní fázi dekódování instrukce rozlišíme především mody adresování a formáty instrukcí (bezprostřední, jednoadresové, skoky, přesuny a pod.). Jednotlivé instrukce každé skupiny rozlišíme dekódováním podle zapamatované části operačního znaku (*JPR, JRL, JLL* funkce).

Je vhodné nejprve průběh mikroprogramu popsat pouze v oblasti mikrooperací operační jednotky a návěstí. Mikrooperace řadiče uvést jen tam, kde se mikroprogram větví (*JPX*), nebo tam, kde se více mikroprogramů sdružuje (*JZR*). Po napsání symbolických mikroprogramů všech instrukcí budeme teprve uvažovat o obsazení řídicí paměti a doplníme příslušné mikrooperace skoků v každé mikroinstrukci.

Všechny větvící stavy musí mít v mikroprogramech východisko — to znamená doplnit nedefinované instrukční kódy prázdnými operacemi. Např. z *obr. 8b* je vidět způsob doplnění grafu mikroprogramů z *příkladu 2* o prázdné mikroinstrukce *NIC 1* ... *NIC 2*. Východiskem z nich může být mikroprogram, který hlásí neplatný instrukční kód, nebo mikroinstrukce *FETCH*. Do ní směřují cílové skoky všech mikroprogramů, tím je zajištěna cyklická práce procesoru s možností přerušení po každém provedení kompletní instrukce programu. Sestavením orientovaného grafu a symbolického zápisu celého souboru instrukcí tato fáze návrhu končí. Jde tady použít řady dalších programovacích obratů, jako využití výstupů *PR* střádače pro modifikaci makroinstrukcí, využití podmíněného hradlování hodin *CPE* pro nedestruktivní testování obsahu registrů apod. Následující příklady jsou uvedeny pouze pro ilustraci techniky mikroprogramování.

Mikroprogram pro nastavení výchozího stavu procesoru

Příklad 3

Zadání: V 16-bitovém procesoru z *příkladu 2* proveď vynulování obsahu registrů a čtení nulté buňky paměti, která obsahuje adresu první instrukce uživatelského programu. Připrav snímací a dekódovací cyklus této makroinstrukce.

Řešení:	Popis operace:
INICIACE:	
CLR(RO)	⇒ $O \rightarrow RO$; nulování *P*
INI 1:	
CLR(R1)	⇒ $O \rightarrow R1$; nulování *A*
INI 2:	
CLR(R2)	⇒ $O \rightarrow R2$; nulování *B*
INI 3:	
LMI(RO)FFO	⇒ $RO \rightarrow MAR$, $RO \rightarrow RO$, $CI = 0$; adresování nulté buňky
INI 4:	
ACM(AC)RM	⇒ $M \rightarrow AC$; čti z paměti poč. adresu programu do *AC*
INI 5:	
SDR(RO), FF1,	
JZR(15)	⇒ $AC \rightarrow RO$; $CI = 1$; počáteční adresa zavedena do *P*, skok na *FETCH*.

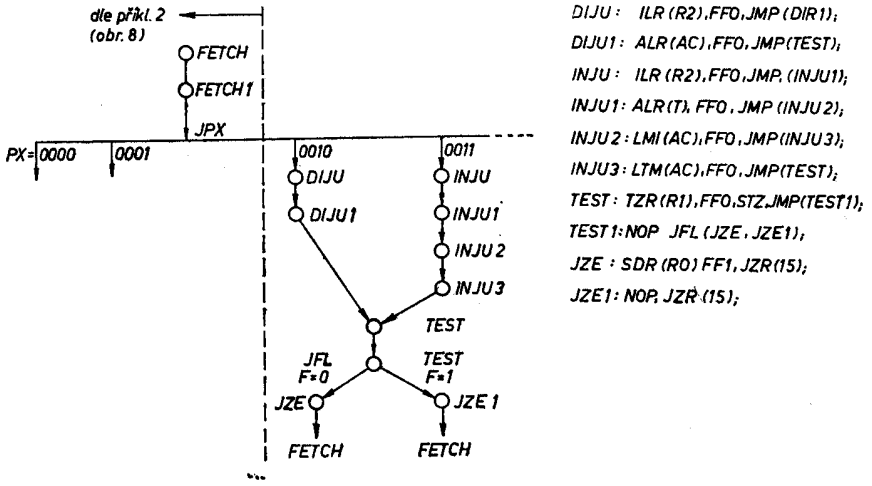
Vyvolání nastavovacího mikroprogramu, které by mělo proběhnout po zapnutí sítě nebo na přání operátora, úzce souvisí s technickým vybavením procesoru. Jednoduchou možnost poskytuje obvod 3001 tím, že lze zavést data ze sběrnice *PX, SX* přímo do registru adresy řídicí paměti a určit tak startovací adresu mikroprogramu.

Mikroprogramy s větvením podle návěstí

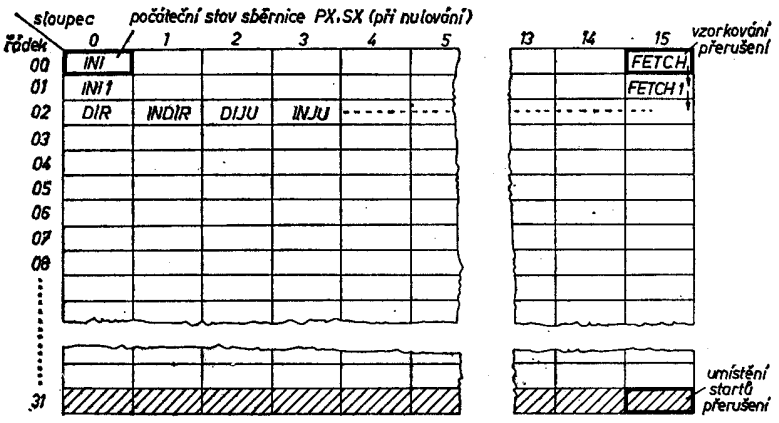
Hlášení operační jednotky předávaná z výstupů *CO, RO* na vstup *FI* řadiče jsou v každém taktu zaznamenána v jednobitovém střádači *F*, jeho stav lze přepsat do indikátorových registrů *C, Z*. Má-li mikroprogram rozhodnout o dalším průběhu podle stavu registrů *F* nebo *C* nebo *Z*, musíme vzít v úvahu, zda procesor pracuje v postupném nebo překrývaném režimu — viz tabulka na *obr. 5*. Mikroinstrukce s podmíněným skokem *JFL, JCF, JZF* musí být umístěny tak, aby v době jejich provedení příslušný registr obsahoval platnou podmínku skoku.

Příklad 4

Zadání: V procesoru z *příkladu 2* navrhněte mikroprogram pro instrukci



Obr. 9. Mikroprogram skoku podle obsahu střádače pro procesor s překrýváním



Obr. 10. Počátek rozmisťování mikroprogramů v řídicí paměti (viz *obr. 8* a *9*)

podmíněného skoku podle obsahu střádače A.
$(B) + a \rightarrow P$ jestli $(A) = 0$; jinak
$(P) + 1 \rightarrow P$ (přímo adresovaný skok)
$/(B) + a/ \rightarrow P$ jestli $(A) = 0$; jinak
$(P) + 1 \rightarrow P$ (nepřímo adresovaný skok).

Řešení: následující kroky mikroprogramu — viz *obr. 9*

a) snímání instrukce, inkrement P;
b) výpočet adresy skoku (přímé, nepřímé);
c) nedestruktivní test obsahu střádače A;
d) větvení mikroprogramu podle výsledku testu, konec.

Ve fázích a), b) bude průběh mikroprogramu odpovídat fázím a), b) *příkladu 2*. Vypočtená adresa skoku je v registru AC. V dalších fázích se bude průběh částečně lišit pro obě konfigurace řadiče *(obr. 5)*.

Bez překrývání:

1. takt:
TEST: *TZR(R1) FFO STZ JFL (JZE, JZE1)*; — větvení podle stavu střádače F, zápis $FI \rightarrow Z$
2. takt:
JZE: *SDR(RO) FF1 HCZ JZR(15)*; $AC \rightarrow P$ pro $(A) = (F) = 0$, skok na *FETCH*
JZE 1: *NOP JZR(15)*; žádná operace pro $(A) \neq 0$, $F = 1$, skok na *FETCH*.

Kdybychom použili větvení podle stavu indikátoru Z, byl by mikroprogram o jednu mikroinstrukci delší (1. takt = test + zápis do Z; 2. takt = = větvení podle Z; 3. takt *JZE* nebo *JZE 1* podle výsledku testu).

S překrýváním:

1. takt:
TEST: *TZR(R1) FFO STZ JMP (TEST1)* — test $(R1) = (A) = 0$?
2. takt:
TEST 1: *NOP JFL (JZE, JZE1)* — větvení podle výsledku testu v střádači F
3. takt:
JZE nebo *JZE 1*

V případě větvení podle stavu C nebo Z by byl mikroprogram opět delší.

Mikroprogramová obsluha přerušení

Standardní realizace přerušení pomocí obvodů 3214 a 3001 je zaměřena na vektorový prioritní přerušovací systém. Přijatý požadavek přerušení způsobí v řadiči skok na startovací adresu, umístěnou ve 31. řádku řídicí paměti. Adresu sloupce je vhodné odvodit od vektoru přerušení, který si v úvodní fázi obsluhy řadič přečte ze sběrnice *PX*, *SX* (výstupy A obvodu 3214). Obslužné mikroprogramy se liší podle účelu obsluhované vnější jednotky, zpravidla však obsahují:

— uklízení přerušeného stavu procesoru,
— vlastní obsluhu (přesun dat z nebo do periférie),
— vybavení přerušeného stavu procesoru,
— snímání nové instrukce programu.

Obsazení řídicí paměti

Na podkladě symbolického a grafického zápisu kompletního souboru mikroprogramů řešíme jejich umístění. Jako pomůcku k tomu použijeme rastr 32 řádků × 16 sloupců, do něhož postupně přepíšeme symbolicky zapsané mikroprogramy. Rastr představuje matici řídicí paměti. Na každé paměťové místo je vhodné zapisovat při postupném zaplňování matice:

— symbolickou adresu (název) mikroinstrukce,
— volací adresu v matici,
— cílovou adresu v matici.

Mikroprogramy musíme doplnit specifikací funkce 3001 v těch místech, kde je vyžadován podmíněný skok (*JFL, JCF, JZF, JPR, JLL, JRL, JPX*). Přitom si musíme poznamenat i počet možných cílových adres takového skoku a zaznamenat zvolený podmíněný skok do grafu mikroprogramu. Pokud není zadán strojový kód, volíme jej podle rozmístění dekódovaných stavů *PX*, *SX*.

Nejprve umístíme základní mikroprogramy stroje — iniciaci, snímání makroinstrukce, přerušení, protože jejich startovací adresy bývají dány zapojením procesoru. Pak rozmisťujeme shluky podmíněných skoků, souvisejících obvykle s dekódováním instrukce. Přitom rozmisťování může ovlivnit zadaný strojový kód a vyžádat si provedení změn zadání (resp. předřazení mapovací paměti PROM). Potom umístíme podmíněná větvení podle návěští *JFL, JCF, JZF*. Posléze krátké a nakonec dlouhé sekvence nepodmíněných skoků, které umístíme nejsnáze v téměř zaplněné matici. Způsob obsazování ilustruje *obr. 10*.

Popsaným postupem se aplikátor dopracuje k zapojení a mikroprogramům navrhované aplikace. Navazující etapy návrhu:

— překlad symbolicky popsaných mikroprogramů do binární formy,
— ladění mikroprogramů a oživování procesoru,
— tvorba aplikačních programů, jejich ladění a oživení aplikace.

Předpokladem úspěšného návrhu je automatizace rutinních prací. K tomu je nutno použít jako nezbytné minimum programové i technické prostředky:

— překladač symbolického jazyka (mikroassembler)
— simulátor řídicí paměti s možností rychlé výměny a oprav mikroprogramů,
— logický analyzátor a další přístrojové vybavení.

Efektivnost práce je možno dále zvýšit použitím programových i technických simulátorů, emulátorů, vývojových systémů apod.

Náš článek popisuje metodiku aplikace stručně a všeobecně. Jeho cílem je inspirovat; nepodává návod na stavbu konkrétního stroje.

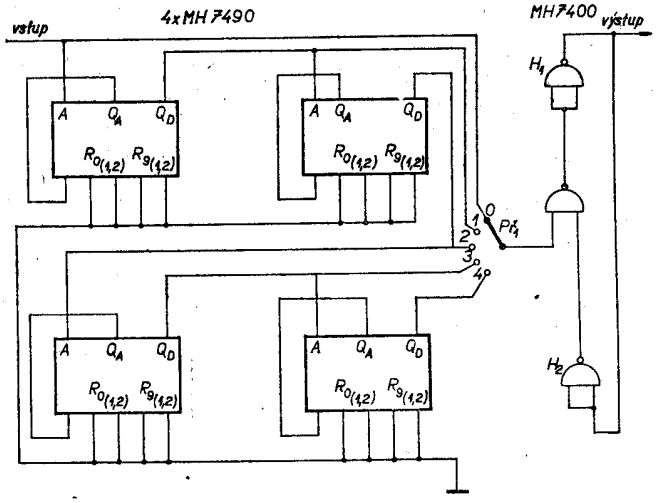
LITERATURA

[1] *Gucký, T.: Prvky mikroprocesorového systému řady 3000, Sdělovací technika*
[2] *Rattner, J., Cornet, J., Hoff, M. E.: Bipolar LSI Computing Elemente Usher in New Era of Digital Design, Electronics, září 5, 1974.*
[3] *Dědina, B.: Zásady návrhu procesoru se stavebnicí I 3000, část I, Výzkumná zpráva ČSAV — ÚTIA Praha, říjen 1976.*
[4] *Kolektiv autorů: Sborník semináře „Mikroprocesorový bipolární systém Intel 3000", DT ČVTS Ostrava, PČVTS TESLA Rožnov, listopad 1977.*
[5] *Series 3000 Reference Manual, Intel Corp. USA, 1976.*

ČÍSLICOVÝ KMITOČTOMĚR S OBRAZOVKOVOU INDIKACÍ

V dílenské a radioamatérské praxi je mnohdy zapotřebí orientačně zjistit kmitočet logického signálu. Nemáme-li právě k dispozici tovární kmitočtoměr, můžeme použít jednoduchý adaptér podle *obr. 1* k televiznímu přijímači. Kmitočet logického signálu je dán vztahem $f \doteq 60.p.10^{n}$, kde p značí počet vodorovných pruhů vytvořených na obrazovce přijímače a n je číslo polohy přepínače $Př_1$ na *obr. 1*, který nastavujeme tak, aby $3 \leq p \leq 33$. Děličky (4 × MH7490) používáme proto, abychom nemuseli identifikovat nepravidelně rozmístěné části svislých pruhů při vysokých kmitočtech. Měřicí rozsah pokrývá oblast přibližně 200 Hz až 20 MHz, u kmitočtů do 2 kHz lze registrovat střídu impulsů logického signálu. Kmitočtoměr napájený z baterie 4,5 V zabudujeme do stíněné krabičky a signál k jedné ze zdířek anténního vstupu televizního přijímače vedeme koaxiálním kabelem. Vyšší harmonické generátoru H_1, H_2 leží ve všech kanálech I. až III. pásma TV, takže při měření stačí naladit přijímač na nejbližší vysílačem neobsazený kmitočet.

D. Němec



Obr. 1. Zapojení jednoduchého adaptéru k televiznímu přijímači

Programové vybavení pro návrh mikropočítačů s mikroprocesorovými elementy řady MH3000

ING. ZBYNĚK CHYTIL

Úvod

V předcházejících článcích [1], [2] byly popsány základní vlastnosti mikroprocesorových elementů řady MH3000 a postup návrhu mikropočítačů s těmito elementy. Následující článek má za úkol seznámit aplikátory mikroprocesorových elementů řady MH3000 s programovými prostředky, které podstatně usnadňují návrh mikroprogramů systému a zejména odstraňují rozsáhlé rutinní práce a kontrolují možné formální chyby návrhu mikroprogramu. Návrh zapojení celkové konfigurace systému byl podrobně diskutován a objasněn v [2]. Návrh mikroprogramu systému byl ve [2] naznačen bez uvedení možností využití programových prostředků.

Z citovaných článků vyplývá, že návrh mikroprogramu mikropočítačového systému je jeho nezbytnou součástí. Návrh mikroprogramu se rozpadá na několik základních kroků:

1. Stanovení souboru makroinstrukcí celého mikropočítače, optimálně odpovídajících předpokládanému aplikačnímu použití.
2. Vytvoření mikroprogramových kroků, popisujících mikroinstrukce, potřebné pro realizaci zadaného souboru makroinstrukcí.
3. Návrh sloučení jednotlivých mikroprogramových kroků a jejich vyjádření šipkovým vývojovým diagramem.
4. Návrh uložení mikroinstrukcí v paměti mikroprogramu a odpovídajících skokových instrukcí.
5. Překlad výsledků kroků 2 a 4 do binárního programu pro zápis do paměti mikroprogramu.
6. Uložení binárního programu do paměti mikroprogramu (obvykle realizovaném pamětmi PROM).

Kroky 1 až 4 zahrnují vlastní tvůrčí práci návrháře mikropočítače. Přitom kroky 2 a 4 se musí řídit řadou pravidel, vyplývajících z vlastností mikroprocesorových elementů. Kroky 5 a 6 obsahují jen rutinní, ale značně rozsáhloučinnost,vykonávanourovněžpodle přesně definovaných pravidel. Naskýtá se tedy možnost s cílem usnadnění a urychlení návrhu využít výpočetní techniku pro kontrolu formálních chyb v krocích 2 a 4 a pro celkové provedení kroků 5 a 6.

Programové vybavení, nabízené výrobci mikroprocesorových elementů

Pro usnadnění programové části návrhu mikropočítačového systému nabízí firma INTEL programový systém „Cromis", jehož základní vlastnosti budou popsány v dalším. Cílem použití programu „Cromis" je usnadnit uživateli výše uvedené kroky návrhu mikroprogramu převedením rutinních činností na výpočetní techniku a maximálním využitím mnemotechniky pro usnadnění, zpřehlednění a urychlení tvůrčích činností, jakož i detailní kontrolou možných formálních chyb návrhu a zjednodušením jejich identifikace. Implementace programu „Cromis" je možná na libovolném počítači s délkou slova minimálně 16 bitů, dostatečnou pamětí (16 K slov operační paměti, 4 magnetopáskové jednotky) a překladačem jazyka FORTRAN IV. V dalším se nebudeme zabývat podrobnostmi programu „Cromis", které jsou obsaženy ve [3], ukážeme si pouze prostředky, které „Cromis" používá pro usnadnění práce návrháře mikropočítače.

Vlastní program „Cromis" se rozpadá na dvě části *XMAS* a *XMAP*. *XMAS* slouží ke čtení vstupních údajů, vyplývajících z výše uvedených kroků 2 a 4 návrhu, k definici uživatelských polí a k deklaraci mnemotechnických označení, sloužících k usnadnění návrhu. Výstupem *XMAS* je pak zkontrolovaný mikroprogram, výpis uložení mikroinstrukcí v paměti mikroprogramu, hlášení chyb ve vstupních údajích a binární data, sloužící jako vstup pro *XMAP*. Vzhledem k náročnosti etapy návrhu, kterou zahrnuje *XMAS*, lze předpokládat její několikeré opakování, ať už pro opravu formálních chyb, nebo pro neuspokojenost dosažených výsledků a z toho vyplývajících změn. To je hlavní důvod pro oddělení *XMAS* od *XMAP*, který provede převedení výsledků *XMAS*, oproštěných od chyb návrhu i od chyb formálních, do údajů, sloužících přímo pro naprogramování konfigurace pamětí PROM, sloužících paměť mikroprogramu. Tato etapa je jako obvykle prosta chyb a provádí se jako závěrečná jen jednou.

Zdrojová data programu „Cromis"

Zdrojový soubor dat obsahuje postupně seřazená data v jazyku FORTRAN,obsahující řídicí příkazy a příkazy jazyka *XMAS*. Soubor se skládá z deklarační a příkazové části. Pro účelný postup výkladu popíšeme nejdříve část příkazovou a postupně objasníme i část deklarační. Ve skutečnosti však deklarační část musí předcházet část příkazovou. Ve [2] bylo objasněno rozdělení mikroinstručního slova na jednotlivá pole a jejich rozdělení na pole standardní a přídavná (uživatelská). Tohoto rozdělení se přidržíme i při dalším výkladu.

Standardní pole příkaz ové části

V příkazové části vstupního souboru jsou mnemotechnicky popsána jednotlivá mikroinstrukční slova, přičemž nezáleží na pořadí, ve kterém jsou uvedena. Příklad popisu standartních polí (kromě pole sběrnice *K*) je

41 : L1 : L2 : ADR(R5) FFO STZ JLL(INP, OUT, DIAR, INAR)

Význam jednotlivých příkazů:

— celé číslo, jehož hodnota je v rozsahu 0 až n—1 (kde n je počet slov paměti mikroprogramu) určuje adresu mikroinstrukce v paměti mikroprogramu, jak bylo určeno krokem 4 návrhu;
— za tímto číslem může (a nemusí) následovat libovolný počet návěští, označujících danou mikroinstrukci a sloužících k odkazům ve skokových příkazech jiných mikroinstrukcí;
— za návěštími následuje vlastní specifikace mikroinstrukčního slova. Pro jednotlivá pole lze definovat obsah
  a) přímým určením obsahu pole v binárním (B), oktalovém (O nebo Q), dekadickém (D nebo bez označení), nebo hexadecimálním (H) vyjádření nebo jako aritmetického výrazu (např. CPE = = 6AH FI = 01B FO = 00B — poslední znak určuje způsob vyjádření);
  b) použitím mnemotechnického označení (nazývaného mikrop), uvedeného pro jednotlivá pole v *tabulkách 2, 3,* a *6* článku [1]. Není-li uveden obsah pole FI, pak se automaticky nastaví FI = 11B (~ HCZ), není-li uveden obsah pole FO, nastaví se FO = 00B (~ FFO).

Způsob ad b) je častější a je také uveden ve výše uvedeném příkladu. Specifikace pole JUMP (pole následující adresy) musí jako parametry v závorce obsahovat odkaz na všechny mikroinstrukce, na něž je prováděn skok. Odkazy musí být ve stejném pořadí, v jakém jsou odpovídající mikroinstrukce uloženy v paměti mikroprogramu. Odkazovat lze buď jedním z návěstí nebo výrazem, určujícím adresu mikroinstrukce. Pro zjednodušení lze navíc místo nepodmíněných skokových mikropů JCC, JCR a JZR psát označení JMP, přitom program sám určí, o který druh skoku se jedná.

Rozšíření příkazové části

Standardní tvar popisu mikroinstrukce lze dále rozšířit o uživatelem definovaná pole (jak uvedeme dále, obdobně se stanoví i obsah pole sběrnice *K*) a o řadu mnemotechnických pomůcek, zvyšujících přehlednost návrhu a usnadňujících práci návrháře.

Deklarace nového pole

Nové pole lze v deklarační části programu definovat příkazem FIELD. Význam objasníme na příkladu:

LOAD FIELD LENGTH = 2 DEFAULT = 0
MIOROPS (LOADA = 1 LOADB = 3)

Hodnotu nového pole lze pak v popisu mikroinstrukce

— neuvést, pak se hodnota nastaví na hodnotu, uvedenou u DEFAULT;
— uvést mikropem (např. LOADA znamená nastavení hodnoty pole LOAD na 01B — obdoba aplikace např. mikropu STZ u pole FI);
— definovat hodnotou (např. LOAD = = 10B — obdoba např. CPE = = 1001101B).

Na tomto místě je nutno se zmínit o poli sběrnice *K*. Jak je uvedeno

v *odst. 3* článku [2], je často účelné definovat v poli *K* mikroinstrukce jen nezbytné konstanty a masky a ty transformovat vhodným dekódováním ke vstupům *K* jednotlivých řezů. Pak ovšem pole *K* mikroinstrukce není totožné se sběrnicí *K* centrální jednotky. Toto pole *K* se pak deklaruje podobně jako nové uživatelské pole s tím rozdílem, že není deklarován DEFAULT. Není-li obsah pole *K* definován, nastaví se v závislosti na mikropu pole CPE na vstupy *K* samé nuly nebo samé jedničky (viz *tab. 6* v článku [1]). Aby platila tato odchylka, je za deklarací nutno uvést např. MASK KBUS (kde MASK je mnemooznačení pole sběrnice *K*).

Mnemotechnická rozšíření

Příkazem IMPLY lze při použití některého mikropu definovat hodnotu i jiných polí. Např. příkaz LOADA IMPLY GATE = 0 CCTL = 1011B znamená, že při použití mikropu LOADA v popisu mikroinstrukce (což při použití výše uvedeného příkladu znamená hodnotu pole FIELD = 1) se současně pole GATE nastaví na hodnotu 0 a CCTL na 1011 binárně. Příkazem VALUE můžeme libovolnému výrazu (jazyka FORTRAN) přiřadit mnemotechnické označení, které pak používáme místo tohoto výrazu (používá se při častém opakování dlouhých výrazů nebo pro snadnou modifikovatelnost programů).

Příklad: PL VALUE A OR 10011B umožní místo výrazu (A OR 10011B) použití mnemotechnického označení PL.

Příkazem STRING je možno delší skupinu popisů jednotlivých polí nahradit mnemotechnickým označením. Např. SRAM STRING 'KB = = 101101B SRA STZ' znamená, že použití v popisu mikroinstrukce SRAM(AC) má stejný účinek, jako SRA(AC) STZ KB = 101101B.

Rozšířené adresování

Samotný obvod I3001 umožňuje adresovat paměť mikroprogramu do 512 slov. Při potřebě většího rozsahu paměti mikroprogramu je možno v deklarační části deklarovat dodatečné pole a příkazem např. ADR ADDRESS (kde ADR je identifikátor dodatečného pole) pak specifikovat, že toto pole má být považováno za další vyšší bity následující adresy mikroprogramu. Není pak nutno specifikovat obsah tohoto pole, ten se automaticky vytvoří obdobně, jako obsah standardního pole následující adresy.

Priority určení obsahu pole

Jak bylo výše objasněno, lze obsah pole určit více způsoby. Přitom jednotlivé způsoby mají tuto prioritu (od nejvyšší k nejnižší):

1. Explicitní vyjádření nebo stanovení mikropem.
2. Mikrop definovaný v IMPLY.
3. V případě sběrnice *K* stanovení, vyplývající z mikropu CPE.
4. Hodnota, uvedená v deklaraci FIELD za DEFAULT.

Výstupy programu „Cromis"

První etapa zpracování dodává 4 výstupy:

Výpis zdrojových dat a binárního překladu

Tento výpis obsahuje vstupní data tak, jak byla zadána, přičemž každý řádek záznamu je průběžně číslován. Současně je u každého řádku vytištěn binární překlad seskupený do standardních polí CPE, FI, FO, JUMP a uživatelských polí, definovaných v deklarační části programu. Příklad výpisu je v *tab. 1.* Je možno rovněž realizovat výpis jen zdrojových dat nebo jen binárního překladu.

Tab. 1. Příklad části výpisu zdrojových dat a binárního překladu

```
109   MASK FIELD LENGTH = 9 DEFAULT = 0;
110   MASK KBUS;
111   MEM FIELD LENGTH = 3 DEFAULT = 0;
112   140: M12: ILR(R2) FI = 0B FO = 0B MASK = 000H JMP(GST);
      (0008CH) 0000010 00 00 0100110 000000000 000
113   6: GST: ADR(R9) MASK = 1FFH JLL (WD1, WI1, BD1, BI1);
      (0006H) 0111001 00 11 110111 111111111 000
                          .
                          .
                          .
                          .
                          .
```

Křížový odkazový slovník

Slovník obsahuje abecední seznam všech návěstí a pořadových čísel záznamů, kde se příslušné návěstí vyskytuje. Pořadové číslo záznamu, kde je návěstí definováno, je v závorkách.

Obraz paměti mikroprogramu

Obraz paměti mikroprogramu znázorňuje graficky obsah paměti mikroprogramu, vytvořený zpracováním vstupních dat. Obraz je vytvořen maticí buněk o 16 sloupcích a tolika řádcích, aby zahrnul celou paměť mikroprogramu. V každé buňce je pak uveden mnemokód skoku v odpovídající mikroinstrukci, hexadecimální adresa 1. cíle skoku, pořadové číslo záznamu ve výpisu programu a počet buněk, ze kterých je proveden skok do dané buňky. Celkový tvar obrazu je velmi blízký výpisu, který je popsán dále jako výstup zjednodušeného programu. Pro jeho rozsáhlost jej neuvádíme zvláště.

Zdrojový soubor pro XMAP

Tento soubor obsahuje jen binární překlad a informace, nezbytné pro oddělení jednotlivých částí. Slouží pouze jako mezilehlé médium mezi dvěma etapami zpracování.

Hlášení chyb

Kromě výše uvedených výpisů jsou ve výpisu zdrojového souboru a binárního překladu vypsány chyby, nalezené ve zdrojovém souboru, zahrnující zejména formální chyby a nesprávně nebo neralizovatelně zadané skoky v mikroprogramu. Je vždy uvedeno číslo chyby, podle kterého lze podrobně určit druh chyby a je specifikováno místo programu, kde se chyba vyskytuje.

Výstup pro programování paměti PROM

Závěrečná etapa zpracování dodává jako nejdůležitější výsledek děrnou pásku nebo jiné médium obsahující data pro přímé programování paměti PROM paměti mikroprogramu. Úvodními příkazy jazyka *XMAP* lze nastavit formát výstupu tak, že pásku lze přímo použít jako vstup zařízení pro programování potřebných pamětí PROM.

Programové vybavení, využívající jednodušší výpočetní techniku

Pro řadu aplikačních použití mikroprocesorových elementů řady I3000 nejsou využity všechny výhody, které program „Cromis" poskytuje. Do popředí vystupují některé nevýhody, spočívající v práci s počítačem, který nebývá na některých pracovištích s dostatečnou operativností přístupný. V TESLA Rožnov n. p. byla jako zlepšovací návrh ZN 546/78 přijata zjednodušená varianta programu „Cromis", aplikovatelná na stolním kalkulátoru HP-9825. Omezení programu nejsou pro běžné aplikace podstatná a přitom je možno plně využít operativnosti kalkulátoru, který je na řadě pracovišť k dispozici. V dalším popisujeme základní vlastnosti tohoto programu, popis vstupů a výstupů, a srovnání s programem „Cromis".

Základní vlastnosti programu

Potřebné technické vybavení: kalkulátor HP-9825 s pamětí 24K bitů; vestavěnými ROM: string, general IO, extended IO, advanced programming; připojený snímač a děrovač děrné pásky a tiskárna.
Maximální velikost paměti mikroprogramu 256 slov.

Pole mikroinstrukčního slova:

JUMP 7 bitů (+1 bit pro externí definici skoku z hlavní paměti mikropočítače aplikací modifikačního vstupu LD)
FI 2 bity
FO 2 bity
CPE 7 bitů (+1 bit pro hradlování hodin centrální jednotky pro nedestruktivní testování registrů)
uživatelské pole (dále řídicí sběrnice) 1 až 8 bitů
K 1 až 8 bitů (pro max. 4 vstupy s možností rozšíření na 8 vstupů).

Vstupy a výstupy programu

Na *obr. 1* je znázorněn průběh činností při zpracování zadaného mikroprogramu. Vstupy programu jsou označeny číslicemi 1 až 3, výstupy písmeny A až H. První fází je definice použitých mnemotechnických zkratek, která odpovídá deklarační části „Cromis". Zadává se:

1. Mnemotechnické označení registrů centrální jednotky dvěma libovolnými znaky. Při dalších krocích lze pak užívat buď označení standardní

(R0 až R9, AC, T), nebo zadané mnemotechnické označení.

2. Mnemotechnické označení (až 6 znaků) kombinace kódu mikroinstrukce a instrukce příznaku (vstupní nebo výstupní) — zjednodušená deklarace STRING.
3. Definice vstupu pole *K* (max. 4 vstupy) a odpovídajícího výstupu na 8 vstupech sběrnice *K*.
4. Mnemotechnické označení (4 znaky) řídicího pole a odpovídající binární kombinace. Pole lze rozdělit až na 4 nezávislá pole, jimž odpovídají znaky nebo skupiny znaků v mnemotechnickém označení.

Poslední bod objasníme blíže. Příklad je v *tab. 2*. První řádek přiřazuje

Tab. 2. Příklad definice řídicí sběrnice

xxAW	xx0110
xxBC	xx1001
xAxx	x0xxxx
Yxxx	1xxxxx
Zxxx	0xxxxx

skupinu AW ve 3. a 4. znaku binární hodnotě 0110 v 3. až 0. bitu. Jakmile jednou bylo toto přiřazení provedeno, musí v dalším vždy skupina složená ze 3. a 4. znaku odpovídat 3. až 0. bitu. Nepoužité znaky a bity nahrazujeme znakem x. Obdobně 3. řádek přiřazuje 2. znak 4. bitu a čtvrtý a pátý řádek 1. znak 5. bitu. Zadání v programu např. ZAAW pak odpovídá binární kombinaci 000110. Současně lze zadat i binární kombinaci (obvykle nejčastěji se vyskytující), která se nastaví automaticky, není-li kód řídicí sběrnice zadán.

Zadání je vyděrováno na děrnou pásku jako vstup ① Tato děrná páska se programem kontroluje, zda neobsahuje formální chyby. Nejsou-li žádné chyby, uloží se zadání na magnetickou pásku (výstup A) a současně pro kontrolu vypíše (výstup B).

Zadání vlastního programu (vstup 2), které odpovídá specifikační části programu „Cromis" (příklad v *tab. 3*),

Tab. 3. Příklad zadání mikroprogramu

```
ARITMETICKE A LOGICKE OPERACE
 = 2. VARIANTA
FETCH: LMI (RO)     FF1;
   F1: NOP(RO)      RXDI;
   F2  LMM(AC)      RPDI;
  LAR:  ALR(T);
        LMI(AC);
        LTM(T)      RDDI;
  SHR: SHRT(T)      STC;
  SHL: MOVE(T);
        ADDE(T)     STC;
  ROR: SRA(T)       FFC STC;
  NIC: NOP(RO);
  CSR: ILR(R5);
        SDR(R6)     FF1;
        ILR(R4);
        SDR(R5)     FF1;
         IRL(R3);
```

se píše ve formě blízké zadání „Cromis". Jako první může být uveden libovolný text (označení programu). Poté následují jednotlivá slova mikroprogramu, počínající návěstím. Není-li návěstí uvedeno (např. u řetězů nepodmíněných skoků — v *tab. 3* např. za návěstím LAR), překládač doplní poslední zadané návěstí a pořadové číslo od posledně zadaného návěstí (za LAR tedy následuje LAR 1, LAR 2). V příkladu u návěstí SHR, SHL je uvedeno použití řetězcových kombinací. Není-li uvedena instrukce příznaku, dosadí se při překladu, stejně jako v „Cromis", HCZ resp. FFO, u pole *K* samé 0 nebo samé 1 podle mikroinstrukce CPE, u řídicí sběrnice zadaná hodnota. Celý program je zakončen znakem; na samostatném řádku. Zadání je opět děrováno do děrné pásky. Požadujeme-li v příslušné instrukci blokování hodin centrální jednotky (viz poznámka u pole CPE), doplníme kód mikroinstrukce CPE o znak P (např. místo TZR (T) píšeme TZRP (T).

Následující zpracování kontroluje formální chyby, seřadí zadání do unifikované formy a řetězové kombinace nahradí standardními kódovými označeními. Jsou-li v programu chyby, vypíše se celý program, chybná místa jsou označena znakem * a současně se na malé tiskárně kalkulátoru vypíše bližší specifikace chyb (výstup C).

Je-li zadání prosto formálních chyb, přejde kalkulátor k dalšímu zpracování. Kalkulátor si vyžádá vstup 3, což je děrná páska, obsahující zadání přiřazení slov mikroprogramu do pamět, mikroprogramu a zadání skoků (vstup 3). Toto zadání obsahuje v libovolném pořadí v každém řádku návěstí slova, uloženého v jedné buňce paměti, hexadecimální adresu této buňky, hexadecimální adresu prvního cíle skoku, kód skoku (místo nepodmíněných skoků JZR, JCC, JCR se použije jednotný kód JMP) a v závorce návěstí všech buněk, na něž je proveden skok, přitom pořadí musí odpovídat pořadí uložení v paměti). Požadujeme-li externí definici skoku (viz pozn. u pole JUMP), použijeme kódu JDI a neuvedeme žádný cíl skoku. Příklad takového zadání je v *tab. 4*.

Následující zpracování kontroluje opět formální chyby, zejména realizovatelnost skoků tak, jak byly zadány a vypíše chyby včetně podrobné identifikace (výstup D) a výpis mapy paměti (zadání obsahující chybu nevypíše).

Část výpisu mapy paměti je v *tab. 5*. Výpis je tvořen maticí 16 × 16 buněk, číslovaných hexadecimálně 0 až F v obou směrech. Na výpisu jsou vyznačeny oblasti, kam lze skákat podmíněnými skoky. V každé buňce je pak následující informace:

1. řádek — návěstí uloženého slova;
2. řádek — kód skoku uložené instrukce;
3. řádek — hexadecimální adresa buňky, ze které byl proveden skok na danou buňku. Je-li takových skoků více, je uveden znak *;
   hexadecimální adresa 1. cíle, na který je proveden skok z dané buňky. Pro kód JDI, kde je skok definován vnějším programem, je uveden znak *;
4. řádek — pořadové číslo slova mikroprogramu v zadání programu.

Jako pokračování výpisu se vypíše přehled všech definic skoků. Od zadání 3 se liší jen tím, že se nevypíše adresa 1. cíle a kód JMP je přeložen do příslušných konkrétních kódů.

V dalším program přeloží celé zadání (vstup 2 a 3) do binární formy a vypíše zadání programu v uspořádaném tvaru současně s binárním výpisem obsazení jednotlivých polí slova mikroinstrukce (výstup F). Část tohoto výpisu je jako příklad v *tab. 6*. Odleva počínaje je uvedeno pořadové číslo slova, návěští, mnemotechnické vyjádření programovaných mikroinstrukcí, kód skoku, hexadecimální adresa uložení slova v paměti mikroprogramu, binární obsah polí slova (A — adresa v paměti; F — pole CPE; — pole FI, FO, K; R — řídicí sběrnice; M — pole JMP). Současně se binární překlad uloží na magnetickou pásku (výst p 6).

Poslední etapa zpracování pak z magnetické pásky vyděruje děrnou pásku

Tab. 4. Příklad zadání přiřazení v paměti mikroprogramu

DIAR	20	60	JPR(NIC6, LDA, LAR, LDC, ADD, AND, IOR, XOR, NEG, SHRO, NIC1, NIC2, NIC3, NIC4, NIC5, STA)
INAR1	41	60	JPR(NIC6, LDA, LAR, LDC, ADD, AND, IOR, XOR, NEG, SHRO, NIC1, NIC2, NIC3, NIC4, NIC5, STA)
INJU2	48	50	JPR(JUN, JAS, JZE, JPL, JEV, JCZ, ISZ, DSZ, CSR, RSR, INP, RSA, NIC7, NIC8, NIC9, OUT)
INAR	21	41	JMP(INAR1)
INJU	29	49	JMP(INJU1)
INJU1	49	48	JMP(INJU2)
DIJU	28	48	JMP(INJU2)
INP	5A	4A	JMP(INP1)
INP1	4A	7A	JFL(NIC15, INP2)
NIC15	7A	5A	JMP(INP)
INP2	7B	7C	JMP(INP3)
INP3	7C	1A	JFL(NIC14, INP4)
SHRO	69	79	JMP(SHRO1)
NIC14	1A	5A	JMP(INP)

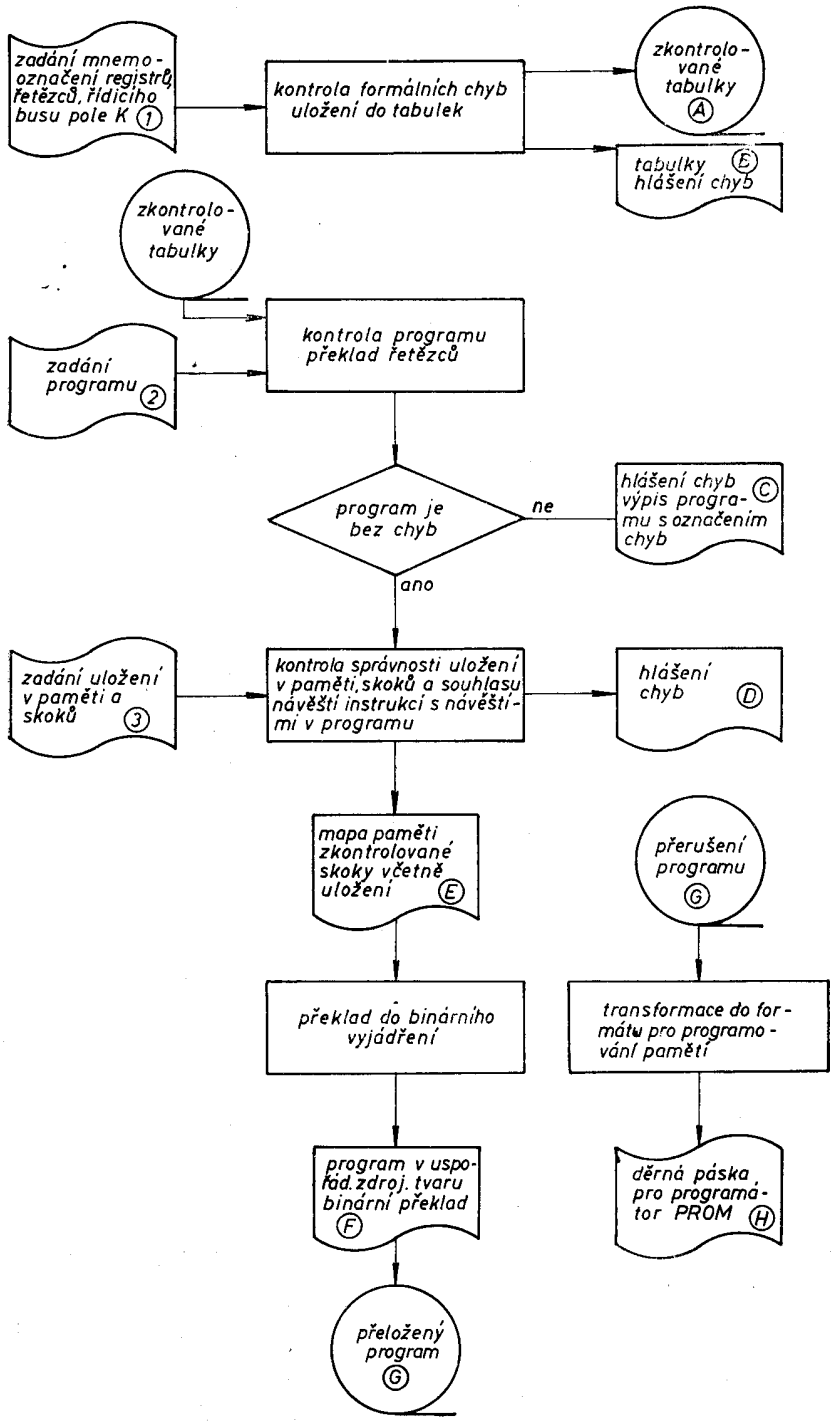
Tab. 5. Příklad části výpisu mapy uložení mikroprogramu v paměti mikroprocesoru

ARITMETICKE A LOGICKE OPERACE BIMBO — 2. VARIANTA — 3. NÁVRH 15/9 78:

	0	1	2	3	4	5	6	7
			+ JCF, JFL, JZF +		——————JLL——————+			
0	INI JCC 60 88				JUM JFL * 52 45			SZB JCC * 17 48
1	STA1 JZR 40 OF 7	LAR1 JCR 41 12 10	LAR2 JZR 11 OF 11		CSR6 JCR 15 16 59	CSR5 JCR 1D 14 58	CSR7 JCC 14 26 60	SZB1 JFL 07 22 49
2	INI9 JCC 21 70 97	INI8 JCR 81 20 96	SZB2 JZR 17 OF 50	SZB3 JZR 17 OF 51	CSR13 JZR 25 OF 66	CSR12 JCR 28 24 65	CSR8 JCR 16 27 61	CSR9 JCR 26 29 62
3	DIAR JPR 2F 40 4	INAR JPR 2F 40 5	T1 JZR 2F OF 102	T2 JZR 2F OF 103	T3 JZR 2F OF 104	T4 JZR 2F OF 105	T5 JZD * OF 106	T6 JZR 2F OF 107

Tab. 6. Úsek výpisu zkontrolovaného mikroprogramu a jeho binární funkce

30	ROR	:SRA	(T)	STC FFC			JCR (7B) 7E	0IIIIII0A	0000III0F	0II0IIIIK	00I0II0IR	00III0IIM
31	NIC	:NOP	(RO)				JZR (OF) 4B	0I00I0IIA	0II00000F	00IIIIIIK	00I0II0IR	00I0IIIIM
32	N1	:NOP	(RO)				JZR (OF) 4D	0I00II0IA	0II00000F	00IIIIIIK	00I0II0IR	00I0IIIIM
33	N2	:NOP	(RO)				JZR (OF) 4E	0I00III0A	0II00000F	00IIIIIIK	00I0II0IR	00I0IIIIM
34	N3	:NOP	(RO)				JZR (OF) 4F	0I00IIIIA	0II00000F	00IIIIIIK	00I0II0IR	00I0IIIIM
35	DIJU	:SDR	(R8)	FF1			JPR (50) 38	00III000A	00I0I000F	IIII0000K	00I0II0IR	0II00I0IM
36	INJU	:LMM	(AC)		RDDI		JCR (38) 39	00III00IA	000II0IIF	00IIIIIIK	00III00IR	00III000M
37	JUN	:SDR	(RO)	FF1			JZR (OF) 52	0I0I00I0A	00I00000F	IIII0000K	00I0II0IR	00I0IIIIM
38	JAS	:SDR	(RO)	FF1	HLDI		JZR (OF) 5B	0I0II0IIA	00I00000F	IIII0000K	00IIIII0R	I00I0IIIM
39	JCZ	:MOP	(RO)				JCF (52) 51	0I0I000IA	0II00000F	00IIIIIIK	00I0II0IR	0I0I0I0IM
40	ROTA	:LTM	(AC)		RDDI		JZR (OF) 4A	0I00I0I0A	0I0II0IIF	00II0000K	00III00IR	00I0IIIIM
41	SHLE	:LTM	(AC)		RDDI		JZR (OF) 49	0I00I00IA	0I0II0IIF	00II0000K	00III00IR	00I0IIIIM
42	JZE	:TZR	(T)				JZR (04) 54	0I0I0I00A	0I0III00F	00II0000K	00I0II0IR	00I00I00M
43	JPL	:TZRP	(T)			K 80	JZR (04) 55	0I0I0I0IA	II0III00F	00II0IIIIK	00I0II0IR	00I00I00M
44	JEV	:TZRP	(T)			K 01	JZR (04) 56	0I0I0II0A	II0III00F	00IIIII0K	00I0II0IR	00I00I00M
45	JUM	:NOP	(RO)				JFL (52) 04	00000I00A	0II00000F	00IIIIIIK	00I0II0IR	0I000I0IM
46	N4	:NOP	(RO)				JZR (OF) 53	0I0I00IIA	0II00000F	00IIIIIIK	00I0II0IR	00I0IIIIM
47	ISZ	:ACM	(AC)	FF1	RDDI		JZR (07) 57	0I0I0IIIA	0000I0IIF	IIIIIIIIK	00III00IR	00I00IIIM
48	SZB	:TZA	(AC)		WDKI		JCC (17) 07	00000IIIA	0I0IIIIIF	00II0000K	00I0000IR	0000000IM
49	SZB1	:NOP	(RO)				JFL (22) 17	000I0IIIA	0II00000F	00IIIIIIK	00I0II0IR	0I0000I0M
50	SZB2	:ILR	(RO)	FF1			JZR (OF) 22	00I000I0A	00000000F	IIIIIIIK	00I0II0IR	00I0IIIIM
51	SZB3	:NOP	(R0)				JZR (OF) 23	00I000IIA	0II00000F	00IIIIIIK	00I0II0IR	00I0IIIIM
52	DSZ	:LDI	(AC)		RDDM		JZR(07) 58	0I0II000A	00I0IIIIF	00II0000K	000II00IR	00I00IIIM
53	CSR	:ILR	(R5)				JCC (19) 59	0I0II00IA	00000I0IF	00IIIIIIK	00I0II0IR	000000IM



Obr. 1. Průběh činností při zpracování mikroprogramu na HP-9825

která slouží k přímému zavedení celého mikroprogramu do programátoru pamětí PROM nebo při vývojových pracech do pamětí RAM pro ověření navrženého programu. Lze přitom vytvořit děrnou pásku pro celý program nebo i pro libovolné úseky (potřebné zvláště při ověřovacích a vývojových pracech).

Srovnání programu pro HP-9825 a programu „Cromis"

Nejzávažnějším omezením programu pro HP-9825 je jeho omezení na paměť mikroprogramu do 256 slov. Pro úvodní práce a pro relativně jednodušší aplikace však toto omezení není příliš závažné. Firma Intel ve svých aplikačních materiálech uvádí příklad návrhu mikropočítače s dosti rozsáhlým souborem makroinstrukcí (36 registrových, 24 skokových, 6 instrukcí subrutin, 4 bitové, 4 vstupní-výstupní, 3 speciální), který je realizován pamětí do 256 slov. Obdobné jsou i zkušenosti z prvních prací, realizovaných v TESLA Rožnov. Kromě toho možnosti kalkulátoru HP-9825 nejsou ještě vyvinutým programem vyčerpány.

V srpnu 1979 byl TESLA Rožnov přihlášen ZN 502/79, jehož předmětem je program pro HP 9825, umožňující zpracování mikroprogramu do 2048 mikroinstrukčních slov, který obsahuje prakticky všechny možnosti programu CROMIS.

Některé příkazy programu „Cromis", jako úplná aplikace STRINGU, dále VALUE, IMPLY, možnost psát program jak s použitím návěstí, tak adres v libovolném tvaru (dekadicky, binárně, hexadecimálně), dále tatáž možnost při zadání obsahu pole usnadňují programování, zdaleka ne však v takovém rozsahu, aby je bylo nutno zařadit do zjednodušeného programu. Praktické zkušenosti ukázaly, alespoň v úvodních a relativně jednodušších aplikacích, že potřeba takových rozsáhlých možností je velmi vzácná a lze ji snadno obejít ponechanými jednoduššími prostředky.

Rovněž jsme nepoužili univerzální možnost zadání formátu konečného výsledku na výstupní děrné pásce pro programátor PROM. Uživatel má obvykle k dispozici velmi úzký sortiment zařízení pro programování, spíše však jen jedno. V tom případě je daleko efektivnější napsat jednoúčelový program pro přepis do výstupní děrné pásky. V našem případě je tento program dlouhý 12 řádek programu pro HP-9825 a jeho návrh si vyžádá mini-

MEZINÁRODNÍ VÝSTAVA TELECOM '79

Třetí výstava sdělovací techniky Telecom, která se z podnětu Mezinárodní telekomunikační unie (ITU) konala ve dnech 20. až 26. září 1979 v Ženevě, přinesla díky účasti předních světových výrobců mnoho zajímavých dokladů o dnešní pozoruhodné vyspělosti a trendech vývoje tohoto oboru. Exponáty vystavované v Ženevě byly důkazem neustále se prohlubující elektronizace nejrůznějších telekomunikačních přístrojů a digitalizace používaných přenosových technik, i toho, že tyto dva trendy již dnes plně ovlivnily koncepci sdělovacích systémů špičkových výrobců.

Např. GEC Telecommunications Lim. of England, která velmi úzce spolupracuje s britskou poštovní správou při přípravě na zavedení tzv. systému X (plně elektronické telefonní ústředny, číslicový přenos), předvedla na výstavě Telecom 1979 nové mikrovlnné radioreléové světlovodové spoje a hierarchický soubor systémů PCM s 30 kanály s rychlostmi 8, 34 a 140 Mbitů/s. Dvě pobočkové ústředny, využívající digitálního principu při spojování hovorů a programového řízení, byly příznačné pro funkční bohatost dnešních číslicových komunikačních soustav (např. jde o zkrácenou volbu, registraci příchozích a odchozích volání, automatickou signalizaci obsazené linky a disperzi — přesměrování příchozích volání na jiného účastníka, pokud se volaný nehlásí). Velmi perspektivní soustavou je nesporně Viewdata, umožňující zobrazit na obrazovce bytového televizoru libovolnou stránku z informačního magazínu, uloženého v paměti ústředního počítače. Jak ukázaly kancelářské i veřejné telefonní přístroje této firmy, přineslo řadu nových vlastností především zavedení mikroprocesorového řízení.

Obr. 1. Kabelový regenerační zesilovač

Těžištěm stánku společnosti Marconi Communication Systems Lim. byl její nový 30kanálový systém PCM, jehož výhodou je slučitelnost se stávající britskou telefonní soustavou i připravovaným systémem X. Velmi dobrou pověst si tento výrobce získal zejména v oboru mikrovlnných spojů, pracujících na základě rozptylu v troposféře (soustavy umožňující spojení s těžaři ropy v Severním moři) a pozemských stanic, udržujících spojení s telekomunikačními družicemi (v současné době je to dodávka zařízení pro jednu stanici západoevropského systému OTS s přenosem v pásmech 11/14 GHz a pro dvě stanice, které zprostředkují spojení s družicemi nové řady Intelsat). Pozornost v Ženevě budily i vysílače s výkonem 1 a 10 kW a nejrůznější telekomunikační přístroje této firmy.

Velkého pokroku bylo, jak ukázaly exponáty dalšího závodu koncernu Marconi-Marconi Instruments Lim., v poslední době dosaženo i v oblasti měřicí techniky. Nová, automaticky pracující zařízení, umožňují provádět rychlou a přitom komplexní diagnostiku analogových i číslicových sdělovacích soustav a přispívají tak ke zvýšení jejich spolehlivosti.

Zaslouženou pozornost budily na výstavě Telecom 1979 i výrobky firmy Eddystone Radio Lim., která je rovněž součástí stejného koncernu a specializuje se na komunikační přijímače, přístroje pro měření šumu v rozsahu 10 kHz až 1000 MHz a různá doplňková zařízení. Zejména to byl měřicí systém 40A, který splňuje náročné požadavky britské pošty a Zvláštního mezinárodního výboru pro vysokofrekvenční interference (CISPRI).

Výběr ze svého rozsáhlého výrobního sortimentu, sahajícího od kapesních radiostanic až po vysoce výkonné vysílače, od směrových spojů po družicové opakovače a od solárních generátorů po kompletní elektrické vybavení lokomotiv, lodí apod., předvedla v Ženevě firma AEG-Telefunken. Z jejich posledních úspěchů lze uvést např. přístrojovou techniku pro první číslicový směrový spoj instalovaný Spolkovou poštou NSR a krátkovlnné vysílače s výkonem 600 kW a středovlnné s výkonem 500 kW, pracující s modulací délkou impulsu. AEG-Telefunken je zatím monopolním dodavatelem těchto typů vysílačů, které mají ve srovnání s běžnými vysílači mnohem nižší energetické nároky a amortizují se proto již za několik málo let.

Stejný výrobce je úspěšný např. i v oboru komunikačních přijímačů pro pošty, zpravodajské agentury apod. Nejcharakterističtějším technickým znakem těchto zařízení (řada E 1500) je rozsah 10 kHz až 30 MHz a rozlišení syntezátoru 10 Hz. K novinkám předvedeným poprvé na zmíněné výstavě patřily i dva směrové spoje. První umožňuje přenos 960 popř. 1260 telefonních kanálů nebo televizního signálu se zvukovým doprovodem, druhý je číslicový spoj střední kapacity (1 televizní a 5 zvukových kanálů). Velký zájem návštěvníků budila v Ženevě i telekomunikační zařízení z přenosového systému družice OTS-1, která po nezdařeném startu bez úhony přežila její pád z výše asi 20 km. Přístroje fungují i nadále a nejsou na nich vidět deformace ani jiné změny.

Důstojné zastoupení měla na letošní výstavě Telecom i firma Siemens, která ukázala různé verze elektronického spojovacího systému s počítačovým řízením EWS a pobočkových ústředen řízených mikroprocesorem (EMS), nový systém pro měření na účastnických vedeních SULIM, stolní a mincovní telefonní přístroje s řadou nových funkcí i mnoho jiných zařízení. U kabelových přenosových systémů se, stejně jako u směrových spojů, velmi důrazně prosazuje číslicová technika přenosu. Firma Siemens rozšířila

Obr. 2. Bezšňůrový telefonní přístroj

v nedávné době svůj výrobní program natolik, že může pro regionální i celostátní sítě nabídnout soustavy pracující s různým tokem informací v několika frekvenčních pásmech, mj. i radioreléové zařízení pro číslicový přenos v regionálních sítích.

Technologický rozvoj vytvořil v posledních letech předpoklady i pro výrobu různě výkonných soustav, umožňujících přenos informací světlovody. Siemens v tomto oboru úspěšně realizoval např. optický systém pro přenos 480 telefonních hovorů, jehož součástí je i regenerační zesilovač, pracují rychlostí 34 Mbity/s. Tento přístroj v provedení pro uložení pod zemským povrchem je znázorněn na *obr. 1.*

K nejzajímavějším novinkám, předvedeným touto západoněmeckou firmou poprvé v Ženevě, patřil nesporně bezšňůrový ředitelský telefonní přístroj, u něhož je pro dvoukanálový přenos řídicích signálů a vlastního hovoru mezi stacionárním dílem, instalovaným na zdi, a leh-

(Dokončení ze str. 448)

mum práce (1 až 3 hodiny včetně odladění).

Článek popisuje vlastnosti programových prostředků pro urychlení a kontrolu návrhu mikroprogramu mikropočítačů, sestavených z mikroprocesorových elementů řady Intel 13000. Úvodem popisuje vlastnosti programu „Cromis", nabízeného firmou Intel. Dále popisuje základní vlastnosti zjednodušené ověřené varianty, implantovatelné na stolním kalkulátoru HP-9825A.

LITERATURA

[1] Gucký, T., ing.; Prvky mikroprocesorového systému řady 3000, Sdělovací technika 8/1979, str. 279.

[2] Gucký, T., ing.; Aplikace mikroprocesorových prvků řady 3000, Sdělovací technika 10/1979, str. 362

[3] Preliminary Specification Intel Series 3000 Cross Microprogramming System CROMIS, Revision A