VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

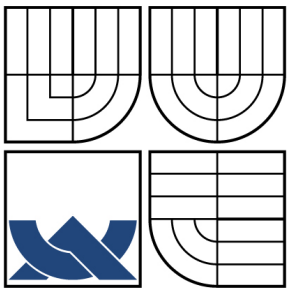
FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

NÁVRH A REALIZACE UČEBNÍ POMŮCKY „LOGIK“

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

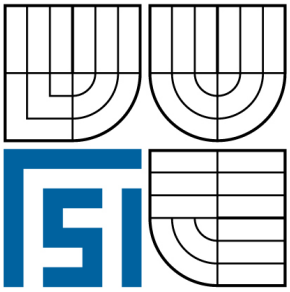
AUTOR PRÁCE PAVEL PUČEGL
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA STROJNÍHO INŽENÝRSTVÍ

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

NÁVRH A REALIZACE UČEBNÍ POMŮCKY „LOGIK“

DESIGN AND REALIZATION OF TEACHING INSTRUMENT "LOGIC"

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — PAVEL PUČEGL

AUTHOR

VEDOUCÍ PRÁCE — Ing. TOMÁŠ MARADA, Ph.D.

SUPERVISOR

BRNO 2008

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav mechaniky těles, mechatroniky a biomechaniky
Akademický rok: 2007/08

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Pučegl Pavel

který/která studuje v **bakalářském studijním programu**

obor: **Mechatronika (3906R001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Návrh a realizace učební pomůcky „LOGIK“

v anglickém jazyce:

Design and realization of teaching instrument "LOGIC"

Stručná charakteristika problematiky úkolu:

Učební pomůcka „LOGIK“ je stavebnice sloužící k podpoře výuky předmětu „Logické řízení a programovatelné automaty“. Umožní praktické ověřování teoreticky navržených úloh pomocí kombinačních a sekvenčních logických obvodů. V podstatě se bude jednat o mikroprocesor s mnoha vstupy a výstupy, které se budou na základě programu chovat jako jednoduchá hradla NOT, OR, AND, NOR, NAND atd..
Cílem této práce je provést návrh a realizaci učební pomůcky „LOGIK“ pomocí vhodného mikro-kontroléru a přizpůsobovacích obvodů tak, aby bylo možno na panelu realizovat zapojení kombinační a sekvenčních logických obvodů. Dále je cílem provést návrh několika laboratorních úloh, vhodných k zapojení na panelu.

Cíle bakalářské práce:

1. Seznamte se stávajícími panely používanými pro výuku a s laboratorníma úlohami které se na nich realizují.
2. S použitím předepsané literatury a souvisejících informací na Internetu proveďte návrh budoucího panelu, jak z hlediska počtu a typu hradel tak z hlediska přehlednosti rozmístění jednotlivých logických obvodů.
3. Realizujte navržený logický panel.
4. Proveďte návrh několika laboratorních úloh a po dohodě se školitelem tyto úlohy realizujte a ověřte jejich funkčnost.
5. Ke každé úloze vypracujte podrobný popis, dále popis možných základních modifikací a sestavte pořadí obtížnosti.

Seznam odborné literatury:

[1] http://www.hw.cz/
[2] http://www.kpsec.freeuk.com/gates.htm
[3] http://www.fm.vslib.cz/~kes/data/rs_ko.pdf

Vedoucí bakalářské práce: Ing. Tomáš Marada, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2007/08.

V Brně, dne

L.S.

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA STROJNÍHO INŽENÝRSTVÍ DĚKANÁT

prof. Ing. Jindřich Petruška, CSc.
Ředitel ústavu

doc. RNDr. Miroslav Doupovec, CSc.
Děkan fakulty

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Pavel Pučegl

Bytem:

Narozen/a (datum a místo): 4.10.1983, České Budějovice

(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta strojního inženýrství

se sídlem Technická 2896/2, 61669 FSI VUT v Brně

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

prof. Ing. Jindřich Petruška, CSc.

(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☐ diplomová práce
   - ☑ bakalářská práce

   jiná práce, jejíž druh je specifikován jako .....

(dále jen VŠKP nebo dílo)

Název VŠKP: Návrh a realizace učební pomůcky „LOGIK“

Vedoucí/školitel VŠKP: Ing. Tomáš Marada, Ph.D.

Ústav: Ústav mechaniky těles, mechatroniky a biomechaniky

Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

- ☑ tištěné formě - počet exemplářů 2
- ☐ elektronické formě - počet exemplářů .....

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☑ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

PODĚKOVÁNÍ

Na tomto místě bych rád poděkoval všem, kteří mi poskytli svůj čas a svými cennými radami přispěli k vypracování této bakalářské práce. Především pak vedoucímu mé bakalářské práce panu Ing. Tomáši Maradovi Ph.D.

Dále bych rád poděkoval svým rodičům a svému zaměstnavateli, kteří mě po celou dobu studia podporovali.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že předložená bakalářská práce je mou vlastní autorskou prací, kterou jsem vypracoval pod vedením vedoucího mé diplomové práce a s použitím literatury uvedené v seznamu literatury.

V Brně dne 23.5.2008 Pavel Pučegl

.....

ANOTACE

Postupné dosluhování a nepříliš velká spokojenost s provedením stávajících panelů pro výuku předmětu „Logické řízení a programovatelné automaty“ vedlo Ústav automatizace a informatiky Vysokého učení technického v Brně k rozhodnutí ke zkonstruování nové učební pomůcky „LOGIK“.

Tato pomůcka by měla pomoci při praktickém ověřování znalostí nabytých při výuce předmětu. Studenti by měli mít možnost na této pomůcce realizovat zapojení úloh pomocí kombinačních a sekvenčních logických obvodů.

Cílem této práce je návrh a následná realizace této učební pomůcky tak, aby mohla být v budoucnu skutečně použita při výuce.

ANNOTATION

Gradual wear and growing dissatisfaction with current panels for purposes of teaching the course „Logical control and programmable controllers“ has lead the Institute of Automation and Computer Science at Brno University of Technology to a decision to establish a new educational tool „LOGIK“.

The tool is intended to facilitate practical verification of knowledge gained by students during this course. Students will have access to this device and will carry out various practical projects concerning combinational and sequential circuits.

The general objective of this thesis is to design and subsequently construct this educational device so that is could be fully utilized during the course in near future.

OBSAH

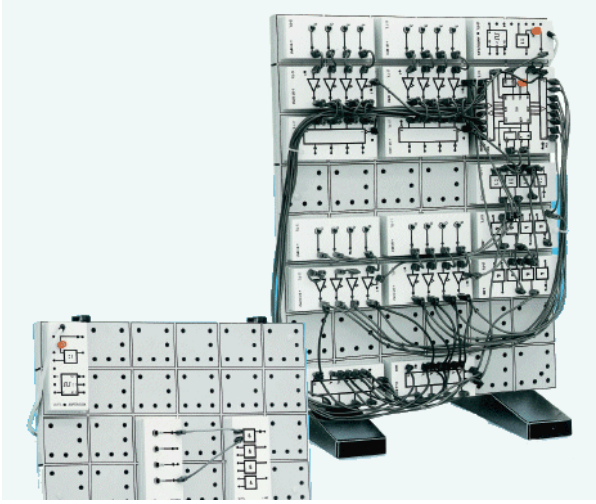
1.ÚVOD

Učební pomůcka „LOGIK“ slouží k procvičení úloh číslicové techniky. Jedná se o zařízení obsahující elektronické obvody, které simulují chování základních funkcí dvoustavové logiky, jejichž vzájemným pospojováním umožňují simulaci chování složitějších logických funkcí.

Podle vnějšího provedení a zapojení elektroniky můžeme učební pomůcky rozdělit.

1.1 Modulové učební pomůcky

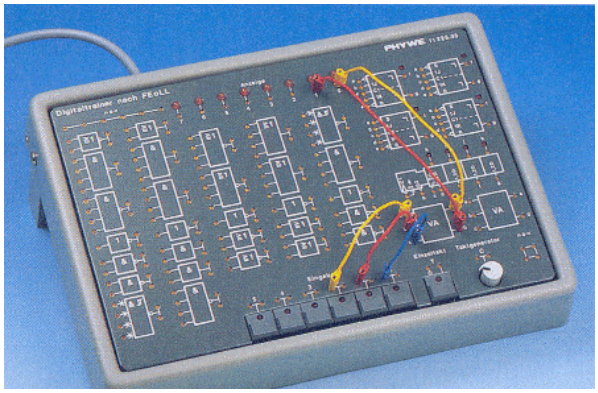
Modulové učební pomůcky se skládají z několika částí (modulů). Na každém z modulů je realizována jedna nebo několik příbuzných funkcí a zapojení úlohy tak probíhá na několika modulech současně. Výhodou tohoto provedení je možnost poskládat si k sobě v podstatě nekonečný počet modulů s různými funkcemi a tím realizovat neomezeně rozsáhlou a složitou úlohu. Nevýhodou je většinou podstatně vyšší cena, problém s nutností napájet každý panel a při rozsáhlejších úlohách větší prostorová náročnost a tím i nepřehlednost zapojení.



Obr. 1.: Příklad modulové učební pomůcky

1.2 Panelové učební pomůcky

Panelové učební pomůcky jsou celé realizované na jednom panelu. Většina pomůcek obsahuje pouze jedno rozložení logických funkcí, některé však umožňují přepínání mezi několika rozloženími. Výhodou tohoto provedení je menší prostorová náročnost, větší přehlednost a vyšší odolnost vůči hrubému zacházení uživatele. Hlavní nevýhodou je omezenost počtu logických funkcí na panelu.



Obr. 2.: Příklad panelové učební pomůcky

1.3 Učební pomůcky realizované pomocí hradel TTL

Logické funkce na těchto pomůckách jsou realizovány pomocí logických hradel TTL řady 74XX. Výhodou tohoto zapojení je jeho jednoduchost, vysoká spolehlivost a odolnost, která však v dnešní době u nejrozšířenějších hradel HC a HCT není tak vysoká. Mezi hlavní nevýhody patří vcelku veliký počet součástek použitých uvnitř pomůcky, což dosti znesnadňuje případné opravy, a prakticky nemožnost realizace více rozložení logických funkcí na panelu a jejich následné přepínání.

1.4 Učební pomůcky realizované pomocí mikroprocesoru

O výpočty logických funkcí v těchto pomůckách se stará mikroprocesor, který pomocí multiplexerů a demultiplexerů přepíná mezi jednotlivými vstupy a výstupy na pomůcce. Výhodou tohoto provedení je možnost realizace několika rozložení logických funkcí na pomůcce, menší počet součástek použitých pro realizaci a možnost vytvoření dostatečné odolnosti vstupů/výstupů. Hlavní nevýhodou je větší náchylnost mikroprocesoru na případné nedokonalosti výpočtového programu a tím způsobená zacyklení a následná nefunkčnost pomůcky.

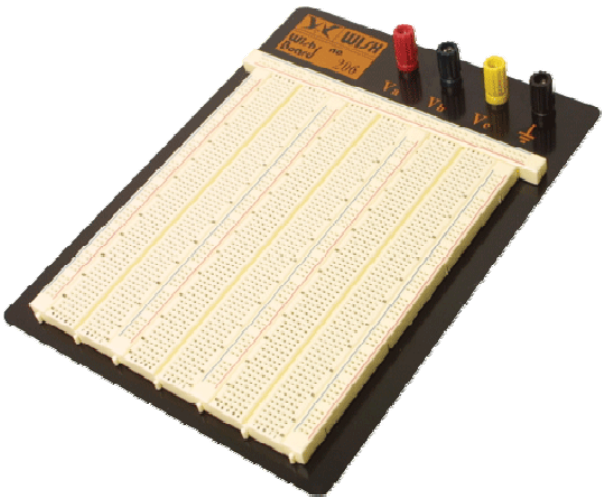
2. ZHODNOCENÍ SOUČASNÉHO STAVU V OBLASTI UČEBNÍCH POMŮCEK PRO VÝUKU ČÍSLICOVÉ TECHNIKY

V dnešní době se na trhu vyskytuje několik pomůcek pro výuku číslicové techniky. Mezi nejvýznamnější zástupce patří DOMINOPUTER a LOGO!. DOMINOPUTER je typický případ modulárního systému, LOGO! pak panelového.

Velmi vysoká cena těchto pomůcek však mnoho škol a jiných organizací využívajících takovéto výukové pomůcky vede k realizaci logických funkcí pomocí TTL hradel řady 74XX na nepájivém kontaktním poli nebo k návrhu a následné výrobě vlastních pomůcek, což je i případ našeho ústavu.

2.1 Realizace logických funkcí na nepájivém kontaktním poli

Realizace logických funkcí pomocí TTL hradel řady 74XX na nepájivém kontaktním poli je sice konstrukčně nejjednodušší, avšak dle mého názoru nepříliš názorná. Vyučovaný přímo nevidí symboly jednotlivých logických funkcí a zapojuje podle popisu zapojení integrovaného obvodu. Toto činí výuku silně nepřehlednou. Navíc dochází k problémům s kontakty propojovacích drátků, což zpravidla působí dost velké problémy s hledáním chyb v zapojení. Tento způsob výuky však vede vyučované k větší praktičnosti a schopnosti použít logické funkce realizované pomocí hradel TTL 74XX v praxi.



Obr. 3.: Nepájivé kontaktní pole

2.2 Výroba vlastních pomůcek pro výuku číslicové techniky

K výrobě vlastních pomůcek, jak již bylo zmíněno, vede školy a organizace nejčastěji značná finanční náročnost nákupu profesionálně vyráběných pomůcek. Při výrobě vlastní učební pomůcky má organizace výhodu, že se pomůcka navrhne a následně vyrobí na míru potřebám výuky v dané organizaci. Lze přizpůsobit rozměr, odolnost, počet a typ logických funkcí obsažených v pomůcce. Toto též činí vlastní výrobu učební pomůcky dost výhodnou. Nevýhodou však je relativní náročnost návrhu a výroby.

3. NÁVRH UČEBNÍ POMŮCKY

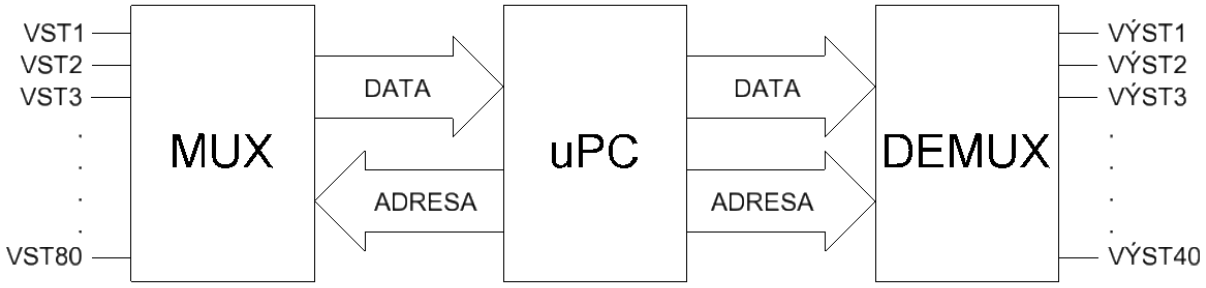
3.1 Návrh konstrukce pomůcky, počtu, typu a rozmístění hradel

Zhodnocením zkušeností s dosavadními panely, s výukou na stávajících panelech, finanční náročností a přihlédnutím ke konstrukci profesionálně vyráběných pomůcek vyplynula konstrukce pomůcky jako panelové s elektronickým obvodem realizovaným pomocí mikroprocesoru.

Dle druhu, náročnosti a rozsahu laboratorních úloh realizovaných v budoucnu na navrhované učební pomůcce bylo navrženo základní uspořádání konektorů, spínačů a indikačních LED diod na panelu a k tomuto základnímu uspořádání byla navržena tři rozmístění logických funkcí (hradel). Výkres základního rozmístění lze nalézt v příloze jako list P1, rozmístění hradel jako listy P2, P3 a P4.

3.2 Návrh hardwarové části

Jak již bylo zmíněno, základní část učební pomůcky tvoří mikroprocesor. Mikroprocesor zajišťuje veškeré výpočty logických funkcí. Se vstupy/výstupy na panelu komunikuje pomocí multiplexerů/demultiplexerů. Blokové schéma elektronické části je zobrazeno na obr. 4.



Obr. 4.: Blokové schéma elektronické části

Kompletní schéma zapojení obvodu, návrh desky plošných spojů včetně osazovacího schématu a schématu zapojení konektorů vstupů a výstupů se nacházejí v příloze jako listy P5 – P9.

3.2.1 Mikroprocesor

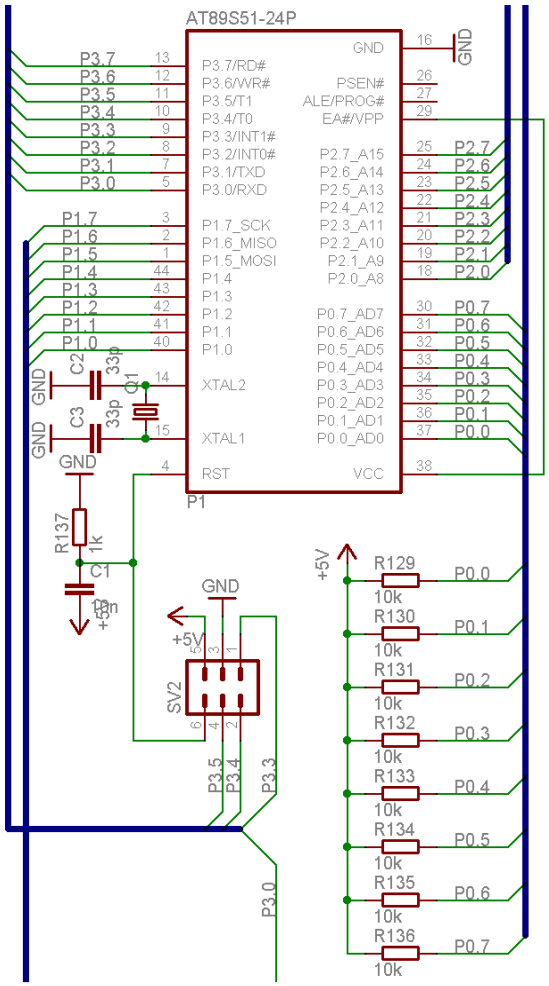
Vzhledem k nabitým znalostem ze střední školy, z tím plynoucí sympatizace s osmibitovými mikroprocesory Intel řady 8051 a z ní plynoucí řady 8052 jsem k návrhu použil jednoho z nejlevnějších zástupců této řady mikroprocesorů obvod AT 89C52 od firmy ATMEL.

Stručná charakteristika procesoru ATMEL 89C52:
8 bitový mikroprocesor s kompletní instrukční sadou (CISC)
Napájecí napětí 4 - 5.5V
Frekvence oscilátoru závisí dle konkrétního typu, univerzální rozmezí pro všechny používané typy je 4 - 24MHz
Velikost paměti dat (RAM) 256 x 8bitů
Velikost paměti programu 8kB organizovaných po 8 bitech
Počet programovatelných vstupů/výstupů 32 se zatižitelností 8 vstupů TTL-LS obvodů v případě brány P0 a 4 TTL-LS v případě bran P1 - P3.

Ze základních charakteristik je patrné, že zvolený mikroprocesor svými parametry bez problémů splňuje požadavky dané pomůcky.

Zapojení mikroprocesoru

Zapojení mikroprocesoru je znázorněno na obrázku 5.



Obr. 5.: Zapojení mikroprocesoru

Pro časování mikroprocesoru bylo použito standardního zapojení krystalového rezonátoru se dvěma kondenzátory o velikosti 33pF. Oscilační frekvence krystalu byla vzhledem k minimální rychlostní náročnosti aplikace zvolena na spodní hranici taktu mikroprocesoru, tj. 4 MHz.

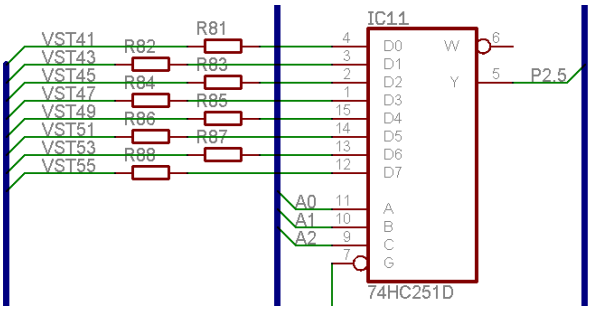
Reset mikroprocesoru zajišťuje derivační RC článek s časovou konstantou $\tau = R \cdot C = 10\ \mu s$. Vzhledem k době trvání strojového cyklu 3 µs tento článek zajistí výrobcem požadovanou dobu 2 strojových cyklů pro správné počáteční nastavení SFR registrů a čítače instrukcí. Vstup resetovacího obvodu je vyveden na konektor pro připojení tlačítka RESET na čelní straně přístroje.

Na stejném konektoru jsou vyvedeny též vstupy bran P3.3, P3.4 a P3.5 pro připojení přepínače volby rozvržení součástek na panelu.

Brána P0 může sloužit pro připojení externí paměti programu. Z tohoto důvodu jsou její výstupy zapojeny s otevřeným kolektorem. V případě použití brány P0 jako výstupní je nutno připojit vnější pull-up rezistory. Tuto funkci zajišťují rezistory R129 – R136.

3.2.2 Multiplexer

Multiplexer slouží k rozšíření počtu vstupů. Mívá zpravidla několik datových vstupů, jeden výstup a několik adresových vstupů, jejichž počet je odvislý od počtu datových vstupů. Funguje tak, že po navolení adresy patřičného datového vstupu na adresových vstupech vytvoří na svém výstupu hodnotu odpovídající hodnotě vstupu na patřičné adrese.

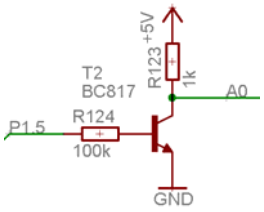


Obr. 6.: Zapojení multiplexeru

Jako multiplexeru bylo v našem zapojení použito obvodů 74HC251. Jedná se o 8 vstupové TTL multiplexery. Z počtu vstupů (80) jasně vyplývá počet použitých obvodů tj. 10. Vstupy jednotlivých multiplexerů jsou připojeny přes ochranné rezistory přímo na vstupní konektory na panelu. Ochranné rezistory mají za úkol omezit případný zkratový proud při neodborném zacházení s panelem. Byla zvolena hodnota maximálního proudu na 1mA, čemuž při pracovním napětí panelu 5V odpovídá hodnota odporu rezistorů 5kΩ. Budeme li vycházet z řady E12, pak je to 4,7kΩ.

Výstupy multiplexerů jsou připojeny každý na jeden bit brány P2, zbylé dva pak na bity brány P3, konkrétně P3.6 a P3.7.

Adresové vstupy multiplexerů jsou navzájem spojeny a připojeny na bránu P1, bity P1.5, P1.5 a P1.7. Vzhledem k počtu obvodů (10), typu (HC-TTL) a zatižitelnosti brány P1 (4 vstupy LS-TTL) nejsou adresové vstupy připojeny na bránu přímo, nýbrž přes tranzistorový zesilovač.



Obr. 7.: Zapojení zesilovače adresových signálů multiplexerů

Bylo použito jednoduchého zesilovače – zapojení NPN tranzistoru se společným emitorem. Je potřeba si uvědomit, že se jedná o invertující zesilovač, tudíž skutečná adresa posílaná multiplexerům bude proti adrese vysílanou programem invertovaná a s tímto počítat v programu pro mikroprocesor.

Proud protékající tranzistorem volíme na cca 5mA tj. odpor v kolektoru tranzistoru na 1kΩ, tím by mělo být dostatečně zajištěno správné zachycení logických úrovní „0“ a „1“ adresovými vstupy multiplexerů. Proudový zesilovací činitel tranzistoru je okolo 300, požadujeme, aby tranzistor byl zcela otevřený, tj. v saturaci, budeme tedy počítat se zesílením 100. Bázový proud potřebný pro otevření tranzistoru bude potom

$$I_B = \frac{I_C}{h_{21E}} = \frac{5}{100} = 0{,}05mA \qquad (1)$$

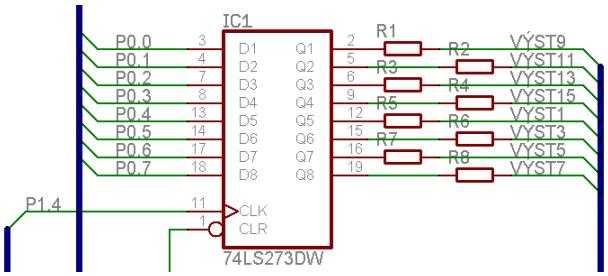
a z toho vyplývající odpor v bázi tranzistoru

$$R_B = \frac{U_{uPC} - 0.6}{I_B} = \frac{5 - 0.6}{0.05} \doteq 100k\Omega \qquad (2)$$

3.2.3 Demultiplexer

Demultiplexer slouží k rozšíření počtu výstupů. Mívá zpravidla jeden datový vstup, několik výstupů a několik adresových vstupů, jejichž počet je opět závislý na počtu výstupů.

Funguje tak, že po navolení adresy patřičného výstupu na adresových vstupech vytvoří na tomto výstupu hodnotu odpovídající hodnotě na svém vstupu.



Obr. 8.: Zapojení demultiplexeru

Jako demultiplexeru bylo v našem zapojení použito obvodů 74LS273. Jedná se o 8-bitové D klopné obvody. Tyto obvody samy o sobě nejsou demultiplexery, funkci demultiplexeru jsou však po vhodném zapojení plnit. Vstupy všech D klopných obvodů jsou navzájem spojeny a připojeny na bránu P0. Vhledem k tomu, že obvodů je jen 5 a zatižitelnost brány P0 je 8 vstupů TTL-LS, můžeme je přímo zapojit bez problémů. Z tohoto důvodu také byly použity odvody TTL-LS se sníženou spotřebou.

Brána P0 však ve svém vnitřním zapojení postrádá odpor v kolektoru výstupního tranzistoru, tudíž není schopna při použití jako výstupní reprezentovat log. „1“. Musíme proto tento odpor realizovat vnějším zapojením. Z toho důvodu byly na bránu P0 připojeny rezistory s odporem 10kΩ připojené svým druhým koncem na napájecí napětí +5V. Tyto rezistory nám již zaručí správnost hodnot na výstupu brány P0.

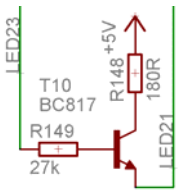
Výstupy všech D klopných obvodů ve všech obvodech 74LS273 jsou připojeny přes ochranné rezistory (opět 4k7) na výstupní konektory na panelu.

Adresa demultiplexerů v tomto případě není realizována několikabitovou kombinací jako v případě multiplexerů avšak je realizována jednobitově. Počet vstupů/výstupů mikroprocesoru to umožňuje a je to tak jednodušší. Celý demultiplexer tedy funguje tak, že z brány P0 se na všechny D klopné obvody pošle hodnota 8 bitů, kterou chceme promítnout na patřičný výstup a poté se provede takt hodinového signálu D klopného obvodu ale nyní již jen jednoho příslušného podle toho, kam chceme dané bity poslat. Tím se dané hodnoty zobrazí na výstupu D klopných obvodů a hodnoty zbylých obvodů zůstanou nezměněny. Adresu tedy určují hodinové vstupy D klopných obvodů a jsou proto připojeny na bity P1.0 – P1.4 brány P1.

3.2.4 Zobrazovací LED diody

Indikace logické úrovně je tvořena pomocí LED diod. Proud potřebný k maximálnímu svitu LED diody je 20mA. Takovýto proud nám obvody 74LS273 realizující výstupy hradel nejsou schopny dát, jsou proto na desce elektronického obvodu navrženy zesilovače pro LED diody, zajišťující jejich dostatečný svit.

Jako zesilovačů bylo opět použito NPN tranzistorů v zapojení se společným emitorem a LED diodami v obvodech emitorů.



Obr. 9.: Zapojení zesilovače indikačních LED diod

Po zahrnutí úbytku napětí na LED diodě a na tranzistoru zbyde na kolektorovém odporu napětí cca 3,5V. Proud potřebný pro požadovaný svit LED diod jak již bylo zmíněno je 20mA. Z toho vyplývá velikost odporu kolektorového rezistoru.

$$R_C = \frac{U_C}{I_C} = \frac{3,5}{0,02} \doteq 180\Omega \tag{3}$$

Při uvažování proudového zesilovacího činitele tranzistoru opět 100, bude velikost bázového proudu rovna

$$I_B = \frac{I_C}{h_{21E}} = \frac{20}{100} = 0,2mA \tag{4}$$

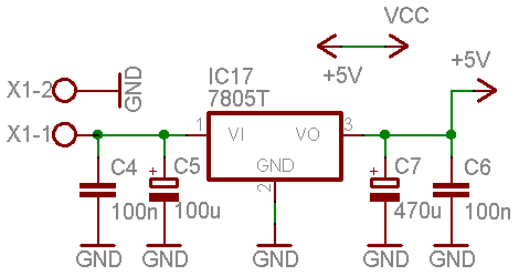
Z toho vyplývá velikost oporu bázového rezistoru

$$R_B = \frac{U_B}{I_B} = \frac{3,5}{0,2} \doteq 27k\Omega \tag{5}$$

3.2.5 Napájení panelu

Napájení panelu je zajištěno síťovým napáječem 9V, 1A, zakoupeným jako finální výrobek. Napětí 9V je na pracovní napětí 5V upraveno stabilizátorem 7805. Tento stabilizátor je schopen přenést proud 1A, což je dostatečné pro celý obvod a maximální zatížení všech vstupů a výstupů na panelu. Bylo použito běžného katalogového zapojení obvodu 7805 s dvěma elektrolytickými a dvěma keramickými filtračními kondenzátory co nejblíže pouzdru obvodu. Stabilizátor je chlazen chladičem V4330N ze sortimentu GM electronic. Tento chladič má tepelnou konstantu 12K/W. Z toho vyplývá, že je schopen při napájecím napětí 9V a odebíraném maximálním proudu tj. 1A a pokojové teplotě 20°C uchladit pouzdro obvodu na 68°C, což nepřesahuje maximální povolenou pracovní teplotu stabilizátoru udávanou výrobcem.

Z výše uvedeného vyplývá, že při použití jiného zdroje napájení je hodnota 9V z hlediska chlazení stabilizátoru horní hraniční, v případě spodní hraniční teploty lze počítat napětí stabilizátoru zvýšené o úbytek na koncovém tranzistoru stabilizátoru. Panel lze tedy napájet stejnosměrným napětím v rozmezí 6 - 9V.



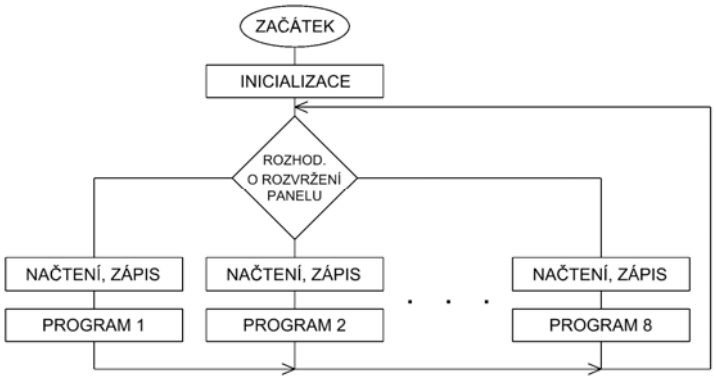
Obr. 10.: Zapojení stabilizátoru napětí

3.3 Návrh softwarové části

Program mikroprocesoru plní čtyři základní funkce:

- rozhodnutí o rozmístění hradel zvoleného přepínači na boku panelu
- načtení hodnot vstupů z konektorů na panelu
- výpočet hodnot logických funkcí představovaných hradly
- zápis hodnot výsledků logických funkcí na výstupy

Z toho vyplývá základní vývojový diagram programu, který zobrazuje obrázek 11.

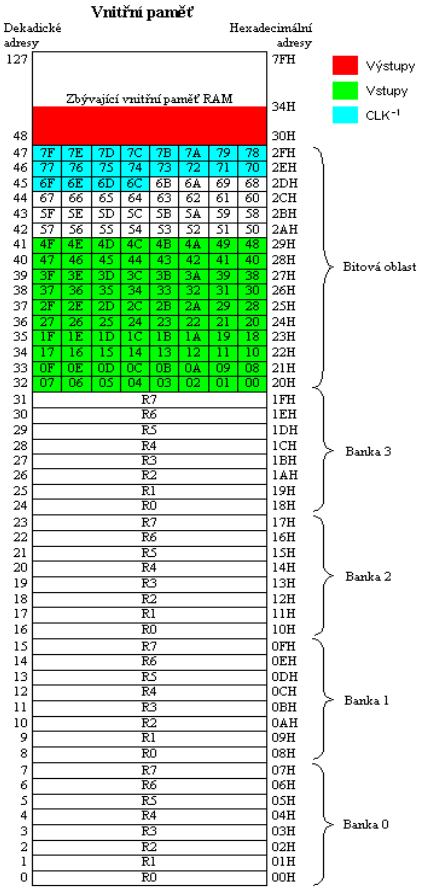


Obr. 11.: Vývojový diagram programu mikroprocesoru

3.3.1 Organizace paměti

Po resetu je ukazatel zásobníku (SP) nastaven těsně nad první banku registrů (RB1) tj. na adresu 07h. Vzhledem k tomu, že nebudeme potřebovat využívat další banky registrů, můžeme ukazatel na této adrese ponechat. Začátek využívané paměti zvolíme na adrese 20h. Tím nám zbyde 25 paměťových míst zásobníku, tj. lze do něho uložit 12 šestnáctibitových absolutních adres. Lze tedy využít až 12 násobně vnořené volání.

Od adresy 20h umístíme hodnoty vstupů postupně až do adresy 29h. Hodnoty výstupů pro jednoduchost umístíme až od adresy 30h do adresy 34h. Mezi paměť vstupů a paměť výstupů (bitově adresovatelná oblast) umístíme paměť předchozích hodnot hodinových signálů CLK^{-1}. Umístění vstupů, výstupů a předchozích hodnot hodinových signálů ve vztahu ke vstupům a výstupům na panelu je znázorněno v příloze na listu P10. Rozmístění v paměti RAM pak na obrázku 12.



Obr. 12.: Umístění dat v paměti RAM

3.3.2 Inicializace

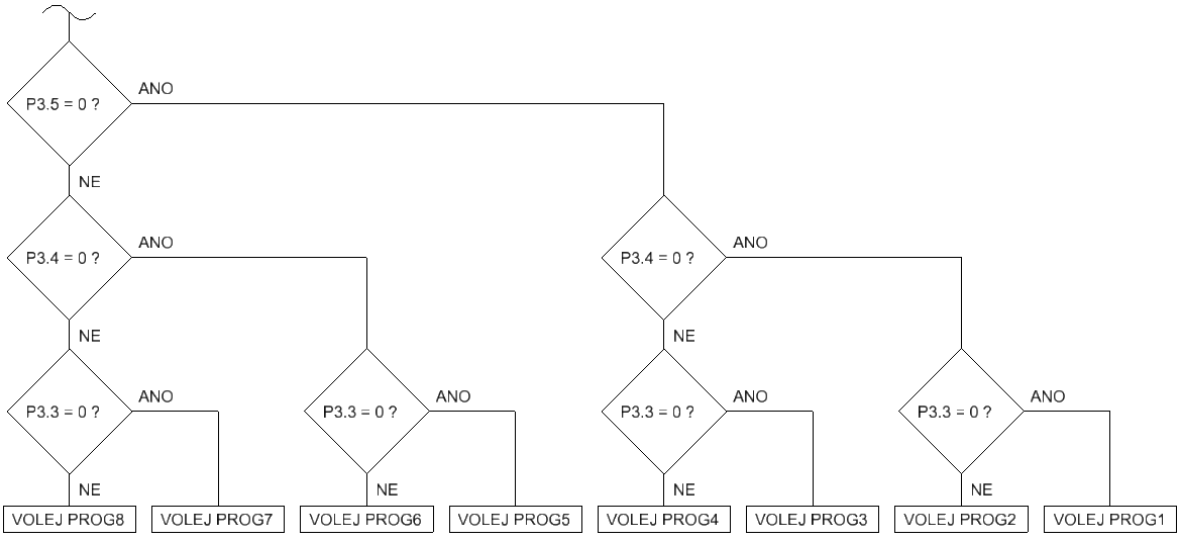
V průběhu inicializace je potřeba vymazat brány, vymazat demultiplexory (D klopné obvody) a nastavit prvotní hodnoty výstupů sekvenčních logických obvodů. Zdrojový kód inicializace bude v jazyce symbolických adres (assembleru) vypadat:

```
        mov P1,#0h              ;vymazani bran
        mov P2,#0FFh
        mov P3,#11111010b       ;reset DKO
        nop
        setb P3.0

        mov 30h,#01010101b      ;nastaveni prvotnich hodnot vystupu
        mov 31h,#01010101b      ;eventuelnich SLO, v pripade KLO nema
        mov 32h,#01010101b      ;tato hodnota vliv, u KLO neni pred.
        mov 33h,#01010101b      ;hodnota podstatna
        mov 34h,#01010101b
```

3.3.3 Rozhodování o rozvržení panelu

Rozhodování probíhá ze tří bitů na bráně P3, konkrétně bitů P3.3, P3.4 a P3.5, které jsou spojeny s přepínači pro volbu programu na čelní straně přístroje. Podle zvolené kombinace se spustí patřičný výpočtový program pro dané rozložení logických funkcí na panelu. Vývojový diagram rozhodování o rozložení logických funkcí na panelu znázorňuje obrázek č. 13.



Obr. 13.: Vývojový diagram rozhodování o rozvržení hradel na panelu

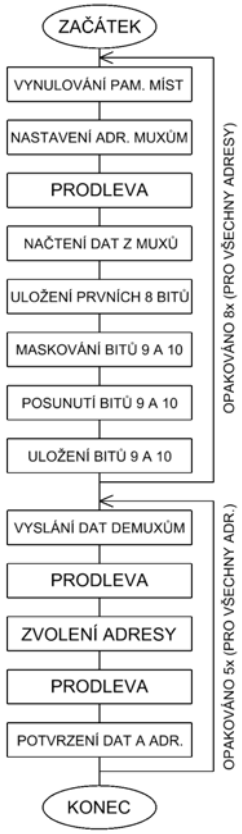
Zdrojový kód programu bude potom podle vývojového diagramu vypadat následovně:

```
                                ;rozpoznani programu z prepinacu
ZAC:        jnb P3.5,PR1234     ;?nejvyssi bit=0 -> program 1-4
            jnb P3.4,PR56       ;program 5-8,?druhy bit=0 -> prog.5-6
            jnb P3.3,PR7        ;program 7 a 8,?treti bit=0 -> prog.7
            call PROG8          ;zbyva program 8, volani programu 8
            sjmp ZAC            ;program 8 provedem -> zacatek
PR7:        call PROG7          ;volani programu 7
            sjmp ZAC
PR56:       jnb P3.3,PR5        ;program 5 a 6,?treti bit=0 -> prog.5
            call PROG6          ;zbyva program 6, volani programu 6
            sjmp ZAC
PR5:        call PROG5          ;volani programu 5
```

```
            sjmp ZAC
PR1234:     jnb P3.4,PR12           ;program 1-4,?druhy bit=0 -> prog.1-2
            jnb P3.3,PR3            ;program 3 a 4,?treti bit=0 -> prog.3
            call PROG4              ;zbyva program 4, volani programu 4
            sjmp ZAC
PR3:        call PROG3              ;volani programu 3
            sjmp ZAC
PR12:       jnb P3.3,PR1            ;program 1 a 2,?treti bit=0 -> prog.1
            call PROG2              ;zbyva program 2, volani programu 2
            sjmp ZAC
PR1:        call PROG1              ;volani programu 1
            sjmp ZAC
```

3.3.4 Načítání vstupů, zápis na výstupy

Načítání vstupů a zápis na výstupy jsou velmi podobné funkce a je proto vhodné provést obě tyto akce najednou. Byly proto spojeny do jednoho podprogramu. Vývojový diagram podprogramu pro načtení a zápis vstupů a výstupů a obsluhu multiplexerů a demultiplexerů zobrazuje obrázek č. 9.



Obr. 14.: Vývojový diagram načítání vstupů a zápisu na výstupy

Zdrojový kód programu podle vývojového diagramu bude v jazyce symbolických adres vypadat:

```
;N A C I T A N I   V S T U P U
NACTI:      mov 28h,#00h            ;nulovani pameti z duvodu pouziteho
            mov 29h,#00h            ;ukladani po bitech
            mov P1,#11111111b       ;nastaveni adresy 0 MUXum
            nop                     ;prodleva pro spravnost zachyceni
            nop                     ;adresy MUXy
            setb P3.2               ;nastaveni OE MUXum
            nop                     ;prodleva pro spravne nacteni dat
```

```
nop
mov 20h,P2              ;nacteni 8 vstupu -> ulozeni do pam.
mov A,P3                ;nacteni zbylych 2 vstupu z P3
clr P3.2                ;nacteno -> vypnuti OE MUXu
anl A,#11000000b        ;maskovani dat vstupu z P3
orl 28h,A               ;ulozeni 2 vstupu z P3 do pameti

mov P1,#11011111b       ;nastaveni adresy 1 MUXum
nop                     ;zbytek stale stejny pro vsechny
nop                     ;adresy s rozdilem ukladani 2 vstupu
setb P3.2               ;nacitanych z brany P3
nop
nop
mov 21h,P2
mov A,P3
clr P3.2
anl A,#11000000b        ;maskovani prebytecnych bitu
rr A                    ;posunuti bitu o dve mista vpravo tj.
rr A                    ;tak, jak se maji ulozit
orl 28h,A               ;ulozeni 2 bitu do pameti

mov P1,#10111111b
nop
nop
setb P3.2
nop
nop
mov 22h,P2
mov A,P3
clr P3.2
anl A,#11000000b        ;maskovani
swap A                  ;prehozeni honiho a dolniho pulbyte
orl 28h,A               ;cimz se 2 bity dostanou na sva mista

mov P1,#10011111b
nop
nop
setb P3.2
nop
nop
mov 23h,P2
mov A,P3
clr P3.2
anl A,#11000000b        ;maskovani
rl A                    ;posunuti dvou bitu o dve mista vlevo
rl A                    ;tj z bitu 7 na 2 a z bitu 6 na 0
orl 28h,A

mov P1,#01111111b       ;zbytek dale stejny, pouze se
nop
nop
setb P3.2
nop
nop
mov 24h,P2
mov A,P3
clr P3.2
anl A,#11000000b
orl 29h,A

mov P1,#01011111b
```

```
            nop
            nop
            setb P3.2
            nop
            nop
            mov 25h,P2
            mov A,P3
            clr P3.2
            anl A,#11000000b
            rr A
            rr A
            orl 29h,A

            mov P1,#00111111b
            nop
            nop
            setb P3.2
            nop
            nop
            mov 26h,P2
            mov A,P3
            clr P3.2
            anl A,#11000000b
            swap A
            orl 29h,A

            mov P1,#00011111b
            nop
            nop
            setb P3.2
            nop
            nop
            mov 27h,P2
            mov A,P3
            clr P3.2
            anl A,#11000000b
            rl A
            rl A
            orl 29h,A

;Z A P I S   N A   V Y S T U P Y
            mov P0,30h              ;poslani hodnoty na vstupy DKO
            nop                     ;prodl. pro spravne zachyceni hodnot
            clr P1.0                ;zachyceni hodnot DKO1 (hodin. sig.)
            nop                     ;trvani hodinoveho signalu
            nop
            setb P1.0               ;konec hodinoveho signalu

            mov P0,31h              ;dale se opakuje, meni se pouze bit
            nop                     ;hodinoveho signalu a adresa pameti
            clr P1.1                ;odkud se data posilaji
            nop
            nop
            setb P1.1

            mov P0,32h
            nop
            clr P1.2
            nop
            nop
            setb P1.2
```

```
mov P0,33h
nop
clr P1.3
nop
nop
setb P1.3

mov P0,34h
nop
clr P1.4
nop
nop
setb P1.4
ret
```

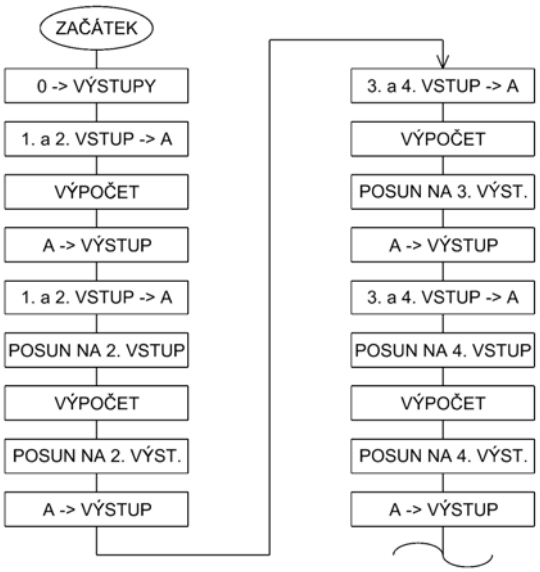
3.3.5 Výpočet logických funkcí

Výpočet logických funkcí lze rozdělit na dvě části, výpočet funkcí kombinačních logických obvodů a výpočet funkcí sekvenčních logických obvodů. Práce s pamětí (čtení, ukládání) probíhá ve vlastním těle programu, výpočet konkrétních logických funkcí probíhá voláním patřičného výpočtového podprogramu. Tělo programu řeší vždy jeden paměťový byte výstupu, tj. čtyři výpočtové bloky, tj. jednu řadu na panelu přístroje. S pouhou obměnou paměťových adres vstupů a výstupů a volaných logických funkcí se opakuje vždy 5x pro každé rozvržení panelu.

Tělo programu pro výpočet kombinačních funkcí

Výpočtové podprogramy konkrétních kombinačních logických funkcí využívají jako vstup akumulátor A ve tvaru VST1, VST2, VST3, VST4, X, X, X, X. Výsledek vrací též v akumulátoru A ve tvaru VÝST1, VÝST2, 0, 0, 0, 0, 0, 0.

Vývojový diagram podprogramu pro výpočet kombinačních logických funkcí je znázorněn na obrázku 15.



Obr. 15.: Vývojový diagram podprogramu pro výpočet kombinačních funkcí

Tělo programu v assembleru:

```
PROG1:      lcall NACTI             ;nacteni/zapis V/V
                                    ;zakladni kostra programu pro KLO
            mov 30h,#0              ;nulovani vystupu pro snazsi ukladani
            mov 31h,#0              ;protoze jde o KLO, neni predchozi
            mov 32h,#0              ;hodnota podstatna
            mov 33h,#0
            mov 34h,#0

;PRVNI RADA
            mov A,20h               ;vlozeni vstupu do akumulatoru
            lcall NEGACE            ;volani vypoctu log. funkce
            orl 30h,A               ;zapis vysledku do pameti vystupu

            mov A,20h               ;vlozeni vstupu do akumulatoru
            swap A                  ;posun vstupu na spravne misto
            lcall NEGACE            ;volani vypoctu log. funkce
            rr A                    ;posun vysledku na spravne misto
            rr A
            orl 30h,A               ;zapis vysledku do pameti vystupu

            mov A,21h               ;vlozeni vstupu do akumulatoru
            lcall NEGACE            ;volani vypoctu log. funkce
            swap A                  ;posun vysledku na spravne misto
            orl 30h,A               ;zapis vysledku do pameti vystupu

            mov A,21h               ;vlozeni vstupu do akumulatoru
            swap A                  ;posun vstupu na spravne misto
            lcall NEGACE            ;volani vypoctu log. funkce
            rl A                    ;posun vysledku na spravne misto
            rl A
            orl 30h,A               ;zapis vysledku do pameti vystupu
```

Tělo programu pro výpočet sekvenčních funkcí

Výpočtové podprogramy konkrétních sekvenčních logických funkcí využívají jako vstupy akumulátor A ve tvaru VST1, VST2, VST3, VST4, X, X, X, X a akumulátor B ve tvaru VÝST1, VÝST2, 0, 0, 0, 0, 0, 0. Výsledek navrací v akumulátoru A ve tvaru VÝST1, VÝST2, 0, 0, 0, 0, 0, 0. Rozhodování o hodinovém signálu probíhá přímo v těle programu, Konkrétní výpočtové podprogramy se volají až při příchodu hodinového signálu.

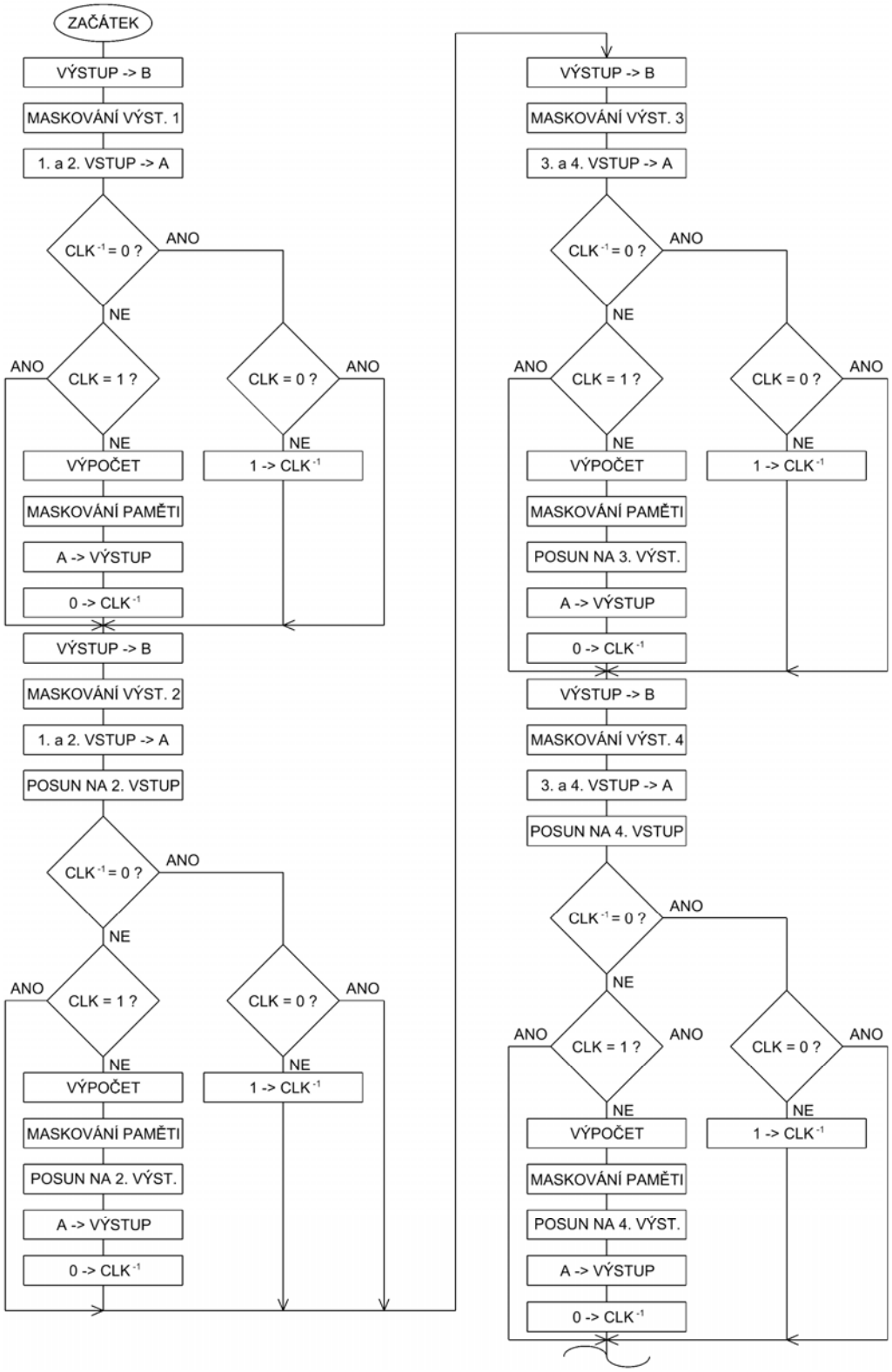
Vývojový diagram podprogramu pro výpočet sekvenčních logických funkcí znázorňuje obrázek 16.

Tělo programu v jazyce symbolických adres pak vypadá následovně:

```
PROG3:      lcall NACTI

;PRVNI RADA                         ;zakladni kostra programu pro SLO
            mov B,30h               ;ulozeni predchoziho vysledku do B
            anl B,#11000000b        ;maskovani predch. vysledku
            mov A,20h               ;ulozeni vstupu do A
            jnb 7Fh,NIC1            ;testovani zda predchozi CLK bylo 0
            jb Acc.6,DALSI1         ;CLK bylo 1, test zda ted je 0
            lcall SLORS             ;CLK bylo 1, je 0 => volej vypocet
            anl 30h,#00111111b      ;maskovani v pameti vystupu
            orl 30h,A               ;zapis vysledku do pameti vystupu
            clr 7Fh                 ;nastaveni minuleho CLK
            sjmp DALSI1             ;preskok na dalsi blok
NIC1:       jnb Acc.6,DALSI1        ;CLK bylo 0, test zda ted je 0
```

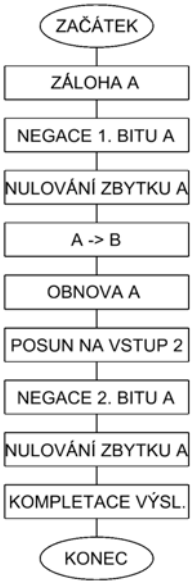
```
            setb 7Fh                ;CLK bylo 0, ted je 1, nast. min.CLK
DALSI1:                             ;dale se opakuje, pouze s drobnymy
                                    ;zmenami kvuli maskovani a posunu dat
                                    ;na patricna mista
            mov A,30h               ;posun a maskovani vysledku patriciho
            rl A                    ;ulozit do B v A, protoze v B nelze
            rl A                    ;rotovat bity
            anl A,#11000000b
            mov B,A
            mov A,20h
            swap A
            jnb 7Eh,NIC2
            jb Acc.6,DALSI2
            lcall SLORS
            rr A
            rr A
            anl 30h,#11001111b
            orl 30h,A
            clr 7Eh
            sjmp DALSI2
NIC2:       jnb Acc.6,DALSI2
            setb 7Eh
DALSI2:
            mov A,30h
            swap A
            anl A,#11000000b
            mov B,A
            mov A,21h
            jnb 7Dh,NIC3
            jb Acc.6,DALSI3
            lcall SLORS
            swap A
            anl 30h,#11110011b
            orl 30h,A
            clr 7Dh
            sjmp DALSI3
NIC3:       jnb Acc.6,DALSI3
            setb 7Dh
DALSI3:
            mov A,30h
            rr A
            rr A
            anl A,#11000000b
            mov B,A
            mov A,21h
            swap A
            jnb 7Ch,NIC4
            jb Acc.6,DALSI4
            lcall SLORS
            rl A
            rl A
            anl 30h,#11111100b
            orl 30h,A
            clr 7Ch
            sjmp DALSI4
NIC4:       jnb Acc.6,DALSI4
            setb 7Ch
DALSI4:
```



Obr. 16.: Vývojový diagram podprogramu pro výpočet sekvenčních funkcí

Výpočtový podprogram funkce NOT

Vývojový diagram podprogramu pro výpočet funkce NOT znázorňuje obrázek 17.



Obr. 17.: Vývojový diagram podprogramu pro výpočet funkce NOT

Tělo programu v jazyce symbolických adres pak vypadá následovně:

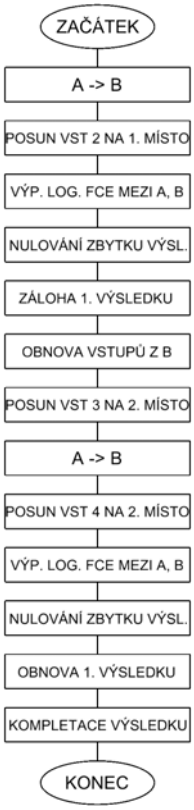
```
NEGACE:     push Acc              ;zaloha akumulatoru
            xrl A,#10000000b      ;vypocet prvni negace
            anl A,#10000000b      ;nulovani zbylych mist
            mov B,A               ;zaloha vysledku do B
            pop Acc               ;obnoveni ze zalohy
            rl A                  ;posun druheho NOTu na patricne misto
            rl A
            xrl A,#01000000b      ;vypocet druhe negace
            anl A,#01000000b      ;maskovani vysledku
            orl A,B
            ret
```

Výpočtový podprogram funkcí 2x dvouvstupový AND, OR, XOR

Výpočtové podprogramy pro funkce AND, OR, XOR jsou obdobné, liší se pouze v použití patřičné logické funkce při výpočtu uvnitř programu. Vývojový diagram podprogramu pro výpočet bloků dvou dvouvstupových nenegovaných funkcí znázorňuje obrázek 18.

Tělo programu v jazyce symbolických adres pak vypadá následovně:

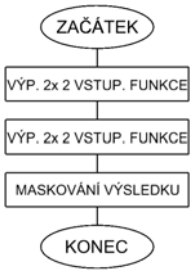
```
AND2:       mov B,A               ;vlozeni vstupu do B
            rl A                  ;posun druheho vst. proti prvnimu v B
            anl A,B               ;logicky soucin A a B
            anl A,#10000000b      ;vysl. v prvnim bitu, ostatni nulujeme
            push Acc              ;zaloha vysledku do zasobniku
            mov A,B               ;obnova vstupu z B
            rl A                  ;posun tretiho vstupu na druhe misto
            mov B,A               ;kvuli umisteni vysledku
            rl A                  ;posun ctvrteho vstupu proti tretimu
            anl A,B               ;logicky soucin tretiho a ctvrteho vst.
            anl A,#01000000b      ;nulovani ostatnich bitu
            pop B                 ;obnova zalohy ze zasobniku do B
            orl A,B               ;kompletace vysledku obou hradel
            ret
```



Obr. 18.: Vývojový diagram podprogramu pro výpočet bloků dvouvstupových nenegovaných log. funkcí

Výpočtový podprogram funkcí čtyřvstupový AND, OR

Vývojový diagram podprogramu pro výpočet bloků čtyřvstupových nenegovaných funkcí znázorňuje obrázek 19.



Obr. 19.: Vývojový diagram podprogramu pro výpočet bloků čtyřvstupových nenegovaných log. funkcí

Tělo programu v jazyce symbolických adres pak vypadá následovně:

```
AND4:       call AND2               ;volani 2x 2-vstupove funkce, tim
            call AND2               ;dostaneme oba vysledky jakoby na vstup
            anl A,#10000000b        ;dalsi funkce a podle distributivniho
            ret
```

Výpočtový podprogram negovaných funkcí NAND, NOR, XNOR

Vývojový diagram podprogramu pro výpočet všech negovaných funkcí znázorňuje obrázek 20.

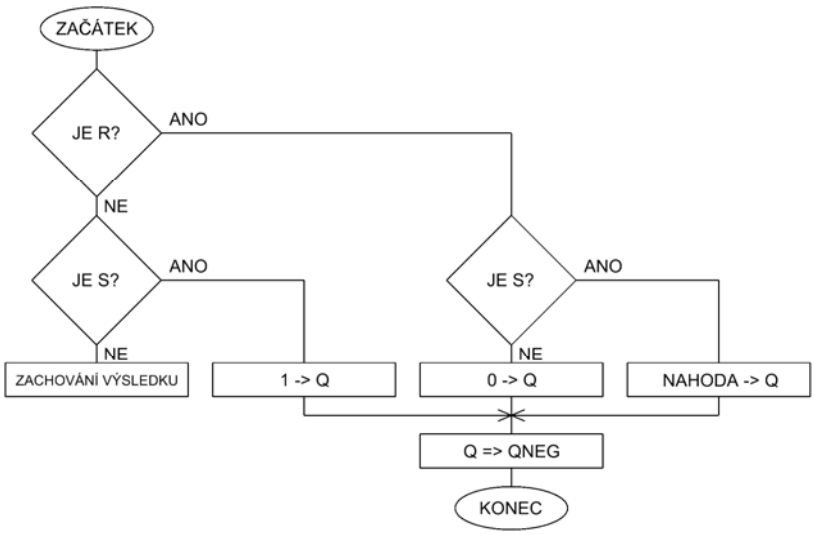
Tělo programu v jazyce symbolických adres pak vypadá následovně:

```
NAND2:      call AND2               ;volani nenegovane funkce
            xrl A,#11000000b        ;negace nenegovane funkce
            ret
```

Obr. 20.: Vývojový diagram podprogramu pro výpočet bloků všech negovaných log. funkcí

Výpočtový podprogram obvodu RS

Stav kdy jsou nastaveny oba vstupy R i S není u RS klopného obvodu definován. Teoreticky je tedy náhodný. Jako náhodná hodnota byla zvolena hodnota bitu na adrese 0 v paměti RAM. Vývojový diagram podprogramu pro výpočet RS klopného obvodu znázorňuje obrázek 21.



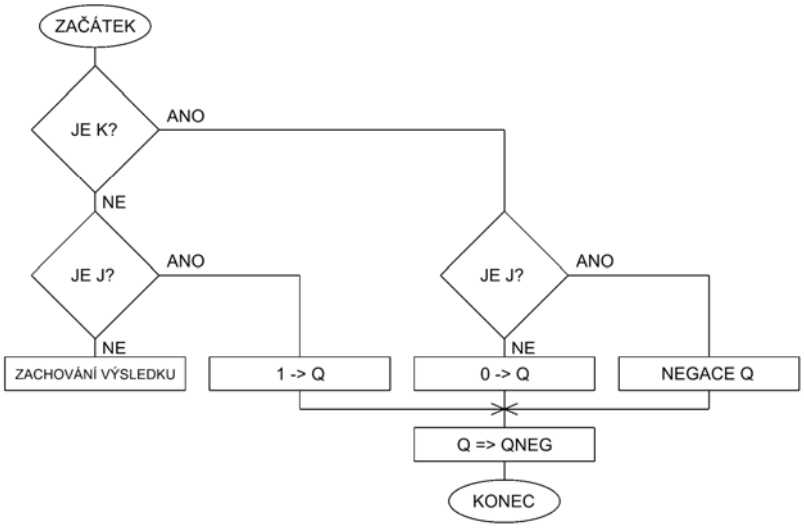
Obr. 21.: Vývojový diagram podprogramu pro výpočet RS klopného obvodu

Tělo programu v jazyce symbolických adres pak vypadá následovně:

```
SLORS:      jb Acc.7,JER          ;pokud je vstup R, skoc na JER
            jb Acc.4,JES          ;pokud je vstup S, skoc na JES
            mov A,B               ;neni ani R ani S vysledek nezmenen
            ret
JER:        jb Acc.4,NAHOD        ;je R,je-li S,neurcity(nahodny) stav
            clr A                 ;je R, neni S, vynuluj Q
            setb Acc.6            ;nastav QNEG
            ret
JES:        clr A                 ;je S, neni R, nuluj QNEG
            setb Acc.7            ;nastav Q
            ret
NAHOD:      jb 00h,NAH1           ;jako nahodna hodnota se bere hdnota
            clr A                 ;bitu 0 v bitove adresovatelne RAM
            setb Acc.6
            ret
NAH1:       clr A
            setb Acc.7
            ret
```

Výpočtový podprogram obvodu JK

Vývojový diagram podprogramu pro výpočet JK klopného obvodu je znázorněn na obrázku 22.



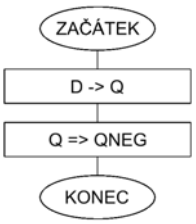
Obr. 22.: Vývojový diagram podprogramu pro výpočet JK klopného obvodu

Tělo programu v jazyce symbolických adres pak vypadá následovně:

```
SLOJK:      jb Acc.7,JEK            ;obdobne jak RS, zmena v miste, kdyz
            jb Acc.4,JEJ            ;je J i K
            mov A,B
            ret
JEK:        jb Acc.4,NEGUJ
            clr A
            setb Acc.6
            ret
JEJ:        clr A
            setb Acc.7
            ret
NEGUJ:      mov A,B                 ;zmena oproti RS, predchozi vysledek
            xrl A,#11000000b        ;ulozeny v B se zneguje a ulozi do A
            ret
```

Výpočtový podprogram obvodu D

Vývojový diagram podprogramu pro výpočet D klopného obvodu je znázorněn na obrázku 23.



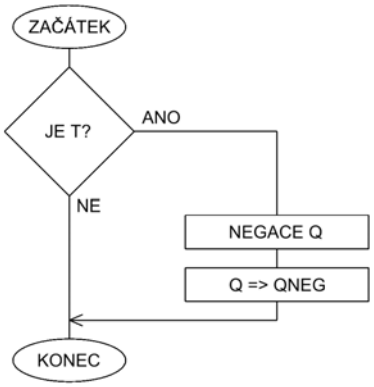
Obr. 23.: Vývojový diagram podprogramu pro výpočet D klopného obvodu

Tělo programu v jazyce symbolických adres pak vypadá následovně:

```
SLOD:       anl A,#10000000b        ;nulovani zbytku vstupu
            mov B,A                 ;ulozeni vysledku do B
            rr A                    ;vypocet QNEG - posun Q do QNEG
            xrl A,#01000000b        ;negace Q v QNEG
            orl A,B                 ;kompletace vysledku Q a QNEG
            ret
```

Výpočtový podprogram obvodu T

V podprogramu bylo využito shody funkce s JK klopným obvodem v případě nastavených obou vstupů J i K. Program proto v případě nastaveného vstupu T skáče dovnitř podprogramu pro výpočet JK klopného obvodu do míst výpočtu negace výsledku. Vývojový diagram podprogramu pro výpočet T klopného obvodu znázorňuje obrázek 24.



Obr. 24.: Vývojový diagram podprogramu pro výpočet T klopného obvodu

Tělo programu v jazyce symbolických adres pak vypadá následovně:

```
SLOT:       jb Acc.7,NEGUJ          ;pokud je vstup T, skoc na vypocet
            mov A,B                 ;negace v JK obvodu
            ret
```

4. NÁVRH LABORATORNÍCH ÚLOH PRO ZAPOJENÍ NA PANELU

4.1 ÚLOHA 1 – Základní kombinační logické obvody

Pro úlohu je navrženo rozmístění panelu č. 1. Úkolem je seznámení se základními kombinačními logickými obvody, jejich značením, matematickou funkcí a ověření této funkce zapojením na panelu.

ZADÁNÍ:

Popište základní kombinační logické obvody, zapojením na panelu ověřte jejich chování a porovnejte jej s pravdivostními tabulkami pro jednotlivé obvody.

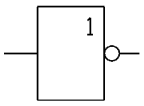
VYPRACOVÁNÍ:

Existují čtyři základní druhy KLO a tři jejich negované modifikace (mají negovaný výstup)

Invertor	NOT	$y = \overline{x}$	(6)
Logický součet	OR	$y = x_1 + x_2$	(7)
Logický součin	AND	$y = x_1 \cdot x_2$	(8)
Logická nonekvivalence	XOR (eXclusive OR)	$y = \overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2}$	(9)
Negovaný logický součet	NOR (Negative OR)	$y = \overline{x_1 + x_2}$	(10)
Negovaný logický součin	NAND (Negative AND)	$y = \overline{x_1 \cdot x_2}$	(11)
Logická ekvivalence	XNOR (eXclusive Negative OR)	$y = \overline{\overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2}}$	(12)

Invertor (NOT)

Jedná se o jednovstupovou funkci. Hodnota výstupu je dána převrácenou hodnotou vstupu.



Obr. 25.: Schematická značka invertoru

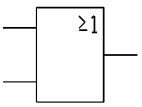
$$y = \overline{x} \tag{6}$$

x	y
0	1
1	0

Tab. 1.: Pravdivostní tabulka invertoru

Logický součet (OR)

Může být obecně n-vstupový. Hodnota na výstupu je dána logickým součtem hodnot vstupů.



Obr. 26.: Schematická značka logického součtu

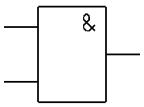
$$y = x_1 + x_2 \tag{7}$$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Tab. 2.: Pravdivostní tabulka logického součtu

Logický součin (AND)

Může být obecně n-vstupový. Hodnota výstupu je dána logickým součinem hodnot vstupů.



Obr. 27.: Schematická značka logického součinu

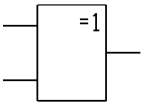
$$y = x_1 \cdot x_2 \tag{8}$$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Tab. 3.: Pravdivostní tabulka logického součinu

Logická nonekvivalence (XOR)

Může být pouze dvouvstupová. Rozhodující je rovnost vstupů. Pokud jsou na obou vstupech shodné hodnoty, vrací tato funkce nulovou hodnotu výstupu, pokud jsou hodnoty na vstupech rozdílné, vrací na výstup hodnotu „1“.



Obr. 28.: Schematická značka logické nonekvivalence

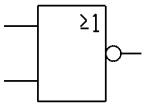
$$y = \overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2} \tag{9}$$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Tab. 4.: Pravdivostní tabulka logické nonekvivalence

Negovaný logický součet (NOR)

Může být obecně n-vstupový. Na výstup vrací převrácenou hodnotu logického součtu hodnot na vstupech.



Obr. 29.: Schematická značka negovaného logického součtu

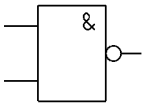
$$y = \overline{x_1 + x_2} \tag{10}$$

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	0

Tab. 5.: Pravdivostní tabulka negovaného logického součtu

Negovaný logický součin (NAND)

Může být obecně n-vstupový. Na výstup vrací převrácenou hodnotu logického součinu hodnot vstupů.



Obr. 30.: Schematická značka negovaného logického součinu

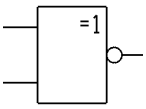
$$y = \overline{x_1 \cdot x_2} \tag{11}$$

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

Tab. 6.: Pravdivostní tabulka negovaného logického součinu

Logická ekvivalence (XNOR)

Může být pouze dvouvstupová. Na výstup vrací „0“ pokud hodnoty na vstupech jsou rozdílné. Pokud jsou hodnoty na vstupech shodné, vrací hodnotu „1“.



Obr. 31.: Schematická značka logické ekvivalence

$$y = \overline{\overline{x_1} \cdot x_2 + x_1 \cdot \overline{x_2}} \tag{12}$$

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

Tab. 7.: Pravdivostní tabulka logické ekvivalence

MODIFIKACE ÚLOHY:

Tato úloha je bez modifikací, pro správné pochopení by si měl každý vyzkoušet všechny základní logické funkce.

4.2 ÚLOHA 2 – Booleova algebra

Pro úlohu je navrženo rozmístění panelu č. 1. Úkolem je seznámení se s pravidly Booleovy algebry a jejich ověření příslušným zapojením na panelu.

ZADÁNÍ:

Popište zákony a pravidla Booleovy algebry a jejich pravdivost ověřte zapojením na panelu.

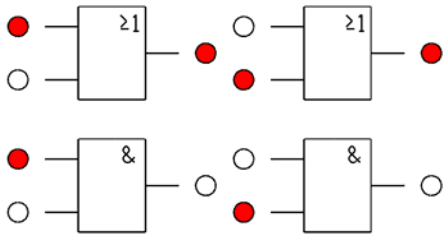
VYPRACOVÁNÍ:

Zákony a pravidla Booleovy algebry:

Komutativní zákon – nezáleží na pořadí prvků

$$\begin{aligned} x_1 + x_2 &= x_2 + x_1 \\ x_1 \cdot x_2 &= x_2 \cdot x_1 \end{aligned} \tag{13}$$

Zapojení na panelu:

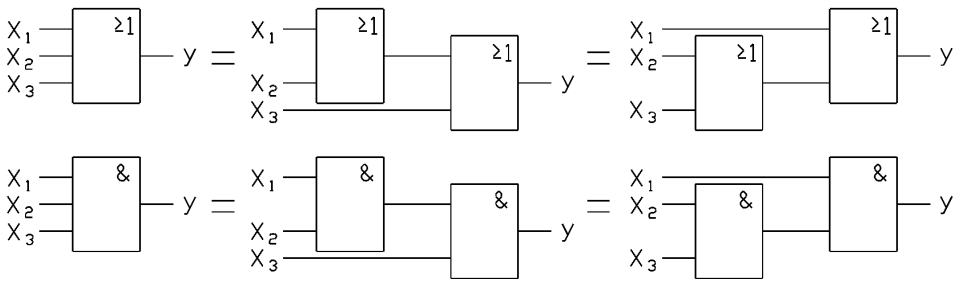


Obr. 32.: Zapojení zkoušky komutativního zákona

Asociativní zákon – sdružování do závorek

$$\begin{aligned} x_1 + x_2 + x_3 &= (x_1 + x_2) + x_3 = x_1 + (x_2 + x_3) \\ x_1 \cdot x_2 \cdot x_3 &= (x_1 \cdot x_2) \cdot x_3 = x_1 \cdot (x_2 \cdot x_3) \end{aligned} \tag{14}$$

Zapojení na panelu:

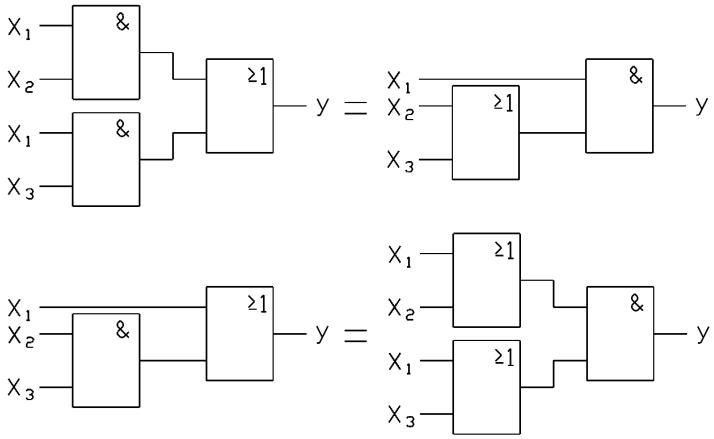


Obr. 33.: Zapojení zkoušky asociativního zákona

Distributivní zákon – vytýkání před závorku

$$\begin{aligned} x_1 \cdot x_2 + x_1 \cdot x_3 &= x_1 \cdot (x_2 + x_3) \\ x_1 + x_2 \cdot x_3 &= (x_1 + x_2) \cdot (x_1 + x_3) \end{aligned} \tag{15}$$

Zapojení na panelu:

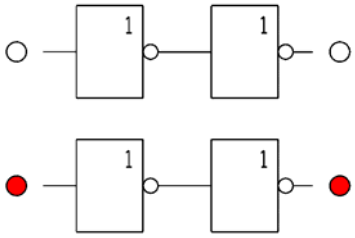


Obr. 34.: Zapojení zkoušky distributivního zákona

Pravidlo dvojí negace

$$\overline{\overline{x}} = x \tag{16}$$

Zapojení na panelu:

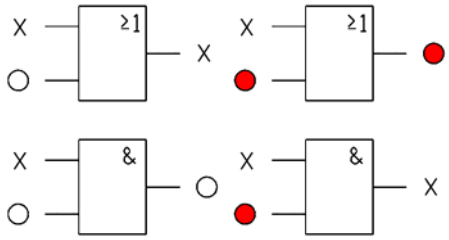


Obr. 35.: Zapojení zkoušky pravidla dvojí negace

Pravidlo neutrality a agresivity „1“ a „0“

$$\begin{aligned} x + 0 &= x \\ x \cdot 1 &= x \\ x + 1 &= 1 \\ x \cdot 0 &= 0 \end{aligned} \tag{17}$$

Zapojení na panelu:

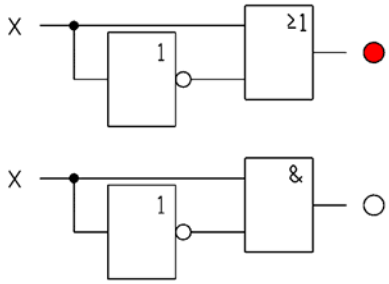


Obr. 36.: Zapojení zkoušky pravidla neutrality a agresivity „1“ a „0“

Pravidlo negace

$$x + \overline{x} = 1$$
$$x \cdot \overline{x} = 0 \tag{18}$$

Zapojení na panelu:

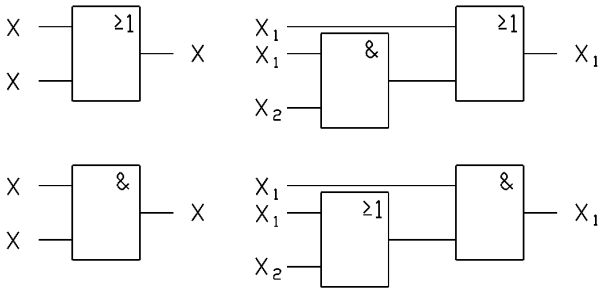


Obr. 37.: Zapojení zkoušky pravidla negace

Pravidlo absorpce

$$x + x = x$$
$$x \cdot x = x$$
$$x_1 + x_1 \cdot x_2 = x_1$$
$$x_1 \cdot (x_1 + x_2) = x_1 \tag{19}$$

Zapojení na panelu:

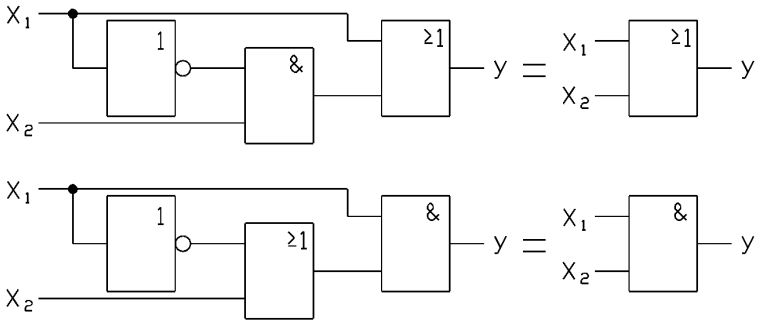


Obr. 38.: Zapojení zkoušky pravidla absorpce

Pravidlo absorpce negace

$$x_1 + \overline{x_1} \cdot x_2 = x_1 + x_2$$
$$x_1 \cdot (\overline{x_1} + x_2) = x_1 \cdot x_2 \tag{20}$$

Zapojení na panelu:

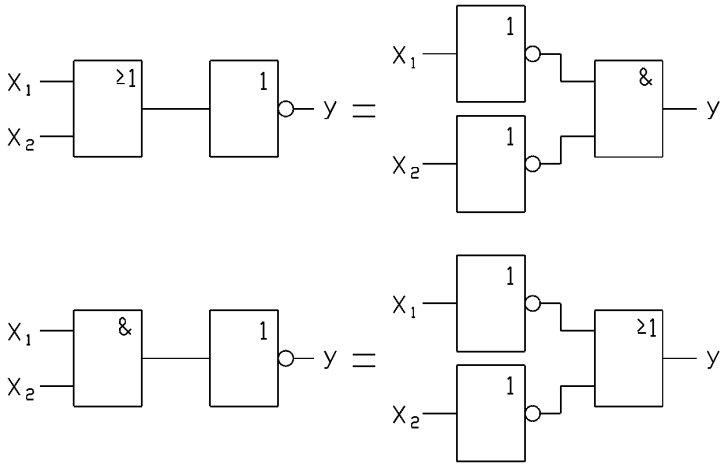


Obr. 39.: Zapojení zkoušky pravidla absorpce negace

De Morganovy zákony

$$\overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}$$
$$\overline{x_1 \cdot x_2} = \overline{x_1} + \overline{x_2} \qquad (21)$$

Zapojení na panelu:



Obr. 40.: Zapojení zkoušky de Morganových zákonů

MODIFIKACE ÚLOHY:

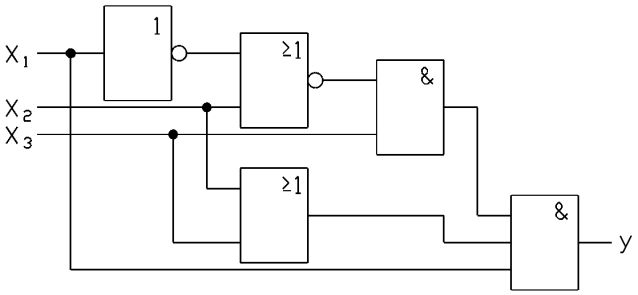
Tato úloha je bez modifikací, pro správné pochopení by si měl každý vyzkoušet všechny zákony a pravidla Booleovy algebry.

4.3 ÚLOHA 3 – Minimalizace logických funkcí na hradla NAND a NOR

Pro úlohu je navrženo rozmístění panelu č. 1 a 2. Úkolem úlohy je procvičení práce se základními kombinačními logickými obvody a minimalizace logické funkce na co nejmenší počet skutečných součástek.

ZADÁNÍ:

Pomocí vhodného pospojování navrhněte a realizujte všechny základní logické funkce pouze s použitím hradel NAND a NOR. Dále s využitím získaných poznatků realizujte zapojení zadané funkce pouze s pomocí 2 a 4 vstupových hradel NAND a ověřte správnost chování.



Obr. 41.: Zadání logické funkce pro minimalizaci

VYPRACOVÁNÍ:

Zapojení jednotlivých obvodů vycházejí ze zákonů Booleovy algebry, zvláště pak z De Morganových zákonů.

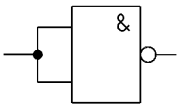
Realizace hradla NOT

Využitím pravidla absorpce, tj. spojením vstupů se využijí pouze 2 řádky pravdivostní tabulky a tím dostaneme námi požadované chování.

Pomocí hradel NAND

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

Tab. 8.: Využití řádků pravdivostní tabulky hradla NAND

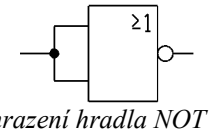


Obr. 42.: Nahrazení hradla NOT hradlem NAND

Pomocí hradel NOR

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	0

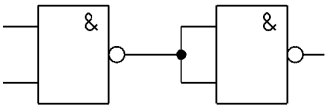
Tab. 9.: Využití řádků pravdivostní tabulky hradla NOR



Obr. 43.: Nahrazení hradla NOT hradlem NOR

Realizace hradla AND

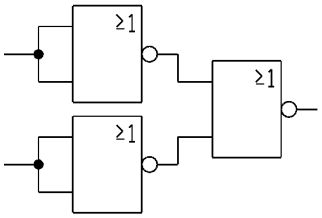
Pomocí NAND – využití pravidla dvojí negace a zároveň pravidla absorpce. Použijeme tedy jedno hradlo NAND, které se na výstupu zneguje druhým hradlem NAND se spojenými vstupy.



Obr. 44.: Nahrazení hradla AND hradlem NAND

Pomocí NOR – využití pravidla dvojí negace a de Morganových zákonů

$$y = x_1 \cdot x_2 = \overline{\overline{x_1 \cdot x_2}} = \overline{\overline{x_1} + \overline{x_2}} \tag{22}$$

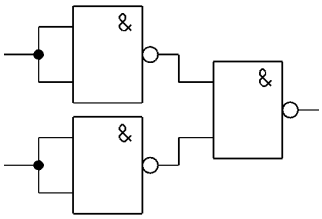


Obr. 45.: Nahrazení hradla AND hradlem NOR

Realizace hradla OR

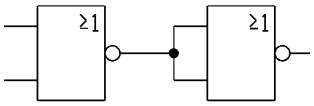
Pomocí NAND – využití pravidla dvojí negace a de Morganových zákonů

$$y = x_1 + x_2 = \overline{\overline{x_1 + x_2}} = \overline{\overline{x_1} \cdot \overline{x_2}} \tag{23}$$



Obr. 46.: Nahrazení hradla OR hradlem NAND

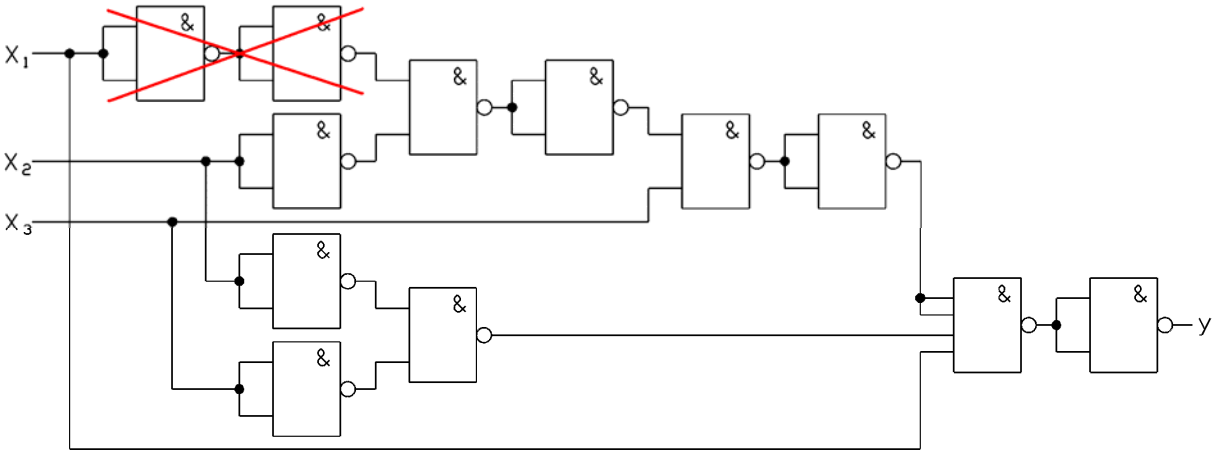
Pomocí NOR – využití jednoho hradla NOR, které se na výstupu zneguje. Podle pravidla dvojí negace se negace na výstupu vyruší.



Obr. 47.: Nahrazení hradla OR hradlem NOR

Realizace zadané funkce pomocí 2 a 4 vstupových hradel NAND

Jednotlivá hradla postupně nahrazujeme ekvivalentním zapojením hradel NAND. Dále vykrátíme zbytečná hradla, v našem případě dvě negace za sebou na vstupu a. Nakonec obě zapojení realizujeme na panelu a zapíšeme pro obě zapojení pravdivostní tabulku. Pravdivostní tabulka pro obě zapojení musí vyjít shodná.



Obr. 48.: Realizace zadané funkce pomocí hradel NAND

MODIFIKACE ÚLOHY:

Tato úloha může mít obrovské množství modifikací. Například lze stejný obvod realizovat pomocí hradel NOR, nebo lze vymyslet v podstatě libovolné zapojení pro minimalizování.

4.4 ÚLOHA 4 – Návrh logické funkce pomocí Karnaughovy mapy

Pro úlohu je navrženo rozmístění panelu č. 1. Úkolem úlohy je procvičení práce se základními kombinačními logickými obvody a minimalizace logické funkce na co nejmenší počet hradel.

ZADÁNÍ:

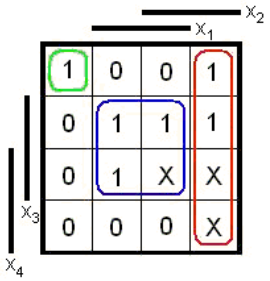
Pomocí Karnaughovy mapy navrhněte a realizujte zapojení logické funkce zadané pravdivostní tabulkou. Ověřte správnost chování.

	x_1	x_2	x_3	x_4	y
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	X
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	X
15	1	1	1	1	X

Tab. 10.: Zadání logické funkce pro minimalizaci

VYPRACOVÁNÍ:

Z pravdivostní tabulky sestrojíme Karnaughovu mapu.

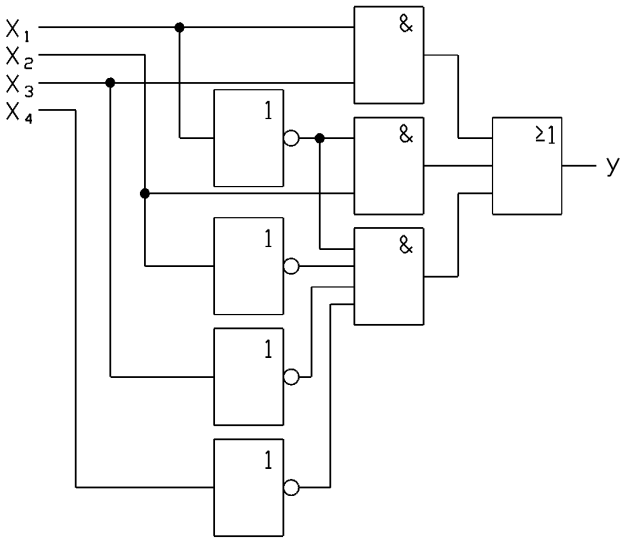


Obr. 49.: Karnaughova mapa zadané logické funkce

Z vytvořené Karnaughovy mapy sestavíme matematický výraz zadané logické funkce.

$$y = x_1 \cdot x_3 + \overline{x_1} \cdot x_2 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} \tag{24}$$

Zapojení zadané funkce bude potom vypadat následovně:



Obr. 50.: Realizace zadané funkce pomocí hradel NAND

MODIFIKACE ÚLOHY:

Tato úloha může mít obrovské množství modifikací. Lze vymyslet v podstatě libovolné zadání pravdivostní tabulky.

4.5 ÚLOHA 5 – Klopné obvody

Pro úlohu je navrženo rozmístění panelu č. 3. Úkolem úlohy je seznámení se základními sekvenčními logickými obvody a ověření jejich funkce.

ZADÁNÍ:

Popište základní sekvenční logické obvody, zapojením na panelu ověřte jejich chování a porovnejte jej s pravdivostními tabulkami pro jednotlivé obvody.

VYPRACOVÁNÍ:

Sekvenční logické obvody dělíme na dvě základní kategorie:
Asynchronní – ke změně hodnoty na výstupu dojde ihned po změně hodnoty na vstupu

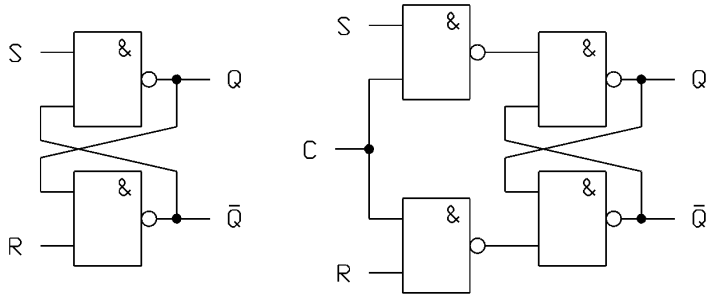
Synchronní – jsou synchronizovány tzv. hodinovým signálem (CLK, vstup C). Obvod při změně hodnoty na vstupu nereaguje ihned výstupní hodnotou, ke změně výstupní hodnoty dojde až s příchodem hodinového signálu. Většina obvodů reaguje na sestupnou hranu hodinového signálu, tj. změnu z „1“ na „0“ na vstupu C.

Rozlišujeme čtyři základní druhy sekvenčních logických obvodů

Klopný obvod RS	Reset Set	$R \neq S$: $Q_{n+1} = S \cdot \overline{R}$ $R = S = 1$: zakázaný stav $R = S = 0$: $Q_{n+1} = Q_n$	(25)
Klopný obvod JK	Jack Kilby	$J \neq K$: $Q_{n+1} = J \cdot \overline{K}$ $J = K = 1$: $Q_{n+1} = \overline{Q_n}$ $J = K = 0$: $Q_{n+1} = Q_n$	(26)
Klopný obvod D	Delay	$Q_{n+1} = D$	(27)
Klopný obvod T	Time switch	$T = 0$: $Q_{n+1} = Q_n$ $T = 1$: $Q_{n+1} = \overline{Q_n}$	(28)

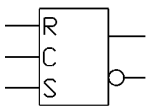
RS klopný obvod

Nejjednodušší klopný obvod, lze jej snadno postavit pomocí dvou hradel NAND



Obr. 51.: Realizace RS klopného obvodu pomocí hradel NAND

Může být asynchronní i synchronní. Má dva vstupy - set (S) a reset (R). Vstup S nastavuje výstup do log. „1“, vstup R do log. „0“. U tohoto obvodu by se nemělo stát, že se oba vstupy nastaví na log. „1“. Dojde totiž k tzv. zakázanému stavu - stavu, v němž není definována hodnota výstupu.



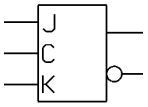
Obr. 52.: Schematická značka RS klopného obvodu

R	**S**	**Q_{n+1}**
0	0	Q_n
0	1	1
1	0	0
1	1	-

Tab. 11.: Pravdivostní tabulka RS klopného obvodu

JK klopný obvod

Pojmenován podle svého vynálezce Jacka Kilbyho, tehdejšího zaměstnance firmy Texas Instruments. Jedná se o vylepšenou variantu RS klopného obvodu s tím rozdílem, že je odstraněn zakázaný stav vstupů a obvod při tomto stavu neguje svou předchozí výstupní hodnotu. Obvod JK se vyskytuje pouze v synchronním provedení.

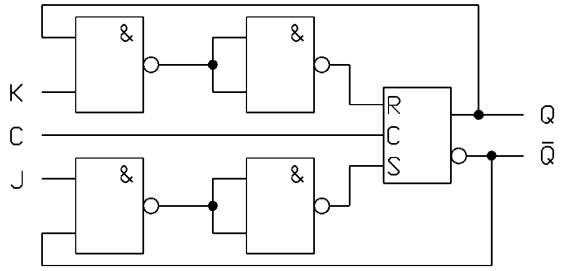


Obr. 53.: Schematická značka JK klopného obvodu

J	**K**	**Q_{n+1}**
0	0	Q_n
0	1	1
1	0	0
1	1	$\overline{Q_n}$

Tab. 12.: Pravdivostní tabulka JK klopného obvodu

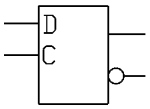
JK klopný obvod lze jednoduše realizovat pomocí RS klopného obvodu



Obr. 54.: Realizace JK klopného obvodu pomocí RS a NAND

D klopný obvod

Může být pouze synchronní. Jedná se o jednobitovou paměť. Při příchodu hodinového signálu se na výstupu objeví aktuální hodnota vstupu D. Tuto hodnotu si obvod „pamatuje“ až do dalšího příchodu hodinového signálu.

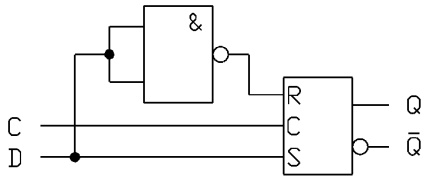


Obr. 55.: Schematická značka JK klopného obvodu

D	**Q_{n+1}**
0	0
1	1

Tab. 13.: Pravdivostní tabulka D klopného obvodu

D klopný obvod realizovaný z RS obvodu

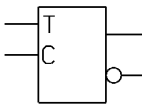


Obr. 56.: Realizace D klopného obvodu pomocí RS a NAND

Stejným způsobem lze D klopný obvod zrealizovat i pomocí JK klopného obvodu, pokud vstup J zapojíme jako vstup S a vstup K jako vstup R.

T klopný obvod

Může být synchronní i asynchronní. Pokud je na vstupu T log. „0“, výstup se s příchodem hodinového signálu nijak nemění, pokud je na vstupu T log. „1“, s příchodem hodinového signálu se hodnota výstupu neguje.

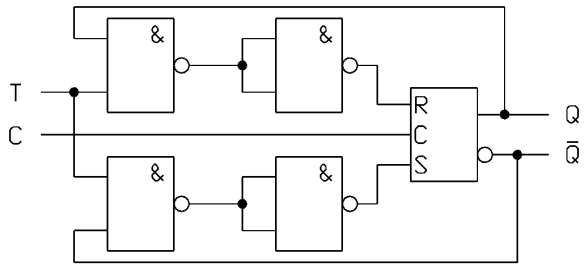


Obr. 57.: Schematická značka T klopného obvodu

T	**Q_{n+1}**
0	Q_n
1	$\overline{Q_n}$

Tab. 14.: Pravdivostní tabulka T klopného obvodu

T klopný obvod realizovaný pomocí RS obvodu



Obr. 58.: Realizace T klopného obvodu pomocí RS a NAND

Z JK klopného obvodu lze T klopný obvod sestrojit tak, že se spojí vstupy J a K.

MODIFIKACE ÚLOHY:

Tato úloha je bez modifikací, pro správné pochopení by si měl každý vyzkoušet všechny základní logické funkce.

4.6 ÚLOHA 6 – Návrh obecného sekvenčního logického obvodu

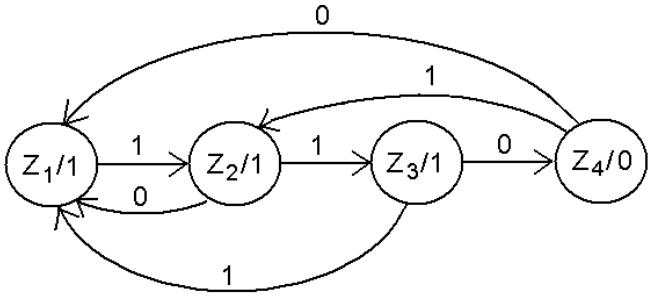
Pro úlohu je navrženo rozmístění panelu č. 3. Úkolem úlohy je vytvoření obecného sekvenčního logického obvodu.

ZADÁNÍ:

Navrhněte automat, který detekuje posloupnost bitů 1, 1, 0. Pokud zachytí tuto posloupnost, na výstupu se objeví logická „0“, v opačném případě bude na výstupu log. „1“.

VYPRACOVÁNÍ:

Dle slovního popisu chování sestavíme graf přechodu. Pro realizaci obvodu použijeme automat Moorova typu u něhož je výstupní stav dán pouze jeho vnitřním stavem.



Obr. 58.: Graf přechodu navrhovaného automatu

Vnitřní stavy vyjádříme pomocí binárních hodnot. Pro čtyři vnitřní stavy potřebujeme dva paměťové bity.

	Q_1	Q_2
Z_1	0	0
Z_2	0	1
Z_3	1	0
Z_4	1	1

Tab. 15.: Vyjádření vnitřních stavů automatu

Paměťovou část obvodu budeme realizovat například pomocí JK klopných obvodů. Pro JK klopný obvod vypadá tabulka přechodů následovně

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

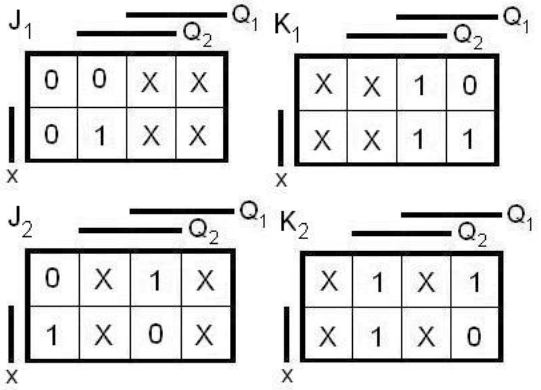
Tab. 16.: Tabulka přechodů pro JK klopný obvod

Dále si sestavíme kompletní tabulku pro zadaný sekvenční logický obvod

	$Q_{1\,n}$	$Q_{2\,n}$	x	$Q_{1\,n+1}$	$Q_{2\,n+1}$	J_1	K_1	J_2	K_2
Z_1	0	0	0	0	0	0	X	0	X
	0	0	1	0	1	0	X	1	X
Z_2	0	1	0	0	0	0	X	X	1
	0	1	1	1	0	1	X	X	1
Z_3	1	0	0	1	1	X	0	1	X
	1	0	1	0	0	X	1	0	X
Z_4	1	1	0	0	0	X	1	X	1
	1	1	1	0	1	X	1	X	0

Tab. 17.: Tabulka stavů a přechodů navrhovaného automatu

Z kompletní tabulky sestavíme Karnaughovy mapy pro všechny vstupy JK obvodů a pro výstupní hodnotu. Z Karnaughových map vyjádříme funkce pro jednotlivé vstupy.



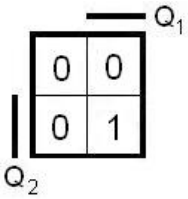
Obr. 59.: Karnaughovy mapy vstupů JK klopných obvodů

$$J_1 = Q_2 \cdot x = \overline{\overline{Q_2 \cdot x}} \tag{29}$$

$$K_1 = Q_2 + x = \overline{\overline{Q_2} \cdot \overline{x}} \tag{30}$$

$$J_2 = Q_2 \cdot \overline{x} + \overline{Q_1} \cdot x = \overline{\overline{Q_2 \cdot \overline{x}} \cdot \overline{\overline{Q_1} \cdot x}} \tag{31}$$

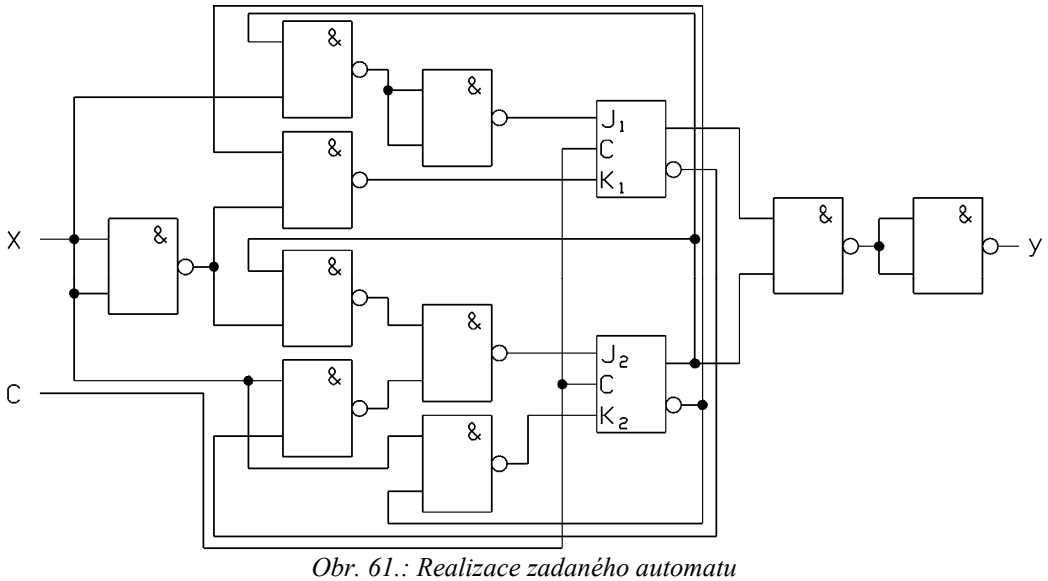
$$K_2 = Q_2 + \overline{x} = \overline{\overline{Q_2} \cdot x} \tag{32}$$



Obr. 60.: Karnaughova mapa výstupu obvodu

$$y = Q_1 \cdot Q_2 = \overline{\overline{Q_1} \cdot \overline{Q_2}} \tag{33}$$

Podle funkcí jednotlivých vstupů a výstupní funkce jsme již schopni navrhnout zapojení automatu.



Obr. 61.: Realizace zadaného automatu

MODIFIKACE ÚLOHY:

Pro tuto úlohu lze vytvořit mnoho modifikací, například lze realizovat automat jako Mealyho typ, nebo jej lze realizovat pomocí jiných klopných obvodů (D, T, RS) možné je též vymyslet úplně jiné chování realizovaného automatu.

5. ZÁVĚR

Návrhem a jeho následnou realizací vzniklo zařízení, které by v budoucnu mohlo skutečně sloužit při výuce. Na zařízení je možné realizovat nejrůznější zapojení kombinačních i sekvenčních logických obvodů a ověřit si tak teoreticky nabyté znalosti z oblasti číslicové techniky v praxi. Při návrhu i konstrukci zařízení bylo k tomuto maximálně přihlíženo, proto je celý přístroj napájen bezpečným stejnosměrným napětím z průmyslově vyráběného zdroje a všechny vývody ze zařízení jsou opatřeny ochrannými rezistory omezujícími maximální zkratový proud. Tímto opatřením bylo téměř vyloučeno riziko poškození přístroje či úrazu v důsledku neodborné manipulace s přístrojem.

V první a druhé části této práce jsem se seznámil s různými druhy přístrojů používaných k výuce číslicové techniky, s produkty na trhu, jejich výhodami, nevýhodami, cenou a provedením.

Třetí část zaměřená na návrh praktické části práce mi pomohla ověřit, procvičit a zdokonalit teoretické i praktické znalosti s návrhem elektronických obvodů a programováním osmibitových mikrořadičů založených na bázi procesoru Intel 8051.

Ve čtvrté části jsem si pak ověřil a zopakoval znalosti z číslicové techniky, zejména chování základních kombinačních a sekvenčních logických obvodů, minimalizace logických funkcí, návrh obecných kombinačních a sekvenčních logických obvodů.

6. SEZNAM POUŽITÉ LITERATURY

Odkazy na internetu:

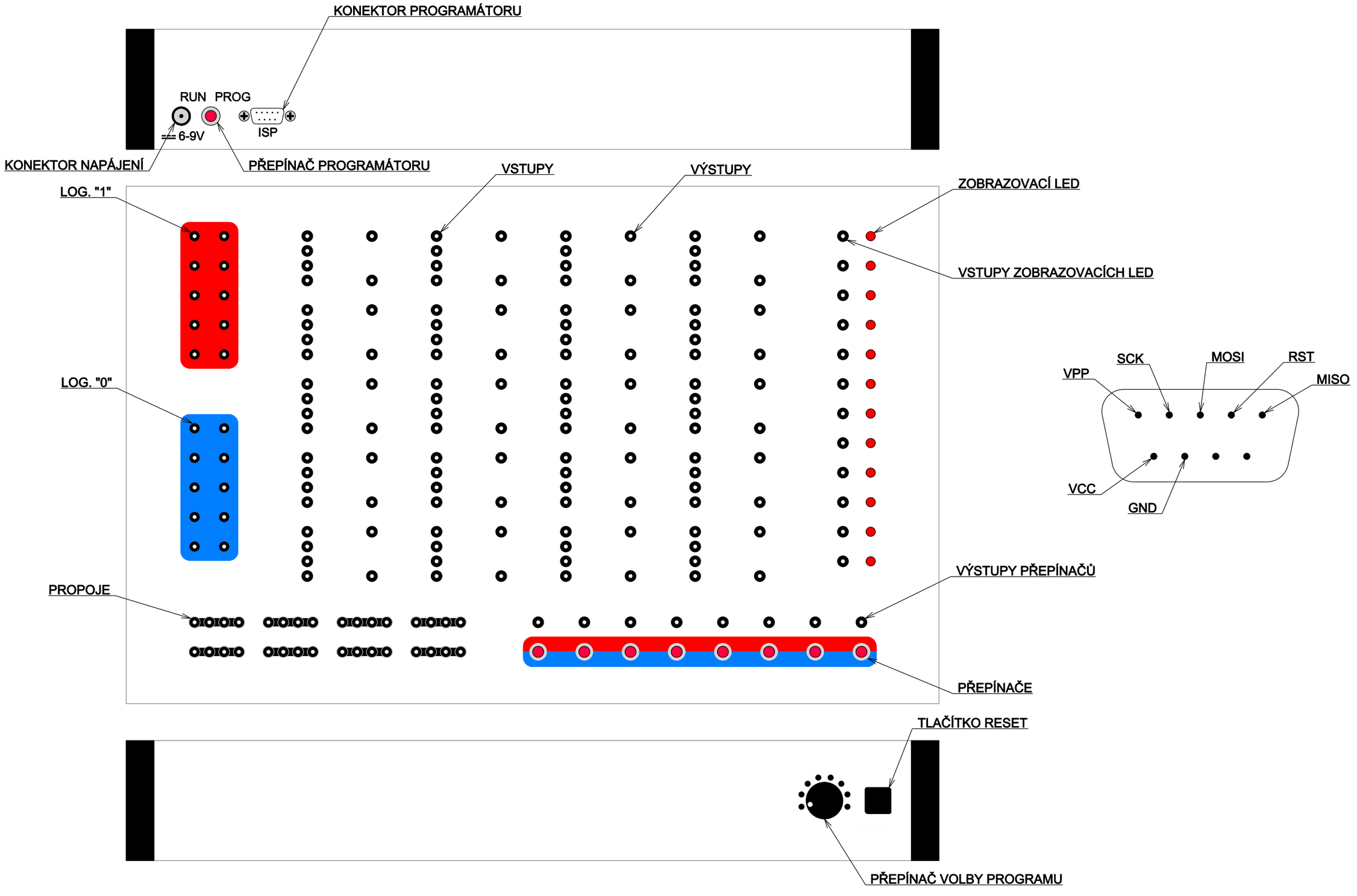
[1] http://www.hw.cz
[2] http://www.kpsec.freeuk.com/gates.htm
[3] http://www.fm.vslib.cz/~kes/data/rs_ko.pdf
[4] http://www.dhservis.cz
[5] http://www.atmel.com
[6] http://www.ti.com
[7] http://www.alldatasheets.com
[8] http://cs.wikipedia.org

7. PŘÍLOHY

SEZNAM PŘÍLOH

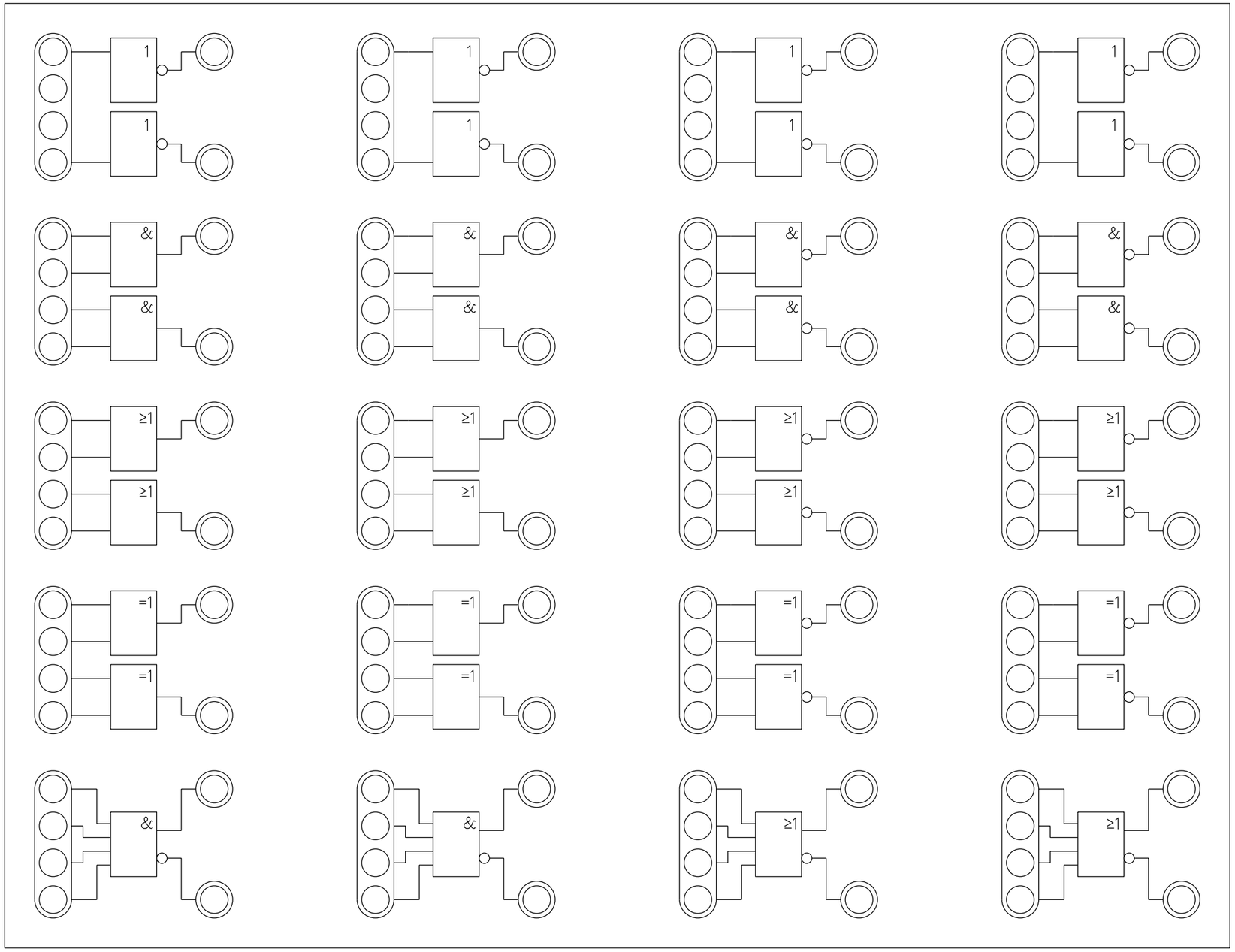
P1 - VZHLED ZAŘÍZENÍ, ZAPOJENÍ ISP KONEKTORU

M 1:2



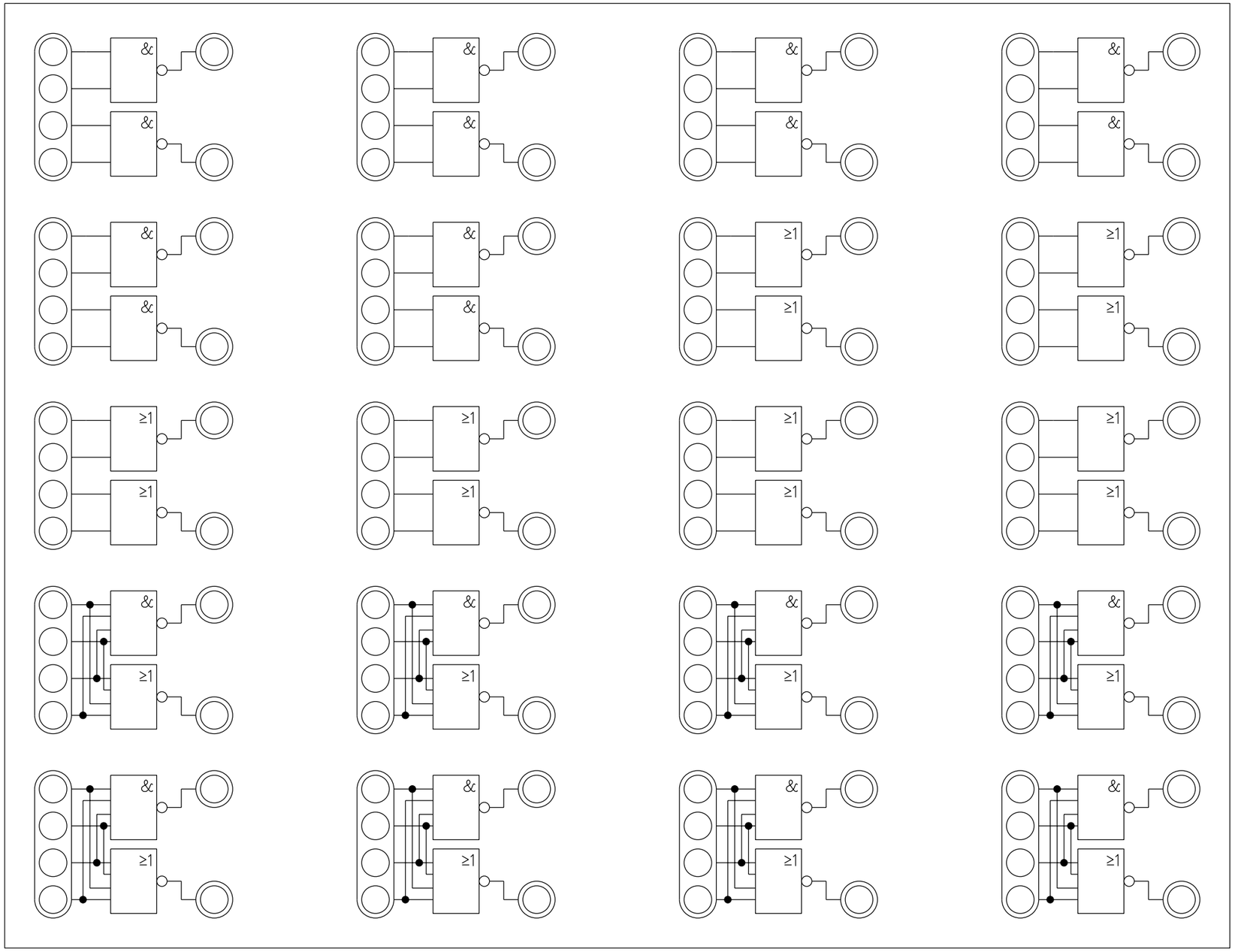
P2 - ROZVRŽENÍ SOUČÁSTEK Č. 1

M 1:1



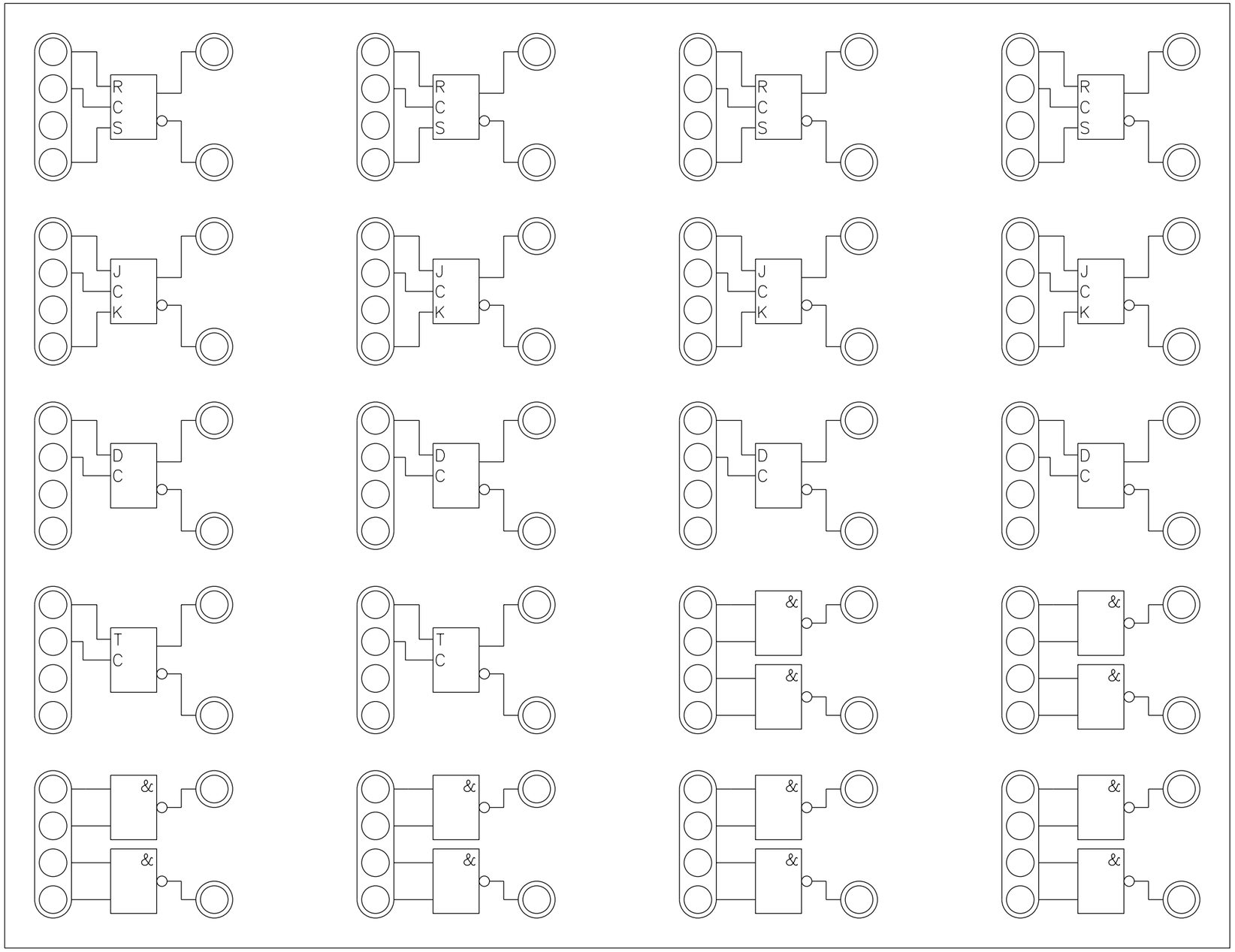
P3 - ROZVRŽENÍ SOUČÁSTEK Č. 2

M 1:1

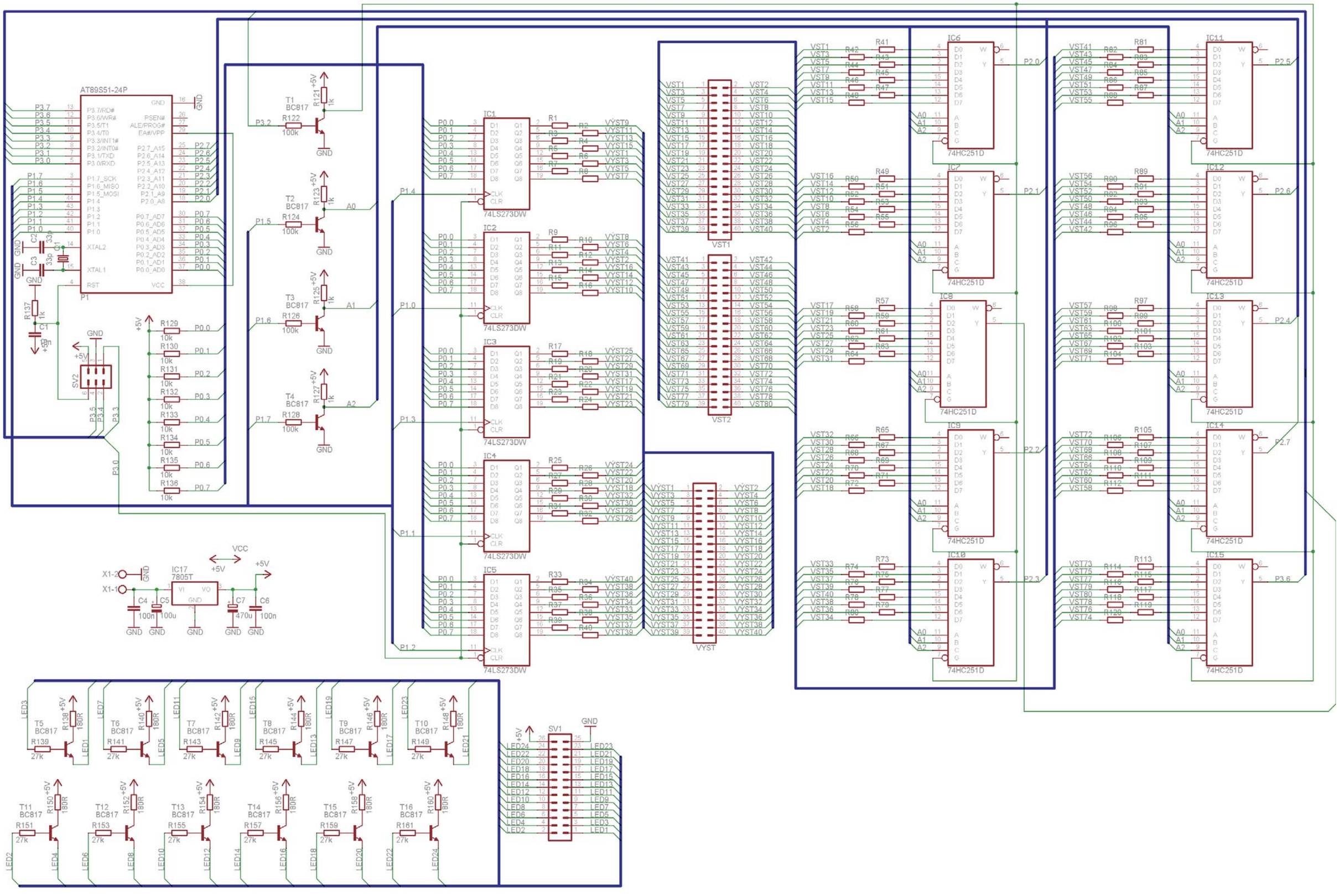


P4 - ROZVRŽENÍ SOUČÁSTEK Č. 3

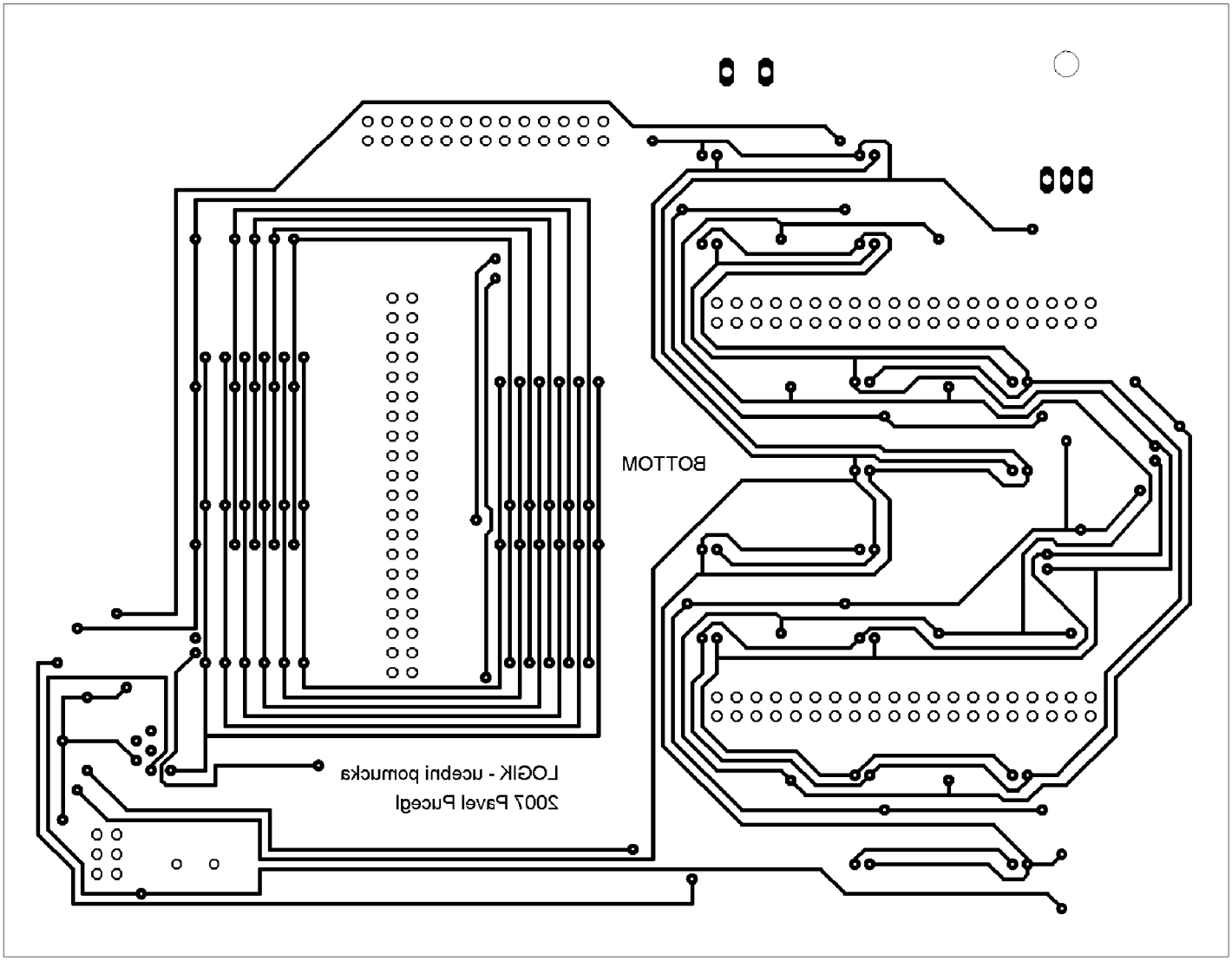
M 1:1



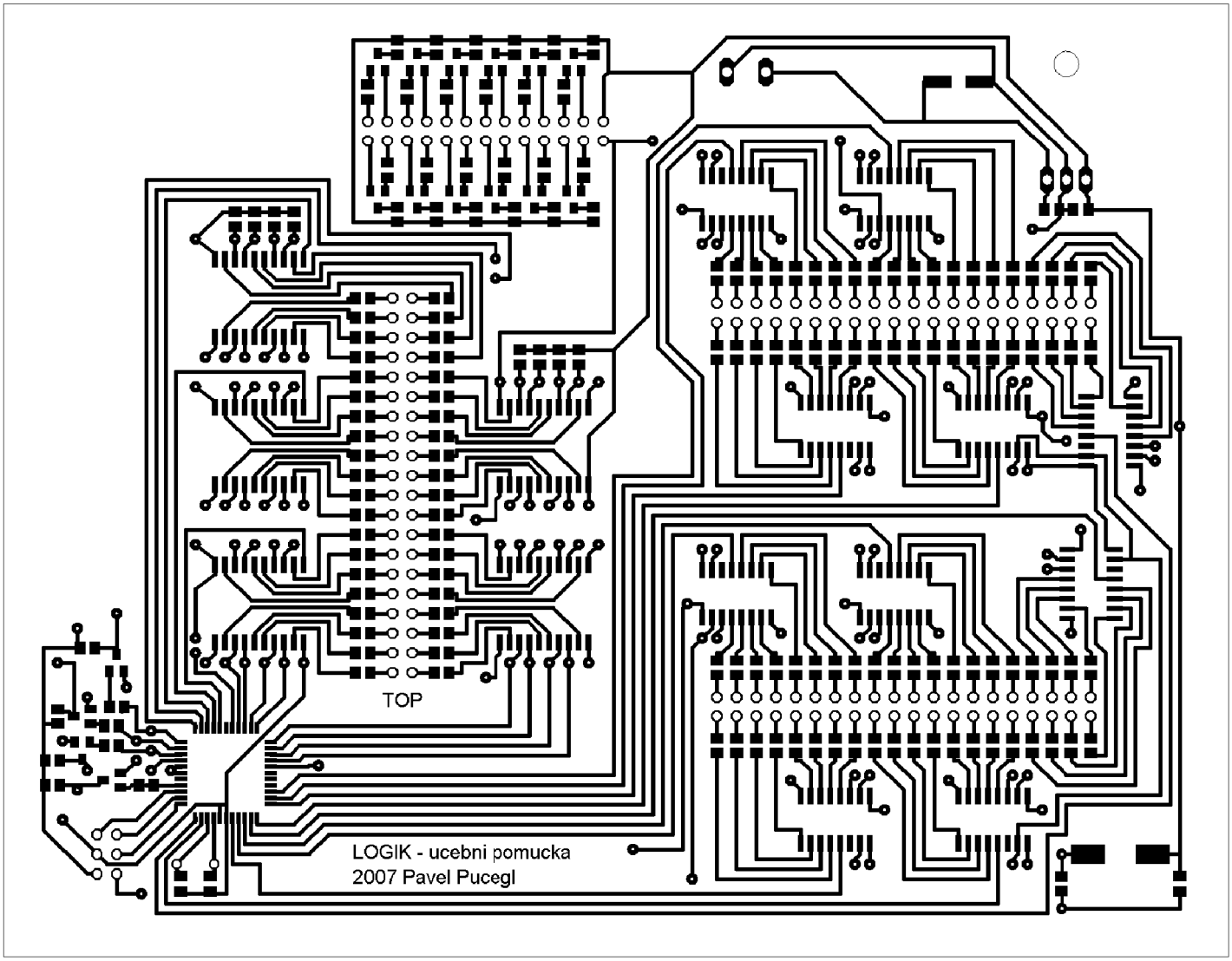
P5 - SCHÉMA ZAPOJENÍ ELEKTRONICKÉ ČÁSTI



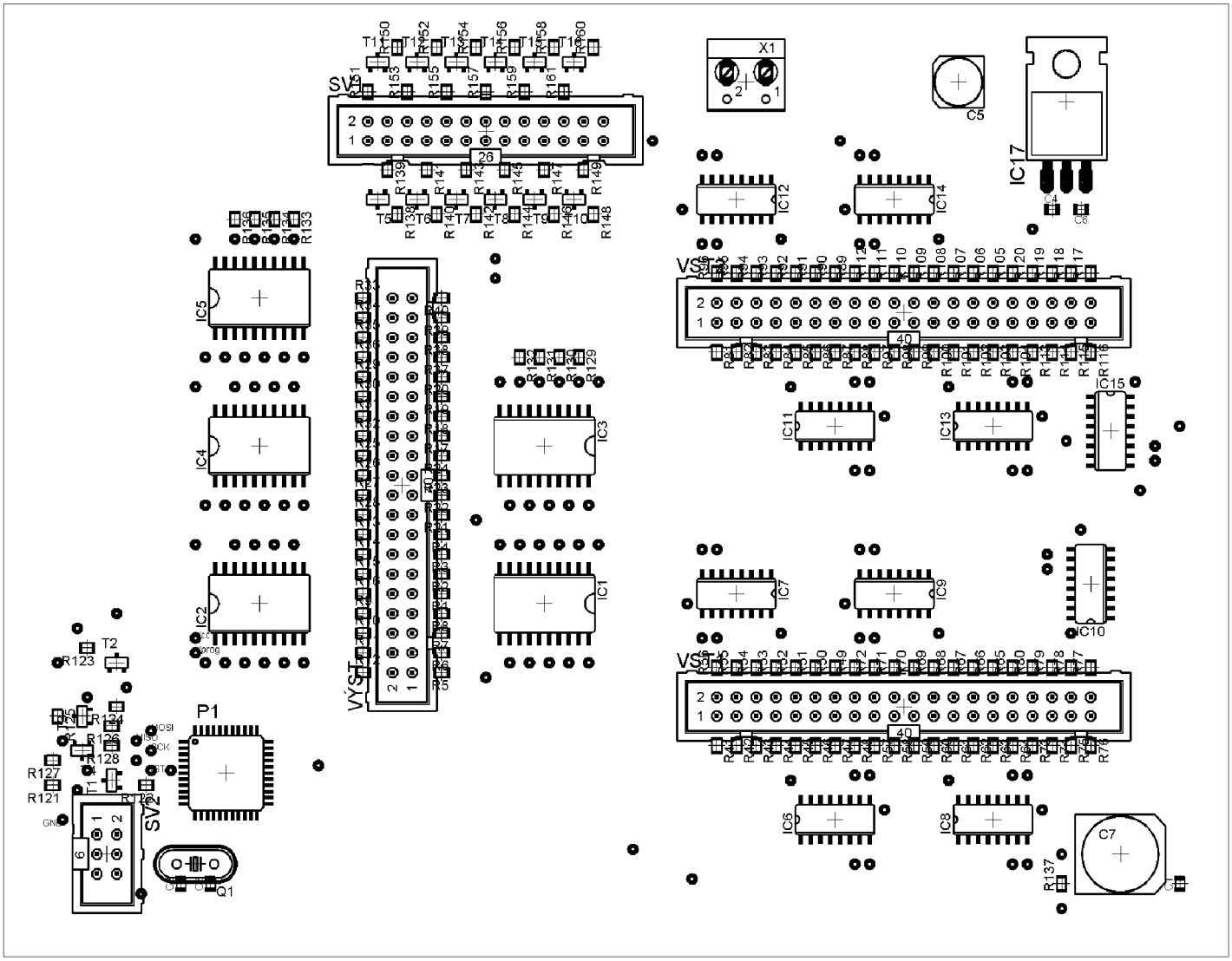
P6 - NÁVRH DESKY PLOŠNÝCH SPOJŮ



P7 - NÁVRH DESKY PLOŠNÝCH SPOJŮ



P8 - OSAZOVACÍ PLÁN



P9 - ZAPOJENÍ KONEKTORŮ VSTUPŮ A VÝSTUPŮ

◯ VST72	◯ VÝST10	◯ VST33	◯ VÝST14	◯ VST70	◯ VÝST2	◯ VST35	◯ VÝST6
◯ VST56		◯ VST32		◯ VST54		◯ VST30	
◯ VST41		◯ VST16		◯ VST43		◯ VST14	
◯ VST57	◯ VÝST12	◯ VST1	◯ VÝST16	◯ VST59	◯ VÝST4	◯ VST3	◯ VÝST8
◯ VST68	◯ VÝST26	◯ VST37	◯ VÝST30	◯ VST66	◯ VÝST18	◯ VST39	◯ VÝST22
◯ VST52		◯ VST28		◯ VST50		◯ VST26	
◯ VST45		◯ VST12		◯ VST47		◯ VST10	
◯ VST61	◯ VÝST28	◯ VST5	◯ VÝST32	◯ VST63	◯ VÝST20	◯ VST7	◯ VÝST24
◯ VST64	◯ VÝST39	◯ VST40	◯ VÝST35	◯ VST62	◯ VÝST34	◯ VST38	◯ VÝST38
◯ VST48		◯ VST24		◯ VST46		◯ VST22	
◯ VST49		◯ VST8		◯ VST51		◯ VST6	
◯ VST65	◯ VÝST37	◯ VST9	◯ VÝST33	◯ VST67	◯ VÝST36	◯ VST11	◯ VÝST40
◯ VST60	◯ VÝST23	◯ VST36	◯ VÝST19	◯ VST58	◯ VÝST31	◯ VST34	◯ VÝST27
◯ VST44		◯ VST20		◯ VST42		◯ VST18	
◯ VST53		◯ VST4		◯ VST55		◯ VST2	
◯ VST69	◯ VÝST21	◯ VST13	◯ VÝST17	◯ VST71	◯ VÝST29	◯ VST15	◯ VÝST25
◯ VST17	◯ VÝST7	◯ VST21	◯ VÝST3	◯ VST25	◯ VÝST15	◯ VST29	◯ VÝST11
◯ VST73		◯ VST77		◯ VST80		◯ VST76	
◯ VST19		◯ VST23		◯ VST27		◯ VST31	
◯ VST75	◯ VÝST5	◯ VST79	◯ VÝST1	◯ VST78	◯ VÝST13	◯ VST74	◯ VÝST9

P10 - UMÍSTĚNÍ HODNOT VSTUPŮ A VÝSTUPŮ V PAMĚTI

◯ 20.7h	◯ 30.7h	◯ 20.3h	◯ 30.5h	◯ 21.7h	◯ 30.3h	◯ 21.3h	◯ 30.1h
◯ 20.6h	7Fh	◯ 20.2h	7Eh	◯ 21.6h	7Dh	◯ 21.2h	7Ch
◯ 20.5h		◯ 20.1h		◯ 21.5h		◯ 21.1h	
◯ 20.4h	◯ 30.6h	◯ 20.0h	◯ 30.4h	◯ 21.4h	◯ 30.2h	◯ 21.0h	◯ 30.0h
◯ 22.7h	◯ 31.7h	◯ 22.3h	◯ 31.5h	◯ 23.7h	◯ 31.3h	◯ 23.3h	◯ 31.1h
◯ 22.6h	7Bh	◯ 22.2h	7Ah	◯ 23.6h	79h	◯ 23.2h	78h
◯ 22.5h		◯ 22.1h		◯ 23.5h		◯ 23.1h	
◯ 22.4h	◯ 31.6h	◯ 22.0h	◯ 31.4h	◯ 23.4h	◯ 31.2h	◯ 23.0h	◯ 31.0h
◯ 24.7h	◯ 32.7h	◯ 24.3h	◯ 32.5h	◯ 25.7h	◯ 32.3h	◯ 25.3h	◯ 32.1h
◯ 24.6h	77h	◯ 24.2h	76h	◯ 25.6h	75h	◯ 25.2h	74h
◯ 24.5h		◯ 24.1h		◯ 25.5h		◯ 25.1h	
◯ 24.4h	◯ 32.6h	◯ 24.0h	◯ 32.4h	◯ 25.4h	◯ 32.2h	◯ 25.0h	◯ 32.0h
◯ 26.7h	◯ 33.7h	◯ 26.3h	◯ 33.5h	◯ 27.7h	◯ 33.3h	◯ 27.3h	◯ 33.1h
◯ 26.6h	73h	◯ 26.2h	72h	◯ 27.6h	71h	◯ 27.2h	70h
◯ 26.5h		◯ 26.1h		◯ 27.5h		◯ 27.1h	
◯ 26.4h	◯ 33.6h	◯ 26.0h	◯ 33.4h	◯ 27.4h	◯ 33.2h	◯ 27.0h	◯ 33.0h
◯ 28.7h	◯ 34.7h	◯ 28.3h	◯ 34.5h	◯ 29.7h	◯ 34.3h	◯ 29.3h	◯ 34.1h
◯ 28.6h	6Fh	◯ 28.2h	6Eh	◯ 29.6h	6Dh	◯ 29.2h	6Ch
◯ 28.5h		◯ 28.1h		◯ 29.5h		◯ 29.1h	
◯ 28.4h	◯ 34.6h	◯ 28.0h	◯ 34.4h	◯ 29.4h	◯ 34.2h	◯ 29.0h	◯ 34.0h

P11 - VÝPIS PROGRAMU MIKROPROCESORU

```
org 0                           ;nastaveni pocatku na adresu 0 v pa-
                                ;meti programu
;I N I C I A L I Z A C E
            mov P1,#0h          ;vymazani bran
            mov P2,#0FFh
            mov P3,#11111010b   ;reset DKO
            nop
            setb P3.0

            mov 30h,#01010101b  ;nastaveni prvotnich hodnot vystupu
            mov 31h,#01010101b  ;eventuelnich SLO, v pripade KLO nema
            mov 32h,#01010101b  ;tato hodnota vliv, u KLO neni pred.
            mov 33h,#01010101b  ;hodnota podstatna
            mov 34h,#01010101b


;T E L O   P R O G R A M U
                                ;rozpoznani programu z prepinacu
ZAC:        jnb P3.5,PR1234     ;?nejvyssi bit=0 -> program 1-4
            jnb P3.4,PR56       ;program 5-8,?druhy bit=0 -> prog.5-6
            jnb P3.3,PR7        ;program 7 a 8,?treti bit=0 -> prog.7
            lcall PROG8         ;zbyva program 8, volani programu 8
            sjmp ZAC            ;program 8 provedem -> zacatek
PR7:        lcall PROG7         ;volani programu 7
            sjmp ZAC
PR56:       jnb P3.3,PR5        ;program 5 a 6,?treti bit=0 -> prog.5
            lcall PROG6         ;zbyva program 6, volani programu 6
            sjmp ZAC
PR5:        lcall PROG5         ;volani programu 5
            sjmp ZAC
PR1234:     jnb P3.4,PR12       ;program 1-4,?druhy bit=0 -> prog.1-2
            jnb P3.3,PR3        ;program 3 a 4,?treti bit=0 -> prog.3
            lcall PROG4         ;zbyva program 4, volani programu 4
            sjmp ZAC
PR3:        lcall PROG3         ;volani programu 3
            sjmp ZAC
PR12:       jnb P3.3,PR1        ;program 1 a 2,?treti bit=0 -> prog.1
            lcall PROG2         ;zbyva program 2, volani programu 2
            sjmp ZAC
PR1:        lcall PROG1         ;volani programu 1
            sjmp ZAC

;N A C I T A N I   V S T U P U
NACTI:      mov 28h,#00h        ;nulovani pameti z duvodu pouziteho
            mov 29h,#00h        ;ukladani po bitech
            mov P1,#11100000b   ;nastaveni adresy 0 MUXum
            lcall ZPO
            setb P3.2           ;nastaveni OE MUXum
            lcall ZPO
            mov 20h,P2          ;nacteni 8 vstupu -> ulozeni do pam.
            mov A,P3            ;nacteni zbylych 2 vstupu z P3
            clr P3.2            ;nacteno -> vypnuti OE MUXu
            anl A,#11000000b    ;maskovani dat vstupu z P3
            orl 28h,A           ;ulozeni 2 vstupu z P3 do pameti

            mov P1,#11000000b   ;nastaveni adresy 1 MUXum
            lcall ZPO
            setb P3.2           ;zbytek stale stejny pro vsechny
            lcall ZPO
```

```
mov 21h,P2              ;adresy s rozdilem ukladani 2 vstupu
mov A,P3                ;nacitanych z brany P3
clr P3.2
anl A,#11000000b        ;maskovani prebytecnych bitu
rr A                    ;posunuti bitu o dve mista vpravo tj.
rr A                    ;tak, jak se maji ulozit
orl 28h,A               ;ulozeni 2 bitu do pameti

mov P1,#10100000b
lcall ZPO
setb P3.2
lcall ZPO
mov 22h,P2
mov A,P3
clr P3.2
anl A,#11000000b        ;maskovani
swap A                  ;prehozeni horniho a dolniho pulbyte
orl 28h,A               ;cimz se 2 bity dostanou na sva mista

mov P1,#10000000b
lcall ZPO
setb P3.2
lcall ZPO
mov 23h,P2
mov A,P3
clr P3.2
anl A,#11000000b        ;maskovani
rl A                    ;posunuti dvou bitu o dve mista vlevo
rl A                    ;tj z bitu 7 na 1 a z bitu 6 na 0
orl 28h,A

mov P1,#01100000b       ;zbytek dale stale obdobny
lcall ZPO
setb P3.2               ;ukladani na 29h (dalsi adresu)
lcall ZPO
mov 24h,P2
mov A,P3
clr P3.2
anl A,#11000000b
orl 29h,A

mov P1,#01000000b
lcall ZPO
setb P3.2
lcall ZPO
mov 25h,P2
mov A,P3
clr P3.2
anl A,#11000000b
rr A
rr A
orl 29h,A

mov P1,#00100000b
lcall ZPO
setb P3.2
lcall ZPO
mov 26h,P2
mov A,P3
clr P3.2
anl A,#11000000b
```

```
            swap A
            orl 29h,A

            mov P1,#00000000b
            lcall ZPO
            setb P3.2
            lcall ZPO
            mov 27h,P2
            mov A,P3
            clr P3.2
            anl A,#11000000b
            rl A
            rl A
            orl 29h,A

;Z A P I S   N A   V Y S T U P Y
            mov P0,30h              ;poslani hodnoty na vstupy DKO
            setb P1.0               ;zachyceni hodnot DKO1 (hodin. sig.)
            clr P1.0                ;konec hodinoveho signalu

            mov P0,31h              ;dale se opakuje, meni se pouze bit
            setb P1.1               ;hodinoveho signalu a adresa pameti
            clr P1.1                ;odkud se data posilaji

            mov P0,32h
            setb P1.2
            clr P1.2

            mov P0,33h
            setb P1.3
            clr P1.3

            mov P0,34h
            setb P1.4
            clr P1.4
            ret

;V Y P O C T O V E   P O D P R O G R A M Y

;P R O G R A M   1
PROG1:      lcall NACTI             ;nacteni/zapis V/V
                                    ;zakladni kostra programu pro KLO
            mov 30h,#0              ;nulovani vystupu pro snazsi ukladani
            mov 31h,#0              ;protoze jde o KLO, neni predchozi
            mov 32h,#0              ;hodnota podstatna
            mov 33h,#0
            mov 34h,#0

;PRVNI RADA
            mov A,20h               ;vlozeni vstupu do akumulatoru
            lcall NEGACE            ;volani vypoctu log. funkce
            orl 30h,A               ;zapis vysledku do pameti vystupu

            mov A,20h               ;vlozeni vstupu do akumulatoru
            swap A                  ;posun vstupu na spravne misto
            lcall NEGACE            ;volani vypoctu log. funkce
            rr A                    ;posun vysledku na spravne misto
            rr A
            orl 30h,A               ;zapis vysledku do pameti vystupu

            mov A,21h               ;vlozeni vstupu do akumulatoru
```

```
            lcall NEGACE            ;volani vypoctu log. funkce
            swap A                  ;posun vysledku na spravne misto
            orl 30h,A               ;zapis vysledku do pameti vystupu

            mov A,21h               ;vlozeni vstupu do akumulatoru
            swap A                  ;posun vstupu na spravne misto
            lcall NEGACE            ;volani vypoctu log. funkce
            rl A                    ;posun vysledku na spravne misto
            rl A
            orl 30h,A               ;zapis vysledku do pameti vystupu

;DRUHA RADA
            mov A,22h               ;dale se vse opakuje stejne,  pouze
            lcall AND2              ;s rozdily adres v pameti vstupu a
            orl 31h,A               ;vystupu

            mov A,22h
            swap A
            lcall AND2
            rr A
            rr A
            orl 31h,A

            mov A,23h
            lcall NAND2
            swap A
            orl 31h,A

            mov A,23h
            swap A
            lcall NAND2
            rl A
            rl A
            orl 31h,A

;TRETI RADA
            mov A,24h
            lcall OR2
            orl 32h,A

            mov A,24h
            swap A
            lcall OR2
            rr A
            rr A
            orl 32h,A

            mov A,25h
            lcall NOR2
            swap A
            orl 32h,A

            mov A,25h
            swap A
            lcall NOR2
            rl A
            rl A
            orl 32h,A

;CTVRTA RADA
            mov A,26h
```

```
            lcall XOR
            orl 33h,A

            mov A,26h
            swap A
            lcall XOR
            rr A
            rr A
            orl 33h,A

            mov A,27h
            lcall XNOR
            swap A
            orl 33h,A

            mov A,27h
            swap A
            lcall XNOR
            rl A
            rl A
            orl 33h,A

;PATA RADA
            mov A,28h
            lcall AND4
            orl 34h,A
            mov A,28h               ;opetovne vlozeni stejneho vstupu do A
            lcall NAND4             ;vypocet druhe log. funkce
            rr A                    ;posun vysledku na patricne misto
            orl 34h,A               ;ulozeni vysledku

            mov A,28h
            swap A
            lcall AND4
            rr A
            rr A
            orl 34h,A
            mov A,28h
            swap A
            lcall NAND4
            swap A
            rl A
            orl 34h,A

            mov A,29h
            lcall OR4
            swap A
            orl 34h,A
            mov A,29h
            lcall NOR4
            swap A
            rr A
            orl 34h,A

            mov A,29h
            swap A
            lcall OR4
            rl A
            rl A
            orl 34h,A
            mov A,29h
```

```
            swap A
            lcall NOR4
            rl A
            orl 34h,A

            ret

;P R O G R A M   2
PROG2:      lcall NACTI

            mov 30h,#0
            mov 31h,#0
            mov 32h,#0
            mov 33h,#0
            mov 34h,#0

;PRVNI RADA
            mov A,20h
            lcall NAND2
            orl 30h,A

            mov A,20h
            swap A
            lcall NAND2
            rr A
            rr A
            orl 30h,A

            mov A,21h
            lcall NAND2
            swap A
            orl 30h,A

            mov A,21h
            swap A
            lcall NAND2
            rl A
            rl A
            orl 30h,A

;DRUHA RADA
            mov A,22h
            lcall NAND2
            orl 31h,A

            mov A,22h
            swap A
            lcall NAND2
            rr A
            rr A
            orl 31h,A

            mov A,23h
            lcall NOR2
            swap A
            orl 31h,A

            mov A,23h
            swap A
            lcall NOR2
            rl A
```

```
            rl A
            orl 31h,A

;TRETI RADA
            mov A,24h
            lcall NOR2
            orl 32h,A

            mov A,24h
            swap A
            lcall NOR2
            rr A
            rr A
            orl 32h,A

            mov A,25h
            lcall NOR2
            swap A
            orl 32h,A

            mov A,25h
            swap A
            lcall NOR2
            rl A
            rl A
            orl 32h,A

;CTVRTA RADA
            mov A,26h
            lcall NAND4
            orl 33h,A
            mov A,26h
            lcall NOR4
            rr A
            orl 33h,A

            mov A,26h
            swap A
            lcall NAND4
            rr A
            rr A
            orl 33h,A
            mov A,26h
            swap A
            lcall NOR4
            swap A
            rl A
            orl 33h,A

            mov A,27h
            lcall NAND4
            swap A
            orl 33h,A
            mov A,27h
            lcall NOR4
            swap A
            rr A
            orl 33h,A

            mov A,27h
            swap A
```

```
            lcall NAND4
            rl A
            rl A
            orl 33h,A
            mov A,27h
            swap A
            lcall NOR4
            rl A
            orl 33h,A

;PATA RADA
            mov A,28h
            lcall NAND4
            orl 34h,A
            mov A,28h
            lcall NOR4
            rr A
            orl 34h,A

            mov A,28h
            swap A
            lcall NAND4
            rr A
            rr A
            orl 34h,A
            mov A,28h
            swap A
            lcall NOR4
            swap A
            rl A
            orl 34h,A

            mov A,29h
            lcall NAND4
            swap A
            orl 34h,A
            mov A,29h
            lcall NOR4
            swap A
            rr A
            orl 34h,A

            mov A,29h
            swap A
            lcall NAND4
            rl A
            rl A
            orl 34h,A
            mov A,29h
            swap A
            lcall NOR4
            rl A
            orl 34h,A

            ret

;P R O G R A M   3
PROG3:      lcall NACTI

;PRVNI RADA                          ;zakladni kostra programu pro SLO
            mov B,30h               ;ulozeni predchoziho vysledku do B
```

```
            anl B,#11000000b        ;maskovani predch. vysledku
            mov A,20h               ;ulozeni vstupu do A
            jnb 7Fh,NIC1            ;testovani zda predchozi CLK bylo 0
            jb Acc.6,DALSI1         ;CLK bylo 1, test zda ted je 0
            lcall SLORS             ;CLK bylo 1, je 0 => volej vypocet
            anl 30h,#00111111b      ;maskovani v pameti vystupu
            orl 30h,A               ;zapis vysledku do pameti vystupu
            clr 7Fh                 ;nastaveni minuleho CLK
            sjmp DALSI1             ;preskok na dalsi blok
NIC1:       jnb Acc.6,DALSI1        ;CLK bylo 0, test zda ted je 0
            setb 7Fh                ;CLK bylo 0, ted je 1, nast. min.CLK
DALSI1:                             ;dale se opakuje, pouze s drobnymy
                                    ;zmenami kvuli maskovani a posunu dat
                                    ;na patricna mista
            mov A,30h               ;posun a maskovani vysledku patriciho
            rl A                    ;ulozit do B v A, protoze v B nelze
            rl A                    ;rotovat bity
            anl A,#11000000b
            mov B,A
            mov A,20h
            swap A
            jnb 7Eh,NIC2
            jb Acc.6,DALSI2
            lcall SLORS
            rr A
            rr A
            anl 30h,#11001111b
            orl 30h,A
            clr 7Eh
            sjmp DALSI2
NIC2:       jnb Acc.6,DALSI2
            setb 7Eh
DALSI2:
            mov A,30h
            swap A
            anl A,#11000000b
            mov B,A
            mov A,21h
            jnb 7Dh,NIC3
            jb Acc.6,DALSI3
            lcall SLORS
            swap A
            anl 30h,#11110011b
            orl 30h,A
            clr 7Dh
            sjmp DALSI3
NIC3:       jnb Acc.6,DALSI3
            setb 7Dh
DALSI3:
            mov A,30h
            rr A
            rr A
            anl A,#11000000b
            mov B,A
            mov A,21h
            swap A
            jnb 7Ch,NIC4
            jb Acc.6,DALSI4
            lcall SLORS
            rl A
            rl A
```

```
            anl 30h,#11111100b
            orl 30h,A
            clr 7Ch
            sjmp DALSI4
NIC4:       jnb Acc.6,DALSI4
            setb 7Ch
DALSI4:

;DRUHA RADA
            mov B,31h
            anl B,#11000000b
            mov A,22h
            jnb 7Bh,NIC5
            jb Acc.6,DALSI5
            lcall SLOJK
            anl 31h,#00111111b
            orl 31h,A
            clr 7Bh
            sjmp DALSI5
NIC5:       jnb Acc.6,DALSI5
            setb 7Bh
DALSI5:
            mov A,31h
            rl A
            rl A
            anl A,#11000000b
            mov B,A
            mov A,22h
            swap A
            jnb 7Ah,NIC6
            jb Acc.6,DALSI6
            lcall SLOJK
            rr A
            rr A
            anl 31h,#11001111b
            orl 31h,A
            clr 7Ah
            sjmp DALSI6
NIC6:       jnb Acc.6,DALSI6
            setb 7Ah
DALSI6:
            mov A,31h
            swap A
            anl A,#11000000b
            mov B,A
            mov A,23h
            jnb 79h,NIC7
            jb Acc.6,DALSI7
            lcall SLOJK
            swap A
            anl 31h,#11110011b
            orl 31h,A
            clr 79h
            sjmp DALSI7
NIC7:       jnb Acc.6,DALSI7
            setb 79h
DALSI7:
            mov A,31h
            rr A
            rr A
            anl A,#11000000b
```

```
            mov B,A
            mov A,23h
            swap A
            jnb 78h,NIC8
            jb Acc.6,DALSI8
            lcall SLOJK
            rl A
            rl A
            anl 31h,#11111100b
            orl 31h,A
            clr 78h
            sjmp DALSI8
NIC8:       jnb Acc.6,DALSI8
            setb 78h
DALSI8:

;TRETI RADA
            mov B,32h
            anl B,#11000000b
            mov A,24h
            jnb 77h,NIC9
            jb Acc.6,DALSI9
            lcall SLOD
            anl 32h,#00111111b
            orl 32h,A
            clr 77h
            sjmp DALSI9
NIC9:       jnb Acc.6,DALSI9
            setb 77h
DALSI9:
            mov A,32h
            rl A
            rl A
            anl A,#11000000b
            mov B,A
            mov A,24h
            swap A
            jnb 76h,NIC10
            jb Acc.6,DALSI10
            lcall SLOD
            rr A
            rr A
            anl 32h,#11001111b
            orl 32h,A
            clr 76h
            sjmp DALSI10
NIC10:      jnb Acc.6,DALSI10
            setb 76h
DALSI10:
            mov A,32h
            swap A
            anl A,#11000000b
            mov B,A
            mov A,25h
            jnb 75h,NIC11
            jb Acc.6,DALSI11
            lcall SLOD
            swap A
            anl 32h,#11110011b
            orl 32h,A
            clr 75h
```

```
            sjmp DALSI11
NIC11:      jnb Acc.6,DALSI11
            setb 75h
DALSI11:
            mov A,32h
            rr A
            rr A
            anl A,#11000000b
            mov B,A
            mov A,25h
            swap A
            jnb 74h,NIC12
            jb Acc.6,DALSI12
            lcall SLOD
            rl A
            rl A
            anl 32h,#11111100b
            orl 32h,A
            clr 74h
            sjmp DALSI12
NIC12:      jnb Acc.6,DALSI12
            setb 74h
DALSI12:

;CTVRTA RADA
            mov B,33h
            anl B,#11000000b
            mov A,26h
            jnb 73h,NIC13
            jb Acc.6,DALSI13
            lcall SLOT
            anl 33h,#00111111b
            orl 33h,A
            clr 73h
            sjmp DALSI13
NIC13:      jnb Acc.6,DALSI13
            setb 73h
DALSI13:
            mov A,33h
            rl A
            rl A
            anl A,#11000000b
            mov B,A
            mov A,26h
            swap A
            jnb 72h,NIC14
            jb Acc.6,DALSI14
            lcall SLOT
            rr A
            rr A
            anl 33h,#11001111b
            orl 33h,A
            clr 72h
            sjmp DALSI14
NIC14:      jnb Acc.6,DALSI14
            setb 72h
DALSI14:
            anl 33h,#11110000b      ;odtud jsou funkce kombinacni
            mov 34h,#0              ;program pokracuje podle kostry
                                    ;podprogramu pro vypocet KLO
            mov A,27h
```

```
            lcall NAND2
            swap A
            orl 33h,A

            mov A,27h
            swap A
            lcall NAND2
            rl A
            rl A
            orl 33h,A

;PATA RADA
            mov A,28h
            lcall NAND2
            orl 34h,A

            mov A,28h
            swap A
            lcall NAND2
            rr A
            rr A
            orl 34h,A

            mov A,29h
            lcall NAND2
            swap A
            orl 34h,A

            mov A,29h
            swap A
            lcall NAND2
            rl A
            rl A
            orl 34h,A

            ret

;P R O G R A M Y   4 - 8
PROG4:      ret                     ;prazdne programy, pocitany jako
                                    ;moznost rozsireni do budoucna
PROG5:      ret

PROG6:      ret

PROG7:      ret

PROG8:      ret

;V Y P O C T Y   L O G I C K Y C H   F U N K C I
NEGACE:     push Acc                ;zaloha akumulatoru
            xrl A,#10000000b        ;vypocet prvni negace
            anl A,#10000000b        ;nulovani zbylych mist
            mov B,A                 ;zaloha vysledku do B
            pop Acc                 ;obnoveni ze zalohy
            rl A                    ;posun druheho NOTu na patricne misto
            rl A
            xrl A,#01000000b        ;vypocet druhe negace
            anl A,#01000000b        ;maskovani vysledku
            orl A,B
            ret
```

```
AND2:       mov B,A             ;vlozeni vstupu do B
            rl A                ;posun druheho vst. proti prvnimu v B
            anl A,B             ;logicky soucin A a B
            anl A,#10000000b    ;vysl. v prvnim bitu, ostatni nulujeme
            push Acc            ;zaloha vysledku do zasobniku
            mov A,B             ;obnova vstupu z B
            rl A                ;posun tretiho vstupu na druhe misto
            mov B,A             ;kvuli umisteni vysledku
            rl A                ;posun ctvrteho vstupu proti tretimu
            anl A,B             ;logicky soucin tretiho a ctvrteho vst.
            anl A,#01000000b    ;nulovani ostatnich bitu
            pop B               ;obnova zalohy ze zasobniku do B
            orl A,B             ;kompletace vysledku obou hradel
            ret

OR2:        mov B,A             ;OR i XOR jsou obdobne jako NAND,
            rl A                ;pouze se lisi log. funkce
            orl A,B
            anl A,#10000000b
            push Acc
            mov A,B
            rl A
            mov B,A
            rl A
            orl A,B
            anl A,#01000000b
            pop B
            orl A,B
            ret

XOR:        mov B,A
            rl A
            xrl A,B
            anl A,#10000000b
            push Acc
            mov A,B
            rl A
            mov B,A
            rl A
            xrl A,B
            anl A,#01000000b
            pop B
            orl A,B
            ret

NAND2:      call AND2           ;volani nenegovane funkce
            xrl A,#11000000b    ;negace nenegovane funkce
            ret

NOR2:       call OR2            ;obdobne jako NAND2
            xrl A,#11000000b
            ret

XNOR:       call XOR            ;obdobne jako NAND2
            xrl A,#11000000b
            ret

AND4:       call AND2           ;volani 2x 2-vstupove funkce, tim
            call AND2           ;dostaneme oba vysledky jakoby na vstup
            anl A,#10000000b    ;dalsi funkce a podle distributivniho
            ret                 ;zakona opet provedeme log. funkci
```

```
OR4:        call OR2              ;obdobne jako AND4
            call OR2
            anl A,#10000000b
            ret

NAND4:      call AND4             ;obdobne jako NAND2
            xrl A,#10000000b
            ret

NOR4:       call OR4              ;obdobne jako NAND2
            xrl A,#10000000b
            ret

SLORS:      jb Acc.7,JER          ;pokud je vstup R, skoc na JER
            jb Acc.4,JES          ;pokud je vstup S, skoc na JES
            mov A,B               ;neni ani R ani S vysledek nezmenen
            ret
JER:        jb Acc.4,NAHOD        ;je R,je-li S,neurcity(nahodny) stav
            clr A                 ;je R, neni S, vynuluj Q
            setb Acc.6            ;nastav QNEG
            ret
JES:        clr A                 ;je S, neni R, nuluj QNEG
            setb Acc.7            ;nastav Q
            ret
NAHOD:      jb 00h,NAH1           ;jako nahodna hodnota se bere hdnota
            clr A                 ;bitu 0 v bitove adresovatelne RAM
            setb Acc.6
            ret
NAH1:       clr A
            setb Acc.7
            ret


SLOJK:      jb Acc.7,JEK          ;obdobne jak RS, zmena v miste, kdyz
            jb Acc.4,JEJ          ;je J i K
            mov A,B
            ret
JEK:        jb Acc.4,NEGUJ
            clr A
            setb Acc.6
            ret
JEJ:        clr A
            setb Acc.7
            ret
NEGUJ:      mov A,B               ;zmena oproti RS, predchozi vysledek
            xrl A,#11000000b      ;ulozeny v B se zneguje a ulozi do A
            ret

SLOD:       anl A,#10000000b      ;nulovani zbytku vstupu
            mov B,A               ;ulozeni vysledku do B
            rr A                  ;vypocet QNEG - posun Q do QNEG
            xrl A,#01000000b      ;negace Q v QNEG
            orl A,B               ;kompletace vysledku Q a QNEG
            ret

SLOT:       jb Acc.7,NEGUJ        ;pokud je vstup T, skoc na vypocet
            mov A,B               ;negace v JK obvodu
            ret
```

```
;Z P O Z D O V A C I   P O D P R O G R A M
ZPO:        mov R7,#10                  ;klasicky zpozdovaci podprog.
            djnz R7,$                   ;s postupnou dekrementaci
            ret                         ;registru

            end
```