Středoškolská technika 2017

Setkání a prezentace prací středoškolských studentů na ČVUT

Návrh procesorového jádra

Lukáš Pácl

Střední průmyslová škola, Česká Lípa
Havlíčkova 426, Česká Lípa

Licenční ujednání

Ve smyslu zákona č. 121/2000 Sb., O právu autorském, o právech souvisejících s právem autorským, ve znění pozdějších předpisů (dále jen autorský zákon) jsou práva k maturitním nebo ročníkovým pracím následující:

Zadavatel má výhradní práva k využití práce, a to včetně komerčních účelů.

Autor práce bez svolení zadavatele nesmí využít práci ke komerčním účelům.

Škola má právo využít práci k nekomerčním a výukovým účelům i bez svolení zadavatele a autora práce.

Prohlášení

Prohlašuji, že jsem svou ročníkovou práci vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze práce jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V České Lípě dne ……

Jméno a příjmení autora

Anotace

Ve své ročníkové práci jsem navrhoval 8-bitové procesorové jádro. Jádro muselo obsahovat aritmeticko-logickou jednotku, registry, instrukční dekodér, řadič a vstup uživatele. Procesor jsem navrhoval v open source programu Logisim verze 2.7.1.

Klíčová slova

Procesor; Aritmeticko-logická jednotka; Řadič; Registr; Instrukční dekodér

Annotation

In my annual work i designed 8-bit proccessor core. The core have to had arithmetic logic unit, registers, instruction decoder, control unit and user input. I designed proccessor using open source program Logisim version 2.7.1.

Keywords

Proccessor; Arithmetic logic unit, Control unit, Register; Instruction decoder

Obsah

1 ÚVOD

Cílem této ročníkové práce je navrhnout jádro procesoru. Jádro bude obsahovat aritmeticko-logickou jednotku, registry, pomocné registry aritmeticko-logické jednotky, řadič a dekodér instrukcí se vstupem pro zadávání operací. Jako vstup bude použito deset bitů pro výběr instrukce a registrů, hodiny a čtyři bity pro reset všech součástí. Jádro procesoru bude jen v základním stavu, kdy nebude použita žádná vnější paměť, čítač a zásobník.

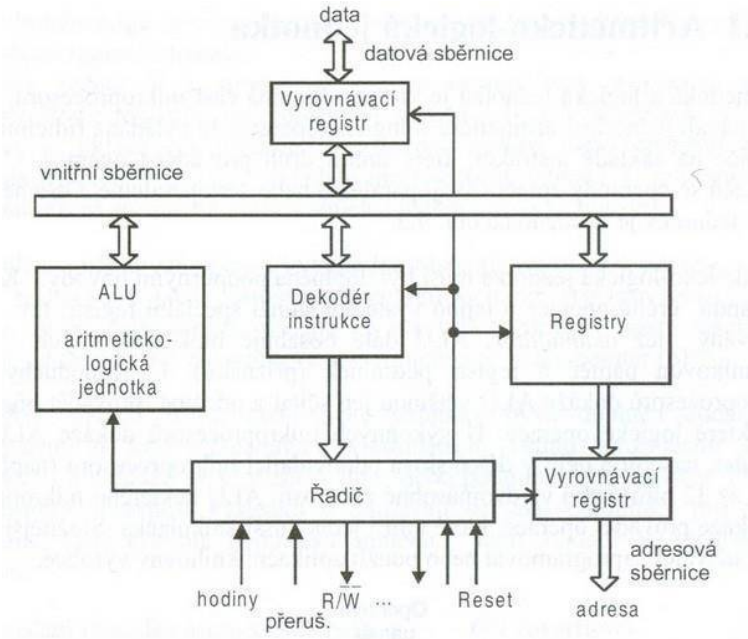
Procesor budu navrhovat v open-source programu Logisim verze 2.7.1. Budu používat předem připravené prvky z tohoto programu tudíž nebudu celý procesor navrhovat ze základních logických hradel. Jednotlivé části procesoru budou navrhnuté do bloků pro větší přehlednost. Součástí práce bude i získání vědomostí o výše zmíněném programu abych měl dostatek znalostí pro realizaci samotného návrhu.

2 PROCESOR

Procesor je integrovaný obvod složený převážně z mnoha tranzistorů. Je to součást, která propojuje veškeré periferie a umožňuje tak chod celého stroje. Toto je důvod, proč je nazýván „mozkem“. Procesory se mohou lišit architekturou, výkonem, instrukční sadou, množstvím registrů, pracovní frekvencí a mnoha dalšími kriterii podle využití, tudíž do mikrovlnky a počítače budou použity naprosto jiné architektury procesoru kde v mikrovlnce bude mnohem jednodušší mnohem míň výkonný procesor s mnohem jednodušší instrukční sadou, jelikož tak univerzální procesor jako je v počítači by byl v mikrovlnce naprosto nevyužit.

2.1 Stavba procesoru

Procesor se skládá z více různých částí, které jsou navzájem propojené a společně umožňují chod procesoru. Každá část má jiný účel a funkci. Zde je schéma jednoduchého procesoru, jak může vypadat.



Obrázek 1

2.1.1 Registry

Registry jsou paměťové jednotky, které jsou schopné uložit jedno „slovo“. Tímto slovem je myšlen sled nul a jedniček o velikosti většinou stejné jako je vstup/výstup aritmeticko-logické jednotky tudíž když aritmeticko-logická jednotka bude mít osmi bitové vstupy tak registr má zpravidla schopnost v sobě uchovat 8 bitů informace (nul a jedniček). Dělí se na více druhů například příznakový registr, pracovní registry, datové registry atd. Některé registry jako například příznakový registr používá procesor interně a programátor k němu nemá přístup, ale jsou i registry ke kterým přístup má, těchto registrů je většinou více aby měl programátor možnost ukládat mezivýsledky operací procesoru. Registry jsou připojeny ke sběrnici pro zápis/čtení dat a výběr registru (adresace) pro zápis/čtení je prováděn řadičem.

2.1.2 ALU

ALU neboli aritmeticko-logická jednotka je část procesoru kde se provádějí všechny aritmetické a logické operace. Princip ALU je poměrně jednoduchý tato jednotka se skládá z dvou n-bitových vstupů, jednoho n-bitového výstupu, většinou ještě jedním příznakovým výstupem o velikosti jednoho bitu a ovládání z řídící jednotky. Na n-bitové vstupy se přivedou skrz pomocné registry (často také nazývané pracovní registry či akumulátory) operandy, které ALU zpracuje podle toho, jaké dostane řídící signály od řadiče čímž se vybere operace. Po zpracování operandů se data pošlou na další pomocný registr u výstupu ALU, který je dále napojen na sběrnici pro zápis dat do paměti či registrů a nastaví se příznak do příznakového registru, což také záleží na operaci, která se provádí. Počet operací, kterými ALU disponuje je dán instrukční sadou procesoru.

2.1.3 Řadič

Řadič je „mozek“ celého procesoru který ovládá jeho chod. Je napojen na všechny části procesoru a řídí jaké části jsou kdy a jak zapnuté. Například jakou operaci má ALU provádět či řízení zápisu/čtení dat do paměti nebo registrů neboli průchod dat, ovládání pomocných registrů atd.

2.1.4 Instrukční dekodér

Instrukční dekodér pracuje úzce ve spojení s řadičem. Princip instrukčního dekodéru je takový, že z paměti či nějakého jiného vstupu jsou přivedeny instrukce psané ve strojovém kódu a dekodér instrukci rozebere na jednotlivé mikroinstrukce, které už jsou zavedeny pomocí řadiče přímo do součástek procesoru. Jedna instrukce se totiž nerovná jeden instrukční cyklus například instrukce pro součet dvou čísel načtených v paměti dekodér rozebere na 4 mikroinstrukce tudíž instrukce součtu trvá čtyři instrukční cykly.

2.1.5 Paměti

Paměť je součást procesoru (nikoliv už jádra), kam se ukládají data. Často se zde nacházejí programy, které se přes sběrnici pošlou do procesoru, který pak daný program provede. Paměť je většinou mnohonásobně pomalejší než registry, a proto pro zpracování přímo instrukcí se používají právě registry a paměť se využívá jen pro kód či pro zápis méně důležitých mezivýsledků, které nejsou tak často potřeba. Paměti se rozdělují na mnoho druhů, zde budou popsány pouze základní druhy pamětí.

2.1.5.1 RAM

RAM neboli „random acces memory“ je polovodičová paměť, která má vlastnost zápisu/čtení dat do/z jednotlivých bytu paměti. Je to volatilitní paměť tudíž po ztrátě napětí ztratí hodnoty v ní zapsané, proto se využívá jako operační paměť, a ne jako dlouhodobé úložiště. Z paměti typu RAM můžeme, jak číst, tak zároveň i zapisovat a to „nekonečně“ mnohokrát a přístup k paměti je opravdu velmi rychlý. Jsou dva základní tipy RAM pamětí, a to DRAM a SRAM neboli statická a dynamická RAM paměť. Rozdíl mezi těmito paměťmi je především ve stylu uchovávání informace kdy statická RAM uchovává informace pomocí bistabilních klopných obvodů kdežto dynamická RAM uchovává informace pomocí malých „kondenzátorů“ tudíž pomocí nábojů.

2.1.5.2 ROM

ROM neboli „read only memory“ je paměť nevolatilní tudíž neztrácí informaci po odpojení napětí, která jak již název napovídá neumožňuje zápis dat. Data jsou zapsány pouze jednou při výrobě paměti, a proto se dříve používali jako paměti pro BIOS v počítačích. Dnes už jsou k vidění dost zřídka kvůli jejich nemožnosti přepsat data. Nahradili ji časem paměti jako PROM „programable read only memory“ kde pomocí speciálního programátoru PROM pamětí může být jednou naprogramován obsah paměti, dále nastoupil EPROM „erasable programable read only memory“, která se nejen dala programovat, ale i pomocí UV světla smazat na samé nuly a znova programovat bohužel takováto paměť se nedá přepisovat věčně a opakovaným

přepisováním se poškodí. Posledním nástupcem je paměť EEPROM „electrically erasable programable read only memory“ kdy paměť může být elektricky smazána a znovu naprogramována, tento typ pamětí se používá dodnes.

2.1.6 Sběrnice

Sběrnice je část procesoru, která zprostředkovává přenos dat mezi jednotlivými částmi procesoru. Je to n-bitů s pár připojenými multiplexory od jednotlivých částí procesoru podle kterých se určí jaká cesta je zrovna propustná a jaká část procesoru má zrovna na sběrnici přístup. Chod sběrnice řídí řadič pomocí řídících signálů.

2.2 Princip fungování procesoru

Hlavní princip fungování procesoru je čtení z paměti program, který řekne co dál dělat. Jakmile se načte program dekodér program rozloží do mikroinstrukcí a podá ho řadiči. Řadič mikroinstrukce pustí do částí procesoru čímž nastaví ALU na určitou potřebnou operaci. Dále se načtou operandy ze zadaných úložišť (většinou registrů) a ALU provede operaci. Po provedení operace se zapíše na dané místo výsledek operace a procesor může provést další instrukci.

2.2.1 Základní instrukce

Základní instrukce každého procesoru jsou průchod informace (většinou z prvního vstupu), negace, logické operace jako AND, OR, XOR atd. a minimálně základní aritmetické operace jako je sčítání, odčítání. V tabulce níže jsou uvedeny tyto základní instrukce a jejich ukázka, jak takové instrukce vypadají a co přesně znamenají.

2.2.2 Řídící signály řadiče

Hlavní práci na fungování procesoru mají řídící signály řadiče, které ovládají, co a jak se má dít. Na obrázku níže je schéma popisující průběh sčítání dvou čísel z registrů. Nejdříve se instrukce z paměti musí dekódovat například z assemblerovského ADD AX, BX se stane v paměti „001 01 11“. Toto je ale stále pouze strojový kód, který se dále musí rozložit na mikroinstrukce, proto se zavede tento strojový kód do dekodéru, který z tohoto kódu udělá například „001 0110101 01“ kde první část kódu je instrukce pro ALU další část nastaví cestu a načte se první operand z registru což je poslední část kódu. Dále proběhne Clk kdy vše, co se nastavilo proběhne. Následně dekodér podá druhou mikroinstrukci, například „001 1011011 10“ kdy se načte druhý operand atd. takže překlad instrukce nakonec může vypadat nějak takto:

001 01 11= 001 0110101 01 Clk 001 0110101 11 Clk 001 0111101 01 Clk 001 0111000 01 Clk

2.2.3 Čtení/zápis do paměti/registrů

Čtení a zápis probíhá pomocí řídících signálů kdy se nastaví sběrnice a vybere se registr nebo část paměti z které se čte nebo se do ní zapisuje. Dále se musí pomocí řídících signálů nastavit,

jestli se bude číst či zapisovat. Jako poslední se jen provede Clk hodin, kdy se zapíše/načte hodnota.

2.2.4 Práce ALU

Práce ALU je řízena řídícími signály z řadiče kdy řadič nastaví operacia podle toho se přepne v ALU cesta a operandy projdou již sestaveným obvodem, který provede danou operaci.

2.3 Architektury procesorů

Jak už bylo zmíněno procesory se mohou lišit a asi nejvýznamnější rozdíly mezi procesory je jejich architektura. Určité přesně dané normované architektury procesorů se vytvořili hlavně proto aby se nemusel psát program pro každý procesor zvlášť tudíž se umožnil přenos programů mezi jednotlivými procesory, které disponovali stejnou architekturou a nemusel se kód upravovat či celý přepsat. Architektur procesorů je opravdu mnoho, a proto tu budou zmíněny pouze jen ty nejznámější používané architektury.

2.3.1 CISC

CISC architektura procesoru je taková, která využívá velmi obsáhlou a složitou sadu instrukcí. Má mnoho instrukcí již složených přímo v hardwaru tudíž nemusí být vymýšleno pomocí jiných menších instrukcí což velmi usnadnilo psaní programů pro takovéto procesory. Po čase používání této architektury se ale zjistilo, že se využívá většinou pouze cca třicet procent instrukcí a ostatní jsou naprosto nevyužity.

2.3.2 RISC

RISC je přesný opak architektury CISC má velmi jednoduchou instrukční sadu složenou z co nejjednodušších instrukcí, které se následně skládají dohromady pro vytvoření složitějších operací tudíž je pouze emuluje což je pro programátora nevýhoda, jelikož si musí sám vytvořit operace, které chce dále používat, ale zase je program efektivnější a více optimalizovaný. Tato architektura disponuje více registry, jelikož se v té době vytvářeli moc pomalé paměti k procesorům, a proto se registry využívali jako nejrychlejší cache paměť.

2.3.3 MISC

Architektura MISC obsahuje podobně jako RISC jednoduché instrukce. Rozdíl oproti RISC je ale takový, že MISC používá velmi malou instrukční sadu a místo registrů využívá hlavně zásobníky tudíž má instrukce zaměřeny právě na práci s nimi a často obsahují zásobníky rovnou v jádře procesoru. Výhodou je zmenšení počtu operandů kde se nemusí adresovat jednotlivé registry, ale pomocné hodnoty si ukládá procesor právě do zásobníků.

3 NÁVRH

Samotný návrh procesoru spočíval zezačátku hlavně z vymyšlení jeho parametrů. Rozhodlo se, že se bude navrhovat 8 bitový procesor s instrukční sadou o deseti instrukcích. Dále proběhlo rozhodnutí, kdy se řeklo, že se bude navrhovat pouze jádro procesoru, tudíž ALU, registry, instrukční dekodér a řadič a nebude se připojovat žádná operační paměť nebude použit čítač ani zásobník, a tudíž instrukce budou zadávány ručně jedna po druhé. Po vymyšlení parametrů se muselo vymyslet základní schéma propojení všech součástí, které je zobrazeno níže. Po základním schématu se přešlo na návrh jednotlivých částí.

3.1 První rozvrhnutí

První rozvrhnutí bylo schematické navrhnutí na papír bez použití programu. Procesorové jádro se rozložilo na jednotlivé bloky, které se dále rozkládaly až do nejmenších logických hradel. Jakmile bylo jádro schematicky rozvrhnuté začal výběr vhodného softwaru pro realizaci návrhu.

3.1.1 Logisim x Deeds

Pro návrh procesorového jádra byl původně vybrán software Deeds. Deeds je simulační program, který dokáže simulovat zapojení logických hradel a kombinačních obvodů. Bohužel po prvotní práci s tímto softwarem se zjistilo mnoho nedostatků. Nedostatečná velikost plochy či nemožnost vytvoření vlastních kombinačních obvodů se stali velkou překážkou ve splnění zadané práce. Pro tyto nedostatky se rozhodlo najít nový simulační program. Po projití mnoha freeware programů se došlo k závěru, že Logisim je nejlepší program pro tuto úlohu. Výhody tohoto programu spočívali v nekonečně velkém návrhovém prostoru, možnosti navrhnutí vlastních kombinačních obvodů, možnost použití zkratky Ctrl+ Z pro vrácení jednoho kroku zpět, automatické vypočítávání změny a propojení součástek při jejím posunu, možnost mít zároveň spuštěnou simulaci při upravování signálů, zobrazování hodnot na sběrnicích a mnoho dalších. Logisim nejen, že velmi zjednodušil návrh, ale dokonce ho až umožnil, jelikož navrhnout procesorové jádro v programu Deeds se ukázalo být nemožné.

3.1.2 Použitá hradla

V celém návrhu se použila pouze hradla NOT, AND, OR a XOR. Ostatní logická hradla by se dala použít také a dokázala by i nahradit již zmíněné použité hradle tudíž by jádro obsahovalo méně druhů hradel, ale o to více by jich bylo potřeba tudíž se usoudilo, že použití těchto hradel je lepší volbou. Níže je uvedena tabulka s chováním jednotlivých hradel.

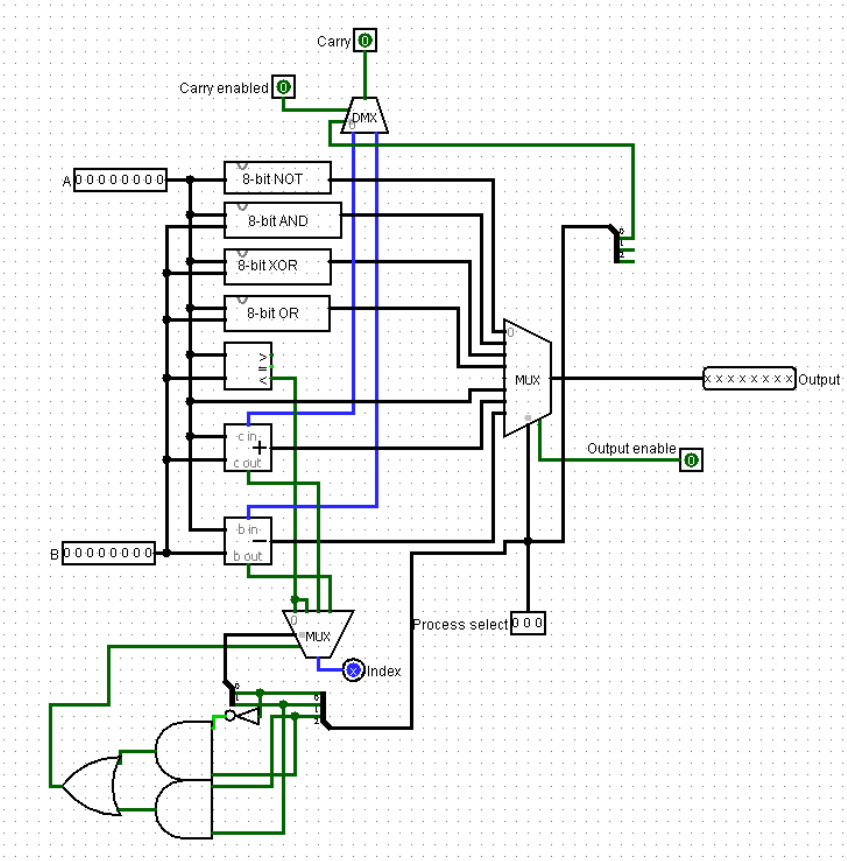
3.1.3 Použité kombinační obvody

Kombinační obvody použité pro návrh zadaného procesorového jádra byly zejména registry, multiplexory, demultiplexory, klopný obvod typu D, sčítačka, odčítačka a komparátor. Registry jejich funkce a využití je uvedeno výše. Multiplexory a demultiplexory jsou kombinační obvody, které slouží pro výběr „cesty pro informaci“. Multiplexor má mnoho n-bitových vstupů

z kterých se pomocí signálu vybírá jaký projde dál do jednoho n-bitového výstupu. Demultiplexor funguje přesně naopak, než multiplexor tudíž je přiveden na jeden n-bitový vstup informace a podle pomocného signálu o n-bitech se rozhodne do jakého z n-výstupů (záleží na počtu bitů pomocného signálu) signál ze vstupu přijde. Sčítačka a odčítačka jsou kombinační obvody kde jeden obvod sečte a druhý odečte od sebe dva n-bitové vstupy a výsledek vyjde z jednoho n-bitového výstupu, přičemž ještě mají vstup pro jeden bit při výpůjčce a jeden byt nastavení příznaku přetečení. Komparátor je kombinační obvod, který porovná dva n-bitové vstupy a výsledek. Klopný obvod typu D je kombinační obvod pro uchování jedné jednobitové informace. Je to hlavní součást registrů.

3.2 ALU

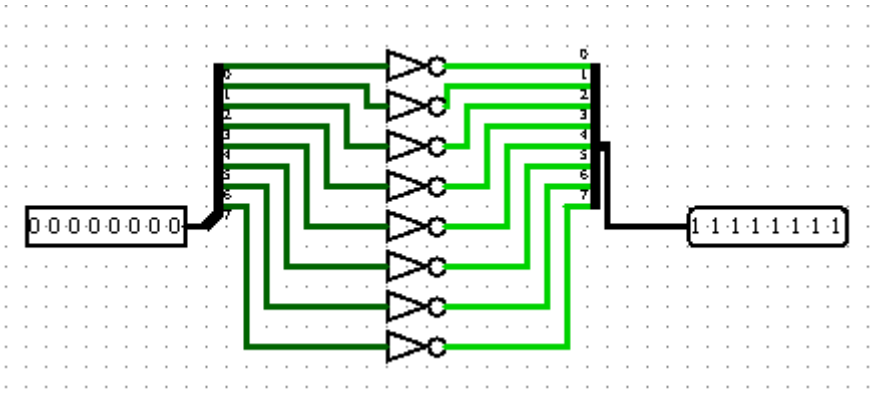
Aritmeticko-logická jednotka byla velmi jednoduchá na návrh. Stačila logická hradla vzájemně propojená pro osm bitů, jelikož má být procesorové jádro 8-bitové. Jelikož všechny kombinační obvody pro instrukce, které byly zadány byly již měl předpřipraveny tak pouze pomocí multiplexoru a demultiplexoru byly spojeny a vyvedeny signály pro příznaky ALU.



Obrázek 2

3.2.1 NOT

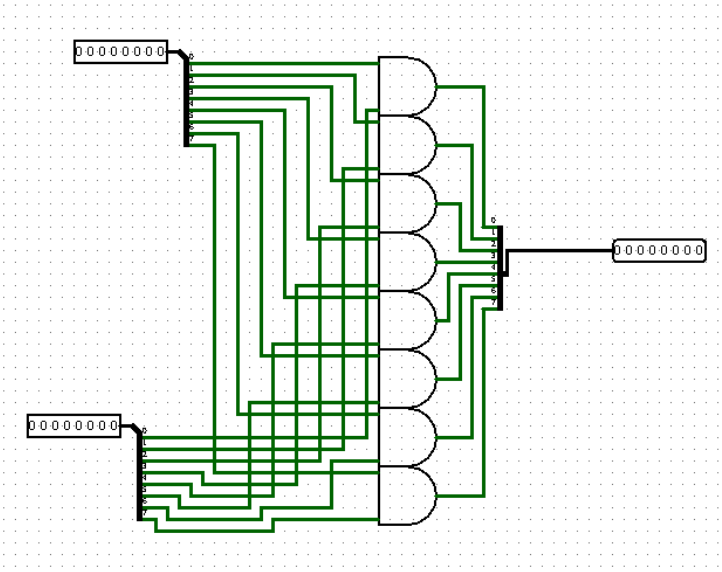
Instrukce NOT byla navrhnuta pomocí osmi hradel NOT kde vstup má osm bitů a výstup také. Při průchodu touto částí aritmeticko-logické jednotky se informace ze vstupu A zneguje. Tudíž například z bitového slova „00011010“ se po průchodu přemění na „11100101“.



Obrázek 3

3.2.2 AND

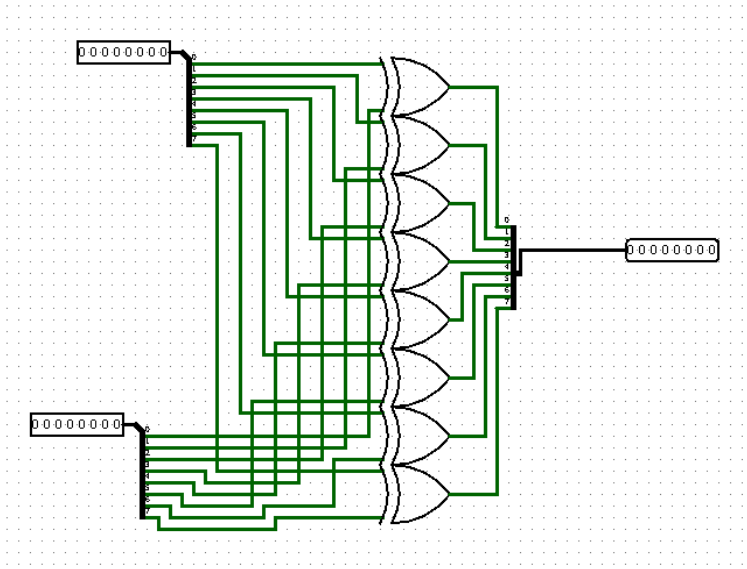
Instrukce AND byla podobně jako instrukce NOT navrhnuta z osmi hradel a disponuje 8-bitovými vstupy a výstupem kde jediný rozdíl jsou dva stupy a použití logického hradla AND. Po průchodu touto částí aritmeticko-logické jednotky se mezi dvěma vstupy provede logická operace AND.



Obrázek 4

3.2.3 XOR

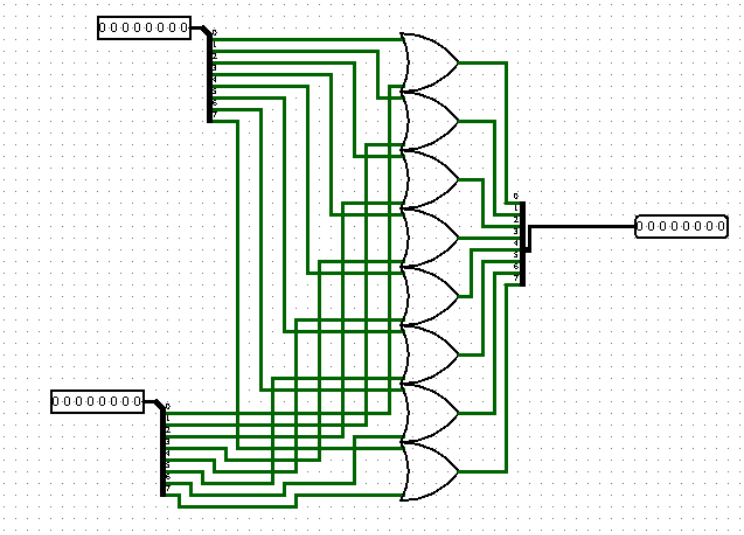
Instrukce XOR byla navrhnuta z osmi hradel a disponuje dvěma 8-bitovými vstupy a jedním výstupem za použití logického hradla XOR. Po průchodu touto částí aritmeticko-logické jednotky se mezi dvěma vstupy provede logická operace XOR.



Obrázek 5

3.2.4 OR

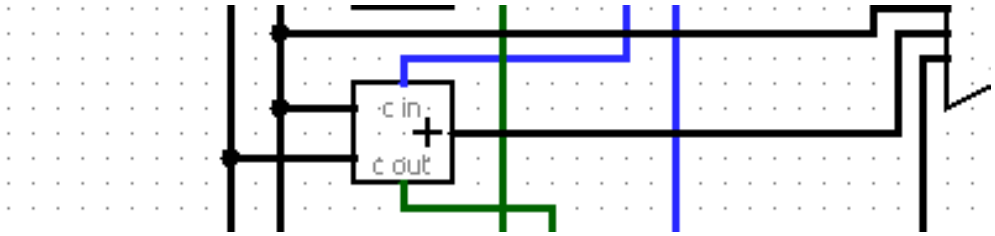
Instrukce OR byla navrhnuta z osmi hradel a disponuje dvěma 8-bitovými vstupy a jedním výstupem za použití logického hradla OR. Po průchodu touto částí aritmeticko-logické jednotky se mezi dvěma vstupy provede logická operace OR.



Obrázek 6

3.2.5 CMP

Instrukce CMP disponuje také dvěma 8-bitovými vstupy, ale na rozdíl od ostatních operací disponuje třemi 1-bitovými výstupy, které vedou přímo do příznakového registru. Tato část obvodu je tvořena kombinačním obvodem zvaným „komparátor“. Po průchodu touto částí aritmeticko-logické jednotky se dva vstupy mezi sebou porovnají a podle toho, jestli bylo první číslo větší, menší či se rovnalo se nastaví příznak.



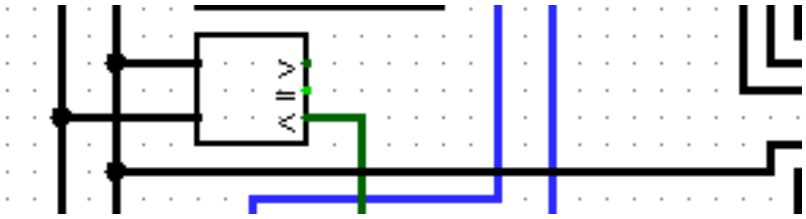
Obrázek 7

3.2.6 NOP

Instrukce NOP je pouze průchod informace ze vstupu A tudíž nebylo zapotřebí využití žádných logických hradel či kombinačních obvodů.

3.2.7 ADD/ADC

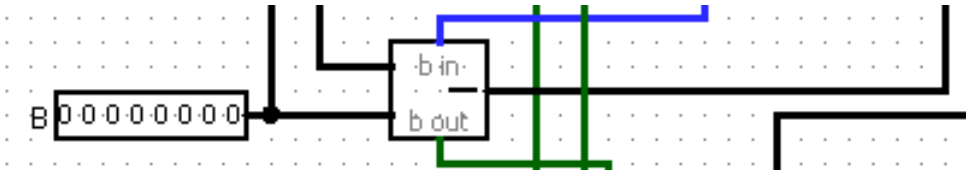
Instrukce ADD či ADC je tvořena jedním a tím samým kombinačním obvodem, kterým je sčítačka. Ta disponuje klasicky jedním 8-bitovým výstupem a dvěma 8 bitovými vstupy. Po průchodu touto částí aritmeticko-logické jednotky se dva vstupy sečtou a výsledek je přenesen na výstup, přičemž při přetečení se nastaví příznak a pokud e jedná o operaci ADC tak se také načte z příznakového registru příznak.



Obrázek 8

3.2.8 SUB/SBB

Tato část ALU je naprosto shodná s ADD/ADC až na použití kombinačního obvodu zvaného odčítačka a místo sečtení vstupů se vstupy od sebe odečtou.



Obrázek 9

3.3 Registry

Návrh registrů se lišil podle druhů registrů. Nejdříve se navrhovaly jednotlivé registry pomocí klopných obvodů typu D, dokud nebylo zjištěno, že jsou předdefinované jako již hotový kombinační obvod v programu Logisim.

3.3.1 Datové

Toto konkrétní procesorové jádro disponuje osmi registry vzájemně propojenými pomocí multiplexorů, kde pomocné signály vedoucí do multiplexoru určují enable na zápis/čtení a zbytek adresuje jaký registr bude využíván. Zároveň jsou připojeny i signály pro hodiny a reset registrů.

3.3.2 Pomocné/pracovní

Pomocné neboli také pracovní registry jsou pouze 8-bitové registry napojeny na vstupy aritmeticko-logické jednotky kam se ukládají vybrané hodnoty z registrů pro zpracování určitou operací.

3.3.3 Příznakové

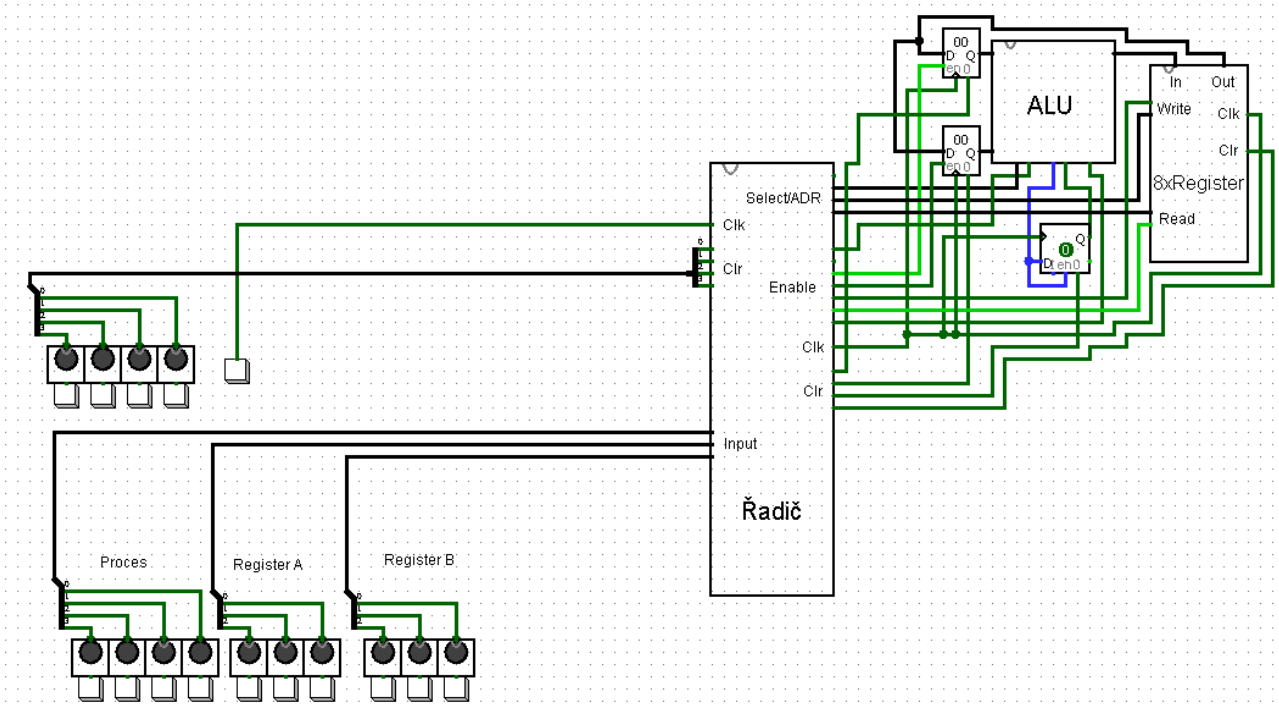
Příznakové registry jsou jednobitové registry vytvořené pomocí jednoho klopného obvodu typu D, kdy je tento registr napojen na ALU a využíván při přetečení při sčítání/odčítání nebo jako příznak po porovnání pomocí instrukce CMP.

3.4 Řadič a dekodér instrukcí

Řadič a dekodér instrukcí byl navrhnut pomocí analýzy jednotlivých instrukcí a následného navrhnutí instrukcí a rozvrhnutí instrukčních cyklů. Nejdříve se analyzovalo například instrukce NOT potřebuje pouze jednu hodnotu tudíž pomocný registr B se nebude vůbec využívat a pouze se načte při prvním taktu do pomocného registru A hodnota vybraného registru. Tudíž při prvním taktu zašle dekodér prováděnou instrukci, která se má v ALU provádět, nastaví datové registry na čtení, povolí pouze pomocný registr A, do kterého se načte hodnota a provede tik hodin. Na druhý takt se nastaví povolení pro zápis do datového registru povolí se výstup z ALU a při tiku se uloží výsledná hodnota instrukce NOT do registru. Po takovéto analýze všech instrukcí se vytvořil kombinační obvod, který takto převádí vstup uživatele do mikroinstrukcí pro jednotlivé části jádra procesoru.

4 PROPOJENÍ SOUČÁSTÍ

Po návrhu všech částí procesorového jádra již stačilo pouze propojit tyto součásti do jednotného celku a připojení uživatelského vstupu složeného z tlačítek pro zadávání instrukcí, reset a hodiny.


ALU
In
Out
Write
Clk
Clr
8xRegister
Read
Select/ADR
Clk
Clr
Enable
Clk
Clr
Input
Řadič
Proces
Register A
Register B

Obrázek 10

5 ZÁVĚR

Mým cílem v této práci bylo navrhnout jádro procesoru s deseti předem zadanými instrukcemi což se povedlo. Až na pár drobností je procesorové jádro plně funkční. Největší problémy, co nastaly byly při návrhu dekodéru instrukcí. Bylo zde mnoho karnaghuových map na zpracování, ve kterých se dalo velmi snadno chybovat. Při navrhování jsem se dověděl mnoho nového o architekturách procesorů a naučil se s novým programem logisim což hodnotím jako velmi pozitivní zkušenost. Celkovou práci bych hodnotil jako částečně úspěšnou, kvůli menší přehlednosti navrhnutého schématu, kde nevidí uživatel do registrů atd. V práci na procesorovém jádře rozhodně nemíním přestat a pokud bude možno tak hodlám rozšířit instrukční sadu, připojit operační paměť, zásobník a čítač a tím si vzít návrh celého procesoru jako maturitní práci.

POUŽITÁ LITERATURA

TIŠNOVSKÝ, Pavel. Od logických obvodů k mikroprocesorům. *Co se děje v počítači* [online]. 2008, **2008**, 1 [cit. 2017-01-19]. Dostupné z: https://www.root.cz/clanky/od-logickych-obvodu-k-mikroprocesorum/

TIŠNOVSKÝ, Pavel. Učíme trpaslíky počítat. *Co se děje v počítači* [online]. 2008, **2008**, 1 [cit. 2017-01-19]. Dostupné z: https://www.root.cz/clanky/ucime-trpasliky-pocitat/

TIŠNOVSKÝ, Pavel. Učíme trpaslíky počítat podruhé. *Co se děje v počítači* [online]. 2008, **2008**, 1 [cit. 2017-01-19]. Dostupné z: https://www.root.cz/clanky/ucime-trpasliky-pocitat-podruhe/

TIŠNOVSKÝ, Pavel. Architektury mikroprocesorů. *Co se děje v počítači* [online]. 2008, **2008**, 1 [cit. 2017-01-19]. Dostupné z: https://www.root.cz/clanky/architektury-mikroprocesoru/

6 SEZNAM OBRÁZKŮ A TABULEK