STREDNÁ PRIEMYSELNÁ ŠKOLA ELEKTROTECHNICKÁ
PREŠOV

IV.B - 06

ŠK. R. 2021 – 2022

PČOZ - 8 BITOVÝ PROCESOR POMOCOU TTL LOGIKY

Matej Dinis

Konzultant: Ing. Martin Ambrozy

ANOTÁCIA V SLOVENSKOM JAZYKU

Cieľom tejto práce bolo vytvoriť 8-bitový procesor. Tento projekt má slúžiť ako učebná pomôcka pre vysvetlenie princípu fungovania procesora a taktiež ako nastroj pre vyučovanie programovacieho jazyka Assembly. Procesor disponuje všetkými základnými matematickými a logickými funkciami. Procesor počas výkonu programu zobrazuje aktuálny stav všetkých časti. Výstupný register dovoľuje použiť procesor ako Arduino.

ANOTÁCIA V ANGLICKOM JAZYKU

The objective of the work was to create an 8-bit processor. This project can be used as an educational resource for explaining the principle of how a processor works, and also as a tool for teaching programming language Assembly. The processor has all the basic mathematical and logical functions. The processor during the execution of the program displays the current state of all parts. The output register allows the processor like an Arduino.

Čestné vyhlásenie

Vyhlasujem, že celú prácu s názvom „8 bitový procesor“ som vypracoval samostatne, s použitím uvedenej literatúry.

Som si vedomý zákonných dôsledkov, ak v nej uvedené údaje nie sú pravdivé.

Prešov, 9. mája 2022 .....

vlastnoručný podpis

Poďakovanie

Týmto by som chcel vyjadriť poďakovanie konzultantovi Ing. Martinovi Ambrozymu za jeho odbornú pomoc pri vytváraní tejto práce.

Taktiež by som chcel poďakovať všetkým ľuďom, ktorý mi povedali svoje pripomienky a vylepšenia, ktorými by som mohol vylepšiť môj projekt.

Obsah

ÚVOD

Tento projekt som vybral z dôvodu, že mojim koníčkom od prvého ročníka strednej školy je digitálna elektronika a programovanie v jazykoch C/C++ a Assembly. Fungovanie procesorov ma stále fascinovalo, ako niečo čo dôkaze len sčítať a odčítať čísla dôkaze spustiť hry, upravovať fotky, ...

Hlavnou inspiráciou pre tento projekt bola kniha Digitálna elektronika od P. A Malvino [1]a videa od Ben Eater[2].

Účelom tejto práce je vytvoriť 8-bitovy procesor, ktorý bude schopný vykonávať základne matematické a logické funkcie a prekladač, ktorý z programovacieho jazyka podobnému Assembleru výtvory strojový kód pre procesor.

[1]https://archive.org/details/367026792DigitalComputerElectronicsAlbertPaulMalvinoAndJeraldABrownPdf1
[2] https://www.youtube.com/watch?v=HyznrdDSSGM&list=PLowKtXNTBypGqImE405J2565dvjafglHU

1 CIEĽ PRÁCE

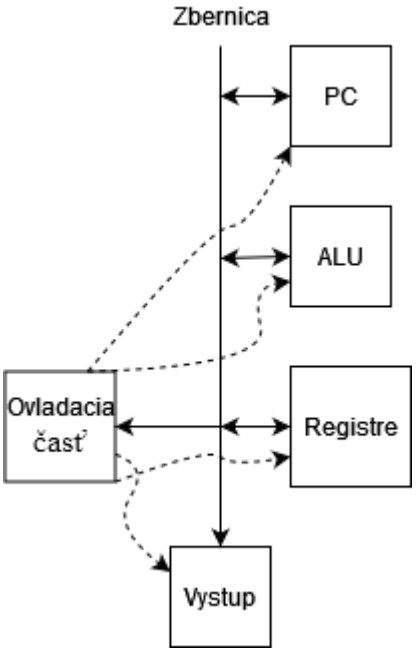
Cieľom práce je vytvoriť funkčný 8-bitovy procesor, ktorý slúži ako vizuálna demonštrácia fungovania jednoduchého procesora.

Hlavným cieľom je vytvoriť jednoduchý procesor, ktorý bude schopný vykonávať rovnakú funkciu ako Arduino.

Ďalším základným cieľom je vytvoriť jednoduchý programovací jazyk na princípe Assembleru.

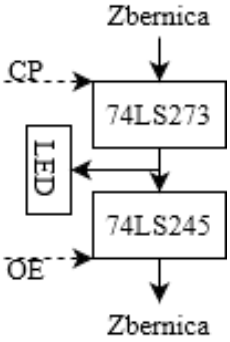
2 MATERIAL A METODIKA PRÁCE

Cela práca pozostáva z navrhnutia a vytvorenia 5 základných časti a to sú ovládacia časť, registre, aritmetická a logická jednotka, program counter a zbernica s vytupeným registrom. V prílohe D sú jednotlivé schémy a návrhy plošných spojov. Bloková schéma je upravenou verziou schémy z knihy Digitálna počítačová elektronika. (1)



Obrázok 1: Bloková schéma procesor

2.1 REGISTRE

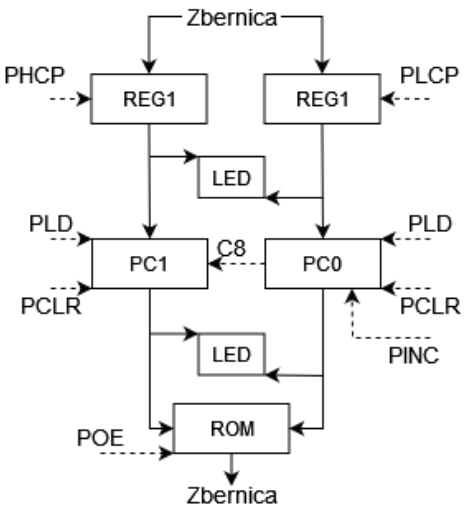


Obrázok 2: Bloková schéma registra

Procesor disponuje dvoma registrami X a Y. Registre je schopný uložiť 8bitove číslo. Ako register je použití 8 bitový D preklápací obvod 74LS273. Pretože použití preklápací

obvod nemá funkciu ovládania výstupov ako výstup z jednotlivých registrov sme použili ovládací obvod 74LS245, ktorý ma na vyspaných pinoch trojstavovú logiku. Pre jednoduché zisťovanie hodnoty čísla uloženého v registri je na výstup pripojene LED diódy.

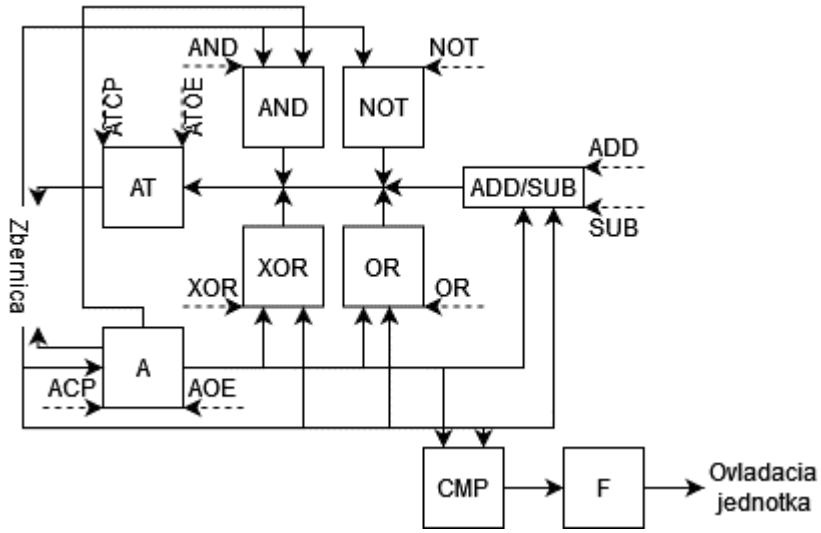
2.2 PROGRAM COUNTER



Obrázok 3: Bloková schéma program counter

Program je uložený v EEROM pamäti AT28C64, ktorú sme zapojili ako ROM pamäť. Adresu pre ROM udávajú štyri počítadla 74LS193. Tieto počítadla nám dávajú rozsah pre ROM pamäť od 0x0000 po 0xFFFF, ale použitá ROM pamäť rozsah adries od 0x0000 po 0x1FFF. Počítadla majú na vstup pripojený registre, ktoré dovoľujú načítavanie adries počas výpočtu programu. Na výstup registrov a výstup počítadiel sú pripojene LED diódy.

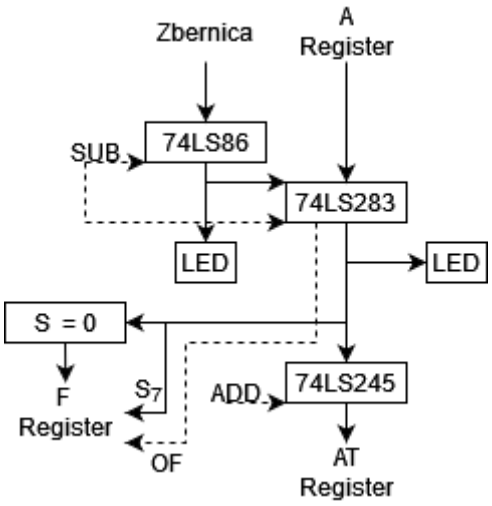
2.3 VÝPOČTOVÁ ČASŤ



Obrázok 4: Obloková schéma ALU

Výpočtová časť pozostáva z troch časti matematickej, logickej časti a registrov A, AT a F. Register A sa môže požívať rovnako ako registre X a Y. Výsledok každej matematickej a logické operácie sa uloží do registra A. Register AT slúži ako dočasný register pre ukladanie čísel počas výpočtu matematickej alebo logickej operácie. Register F slúži na ukladanie výsledku porovnávacieho obvodu.

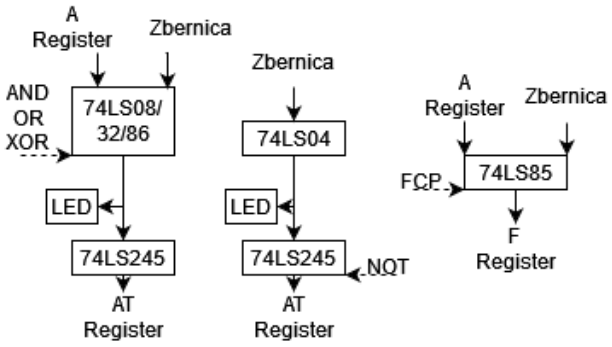
2.3.1 MATEMATICKÁ ČASŤ



Obrázok 5: Bloková schéma sčítačky

Poskytuje základne matematické funkcie a to sú sčítavanie a odčítavanie. Na vstup A 8 bitovej sčítačky 74LS283 je pripojený výstup registra A, na vstup B je pripojená zbernica cez XOR hradla 74LS86, ktoré poskytujú možnosť odčítavať čísla. Pridávna logika pozostávajúca z AND 74LS08 a NOT 74LS02 hradiel vyhodnocuje tri logické stavy pre register F a tie sú ci výsledok sa rovná nule (ZF), výsledok je záporný (NF) to udáva bit na siedmej pozícii a ci nastalo pretečenie výsledku (OF) keď je výsledok väčší ako 255. LED diódy zobrazujú aktuálny výstup sčítačky a XOR hradiel.

2.3.2 LOGICKÁ ČASŤ



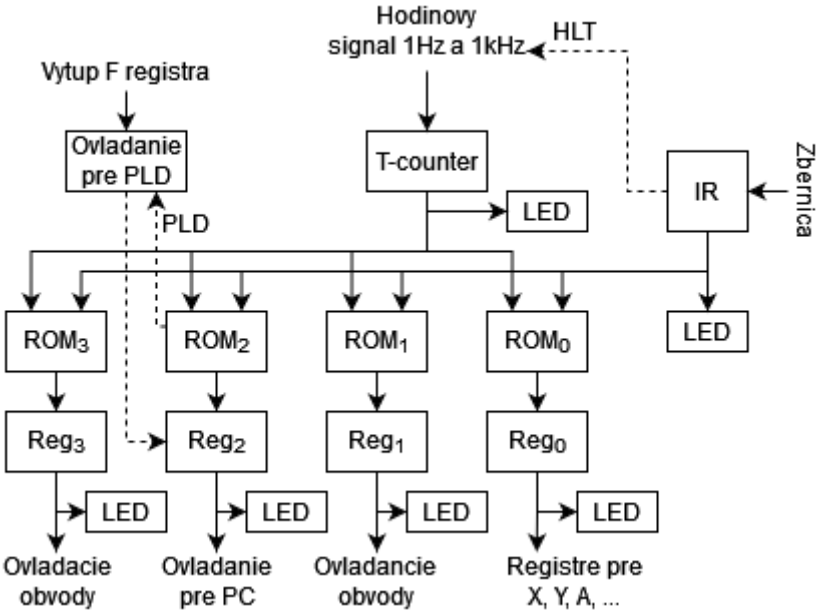
Obrázok 6: Bloková schéma logických obvodov

Disponuje štyrmi základnými funkciami a to sú logicky súčin, súčet, negácia a exkluzívny súčet. Logicky súčet (74LS32), exkluzívny súčet (74LS86) a súčin (74LS08) ma na vstupe A pripojený register A, na stranu B je pripojená zbernica. Logická negácia (74LS04) ma len jeden vstup a ten je pripojený na zbernicu. Všetky logické operácie nemajú na výstupe trojstavovú logiku preto na ovládanie je použití ovládací obvod 74LS245. Tiež obsahuje obvod na porovnávanie hodnôt (74LS85), čo poskytuje stavové bity ako menšie (LF), väčšie (GF) a rovne (EF). Výsledok porovnávanie je uložený do registra F, ktorý je pripojený ku ovládacej jednotke.

2.4 ZBERNICA

Spája všetky časti procesora, pomocou 8 bitovej dátovej zbernice a napájanie pre každú časť procesora. Zelena LED udáva ci je pripojený zdroj napätia. Výstupný register je jednosmerný je možne do neho len načítavať čísla. Slúži na ovládanie veci mimo procesora.

2.5 OVLÁDANIE



Obrázok 7: Bloková schéma ovládača

Najdôležitejšia časť celého procesora. Pozostáva z registra IR, kde je uložená momentálne vykonávaná inštrukcia. Hodnota uložená v IR a hodnota T-counter udáva adresu pre ROM pamäte, ktoré udávajú momentálny stav všetkých ovládacích pinoch. Sirka T-counter je 4 bity to nám dáva pre každú inštrukciu 16 cyklov. Prvý (T0) a posedný (T15) cyklus sú nastavene tak aby nenastala žiadna zmena ovládacích pinov pri zmene inštrukcie. Cykly T13 a T14 sú pre každú inštrukciu rovnaké a to inkrementacia PC počítadla a načítanie inštrukcie do registra IR.

2.6 INŠTRUKCIE

Inštrukcia je daná ako 8 bitové číslo. Procesor ma 44 rôznych inštrukcii.

2.6.1 POHYBOVÉ INŠTRUKCIE

Patria tu LDR, MOV, OUT inštrukcie. Slúžia na presun čísel medzi registrami a na načítavanie čísle z ROM pamäte. Do registra O je možne len načítavať čísla slúži ako výstup pre procesor.

```
ldr x, 0x2F
mov a, y
out y
```

Skript 1: Syntax pohybových inštrukcii

2.6.2 LOGICKÉ A ARITMETICKÉ INŠTRUKCIE

Patria tu ADD, SUB, AND, OR, XOR, NOT, CMP inštrukcie. Všetky inštrukcie pracujú register A *inštrukcia* (X, Y, číslo). NOT inštrukcia funguje len na registroch X, Y a A.

```
add a
sub y
and 0x64
or  x
xor 0x01
not x
cmp y
```

Skript 2: Syntax ALU inštrukcii

2.6.3 KONDICIONÁLNE SKOKY

Tiež nazývane podmienene a nepodmienene skoky. Patria tu JMP, JZ, JN, JO, JL, JG, JE inštrukcie. Umožňujú využívať v programe značky. Adresa skoku je stále načítaná do program counter registrov, načítanie do počítadiel závisí do stavových bitov.

```
jmp loop
jz  lable1
jn  lable1
jo  start
jg  end
je  lable2
```

Skript 3: Syntax kondicionálnych skokov

2.6.4 ŠPECIÁLNE INŠTRUKCIE

CLR – nastaví hodnotu počítadla na 0x0000. Využitie na reštartovanie programu.

NOP – nevykoná sa žiadna inštrukcia. Môže byt použitá na spomalenie programu.

HLT – zástavy obvod hodinového signálu.

```
clr
nop
hlt
```

Skript 4: Syntax špeciálnych inštrukcii

2.7 PREKLADAČ – ROSETTA STONE

Jednoduchý program v jazyku C++, ktorý prelozí program v jazyku podobnému Assembleru do strojového kódu, ktorý sa potom naprogramuje do ROM pamäte. Prekladač premení odkazy na adresy kde ma program pokračovať, čísla sa zadávajú v hexadecimálnej podobe. Prekladač nedisponuje žiadnou kontrolou pre zisťovanie logických chyb, upozorňuje len základne syntaktické chyby. Inštrukcie a značky môžu byt písane veľkými aj malými písmenami.

```
start:
        ldr x, 0x00
        ldr y, 0x01
loop:
        mov a, x
        add y
        mov x, y
        mov y, a
        cmp 0x64
        jg end
        jmp loop

end:
        out a
        hlt
```

Skript 5: Fibonacciho postupnosti v Assemblery

```
bool X0[][8] = {
{ 0,0,0,0,0,0,0,0 }, // CLR
{ 0,0,0,0,0,1,0,1 }, // LDR x
{ 0,0,0,0,0,0,0,0 }, // 0x00
{ 0,0,0,0,0,1,1,0 }, // LDR Y
{ 0,0,0,0,0,0,0,1 }, // 0x01
{ 0,0,0,1,0,1,0,0 }, // MOV A, X
{ 0,0,0,0,1,0,0,1 }, // ADD Y
{ 0,0,0,1,0,0,0,0 }, // MOV X, Y
{ 0,0,0,1,0,0,1,1 }, // MOV Y, A
{ 0,0,1,0,0,1,0,0 }, // CMP
{ 0,1,1,0,0,1,0,0 }, // 0x64
{ 0,0,1,0,1,0,1,0 }, // JG
{ 0,0,0,1,0,0,0,0 }, // 0x10
{ 0,0,0,0,0,0,0,0 }, // 0x00
{ 0,0,1,0,1,1,1,0 }, // JMP
{ 0,0,0,0,0,1,0,0 }, // 0x04
{ 0,0,0,0,0,0,0,0 }, // 0x00
{ 0,0,0,0,0,1,0,0 }, // OUT A
{ 1,0,0,0,0,0,0,0 }  // HLT
};
```

Skript 6: Fibonacciho postupnosti v strojovom kóde

3 TEORETICKÁ ČASŤ – ÚVOD DO PROBLEMATIKY

3.1 PROCESOR

CPU (skr. z angl. central processing unit, často prekladané ako centrálna procesorová jednotka) je hlavný procesor počítača. Interpretuje, vykonáva alebo spracúva inštrukcie alebo dáta programu vo forme strojového kódu. Dnes sú centrálne procesorové jednotky takmer vždy realizované vo forme mikroprocesora. CPU sa v slovenčine oficiálne označuje ako procesor základnej jednotky alebo skrátene základná jednotka alebo procesor (tiež: procesor ústrednej jednotky, centrálny procesor, ústredný procesor). (2)

3.2 DVOJKOVÝ KOMPLIMENT

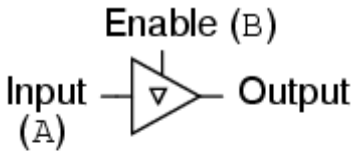
Dvojkový kompliment je matematická operácia na binárnych číslach, a je príkladom koreňových komplimentov. Je používaná v počítačovej vede ako metóda reprezentovania čísel. Ak je MSF je jedna, číslo je záporne. Dvojkový kompliment N-bitového čísla je definovaná ako kompliment pre 2^N; súčet čísla a jeho dvojkového komplimentu je 2^N. Na príklad trojbitového čísla 011_2, dvojkový kompliment je 1012, lebo $011_2 + 101_2 = 1000_2 = 8_{10}$ čo je rovne 2^3. Dvojkový kompliment sa vypočíta investovaním bitov a pripočítaný jednotky. (3)

3.3 INŠTRUKČNÝ SÚBOR

Inštrukčný súbor alebo inštrukčná sada (angl. instruction set architecture) je všeobecný opis organizačných, funkčných a prevádzkových princípov procesora, z pohľadu programátora je to zoznam dostupných mechanizmov pre programovanie. Inštrukčný súbor sa často nazýva aj ako architektúra. (4)

3.4 TROJSTAVOVÁ LOGIKA

Trojstavová logika dovoľuje výstupnému alebo vstupnému pinu zaujať stav vysokej impedancie, čo efektívne odstráni pin z obvodu, pričom poskytuje normálne logické úrovne 0 a 1. Toto dovoľuje viacerým obvodom zdieľať rovnaký výstupný pin (ako napríklad zbernica). Trojstavový výstup je implementovaný v mnohých registroch, zberniciach a preklápacích obvodoch v 7400 a 4000 sérii ako tiež v iných typoch, tiež interne v mnohých integrovaných obvodov. Typické použite je ako interne alebo externe zbernice v mikroprocesore, počítačovej pamäti a periférnych zariadeniach. Mnohé zariadenia sú kontrolovanie pomocou vstupu aktívnej nuly nazývaného OE (Output Enable) čo hovory ci ma výstup byt v móde vysokej impedancie alebo reprezentovať záťaž a to 0 alebo 1 log. úroveň. (5)



Truth table

A	B	Output
0	0	High-Z
0	1	0
1	0	High-Z
1	1	1

Obrázok 8: Príklad trojstavová logika

3.5 REGISTER

Register v procesore je pamäťové miesto, ktoré slúži procesoru na uchovávanie údajov, ktoré práve spracováva. Ide o pomerne malé množstvo veľmi rýchlej pamäte, ktorá je priamo

súčasťou procesorového jadra a prístup k nim je obvykle súčasťou inštrukčného súboru. Registre majú obvykle rovnaký počet bitov ako je základná šírka spracovaného údaju (t. j. napr. u 8-bitového procesora je to 8 bitov atď.). U niektorých procesorov sú k dispozícii aj registre s dvojnásobnou príp. štvornásobnou šírkou, obvykle zložené z viacerých základných registrov (ku ktorým je možné pristupovať aj individuálne). CISC procesory obyčajne obsahujú malý počet registrov (3 – 20), z ktorých niektoré majú špeciálne určené postavenie (napríklad akumulátor, počítadlo cyklov, indexový register a pod.). Naopak, RISC procesory majú typicky veľký počet (16 – 32) navzájom zväčša rovnocenných registrov. Okrem dátových registrov obsahujú procesory aj špeciálne registre (angl. Special Function Register, skr. SFR) určené pre zvláštne funkcie, napríklad počítadlo programu (angl. Program Counter, PC) obsahujúce adresu práve vykonávanej inštrukcie, alebo ukazovateľ vrchola zásobníka (angl. Stack Pointer, SP). Táto skupina registrov je ešte významnejšia (a rozsiahlejšia) u mikrokontrolérov, kde sa pomocou SFR pristupuje k zabudovaným periférnym zariadeniam. (6)

3.6 ARITMETICKÁ A LOGICKÁ JEDNOTKA

Aritmeticko-logická jednotka (skrátene ALU podľa anglického arithmetic logic unit) je centrálna časť procesora (CPU), v ktorej sa vykonávajú základné aritmetické a logické operácie s číslami: sčítanie, odčítanie, násobenie, delenie, logický posun, negácia, komplement, atď. Rozsah operácií závisí od konkrétneho procesora. V minulosti sa aritmeticko-logická jednotka realizovala samostatne pomocou diskrétnych súčiastok (napríklad v starých sálových počítačoch zo 70-tych rokov). Neskôr sa ALU realizovala pomocou tzv. bitových rezov (staršie počítače SMEP). Dnes je ALU súčasťou procesora realizovaného ako integrovaný obvod (čip). (7)

3.7 JAZYK SYMBOLICKÝCH INŠTRUKCII

Jazyk symbolických inštrukcií alebo jazyk symbolických adries (skr. JSI či JSA), nesprávny populárny názov tiež assembler či asembler pozri (asembler) je pre ľudí na čítanie vhodná forma strojového kódu. Zjednodušenie je založené na nahradení binárnych kódov skrátenými názvami (tzv. mnemonikou) „príkazov“ (inštrukcií). Pracovať priamo so strojovým kódom je pre človeka veľmi zložité, pretože programy v strojovom kóde sú postupnosťami čísel. Preto bolo potrebné vytvoriť jazyk, ktorý by bol zrozumiteľný pre človeka a zároveň by bol priamočiaro transformovateľný do strojového kódu. Jazyk symbolických adries je programovací jazyk, v ktorom každý kód inštrukcie generuje jednu inštrukciu strojového jazyka, viazanú na konkrétny procesor počítača, kontrastne k vyššiemu programovaciemu jazyku, kde sa príkaz už neviaže na konkrétny počítač alebo operačný systém. (8)

3.8 EEPROM

EEPROM (Electrically Erasable Programmable Read-Only Memory) je elektricky zmazateľná pamäť ROM. Princíp činnosti je podobný ako u EPROM – pamäťovým prvkom je izolované (plávajúce) hradlo do ktorého sú nainjektované nosiče náboja cez izolačnú oxidovú vrstvu, oxid je tu však tenší a prenos náboja je šetrnejší a je možný v oboch smeroch. Napätia potrebné pre tento jav sú väčšinou generované obvodmi integrovanými na čipe s pamäťou a navonok je pamäť programovaná pri bežnom napájacom napätí. Programovanie i mazanie trvá niekoľko milisekúnd, čítacie doby sú podobné ako u EPROM (cca 100ns). EEPROM znesú typicky 1E4 až 1E5 prepisov (mazacích a programovacích cyklov) a doba uchovania údajov býva pri bežných teplotách (t. j. do 80 °C) zaručovaná na 10 – 20 rokov. EEPROM predstavuje najjednoduchšie riešenie permanentnej pamäte ROM s možnosťou príležitostných zmien. Možno do nej zapisovať činnosťou programu mikropočítača. Zápis vyžaduje čas rádovo milisekundy, kým čítanie rádovo nanosekundy, preto sa nemôže použiť ako pamäte RWM. Bývajú vyrobené technológiou CMOS. (9)

3.9 ENDIANITA

Endianita je v informatike spôsob uloženia čísel v pamäti počítača, ktorý definuje, v akom poradí sa ukladajú jednotlivé jednotky informácie príslušného údajového typu. Ak ide konkrétne o bajty, označuje sa tiež ako poradie bajtov (angl. byte order). Rôzne platformy môžu používať rôznu endianitu a tento problém je potrebné brať do úvahy pri prenose binárnych súborov alebo sieťovej komunikácii medzi platformami s rôznou endianitou. (10)

3.9.1 LITTLE-ENDIAN

V tomto prípade sa na pamäťové miesto s najnižšou adresou uloží najmenej významný bajt (LSB) a zaň sa ukladajú ostatné bajty až po najvýznamnejší bajt (MSB). Architektúry uplatňujúce tento princíp sa nazývajú little-endian (mnemotechnická pomôcka: little end first) a patria medzi ne MOS Technology 6502, Intel x86 a DEC VAX. (10)

3.10 EAGLE

EAGLE aplikácia pre návrh schém, vytvorenie dosiek plošných spojov a podpora vytvárania CAM súborov.

3.11 LOGISIM EVOLUTION

Voľne šíriteľný program, ktorý dôkaze simulovať a minimalizáciu digitálnej diskrétnej logiky.

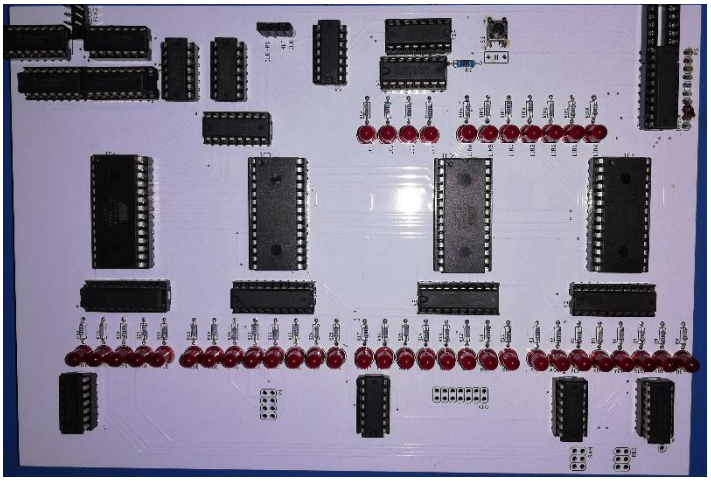
3.12 Logický analyzátor

Logicky analyzátor je elektronicky nastroj, ktorý dovoľuje zaznamenávať a zobrazovať signál z viacerých digitálnych systémov alebo digitálnych obvodov. Logicky analyzátor dovoľuje prevádzač dáta do časových diagramov, dekódovanie pretokov, stavu zariadenia, Assembly jazyk alebo možne rozlúštiť assembly z zdrojových úrovní programu. Logicky analyzátor ma vylepšenú možnosť spuste, čo sú užitočne keď užívateľ potrebuje vidieť časovú závislosť medzi signálmi v digitálnych systémoch. (11)

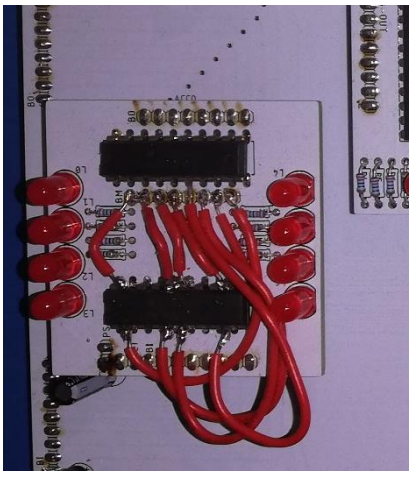
4 PRAKTICKÁ ČASŤ

Praktická časť prace pozostával z vytvorenia jednotlivých plošných spojov potrebných na správne fungovanie procesora. Plošne spoje boli navrhnute v programe Eagle. Funkčnosť plošných spojov bolo odsimulovane v programe Logisim Evolution. Jednotlivé plošne spoje boli vyrobene v Čine. Jednotlivé moduly a prepojovacie plošne spoje boli otestovane pomocou logického analyzátora a Arduina. EEPROM pamäte pre ovládací modul a pre ROM pamäť boli naprogramované pomocou Arduina. Ďalšie fotky a videa z výroby procesora a fungovania sú v prílohe C.

Pri výrobe sme nezrazili na zopár problémov, ktoré boli spôsobene chybou vo výrobe ale aj chybou návrhu ale tie sme veľmi rýchlo opravili.



Obrázok 9: Ovládací modul



Obrázok 10: Register A

5 VÝSLEDKY PRÁCE A DISKUSIA

Pri riešený prace sme sa stretli s rôznymi problémami. Tie však sme rýchlo odhalili a pomocou logického analyzátora a následne opravili. Vyhotovenie prace vychádzalo z teoretických poznatkov nadobudnutých štúdiom, praktických skúsenosti a konzultácie.

Vytvorili sem funkčný projekt s názvom 8-bitový procesor pomocou TTL logiky. Procesor dôkaze po zapnutí vypočítať program uložený v ROM pamäti, bez zásahu používateľa. Vytvorený prekladač vie veľmi rýchlo preloziť používateľom vytvorený program do strojového kódu a pomocou Arduina sa program nahrá do EEPROM pamäte. Ktorá sa vloží do procesora a začne sa výpočet programu. Procesor sa da použiť ako učebná pomôcka pre učenie Assembly jazyka a pre vysvetlenie funkčnosti procesora. Tomuto napomáha že modul hodinového signálu sa da nahradiť modulom s menšou frekvenciu, Arduinom alebo iným generátorom pravouhlého signálu, pre lepšiu vizualizáciu funkčnosti jednotlivých inštrukcii. Programovací jazyk pozostáva len zo zopár inštrukcii a programovať v ňom je veľmi jednoduché. Priložený prekladač, napomáha užívateľovi v tom že nemusí poznať každú číselnú kombináciu pre inštrukcie.

6 ZÁVERY PRÁCE

Procesor sa nám podarili zrealizovať podlá stanovených cieľov. Procesor je jednoduchá a zároveň komplikovaná práca, ktorá dôkaze vysvetliť jednoduchosť prace procesora. Správnosť výpočtu procesora sme otestovali pomocou rôznych programov. Týmto sme nie len otestovali funkčnosť jednotlivých modulov ale aj funkčnosť prekladača.

V teste prekladača sme skúšali správnosť prekladu (Príloha A, TEST 1). V ďalšom teste (Príloha A, TEST 2) sme skúšali správnosť prekladu kondicionálnych skokov a adresu kde ma procesor pokračovať vo výpočte.

Do budúcna by sa procesor dal rozšíriť o ďalšie matematické funkcie ako binárny posun, násobenie a delenie čísel. Modul RAM pamäte, pre ukladanie hodnôt počas výpočtu. Možnosť primania hodnôt mimo ROM pamäte. Modul Zásobníka pre ukladanie hodnôt pri využívaný procedúr. Modul pre primanie a spracovanie prerušený programu. Jednou z hlavných veci čo by sme chceli do budúcna vylepši je skrátiť počet cyklov. Zmena z THT súčiastok na SMD, kvôli momentálnej veľkosti procesora.

Zhrnutie

Stanovený ciel a to postaviť funkčný 8 bitový procesor a naprogramovať funkčný prekladača pre upravený Assembly programovací jazyk. Tento ciel sa nám podaril. Vytvoriť program v Assemblery je jednoduché a vďaka rýchlemu prekladaču a EEPROM programátora aj ľahké vložiť do procesora. Vďaka tomu že procesor nám stále ukazuje stáva všetkých časti je jednoduché pozorovať ako sa vykonáva každá inštrukcia.

Vypracovanie tejto obohatilo moje znalosti digitálnej elektroniky, prácu s EAGLE programom, naučil som sa používať logické analyzátory a logické simulačne programy.

Resumé

The goal that was set namely to build a functional 8-bit processor and program a functional translator for the modified Assembly programming language. We have succeeded in this goal. Creating a program in Assembly is easy and easy to insert into the processor thanks to a quick translator and EEPROM programmer. Thanks to the fact that the processor still shows us becoming all parts it is easy to observe how each instruction is executed.

The elaboration of this enriched my knowledge of digital electronics, working with EAGLE program, I learned to use logical analyzers and logical simulation programs.

ZOZNAM POUŽITEJ LITERATÚRY

[1. **Malvino, Albert Paul a Brown, Jarald.** *Digital Comuter Electronics.* New York : Glenocoe, 1993. ISBN 0-02-800594-5.
2. **Wikipedia.** CPU. [Online] [Dátum: 4. Máj 2022.] https://sk.wikipedia.org/w/index.php?title=CPU&oldid=7085616.
3. —. Two's complement. [Online] [Dátum: 4. Máj 2022.] https://en.wikipedia.org/w/index.php?title=Two%27s_complement&oldid=1085992736.
4. —. Inštrukčný súbor. [Online] Wikipédia, Slobodná encyklopédia. [Dátum: 5. Máj 2022.] https://sk.wikipedia.org/w/index.php?title=In%C5%A1truk%C4%8Dn%C3%BD_s%C3%BAbor&oldid=6700238.
5. —. Three-state logic. [Online] Wikipedia, The Free Encyclopedia. [Dátum: 5. Máj 2022.] https://en.wikipedia.org/w/index.php?title=Three-state_logic&oldid=1074281286.
6. —. Register (procesor). [Online] Wikipédia, Slobodná encyklopédia. [Dátum: 5. Máj 2022.] https://sk.wikipedia.org/w/index.php?title=Register_(procesor)&oldid=6067318.
7. —. Aritmeticko-logická jednotka. [Online] Wikipédia, Slobodná encyklopédia. [Dátum: 5. Máj 2022.] https://sk.wikipedia.org/w/index.php?title=Aritmeticko-logick%C3%A1_jednotka&oldid=5252639.
8. —. Jazyk symbolických inštrukcií. [Online] Wikipédia, Slobodná encyklopédia. [Dátum: 5. Máj 2022.] https://sk.wikipedia.org/w/index.php?title=Jazyk_symbolick%C3%BDch_in%C5%A1trukci%C3%AD&oldid=7302290.
9. —. EEPROM. [Online] Wikipédia, Slobodná encyklopédia. [Dátum: 5. Maj 2022.] https://sk.wikipedia.org/w/index.php?title=EEPROM&oldid=5913931.
10. —. Endianita. [Online] Wikipédia, Slobodná encyklopédia. [Dátum: 7. Maj 2022.] https://sk.wikipedia.org/w/index.php?title=Endianita&oldid=7225481.
11. —. Logic analyzer. [Online] Wikipedia, The Free Encyclopedia. [Dátum: 9. Maj 2022.] https://en.wikipedia.org/w/index.php?title=Logic_analyzer&oldid=1077249124.

ZOZNAM OBRÁZKOV

ZOZNAM SKRIPTOV

PRÍLOHY

Zoznam príloh:

Príloha A: Testovacie programy

Príloha B: Prekladač Rosetta Stone

Príloha C: Foto a video dokumentácia fungovania procesora

Príloha D: Schémy a návrhy jednotlivých modulov procesora

Príloha E: Tabuľky pre ROM pamäte v ovládači