FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Emulátor kufříkového mikropočítače PMI-80
Student:	Vojtěch Straka
Vedoucí:	Dr.-Ing. Martin Novotný
Studijní program:	Informatika
Obor / specializace:	Počítačové inženýrství
Katedra:	Katedra číslicového návrhu
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Seznamte se s architekturou výukového kufříkového mikropočítače Tesla PMI-80. V jazyce VHDL navrhněte a implementujte hardwarový emulátor mikropočítače PMI-80 včetně jeho rozhraní. Cílovou platformou je FPGA, proto zvolte vhodnou vývojovou desku s FPGA. Ověřte správnost návrhu sadou testbenchů a funkčnost Vámi navrženého zařízení pomocí sady softwarových testů. Popište, jakým způsobem by bylo možné emulátor zakomponovat do expozice historie výpočetní techniky, případně konkrétní realizovaná použití.

Elektronicky schválil/a prof. Ing. Hana Kubátová, CSc. dne 30. ledna 2023 v Praze.

Bakalářská práce

IMPLEMENTACE ARCHITEKTURY POČÍTAČE TESLA PMI-80 V JAZYKU VHDL

Vojtěch Straka

Fakulta informačních technologií
Katedra číslicového návrhu
Vedoucí: Dr.-Ing. Martin Novotný
11. května 2023

Obsah

Seznam obrázků

Chtěl bych poděkovat svým kolegům ze spolku Herní historie, bez kterých bych si nevypěstoval zájem o počátky výpočetní techniky, jenž mě vedl až k výběru tohoto tématu. Také bych rád poděkoval svojí rodině a přátelům, kterým jsem na několik týdnů zmizel ze života. Dík si zaslouží i členové komunitního fóra OldComp a vývojáři MiSTera, kteří mi mnohdy poskytli cennou radu. V neposlední řadě musím zmínit podporu a odborné vedení svého vedoucího Dr.-Ing. Martina Novotného, díky kterému byl celý proces mnohem jednodušší.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 11. května 2023 .....

Abstrakt

Práce se zabývá tvorbou emulátoru pro počítač Tesla PMI-80 na bázi FPGA čipu. Popisuje architekturu počítače a druhy emulace, které lze využít. Dále je zdůvodněn výběr cílové platformy, kterou je jádro pro projekt MiSTer na přípravku Terasic DE-10 Nano. Nakonec je popsána samotná implementace a uvedeny některé z možností jejího dalšího využití.

Klíčová slova hardwarová emulace, Altera, Tesla PMI-80, integrace do platformy, hardwarová architektura, jednodeskový počítač, historický počítač, MiSTer, VHDL, FPGA

Abstract

The goal of this thesis is the creation of a FPGA-based emulator for the computer Tesla PMI-80. The thesis describes the architecture of the computer and methods of emulation that could be used. Then follows the reasoning for the final choice, which is a core for the MiSTer project on the Terasic DE-10 Nano board. Finally, the implementation is documented and some possibilities of its further use are shown.

Keywords hardware emulation, Altera, Tesla PMI-80, platform integration, hardware architecture, single-board computer, historical computer, MiSTer, VHDL, FPGA

Seznam zkratek

FPGA	Field-programmable gate array
HLE	High-level emulation
LLE	Low-Level emulation
OSD	On-screen display
PLL	Phase-locked loop
VHDL	VHSIC Hardware Description Language

Úvod

Je až s podivem, jak se výpočetní technika rychle vyvíjí. Nemyslím jen po stránce výkonu, ilustrované panem Moorem v jeho slavném zákonu. Používané technologie a platformy se, až na pár čestných výjimek, mění nezastavitelným tempem. Dokonce takovým, že v některých oborech je zpochybňováno, zda formální vysokoškolská výuka nebude zastaralá, jakmile jí student projde. Existuje ale protipól této překotné evoluce. Základní rámec informatiky se naopak posunuje velmi pomalu. Po teoretické stránce stále vycházíme z algoritmické představy a v širším hledisku z idejí informatiků předpočítačové éry. Významné milníky posledních dekád se dají spočítat na prstech jedné ruky – objektově orientované programování, paralelizace, možná i funkcionální přístup, ale ten se neprosadil tak zásadním způsobem.

Stejně tak je neuvěřitelně konzistentní technologie hardwaru. Od přechodu k binárním číslicovým počítačům a později mikroprocesorům je představa počítače v podstatě nezměněná. Nic to neilustruje lépe, než fakt, že stroj, na kterém píši tuto práci, je kompatibilní se standardem IBM PC z roku 1981. Mimochodem o rok starším, než je počítač, kterým se zabývám v této práci. Říct, že nedochází k vývoji, je trochu zavádějící. Jak dosahujeme fyzikálních limitů křemíkového přístupu na úrovni pouhých nanometrů, jsme nuceni hledat jiné možnosti. A na obzoru je stále příslib kvantového počítače, případně jiné možnosti návratu k analogovému světu. Co jsem ale chtěl tímto úvodem naznačit je, že zdánlivě naprostý archeologický relikt, jakým jistě je počítač starý čtyři dekády, je stále relevantní.

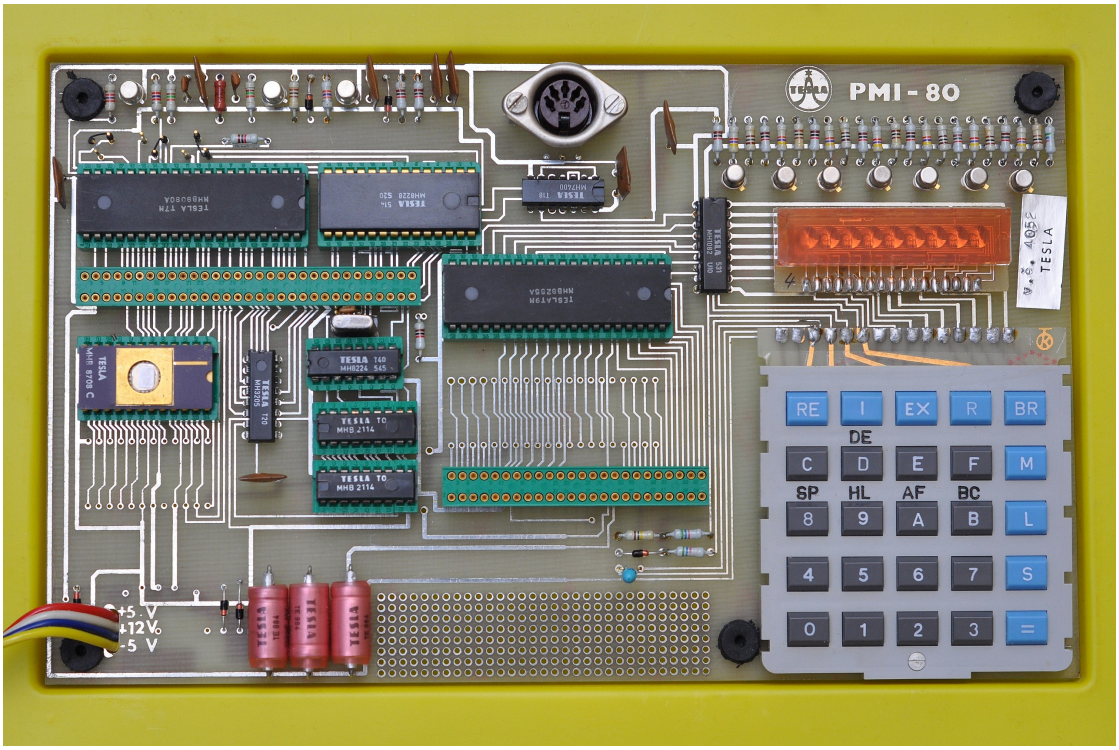
Rozhodl jsem se pro obor počítačového inženýra mimo jiné proto, že mám přirozený sklon věcem rozumět komplexně a v celé šíři. Představa, že si myšlenkově propojím znalosti počínající logikou a číslicovými obvody až po roviny abstrakce vyšších programovacích jazyků, byla lákavá. V tomto ohledu poskytuje náš výběr povinných předmětů velmi dobrý základ a jsem rád, že jsem na něm mohl stavět dál. Jelikož se pohybuji na pomezí hardwaru a softwaru, byly zkušenosti jako psaní strojového jazyka nebo důkladné prozkoumání architektury procesoru velmi přínosné a ve své závěrečné práci jsem je chtěl náležitě využít.

Druhou motivací byl můj vřelý vztah k historii. Již několik let vedu spolek Herní historie a věřím, že dokumentace vývoje výpočetní techniky je pro tak dynamický obor důležitá. V této práci se ale zabývám nejenom popisem, ale hlavně implementací. Tomu odpovídá i zvolená architektura, která není v dostupných zdrojích tak dobře zpracovaná a je vhodná svojí komplexitou na rozsah bakalářské práce. Cílem je vytvořit interaktivní a přístupnou simulaci zdrojové architektury a umožnit tak jednodušší přístup k interakci s ní. Praktické využití má pak výukový přesah, již původní účel počítače bylo seznámit uživatele s principy programově ovládaného stroje.

Kapitola 1
Úvod do problematiky

Téma této práce je nejen implementační, ale také historické. Proto je nejprve představen kontext přístroje, kterému se práce věnuje. Dále je popsána základní architektura daného stroje a přiblíženo, k čemu sloužil. Hlavním cílem práce je pak tvorba emulátoru, a tak jsou v druhé části obecně popsány způsoby emulace počítačových architektur a jejich členění. V průběhu kapitoly jsou také představeny základní termíny, které jsou v práci použity.

1.1 Výukový počítač Tesla PMI-80



Obrázek 1.1 Pohled na jednodeskový počítač Tesla PMI-80 v základní konfiguraci. [1]

Počítač Tesla PMI-80 se začal vyrábět v roce 1982 [2]. Jedná se o jednodeskový mikropočítač,

který měl nalézt uplatnění hlavně ve školách. Jeho cílem bylo zajistit přenosné, odolné a ucelené řešení pro jednoduché programování. Tyto vlastnosti pomáhá splnit uložení do kufříku, v čemž se podobá dřívějším „kufříkovým počítačům“. Inspirací byla pravděpodobně řada počítačů a periférií TEMS, dříve také vyráběná v Tesle, zahrnující školní jednodeskový počítač TEMS 80-03 [3] a také dvouprocesorový TEMS 49 [4]. Informace o těchto dřívějších strojích jsou ale velmi kusé a nepodařilo se mi na toto téma nalézt vhodné zdroje. Jisté je, že PMI-80 byl řádově populárnější a rozšířenější stroj, což i naznačuje množství dochovaných kusů nebo zmínky o distribuci v prodejní síti (za cenu 4770 Kčs [5]).

PMI-80 byl rozšířený na mnoha českých školách technického zaměření (včetně několika kusů na ČVUT) a tak lze nalézt mnoho pamětníků, kteří na něj vzpomínají. Z těchto vzpomínek se dá také zjistit, že byl často využíván pro obsluhu periferií, pro řídící aplikace v průmyslu a na jiné nenáročné výpočetní účely. Jisté pozornosti se mu dostává dodnes. Jde například o fanouškovské stránky s popisy architektury a návody na zprovoznění, nebo stavbu repliky. Zároveň ale vznikly v posledních letech nové programy pro tuto platformu. Několik na internetu dostupných pokusů rozšířila i soutěž v tvorbě programů pro PMI-80 [6] pořádaná v roce 2022 nadšenci okolo diskusního fóra OldComp [7].

1.2 Architektura počítače

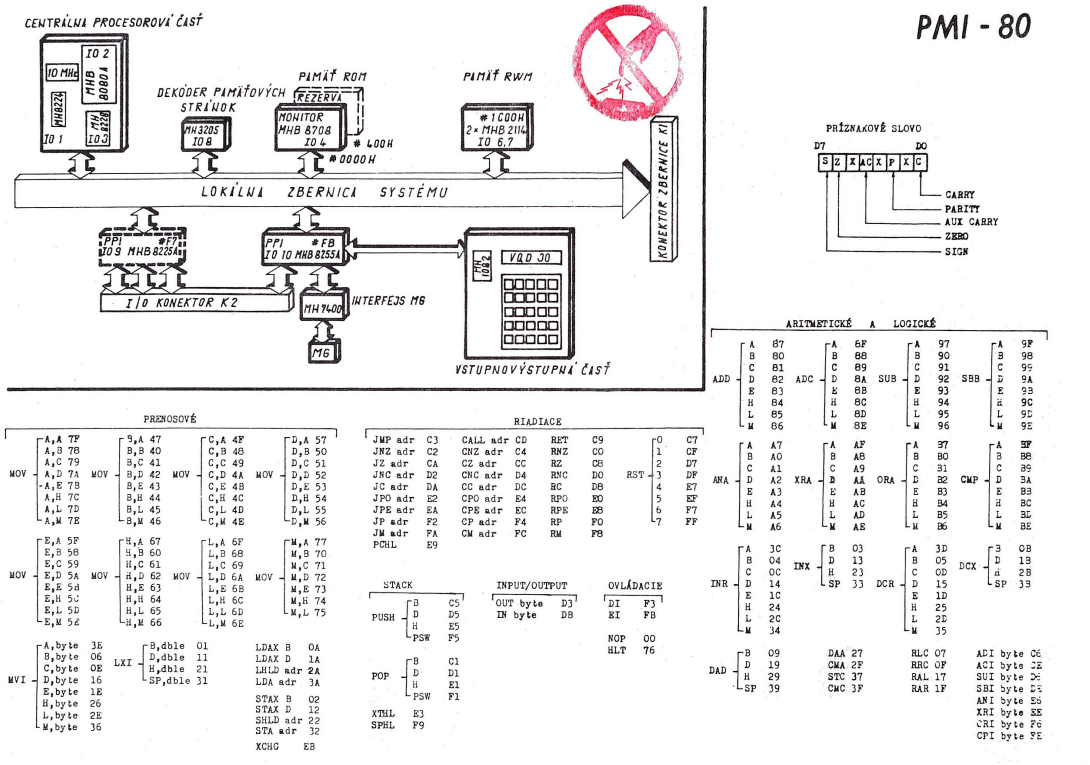
Tesla PMI-80 má základní architekturu postavenou okolo procesoru Tesla MHB8080A, československého klonu procesoru Intel 8080. Ten je 8-bitovým předchůdcem budoucích procesorů standardu x86, na kterém fungují osobní počítače dodnes. Na lokální sběrnici je také napojen tříkanálový programovatelný paralelní interface MHB8255A. Jeden z kanálů je přístupný pro externí periferie a je vyveden na konektor K2, zbylé dva kanály pak obsluhují vstupy a výstupy. Jde především o sedmisegmentový displej o 9 cifrách VQD 30, který využívá technologii LED, používanou primárně v kalkulačkách. Jako vstupní zařízení slouží klávesnice, která ale obsahuje pouze 25 kláves, jednu pro každý z 16 znaků hexadecimálního kódu a několik ovládacích kláves. Konektor K3 slouží k připojení magnetofonové jednotky, na kterou lze ukládat programy, nebo je z ní nahrávat.

K dispozici je také paměť RAM o velikosti 1 kB a v základu stejně velká ROM obsahující monitor, který poskytuje základní obslužné rutiny. Návrh desky zároveň počítá s možností rozšíření o další 1 kB ROM či EPROM s vlastním programem a také s druhým interface MHB8255A, který rozšíří možnosti komunikace s periferiemi. Konektor K1 slouží k připojení na lokální sběrnici a vznikla pro něj okolo roku 2010 moderní rozšiřující karta poskytující další 2 kB RAM a 8 kB ROM se softwarem například pro programování přepisovatelných pamětí [8]. Ta byla později dále rozšířena o 8 kB RAM, čímž pokryla většinu adresního prostoru.

1.3 Metody spouštění softwaru

Obecně se dá říci, že architektura počítačového systému je teprve polotovar, který lze během implementace zpracovat mnoha způsoby. Ve spojitosti s dalšími kroky padají termíny jako emulace a simulace, které nemají jednoznačný výklad a bývají volně zaměňovány. V následující části popíšu základní rozdělení těchto metod a stanovím, jak budou pojmy dále v práci použity.

- Virtualizace
- High-level emulace
- Low-level emulace
  - S přesností na úrovni taktů
  - S přesností na úrovni logických hradel



Obrázek 1.2 Schéma mikropočítače PMI-80, které bylo vložené do víka kufříku. Obsahuje také nápovědu pro programování ve strojovém jazyku Intel 8080 a obsah registru s příznaky. [9]

- Fyzikální simulace

1.3.1 Virtualizace

Základním problémem spouštění softwaru na jiné než cílové platformě je většinou nekompatibilita strojového kódu. Problém přenositelnosti programové výbavy našel mnoho různých řešení a dnes se hojně využívají interpretované jazyky, nebo prohlížečové aplikace. Ale ještě dříve, než narazíme na problémy s rozdílnou architekturou, můžeme narazit na problém s běhovým prostředím. Virtualizace se hojně využívá například jen k oddělení programu od zbytku počítače do tzv. sandboxu, ve kterém nemůže program ovlivnit zbytek systému. Mezi typické zástupce tohoto druhu virtualizace patří Docker a VMWare. Virtualizovaný systém se může i lišit například verzí Windows a lze tak spouštět programy nespustitelné v těch současných.

O krok dále je pak asi nejznámější projekt Wine, který dovoluje spouštění programů určených pro Windows na operačních systémech jako je MacOS a různé varianty linuxových systémů [10]. Je zde také dobře ilustrovaná nejednoznačnost terminologie. Název Wine je podle tvůrců tautologickou zkratkou z věty „Wine is not (an) emulator“, oproti časté interpretaci „Windows emulator“. Autoři jej považují za vrstvu kompatibility, jelikož primárně využívá přímo hardware systému, na kterém běží. Virtualizace se také používá primárně mezi správci systémů a při práci s cloudem a jde tak vymezit do nějaké míry i sférou, ve které se využívá.

1.3.2 High-level emulace

Emulace je dnes většinou vnímána jako proces, ve kterém dochází, alespoň částečně, k překladu instrukcí. Z technologického hlediska je to definice zcela nedostatečná a dá se argumentovat o tom, kde přesně leží hranice. Pro účely této práce ale stačí jednoduchá představa spouštění programů určených pro jinou architekturu. Přídomek high-level pak značí, že jde o formu emulace s vyšší mírou abstrakce. Nesnaží se tedy virtuálně procházet všemi stavy přesně jako cílová platforma, ale pouze počítat na pozadí správné reakce dostačující pro zajištění běhu programu.

High-level emulace (HLE) je klíčová pro jakékoliv moderní platformy, kde je komplexita architektury přílišnou výkonnostní zátěží a navíc je často zbytečná. HLE emulátor může zajišťovat emulaci pouze na úrovni volání knihoven a funkcí operačního systémů, případně překladu grafických instrukcí. Dobrým příkladem je DOSBox, emulátor 16-bitové architektury x86, který nachází široké využití pro zpětnou kompatibilitu moderních Windows s operačními systémy MS-DOS, Windows 3.x a Windows 9x [11]. Vzhledem k tomu, že jde o příbuzné architektury, lze většinu kódu spouštět přímo na CPU hosta a pouze dohlížet na specifické případy a systémová volání.

Není však pravdou, že high-level emulace a příbuznost architektur znamená snadnou práci. Většinou není dostupná kompletní dokumentace a tvorba emulátoru vyžaduje značnou míru reverzního inženýrství, a to občas i v případě spolupráce s výrobci a držiteli práv. Ilustrovat to lze na systému Xbox od Microsoftu, který využívá v základu architekturu x86. Používá ale proprietární grafický hardware a má vlastní operační systém. Díky tomu většina pokusů o HLE selhala a dnešní emulátory od tohoto přístupu upustily, přesto je pokrok pomalý [12]. Velká část dnešních emulátorů pro novější platformy využívá HLE pro některé nekritické komponenty, nebo nabízí možnost nižších nároků na výkon za cenu přesnosti emulace.

1.3.3 Low-level emulace

Na rozdíl od HLE se low-level emulace (LLE) snaží co nejvíce přiblížit cílové architektuře. Tyto emulátory překládají kód instrukci po instrukci a emulují chování procesoru včetně přerušení a stavů registrů. LLE emulátory jsou často používány k emulaci 8 a 16-bitových architektur, na kterých se také vývojáři seznamují s principy tvorby emulátorů.

Pokud je procesor dobře zmapovaný, není příliš obtížné napsat obstojný emulátor pro jen několikanásobně výkonnější počítač. První emulátory tvořené amatéry v devadesátých letech minulého století přistupovaly k problému tímto způsobem. Většinou ale chybí vhodné testovací prostředí a emulátory byly vyvíjeny pro kompatibilitu s nejpopulárnějšími programy a hrami. Části systému, ke kterým nebyla dokumentace (nejzásadněji grafický čip), se braly jako black box a kód emulátoru byl upravován podle viditelného výstupu bez porozumění všem procesů vnitřní logiky.

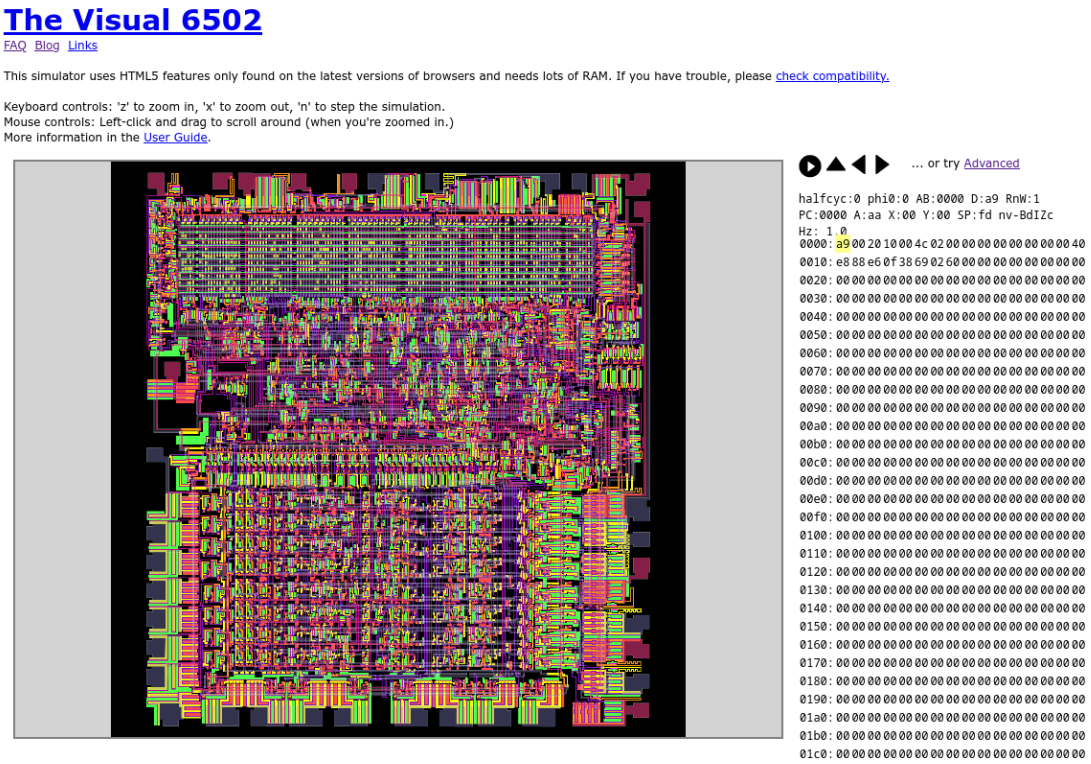
V poslední dekádě ale pokročily metody reverzního inženýrství a díky píli komunity nadšenců se stav dostupné dokumentace pro mnoho nejpopulárnějších architektur dostal do stavu, kdy je možné popsat jejich chování do nejmenších detailů. S tím ale přišly zvýšené požadavky na výkon, a to hlavně díky nutnosti popsat časování práce jednotlivých čipů a jejich vzájemnou synchronizaci. Ta většinou musí být natolik přesná, že celý program musí běžet na jednom jádře hostujícího procesoru, jejichž výkon roste v posledních letech mnohem pomaleji.

Near[1], autor významného emulátoru bsnes pro platformu Super Nintendo Entertainment System, strávil dokumentací a emulací jedné architektury dvě dekády. Ve slavném článku popisuje synchronizaci jako základní kámen úrazu pro přesnější emulaci a objasňuje, proč jeho program vyžaduje desetinásobek výpočetního výkonu oproti starším řešením [13]. Tomuto přístupu se dnes říká *cycle accurate*, jelikož cílem je, aby na konci každého taktu simulovaného procesoru odpovídal stav všech komponent reálné architektuře.

[1] Near o sobě přemýšlel jako o gendrově neutrální osobě. V Češtině pro to zatím neexistuje standard, referuji k němu tedy jako k autorovi v mužském rodě.

Extrémní přístup, prozatím nepraktický i pro ty nejjednodušší architektury, je úplný popis vnitřního rozložení čipů. I výše popsaná metoda se totiž snaží jen dosáhnout stejného výsledku jako původní systém. Pokud bychom ale vzali základní jednotku binární logiky – logické hradlo, mohli bychom popsat celý systém pouze pomocí nich. Takový přístup je z hlediska simulace logiky perfektní a nemůže poskytovat jiný než přesný výsledek. Znamená to ale nárůst komplexnosti o další řád a ani nejvýkonnější počítače dnešní doby nejsou schopné emulovat takové systémy v reálném čase.

Přesto jsou tyto emulátory užitečnou referencí pro případnou korekci jiných řešení, nebo úplnou dokumentaci fungování čipu a zároveň poskytují unikátní vhled do struktury počítačových systémů na nejnižší úrovni. Krásnou ukázkou je projekt Visual6502, emulující hojně používaný mikroprocesor MOS 6502 [14]. Tento emulátor dokonce poskytuje grafické rozhraní, ve kterém lze sledovat takt po taktu aktuální stav všech tranzistorů v celém čipu (viz obrázek 1.3).



Obrázek 1.3 Webové rozhraní projektu Visual6502 ukazující grafický výstup emulátoru. Po pravé straně je vidět obsah paměti. [14]

1.3.4 Fyzikální simulace

Fyzikální simulace jsou založené na matematickém modelování a využívají se běžně k simulaci reálného světa. Specificky ve vztahu k emulaci jde ale o proces, který se snaží popsat jevy mimo digitální logiku zařízení. Za jednoduchou simulaci by se dala považovat implementace specifik některých periferií. Například jeden z emulátorů Game Boye od firmy Nintendo má podporu pro emulaci propojení se speciálním modelem šicího stroje, včetně simulovaného zobrazení vyšívaných vzorů posílaných z emulovaného zařízení [15].

Zdaleka nejvíce se ale fyzikální simulace využívají pro analogové procesy, jako je například simulace zvukových procesorů. Mnoho raných digitálních zařízení používalo i analogové obvody,

nejčastěji právě pro generování zvuku. Simulátory tohoto typu nejsou příliš běžné. Emulátor MAME [16] nabízí framework nazvaný netlist, který využívají některé z jeho systémů pouze pro analogovou část původního hardwaru a který komunikuje s běžným emulátorem. Dalším příkladem je dnes již neaktivní Digital Integrated Circuit Emulator (DICE) [17]. DICE je open-source simulátor, který umožňuje simulovat chování analogových obvodů na úrovni jednotlivých elektrických prvků a podporuje několik arkádových automatů ze 70. let, jejichž TTL logiku kompletně simuluje. Tato cesta je samozřejmě nejnáročnější na výkon a málokdy je opodstatněná.

Kapitola 2
Analýza

Základem této práce je implementace architektury počítače PMI-80. K tomu lze zvolit mnoho způsobů, například tvorbu repliky nebo jeden z mnoha druhů emulace, které jsou představeny v předchozí kapitole. V následujících sekcích jsou rozebrány jednotlivé možnosti a případně popsána současná řešení. Na závěr je zhodnoceno vybrané řešení a důvody, které mě k němu vedly.

2.1 Výběr implementační metody

Nejdůležitější fáze každé implementace nastává před tím, než člověk napíše jediný řádek kódu. Důkladná analýza vede k lepšímu naplnění vytyčených cílů, a tak přestože práce počítá již ve svém zadání s hardwarovým řešením, je potřeba objasnit důvody takového rozhodnutí a porovnat jeho výhody a slabiny s jinými možnými přístupy.

2.1.1 Fyzická replika

Nejpřímější metoda reimplementace je výroba fyzické repliky. Pokud je architektura postavená pouze na běžně dostupných obvodech, je tato možnost poměrně jednoduchá. U architektur využívajících proprietární čipy se často musí spoléhat na dostupnost skladových zásob, nebo se jedná o hybrid mezi replikou a emulací.

Proces tvorby repliky u PMI-80 proběhl již několikrát. Krom snahy o přesnou repliku je nejpopulárnější stavba PMI-80 M16, která je plně kompatibilní s původní architekturou, ale nahrazuje obtížně sehnatelné součástky současnými alternativami (předně se jedná o displej, klávesnici a paměť ROM) [18]. Existují ale i variace architektury využívající jiné soudobé procesory – PMI-85 na základě procesoru Intel 8085 [19] a PMI Z-80 postavené okolo procesoru Zilog Z80 [20]. Tyto snahy jsou ale spíše zajímavým projektem pro kutily a neexistuje okolo nich širší komunita.

Tvorba fyzické repliky umožňuje sestrojit velmi přesnou náhradu původního počítače, ale neodhaluje skoro nic z vnitřního fungování. Zároveň spoléhá na schopnost uživatele zajistit potřebné součástky a mít alespoň základní schopnosti práce s elektronikou. Není tedy vhodným způsobem pro rozšíření platformy mezi veřejnost a bariéra pro práci s počítačem je při tomto postupu poměrně vysoká. Z toho důvodu jsem se rozhodl ve své práci věnovat emulaci.

2.1.2 Softwarová emulace

Po rozhodnutí vytvořit virtuální reprezentaci PMI se nabízely dvě varianty. Jelikož je systém zcela digitální, je možné sestrojit emulátor na úrovni binární logiky a není potřeba simulace fyzikálních

jevů. Typicky se jedná o emulaci softwarovou, která umožňuje překlad chování programů pro PMI-80 na moderních architekturách. Díky podobnosti procesorů by v případě emulátoru pro Windows mohlo jít teoreticky i o HLE, ale takový přístup by byl zcela nepřiměřený. V případě PMI-80 je celková komplexnost velmi nízká, a tak není potřeba například ladit časování pro komunikaci mezi procesorem a grafickým čipem.

Zároveň by softwarový emulátor umožnil snadné napojení na vývojová prostředí a zjednodušil tak programování. Mnoho emulátorů dokonce nabízí zabudovaný debugger či rozhraní pro komunikaci s virtuálním strojem. Stejně jako v případě replik ale PMI-80 již má několik softwarových emulátorů dostupných [21][22]. Dokonce existuje emulátor napsaný v jazyce Java, který si jde snadno spustit v prohlížeči [23].

2.1.3 Emulace za pomocí obvodu FPGA

Druhou možností je využití některého z jazyků pro popis hardwaru a následná emulace za pomocí hardwarového přípravku. K tomu účelu se používají jazyky Verilog a VHDL, které jsou syntetizovatelné za pomocí vývojových nástrojů pro obvody FPGA. Díky desce s FPGA čipem je možné vytvořit fyzickou repliku počítače a zároveň slouží program jako detailní popis jeho vnitřní logiky. Díky standardním vstupně-výstupním možnostem některých desek by bylo možné i připojit původní periferie. Během analýzy jsem navíc nedohledal žádný aktivní projekt implementující PMI-80, přestože jsem později objevil rovnou dva – fpmi [24] a FPGA SBC [25]. Oba ale mají jiný účel, než implementace z této práce. Slouží pouze jako integrované řešení pro zjednodušení tvorby softwarově kompatibilní fyzické repliky.

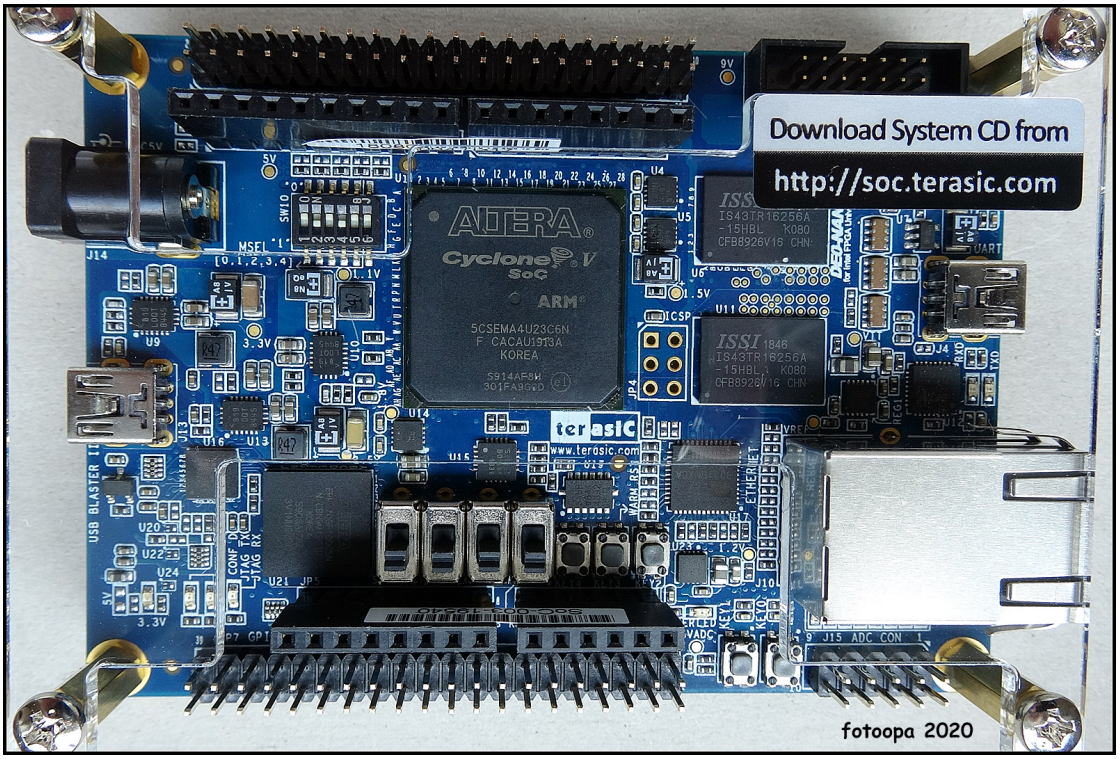
2.2 Vybrané řešení

Rozhodl jsem se pro implementaci v FPGA obvodu. Ze studia mám zkušenosti s jazykem VHDL na přípravku Basys-3 a s jazykem Verilog v simulátoru. Jako primární jazyk jsem zvolil VHDL, jelikož jsem s ním pracoval v nedávné době a cítím se v něm jistější. Při volbě přípravku pak bylo důležitých několik věcí. Cílem bylo ideálně rozšířit povědomí o tomto počítači mezi širší veřejnost, a tak byly kritériem výběru nejenom technické parametry, ale i dostupnost vývojové desky.

Už v minulosti jsem byl seznámen s projektem MiSTer, který sdružuje komunitu okolo vývojové desky DE-10 Nano (viz obrázek 2.1) [27]. Nabízí platformu pro integraci jednotlivých jader (core), které využívají obsažený FPGA čip k emulaci široké škály architektur. Zároveň ale zapojením do ekosystému MiSTera získá vývojář mnoho výhod. Jsou mu dostupné knihovny řešící základní problémy jako digitální i analogový obrazový výstup, nebo připojení ovládacích prvků. Také lze jádro přidat do oficiálních repozitářů, načež se automaticky bude stahovat všem uživatelům a udržovat v aktuální verzi. Rovněž je to asi jediná platforma na bázi FPGA, kterou pravidelně využívají i lidé bez odborného zájmu a které tak lze seznámit s historickým počítačem.

2.3 Projekt MiSTer

Počátky fanouškovských snah, které o mnoho let později vedly ke vzniku MiSTera, leží u platformy MiniMig – FPGA implementace počítače Amiga od značky Commodore, který byl vyvinut a prodáván v 80. letech [28]. Komunita okolo Amigy byla vždy velká a mnoho fanoušků značky dlouhá léta odmítalo přechod na standard IBM PC. I proto dodnes existuje čilá aktivita okolo hardwaru i softwaru počítačů Amiga. MiniMig začal v roce 2005 na vývojové desce Xilinx Spartan-3 a jako projekt funguje dodnes. Zároveň je, jako většina FPGA simulací historických architektur, open source. Tento projekt zaujal i Alexeye Melnikova, budoucího tvůrce MiSTera [29],



Obrázek 2.1 Deska přípravku DE-10 Nano od firmy Terasic obsahující FPGA čip Altera Cyclone® V SE 5CSEMA4U23C6N ve své základní konfiguraci. Na tomto přípravku je založený projekt MiSTer a je cílovou platformou implementace z této práce. [26]

Po seznámení se s principy FPGA emulace se Melnikov přesunul k projektu MiST [30]. Ten využíval mnohem pokročilejší FPGA z řady Cyclone III od Intelu a stejně jako MiniMig byl původně zaměřen na 16-bitové počítače; krom Amigy i na konkurenční Atari ST (odtud i název **Ami**ga a Atari **ST** – MiST). Silnější výkon FPGA a rostoucí komunita ale brzy přinesla vývoj nových jader implementujících architektury rozličných historických počítačů a herních konzolí.

Melnikov ale narážel na dva problémy. Zaprvé, hardware byl vyráběný přímo pro projekt, a tak byl velmi nákladný a závisel na možnostech komunity vyrábět hotové desky za použití samotných FPGA čipů. Druhým problémem bylo, že poskytoval pouze analogový video výstup, a tak byli vývojáři i uživatelé na moderních obrazovkách nuceni používat různé konvertory a převodníky, které ale ne vždy fungovaly správně. Melnikov se tedy v roce 2017 rozhodl, že nástupce bude využívat komerčně dostupnou desku s digitálním video výstupem. Tou se stala DE-10 Nano od firmy Terasic a projekt získal název MiSTer.

V současnosti zažívá rapidní rozvoj díky rozšiřování obecného povědomí o technologii FPGA a zároveň propagaci mezi nadšenci do historické výpočetní techniky. V oficiálním repozitáři je vedeno 62 jader osobních počítačů, 31 jader herních konzolí a 114 jader arkádových automatů. Mimo jiné zde nalezneme i československé mikropočítače Tesla PMD 85 [31] a Ondra [32].

MiSTer dnes získává ohlas hlavně v komunitách okolo softwarové emulace, která oceňuje výhody tohoto řešení, jako je nízká latence ovládacích prvků, snadnější připojení na dobové obrazovky, nebo jen kompaktnost řešení, které se vejde do malé krabičky. Nová vlna zájmu zároveň při tvorbě jader pro platformy s existujícími emulátory přináší nové poznatky o těchto strojích, které pak zlepšují stav dokumentace a tím kvalitu hardwarových i softwarových řešení. Někteří vývojáři podporovaní komunitou dokonce pracují na jádrech pro MiSTer na plný úvazek. K základní desce také existuje nabídka příslušenství, jako rozšiřující desky nebo adaptéry pro

periferie a dokonce je tu možnost pro laické uživatele zakoupit sestavu MiSTera jako samostatný produkt včetně krabičky [33].

Kapitola 3
Popis implementace

V kapitole je popsán finální stav implementace, ale také klíčová zjištění z jejího průběhu a problémy, které bylo potřeba vyřešit. Rozebírá detaily jednotlivých částí architektury a shrnuje proces integrace jádra do platformy MiSTer.

3.1 Vývojové prostředí

Pro vývoj na desce DE-10 Nano se používá prostředí Quartus, specificky ve verzi 17, neboť v dalších verzích už nejsou žádné změny týkající se této desky a jednotné vývojové prostředí umožňuje uniformitu základní struktury jádra a méně chyb způsobených změnami mezi verzemi. Quartus jsem instaloval pod operačním systémem Fedora a díky tomu, že je oficiálně podporován příbuzný linuxový systém RHEL, se nevyskytly žádné obtíže. Na problém jsem ale narazil při simulacích, neboť Quartus pro ně využívá standardně externí program ModelSim-Altera. Ten je v neplacené verzi ale poskytován pouze jako 32-bitový a jeho linuxová verze nejde spustit se současnými systémovými knihovnami. Nalezl jsem nějaké rady, jak si zkompilovat starší verze knihoven a simulaci zprovoznit, ale od té doby se změnilo ještě více závislostí a nakonec jsem se rozhodl simulace pouštět pod Windows.

Nejprve jsem ověřil, zda jde VHDL popis zkompilovat pro danou desku, ale s vývojem jádra jsem pak již pokračoval v šabloně, kterou MiSTer poskytuje. Šablona nabízí top modul `emu`, v rámci kterého je navázané rozhraní MiSTera na jádro a který obsahuje mnoho nastavení. Samotná logika jádra je umístěná ve složce `rtl`, složka `sys` pak obsahuje zdrojové kódy poskytnutých služeb. Šablona vyžaduje, aby jádro využívalo PLL (phase-locked loop) obvod pro generování hodin a krom obsluhy vstupů a výstupů poskytuje také komunikaci s procesorem na desce. Ten může zajišťovat paměťové operace s externí SDRAM či DDR3 RAM a také komunikuje se službou OSD (on-screen display), skrze kterou uživatel mění nastavení, nahrává software do paměti a případně nahraje jiné jádro, nebo se vrátí do úvodní nabídky MiSTera. Seznam souborů jádra, které nejsou součást šablony, se nahrává ze souboru `files.qip`, díky čemuž lze většinou aktualizovat šablonu na nejnovější verzi pouhým přepsáním všech ostatních souborů.

3.2 Architektura PMI-80

Při implementaci jsem hodlal využít již hotových popisů základních komponent architektury ve VHDL či Verilogu. Zatímco jsem dohledával informace o počítači, narazil jsem na již hotovou implementaci PMI-80 pro FPGA s názvem fpmi. Implementace byla určená pro fyzickou repliku a jelikož je open source, chtěl jsem vystavět své řešení na základu této implementace. Po nějakém čase stráveném její úpravou jsem ale zjistil, že k tomuto účelu není příliš vhodná. Z důvodu

Obrázek 3.1 Uvítací obrazovka monitoru PMI-80 zobrazená emulátorem běžícím na DE-10 Nano. Deska s přípravkem je v levé spodní části. Vstup je ovládán standardní klávesnicí. [Autorské foto]

nedostupnosti původních komponent totiž fpmi používá jiný protokol komunikace s displejem a klávesnicí. Zároveň byl projekt naprogramován ve značném spěchu a není příliš vhodný pro snadné úpravy.

Během modifikací fpmi jsem ke svému překvapení nalezl díky zmínce na fóru OldComp druhou hotovou implementaci PMI-80 do FPGA, tentokrát v rámci komplexnějšího projektu FPGA SBC zaměřujícího se na jednodeskové počítače. Modulární design zdrojových kódů je pro úpravy mnohem vhodnější a implementace odpovídá zapojení původního PMI-80 včetně protokolů pro vstupy a výstupy. Navíc dokonce nabízí PMI v sestavě s dobovými i moderními rozšířeními. Pro procesor je použit popis z projektu T80, který jsem plánoval využít i při případné vlastní implementaci, a krom toho obsahuje FPGA SBC jednoduchého správce paměťových přístupů a modul obvodu MHB8255A. Samotné paměťové obvody jsou standardní, generované vývojovým prostředím.

Spuštění PMI-80 z FPGA SBC bylo jednoduché a v simulaci jsem si ověřil, že instrukce monitoru jsou vykonávány a procesor běží správně. Krom toho, že top modul simuluje pull-up porty a očekává aktivní signály na nule, kvůli čemuž jsem mnohokrát kompiloval s trvalým resetem, jsem během práce s FPGA SBC narazil jen na jeden podstatný problém. Když jsem přenesl moduly do projektu s cílovou deskou DE-10 Nano, vyžadoval po mě Quartus aktualizaci obvodů, které generuje za pomocí programu MegaWizard. Všechny paměti jsem tak jen nechal automaticky rekonfigurovat. Jakmile jsem ale poprvé spouštěl na desce jádro s tím, že bych měl vidět obrazový výstup, nic se neukázalo. Po dlouhém testování jsem zjistil, že paměťové obvody na DE-10 Nano mají přístup při čtení trvající dva takty. Procesor ale už po prvním taktu očekával nová data, takže jádro samozřejmě nefungovalo. Zvýšil jsem tedy frekvenci hodin na dvojnásobek

a běh procesoru podmínil povolením.

3.2.1 Frekvence hodin

Na konci implementace jsem se obrátil k problematice hodin a jejich taktu. Po celou dobu jsem pracoval s provizorními hodnotami a dokonce jsem občas využil pomalejší hodiny pro pozorovaní změn stavu počítače lidským okem. Ideální stav je, aby celý systém běžel na jedněch hodinách. To však není možné z důvodu odlišného časování VGA signálu video výstupu – v mém případě 25.116279 MHz. Alespoň veškerá další logika by ale měla být na stejných hodinách. Z důvodu přístupu do paměti jsem už dříve zavedl odvozený signál `clock_enable` pro zpomalení procesoru, aniž by byl na odlišných hodinách. Byl jsem také varován, že při nízkých hodinových frekvencích, ve kterých bych se ideálně rád pohyboval, nemusí fungovat správně komunikace s modulem `hps_io`, kterým nahrávám obsah paměti. Pro generování hodin využívám PLL obvod, který deska DE-10 Nano nabízí. Skrz Altera MegaWizard jsem nakonfiguroval přesnou hodnotu video hodin odvozenou ze základních hodin na frekvenci 50 MHz. Velmi specifický požadavek s přesností na šest desetinných míst ale znamenal, že PLL nebylo schopno generovat další složité frekvence.

Po většinu vývoje jsem požadoval od PLL obvodu osminásobek frekvence procesoru v PMI-80, tedy 8.888888 MHz. Byl mi poskytnut nepřesný signál odpovídající zhruba hodnotě 8.5 MHz, což ale dostačovalo pro testování. Ve chvíli, kdy zbývalo již jen doladit časování mého jádra, jsem začal přemýšlet nad možnými řešeními. Nejprve mě napadlo použít přímo matematický výraz, kterým je frekvence procesoru určená. Odpovídá přesně 10/9, tedy by stačilo vzít 10 MHz signál a pomocí děličky hodin ho vydělit devíti. Dělení signálu lichými čísly je ale problematické a nebyla možnost využít druhý PLL obvod, neboť s tím šablona MiSTera nepočítá. Nakonec jsem se tak rozhodl pro z hlediska návrhu čistší řešení a použil jsem pro všechny obvody 20 MHz signál a pro procesor jsem odvodil `clock_enable` na každý osmnáctý takt, čímž jsem úspěšně dosáhl frekvence původního procesoru 1.111111 MHz. S tím jsem úspěšně ověřil časování oproti reálnému stroji na programu tenis.

3.3 Vstupy a výstupy

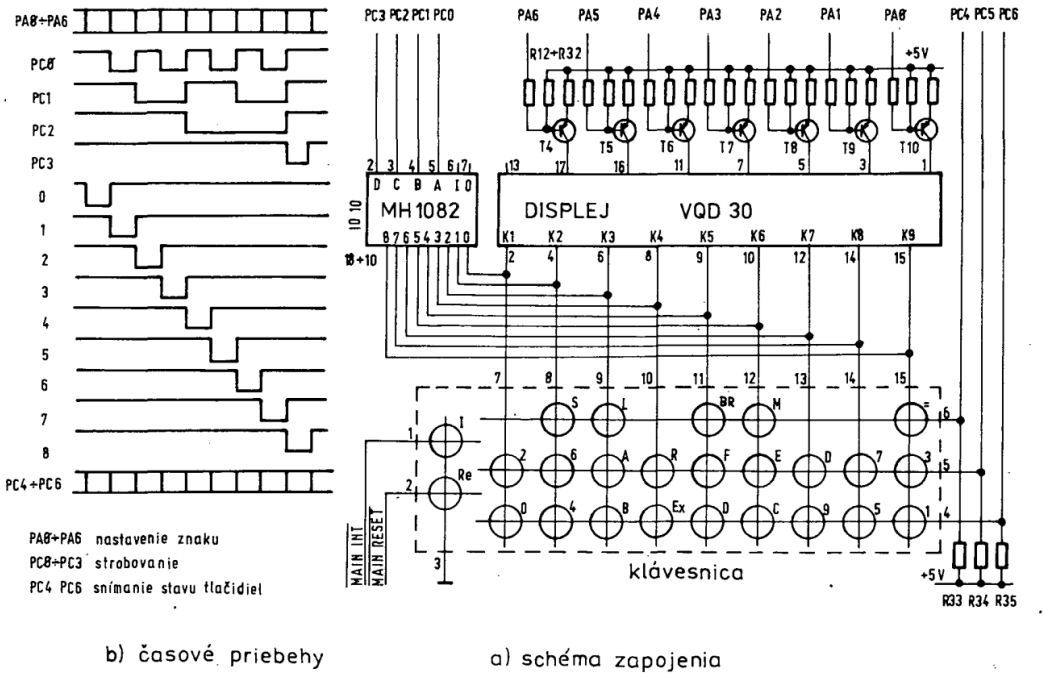
Vstupy a výstupy počítače PMI-80 jsou obsluhovaný obvodem MHB8255A na portech *a* a *c*. Port *b* je tak ponechán pro využití v programech a v současné implementaci není obsluhován. Stejně tak nejsou zapojeny piny pro komunikaci s kazetovou jednotkou určenou k ukládání a nahrávání programů.

3.3.1 Displej

Displej je tvořený devíti pozicemi sedmisegmentových znaků. Nejsou ale ovládány přímo, místo toho se dekodérem MH1082 prochází pozice displeje a vykreslují se jednotlivé znaky. Tento proces běžně zajišťuje podprogram `OUTKE`, který poskytuje monitor. V paměti monitoru je také uložena znaková mapa, kterou lze jednoduše vykreslovat nejužitečnější konfigurace segmentů. Výstup je vyveden na piny PA0 až PA6 obvodu MHB8255A a pozice je určená výstupem dekodéru (nízké napětí na jedné z devíti linek). Stav displeje není ve svém celku nikde ukládán. Počítá se s tím, že program zavolá vykreslovací funkci dříve, než dojde k dosvitu LED segmentů použitého displeje VQD 30. Schéma zapojení ukazuje obrázek 3.2.

V rámci mé implementace je v souboru `display.vhd` popsaný modul, který zachycuje vstupy, které by běžně šly do displeje, a ukládá jejich stav do vlastních vnitřních registrů. Během implementace jsem narazil na problém s rapidním blikáním zapnutých segmentů. Zpomalením hodin jsem zjistil, že výstup není validní po celou dobu, po kterou je daný znak vybrán. Během přesahu signálu z dekodéru byly na výstupu všechny segmenty vypnuté a registry se tak naplnily nulovou hodnotou. Změnil jsem tedy funkci registrů tak, že reagují pouze na aktivní signál, při kterém

zapíšou jedničku a zároveň resetují čítač simulující dosvit LED. Pokud tedy dojde během odpočtu k opětovnému zapsání hodnoty, tak zůstane segment zapnutý po celou dobu a zároveň nijak nereaguje na prázdný výstup v době přesahu dekodéru. Dobu dosvitu jsem nastavil na 10 milisekund velikostí čítače.



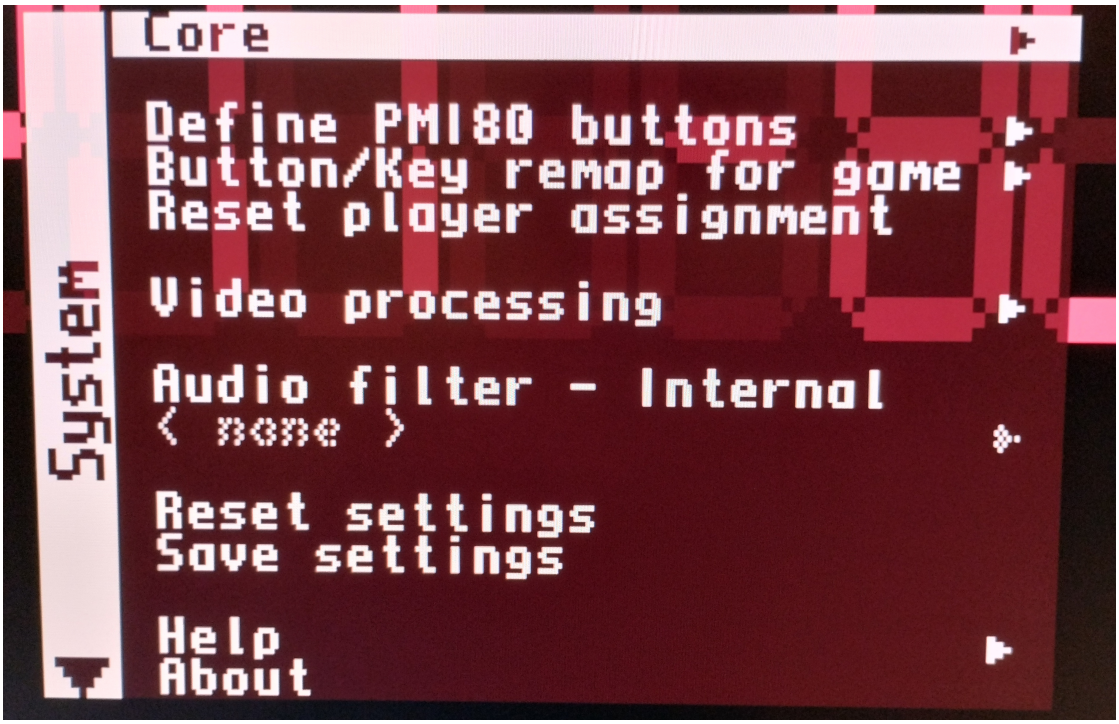
Obrázek 3.2 Schéma ukazující zapojení klávesnice a displeje a funkci dekodéru MH 1082, který prochází devět pozic. Z dané pozice se vždy čte sloupec klávesnicové matice a zapisují segmenty číslice. [34]

Segmenty je potřeba vykreslit do video signálu, na což jsem využil ukázkové implementace generování VGA výstupu (popsané v souboru `vga.v`). MiSTer poskytuje několik různých možností, jak z jádra generovat videosignál, ale způsob použitý v ukázce byl adekvátní mým požadavkům. Jedná se o generátor VGA signálu v rozlišení 640x400 při frekvenci 70 Hz. Video paměť je ale pouze o velikosti 160x100 a každý virtuální pixel je vykreslen čtyřikrát. Jelikož je sedmisegmentový displej nepříliš náročný na rozlišení, omezení nebylo problematické. Stejně tak použitá barevná hloubka (8 bitů na pixel s rozložením RGB barev 3:3:2) bohatě stačila pro zvolené černé pozadí a sytě červené segmenty. Upravil jsem tedy hlavně způsob zápisu do paměti. Kód poskytoval rozhraní pro obsluhu procesorem, ale nakonec jsem se rozhodl napojit registry s hodnotami jednotlivých segmentů přímo na video paměť. Ke generování kódu pro signály jsem použil jednoduchý skript a počáteční souřadnice v paměti jsou volitelné za pomocí parametrů.

3.3.2 Klávesnice

Čtení z klávesnice využívá stejný princip, jako vykreslování na displej. Klávesy jsou umístěné do matice 9x3, kterou prochází podprogram `OUTKE`, zatímco zároveň vykresluje postupně devět znaků displeje. Vstupní signály PC4 až PC6 značí každý stisk jedné ze tří kláves v daném sloupci. Detaily zapojení jsou na obrázku 3.2. Speciálně pak fungují klávesy RST a INT, které jsou přímo napojené na dané piny procesoru a reagují tedy na stisk okamžitě.

Kvůli množství kláves jsem v implementaci zavrhl možnost rozhraní přijímat vstup z ovladače a vytvořil pouze adaptér pro klávesnici. Využil jsem vstupní signál `ps2_key_i`, který poskytuje scancode klávesy v komunikačním protokolu PS/2 a zároveň příznak toho, zda se jedná o rozšířený scancode, zda je klávesa stisknutá a zda došlo ke změně hodnoty signálu. Vzhledem k tomu, že PMI čte vstup z klávesnice jen v určitých intervalech, musel jsem ukládat současný stav všech kláves (teoreticky by uživatel mohl stisknout i všechny klávesy naráz). Vytvořil jsem si tedy registr pro každou klávesu, kterému jsem měnil hodnotu podle toho, zda naposledy přišel kód klávesy s příznakem stisknutí nebo ne. Zároveň entita posílá na výstup klávesy jen právě zvoleného sloupce.



Obrázek 3.3 On-screen display nabídka s nastavením jádra poskytovaná rozhraním MiSTera. Konkrétně se jedná o sekci s obecnými možnostmi, jako je změna mapování kláves nebo audio a video filtry. [Autorské foto]

3.4 Integrace do prostředí MiSTera

Napojení na poskytované rozhraní bylo přímočaré, pokud jsem z dokumentace dostatečně vyrozuměl, jakým způsobem k němu má dojít. Pro OSD jsem nakonec nevytvářel žádná nastavení specifická pro jádro, ale vložil jsem dialogy pro nahrávání programů. Samozřejmě zůstala možnost obecných nastavení, která jsou poskytnutá každému jádru (viz obrázek 3.3). Hodiny jsem odvodil z PLL obvodu a z rozhraní jsem využil vstup pro klávesnici, vstupy pro čtení z SD karty pro nahrávání softwaru a výstupy pro VGA video signál. Také jsem na LED diody na desce DE-10 Nano vyvedl signalizaci přístupu do paměti ROM, paměti RAM a do periferií.

3.4.1 Nahrávání do paměti ze souboru

Přístupy do paměti byly nakonec jediná funkce, která vyžadovala podstatné zásahy do vnitřní logiky počítače. Paměťové obvody byly generované funkcí MegaWizard prostředí Quartus a jed-

nalo se o generické ROM a RAM obvody. Do základní ROM byl při kompilaci vložen monitor za pomocí inicializačního souboru. Jádra vyžadující pro své fungování data v paměti využívají funkce MiSTera pro automatickou inicializaci při zapnutí jádra. Obzvláště pro autorsky chráněný software je potřeba nechat jejich zajištění na uživateli. Tato funkce ale není příliš dobře zdokumentovaná a její implementace tak zabrala delší čas. Rozhraní poskytuje signály `ioctl_wr` (příznak zápisu), `ioctl_addr` (adresa v paměti), `ioctl_data` (přenášená data) a `ioctl_index`. Použitím `ioctl_index` můžeme volit, kam se data nahrají. Inicializační soubory mají index 0, nahrávání programů z menu pak má index zvolený v konfiguračním skriptu OSD. Navíc horní část `ioctl_index` určuje příponu přenášeného souboru, případně u inicializačních dat pořadí souborů. Při spuštění jádro automaticky zapíše data ze souboru `boot.rom`, popřípadě `boot1.rom`, atd.

Paměťové obvody jsem rozdělil podle fyzické reprezentace na desce PMI-80 (dva čipy ROM po 1 Kb místo jednoho 2 Kb paměťového obvodu) a zároveň jsem je vyměnil za dual-port RAM obvody. V případě přenášení velkého množství dat by bylo možné taktovat přístup z rozhraní na vyšší frekvenci, ale pro jednotky kilobajtů stačily hodiny používané pro logiku počítače. Programy pro PMI-80 většinou počítají se zavedením do RAM začínající na adrese 0x1C00, případně jde o programy na EPROM nacházející se za adresním prostorem monitoru. Vedle možnosti nahrání do těchto dvou míst umožňuje jádro i změnu obsahu ROM pro monitor a nahrání programu do rozšířené spodní části RAM.

Kapitola 4
Testování

Následující část poskytuje krátké shrnutí použitých testovacích postupů během práce, rozbor omezení, která určila rozsah testování a výhled na budoucí možné rozsáhlejší testy.

První zkouška funkčnosti typického jádra pro MiSTer proběhla vlastně už když jsem zkoušel hotová jádra. Seznámení se s prostředím, v jakém bude moje práce využívána, mi dalo porozumění toho, jak bude uživatel k mému jádru přistupovat. Co se vnitřní logiky týče, nejpodstatnější je funkčnost procesoru. Projekt T80, který implementace využívá pro modul procesoru, je použit v mnoha jiných projektech a je velmi dobře otestován. Provedl jsem jeho aktualizaci z verze 303 použité v FPGA SBC na nejnovější verzi 351, která mimo jiné právě na základě testů upravuje chování některých instrukcí [35]. Psát tak vlastní, jistě mnohem naivnější, testy mi přišlo zbytečné.

V průběhu ladění programu jsem narazil na problém se spouštěním monitoru. Pro zjištění příčiny jsem si vytvořil jednoduchý testbench (nachází se v souboru `sim/testbench.vhd`) nad celým modulem `pmi80` z projektu FPGA SBC. V rámci simulace jsem pak dohledal, že chyba byla v časování přístupů do paměti. Testbench jsem využil ještě několikrát, protože umožňoval mnohem agilnější vývoj – kompilace do přípravku trvala více jak čtvrt hodiny. Ve chvíli, kdy už bylo jádro funkční, sloužil jako vhodný základní test samotný program monitoru. Později jsem provedl test jednoduchého programu ručně vepsaného do paměti a po zprovoznění nahrávání do RAM i většiny dostupných programů pro PMI, jmenovitě programy Tenis [36], Cosmos [37] a programy ze soutěže v programování pro PMI-80 [6].

Testování ztěžovala komplexita celé nadstavby poskytované MiSTerem. Projekt vytvořený v šabloně pro jádra v podstatě nešlo simulovat, krom nutnosti nahradit PLL obvod vlastní implementací by to vyžadovalo i nutnost obsluhovat velké množství vstupů a výstupů, jejichž funkce není dobře zdokumentována a nesouvisí příliš s funkcionalitou této práce. V budoucnu by ale bylo vhodné rozšířit testbench na celý systém včetně mnou dodaných rozhraní. Psaní testů pro obrazový výstup je obtížné, a proto jsem zvolil pouze vizuální kontrolu, ale v případě nutnosti ověřit například časování s přesností na jednotlivé takty by to bylo nutné. Pro potřeby této práce jsem se spokojil s ověřením časování pozorováním oproti programu spuštěnému na reálném hardwaru.

Kapitola 5

Využití emulátoru ve styku s veřejností

Smysl této práce není jen samotná implementace, ale i možnosti využití hotového produktu. Krom popisu integrace jádra do MiSTera je výsledkem i hotový emulátor. Způsoby jeho uplatnění naznačuje následující text.

Historický počítač, kterým PMI-80 bezesporu je, dokládá svoje omezené možnosti na každém kroku. Ani v době svého vzniku nebyl tento stroj nijak sofistikovaný, naopak měl nabídnout jen nejzákladnější funkce, primárně pro výukové účely. Přímé praktické využití emulátoru tedy rozhodně nespočívá v poskytnutí výpočetních možností, ani v kompatibilitě s drobným množstvím dochovaného softwaru. Hlavní poslání PMI-80, výuka programování, je realizováno zadáváním strojového kódu přímo do paměti ručně a je rozšířeno jen o několik málo prvků usnadňujících práci, jako je trasování programu a modifikace obsahu registrů. Jde tedy o způsob práce, který je ještě primitivnější, než programování v assembleru.

Dnes se již právem zapomenulo na nemožnost vložit novou část kódu doprostřed stávajícího kódu a mnoho dalších problémů, které řeší moderní nástroje. Ale krom toho, že to je uživatelsky nepřívětivé, je to také mnohem blíže fungování počítačů na nejnižší úrovni. Můžeme být o to vděčnější, že je velké množství práce automatizované, když si to uvědomíme při interakci s historickou technikou. Je tedy dobré se seznámit s celý procesem, než začneme spoléhat na funkce vývojového prostředí, jenž nám skryje mnohé detaily probíhajících úkonů.

Ještě jedna klíčová vlastnost programátora je zkoušena prací s PMI-80. Velkou částí vývoje softwaru není jen samotné psaní kódu, ale také čtení a interpretace cizího kódu. Tato schopnost hrála velkou roli v počátcích počítačové kultury u nás, neboť dokumentace v podstatě neexistovala a velká část programů se šířila jako výpisy. Podobně jako při práci s PMI-80 musel tedy uživatel nejprve opsat kód do paměti počítače, což byl sice zdlouhavý proces, ale s ne zcela zjevným edukativním prvkem. Čas strávený touto nutnou procedurou nabízel možnost se zamyslet nad významem zpočátku nesrozumitelných formulí a zkušenějším uživatelům již prozrazoval konkrétní funkce zadávané části. Naprosto zásadním okamžikem pak byla chvíle, kdy byl program vyzkoušen, ale zůstával uložený v paměti a vybízel k otestování nabytých znalostí a experimentaci s hodnotami a následnému pozorování jejich vlivu [38]. Tento způsob učení se za pochodu namísto postupného budování základů znalostí odspodu je zajímavou alternativou a jistě by mohl být vhodný pro použití například v zájmových kroužcích a volitelných aktivitách, kde může posílit přirozené sklony žáků k interakci tímto způsobem.

Co se týče historického významu zařízení, dochází v posledních letech k většímu zájmu o vývoj výpočetní techniky a začínají vznikat sbírky i expozice v rámci paměťových institucí. Lze zmínit například výstavu Národního technického muzea *Česká stopa v historii výpočetní techniky* z roku

2021, v rámci které bylo předvedeno velké množství místního hardwaru [39]. Většina výstavy ale ukazovala stroje vypnuté a bez možnosti je používat, což je pochopitelné, neboť návštěvníci by mohli techniku poničit. Navíc mnoho zařízení nemusí ani být ve funkčním stavu a opravy i následná údržba jsou nákladné a vyžadují expertízu.

Tento problém často řeší použití softwarových emulátorů, většinou s dobovými hrami, jako nejvhodnější ukázkou zábavné interakce. Emulátory jsou však často spuštěné v prostředí moderních operačních systémů na plochých obrazovkách a ovládají se na běžné klávesnici a myši. Většinu těchto omezení řeší právě hardwarová emulace. MiSTer nabízí jednoduchou možnost připojit video výstup na CRT obrazovku a díky vstupním portům lze připojit i repliky dobových periferií. Navíc je celý systém jednoúčelový a nehrozí komplikace s aktualizacemi operačního systému nebo procesy na pozadí. Jednoduchost použití hardwarových emulátorů v muzejním prostředí má potenciál zlepšit zážitek návštěvníků a mohla by pomoci rozšíření interaktivních zastavení do širšího spektra expozic. Počítač lokální výroby od důležitého koncernu spojený s obdobím normalizace totiž může najít uplatnění i mimo výstavy se zaměřením na výpočetní techniku a sloužit tak například k ilustraci snah o intermediální přístup a výzkum vlivu počítačů na kulturu a společnost.

Kapitola 6
Budoucí práce

V této kapitole jsou rozebrány kroky, které budou následovat po odevzdání práce a také rozvedeny směry, kterými lze práci rozšířit, nebo využít její výsledky.

V současné chvíli je jádro v procesu přijímání mezi oficiální jádra v projektu MiSTer, což byl jeden z hlavních cílů této práce. Je možné, že zveřejnění přinese nové podněty na opravy či vylepšení, a mám v plánu dále jádro aktivně spravovat a udržovat. Zároveň je zde možnost doplnit chybějící funkce. Z těch základních jde především o zprovoznění ukládání a načítání z magnetofonové pásky. Díky tomu, že deska DE-10 Nano poskytuje rozhraní pro vstupy a výstupy, šlo by vyvést rozšiřující konektor K2, obsahující uživatelem obsluhovatelné brány obvodů MHB8255A, přímo na fyzický konektor desky. S patřičnou redukcí by pak šlo připojit původní fyzické periferie. Stejně tak se nám dochovalo pár dobových rozšíření a úprav PMI-80, jako například deska poskytující zvukový výstup popsaná v Amatérském rádiu [40]. Vzniklo i několik moderních rozšíření, z nichž jsou částečně implementované větší paměťové obvody z rozšiřující desky popsané na webu Nostalcomp [8], ale část funkcí desky prozatím v jádru chybí.

Ukázka implementace v MiSTerovi bude moct být nadále využívaná v rámci prezentací spolku Herní historie, jehož jsem členem [41]. Stejně tak by mohla najít využití během akcí fakulty. Vzniká zde také snaha vystavit exponáty z historie fakulty v prostorách školy, které by mohla interaktivní ukázka vhodně doplnit. Plánuji i kontaktovat organizátory soutěže v programování pro PMI-80 a navrhnout využití jádra pro vývoj účastníků v dalších ročnících [6].

Během své práce jsem se setkal s několika lidmi, kteří vzpomínali na práci s PMI-80 v rámci výuky. Jejich zkušenosti bych rád shrnul do článku, který se bude zabývat spíše historickým zasazením počítače. Okolnosti jeho vzniku jsou dosud jen letmo popsané a i přes jeho rozšíření se nedochovalo příliš softwaru nebo svědectví o jeho dobovém využití. Své pátrání po těchto informacích hodlám dále rozšířit i na další dobovou techniku, například pokročilejší systém Tesla PMD 85, který ve svém původu přímo navazuje na PMI-80.

Závěr

Cílem práce bylo seznámit se s architekturou počítače Tesla PMI-80 a tu následně vhodně implementovat do emulátoru, dále pak popsat možnosti, jakým výsledný software využít. Architektura na bázi procesoru MHB8080A byla podrobně popsána, včetně dobových i moderních rozšíření. Popis vycházel především z historických materiálů a nadšenecké tvorby okolo počítače a nabízí přehled nejdůležitějších částí a jejich funkcí.

Dále byly prozkoumány současné možnosti emulace počítačových architektur včetně příkladů existujících projektů využívající dané principy. Z nich bylo na základě požadavků vybráno řešení za pomocí FPGA čipu na přípravku Terasic DE-10 Nano a za využití platformy poskytované projektem MiSTer. Byly představeny principy projektu a popsána struktura jádra, jež je v práci implementováno.

Po zjištění stavu již existujících implementací byla jedna z nich, modul PMI-80 z projektu FPGA SBC, zvolena pro integraci a byly vytvořeny moduly pro převod mezi rozhraním PMI-80 a rozhraním MiSTera. Jednalo se především o vstupní obvody klávesnice a výstupní obvody displeje. V rámci integrace bylo potřeba vybrat vhodná řešení problémů s emulací pro fyzické procesy dosvitu LED displeje a pro ukládání mezistavu vstupů a výstupů. Jejich zdůvodnění i popis jsou v práci uvedeny.

Během testování vznikla chyba způsobená přechodem z přípravku EP2C5 na DE-10 Nano – paměťové obvody vyhodnocovaly adresu o takt později, než procesor očekával. V projektu byla chyba ošetřena zrychlením frekvence hodin a dělením signálu povolujícímu běh procesoru. Paměťová struktura byla také integrována s uživatelským rozhraním MiSTera a dovoluje tak nahrávání softwaru do paměti během chodu počítače.

Stručně jsou předvedeny testovací postupy použité během práce a možnosti využití jádra ve výuce a muzejních expozicích. Jádro pro MiSTera nabízí interaktivní zážitek, který není zatížen mnohými problémy dnes využívaných interaktivních zastavení ve výstavách. Stejně tak práce s počítačem je velmi odlišná od té dnešní a během výuky by ji šlo vhodně zapojit jako ukázku počátků programovacích postupů.

Celkově byl naplněn hlavní cíl práce a vznikl vhodný emulátor ve formě jádra pro projekt MiSTer. Jádro splňuje veškerou základní funkcionalitu, umožňuje používat prostředí počítače PMI-80, i spouštět dostupné programy, které nevyužívají periferie mimo základní výbavu. Jádro je vhodně integrováno do prostředí MiSTera a nabízí standardní nastavení díky použití šablony se systémovými funkcemi.

Bibliografie

1. TESLATON. *PMI-80* [Obrázek]. 2018. Dostupné také z: https://commons.wikimedia.org/wiki/File:PMI-80_201802_03.jpg.
2. TESLA PIEŠŤANY. *Školský mikropočítač PMI 80: Užívateľská príručka*. Piešťany: Tesla Piešťany, 1982.
3. NOSTALCOMP. *Mikropočítač TEMS 80-03 A* [online]. 2010 [cit. 2023-05-10]. Dostupné z: https://web.archive.org/web/20190806000712/http://nostalcomp.cz/tems8003.php.
4. NOSTALCOMP. *Mikropočítače TEMS 48, 49 a 51* [online]. 2010 [cit. 2023-05-10]. Dostupné z: https://web.archive.org/web/20190805235919/http://nostalcomp.cz/tems4x.php.
5. TESLATON. *PMI-80 Box label* [Obrázek]. 2018. Dostupné také z: https://commons.wikimedia.org/wiki/File:PMI-80_201803_01_box_label.jpg.
6. VEJO. *PMI-80 Soutěž 2022* [online]. 2022 [cit. 2023-05-09]. Dostupné z: https://www.pmi-80.cz/.
7. STAREJ_MRAF; MISTICJOE. *OldComp.cz: Komunitní diskuzní fórum pro fanoušky historických počítačů* [online]. 2023 [cit. 2023-05-09]. Dostupné z: https://oldcomp.cz/.
8. NOSTALCOMP. *Úpravy a doplňky pro PMI - 80* [online]. 2010 [cit. 2023-05-10]. Dostupné z: https://web.archive.org/web/20190730070129/http://www.nostalcomp.cz/pmi80_upr.php.
9. TESLATON. *PMI-80* [Obrázek]. 2018. Dostupné také z: https://commons.wikimedia.org/wiki/File:PMI-80_201802_04.jpg.
10. SOFTWARE FREEDOM CONSERVANCY. *Wine* [Software]. 2023. Ver. 8.7. Dostupné také z: https://www.winehq.org/.
11. DOSBOX. *DOSBox* [Software]. 2019. 0.74-3. Dostupné také z: https://www.dosbox.com/.
12. BLUESHOGUN96. *Why is XBOX emulation premature?* [Online]. 2010 [cit. 2023-04-23]. Dostupné z: https://www.ngemu.com/threads/why-is-xbox-emulation-premature.132032/.
13. NEAR. *Accuracy takes power: one man's 3GHz quest to build a perfect SNES emulator* [online]. 2011 [cit. 2023-04-24]. Dostupné z: https://arstechnica.com/gaming/2011/08/accuracy-takes-power-one-mans-3ghz-quest-to-build-a-perfect-snes-emulator/.
14. JAMES, Greg; SILVERMAN, Barry. *The Visual 6502* [Software]. 2010. Dostupné také z: http://www.visual6502.org/JSSim/index.html.

15. SHONUMI. *Edge of Emulation: Game Boy Sewing Machines* [online]. 2020 [cit. 2023-05-08]. Dostupné z: https://shonumi.github.io/articles/art22.html.
16. EMBER, Gregory. *MAME* [Software]. 2023. Ver. 0.254. Dostupné také z: https://www.mamedev.org/.
17. DICE TEAM. *DICE* [Software]. 2014. Ver. 0.9. Dostupné také z: https://adamulation.blogspot.com/.
18. NOSTALCOMP. *PMI-80 M16: "postavitelná replika"počítače PMI-80* [online]. 2011 [cit. 2023-05-10]. Dostupné z: https://web.archive.org/web/20190731133225/http://www.nostalcomp.cz/pmi80m16.php.
19. NOSTALCOMP. *Mikropočítač PMI - 85* [online]. 2010 [cit. 2023-05-10]. Dostupné z: https://web.archive.org/web/20190727162506/http://nostalcomp.cz/pmi85.php.
20. NOSTALCOMP. *Mikropočítač PMI Z-80* [online]. 2010 [cit. 2023-05-10]. Dostupné z: https://web.archive.org/web/20190731150734/http://www.nostalcomp.cz/pmi_z80.php.
21. NOSTALCOMP. *emulátor počítače PMI-80 pro Windows* [online]. 2010 [cit. 2023-05-10]. Dostupné z: https://web.archive.org/web/20190727162747/http://nostalcomp.cz/pmi_emul.php.
22. FREE8BIT. *Emulation.Free8bit.net* [Software]. 2020. Dostupné také z: https://emulation.free8bit.net/.
23. MALÝ, Martin. *ASM80 PMI-80* [Software]. 2013. Ver. 2.9.4. Dostupné také z: https://www.asm80.com/pmi80.html.
24. MALÝ, Martin. *fpmi* [Software]. 2015. Dostupné také z: https://github.com/maly/fpmi.
25. BLASKO, Marek. *FPGA SBC* [Software]. 2017. Dostupné také z: https://zz-indigo.mavipet.sk/?page_id=1846.
26. FOTOOPA. *DE0-nano-soc-5623* [Obrázek]. 2020. Dostupné také z: https://www.flickr.com/photos/fotoopa_hs/49994585953.
27. MELNIKOV, Alexey. *MiSTer* [Software]. 2023. Dostupné také z: https://github.com/MiSTer-devel/Main_MiSTer.
28. EMBEDDEDSOFT CANADA. *MiniMig* [Software]. 2022. Ver. 1.96. Dostupné také z: https://www.minimig.ca/.
29. CROOKES, David. *The FPGA retro revolution* [online]. 2022 [cit. 2023-05-09]. Dostupné z: https://whynowgaming.com/the-fpga-retro-revolution/.
30. HARBAUM, Till. *MiST* [Software]. 2023. Dostupné také z: https://github.com/mist-devel/mist-board/wiki.
31. PETRM1. *PMD85_MiSTer* [Software]. 2022. Dostupné také z: https://github.com/MiSTer-devel/PMD85_MiSTer.
32. PETRM1. *OndraSPO186_MiSTer* [Software]. 2022. Dostupné také z: https://github.com/MiSTer-devel/OndraSPO186_MiSTer.
33. MISTER ADDONS. *Mister Addons* [online]. 2023 [cit. 2023-05-09]. Dostupné z: https://misteraddons.com/.
34. TÓTH, Štefan. PMI-80. *Amatérské rádio řada A*. 1984, roč. 33, č. 8, s. 297–301.
35. WALLNER, Daniel. *T80* [Software]. 2021. Ver. 351. Dostupné také z: https://github.com/MiSTer-devel/T80.
36. GEROČ, Marián. *Tenis-1* [Software]. 198X. Dostupné také z: https://web.archive.org/web/20190727165930/http://www.nostalcomp.cz/pmi_hry.php.
37. ŠUTERA, Jiří. *Cosmos* [Software]. 2013. Dostupné také z: https://oldcomp.cz/viewtopic.php?f=48&t=11192.

38. ŠVELCH, Jaroslav. Kdo se bojí her: Československý diskurz o počítačových hrách. In: *Jak obehrát železnou oponu: Počítačové hry a participativní kultura v normalizačním Československu*. Praha: Akropolis, 2023, kap. 4, s. 165. ISBN 978-80-7470-317-1.

39. NÁRODNÍ TECHNICKÉ MUZEUM. *Česká stopa v historii výpočetní techniky* [Výstava]. 2021. Dostupné také z: `https://www.ntm.cz/aktualita/2511-2020-95-2021-ceska-stopa-v-historii-vypocetni-techniky`.

40. BENEŠ, Jan; WOLF, Miloslav. Zvukový výstup mikropočítače PMI-80. *Amatérské rádio řada A*. 1987, roč. 36, č. 1, s. 22.

41. HERNÍ HISTORIE, Z.S. *Herní historie, z.s.* [Online]. 2023 [cit. 2023-05-09]. Dostupné z: `https://www.hernihistorie.cz/`.

Obsah přiloženého média

Projekt bude v budoucnu dostupný na následující adrese:
`https://github.com/MiSTer-devel/PMI80_MiSTer`

- `README.txt` ..... stručný popis obsahu média
- `pmi80.rbf` ..... soubor kompilovaného jádra
- `src/`
  - `pmi80_mister/` ..... zdrojové kódy implementace
  - `bakalarska_prace_text/` ..... zdrojová forma práce ve formátu LaTeX
- `text/` ..... text práce
  - `BP_Straka_Vojtech_2023.pdf` ..... text práce ve formátu PDF