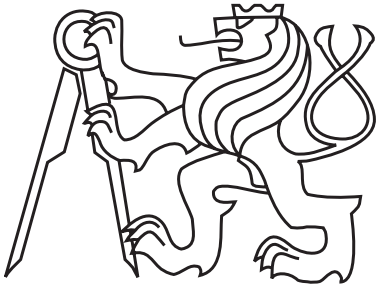
PLACEHOLDER

- Zadání podepsané děkanem a vedoucím katedry.

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Emulátor systému NEC PCEngine

Ondřej Baláž

Vedoucí práce: Ing. Tomáš Davidovič

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

8. ledna 2010

Poděkování

Na tomto místě bych chtěl poděkovat svým přátelům za jejich podporu, trpělivost a dávky sebevědomí, kterými jsem byl zahrnut v období kdy jsem tuto práci psal. Slova díku též patří mému zaměstnavateli, který vycházel vstříc mým časovým potřebám týkajícím se této práce.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.
Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 8. ledna 2010 .....

Abstract

This thesis describes emulation of video gaming system NEC PCEngine and implementation of basic functionality of the emulator, with focus on modularity and portability of source code. In first chapter we present system specification compiled from various sources, then existing emulators of this system are reviewed. Then we introduce basic ideas and algorithms used in our implementation. In last chapter we summarize our results and discuss options for further improvements, like adding support for other video gaming system emulators or user interfaces.

Abstrakt

Bakalářská práce se zabývá problematikou emulace videoherního systému NEC PCEngine a implementací základní funkčnosti emulátoru s ohledem na modularitu a přenositelnost kódu. Čtenář je v úvodní části seznámen se zpracovanou specifikací systému, dále jsou představena existující řešení v oblasti emulace tohoto systému a rozebrány postupy využívané při implementaci emulátorů. Následně jsou nastíněny základní myšlenky a postupy využité při implementaci modulárního přenositelného emulátoru. Závěrečná část pak pojednává o dosažených výsledcích a věnuje se možnostem rozšíření navrženého programu o další funkce, emulátory dalších videoherních systémů a uživatelská rozhraní.

Obsah

Seznam obrázků

Seznam tabulek

Typografické konvence

V následujícím textu se vyskytuje řada významných klíčových slov. Pro vyšší přehlednost jsou použity následující typografické konvence:

- identifikátory a názvy funkcí budou sázeny *kurzívou*
- adresy, offsety a datové konstanty budou uvedeny v hexadecimální soustavě sázeny `strojopisem` s použitím prefixu `$` (v souladu s assemblerem pro procesory WDC 65c02 a HuC6280)
- názvy registrů budou sázeny groteskem
- jména instrukcí budou sázena KAPITÁLKAMI
- názvy souborů budou sázeny `strojopisem`, v případě uvedení cesty, v UNIXové notaci vzhledem ke kořenovému adresáři projektu

Kapitola 1

Úvod

Neuvěřitelné tempo, kterým se v dnešní době vyvíjejí nové technologie v oblasti výpočetní techniky, nechává mnohem rychleji, než v jiných oblastech, zestárnout ty včerejší. Čekání s vývojem software na finální verzi hardware dnes téměř implikuje vývoj pro zastaralý hardware a v některých odvětvích, jako je třeba videoherní průmysl, může mít pro autora kritické následky. Emulace je postup, který pomáhá vývojářům software udržet krok s vývojáři hardware a umožňuje jim snadno vytvářet software pro hardware, který ještě ve finální podobě neexistuje. Kromě toho, že je díky emulaci paralelní vývoj software a hardware běžnou praxí, už i někteří výrobci zpřístupňují ve svých nejmodernějších výrobcích emulaci těch starších, které si jejich zákazníci oblíbili.

Emulace ale není jen nástroj, který nám pomáhá dostat se s vývojem rychle kupředu. Můžeme se díky ní ohlédnout i do minulosti, což je jistě zajímavé právě v oblasti zmíněného herního průmyslu. Za posledních 40 let vznikl nespočet herních titulů pro 7 generací videoherních konzolí, které si díky nízkým pořizovacím nákladům a jednoduchosti používání našly své neohrozitelné místo na trhu. Většina z těchto titulů dávno upadla do zapomění nahrazena novějšími a emulace videoherních konzolí je tak, vzhledem k nedostupnosti potřebného hardware, jedinný způsob jak si tyto tituly připomenout.

Jednou z herních konzolí, pro kterou některé z těchto titulů vznikly je NEC PCEngine. Jedná se o poměrně revoluční koznoli 3. generace, která použitím 16-ti bitové technologie pro obrazový výstup a velice kvalitním zvukovým výstupem předčila technické možnosti své konkurence. Snad vlivem špatného marketingu společnosti NEC se nikdy nestala tak populární jako ostatní konzole této generace, což je společně s tím, že společnost NEC šla cestou použití komponent navržených výhradně pro tento systém (např. procesoru), důvodem, proč existuje pouze omezené množství emulátorů této konzole. Právě zajímavá architektura tohoto systému a poměrně malý počet dostupných emulátorů jsou hlavním důvodem, proč se tento systém stal předmětem této práce.

Cílem této práce je prostudování architektury videoherního systému NEC PCEngine, její shrnutí do ucelené technické specifikace (pravděpodobně jedinné v českém jazyce) a následná analýza, návrh a implementace emulátoru tohoto systému na jejím základě. Výsledný emulátor by měl, na rozdíl od většiny existujících implementací, umožňovat snadné rozšíření o podporu dalších videoherních systémů a být jednoduše přenositelný i na jiné platformy, než jsou operační systémy osobních počítačů.

Kapitola 2

NEC PCEngine

Znalost principů fungování emulovaného systému a systému samotného je jedním z klíčových předpokladů pro úspěšnou implementaci emulátoru. U videoherních systémů je získání technické specifikace většinou obtížné, protože se jedná o uzavřené produkty a jejich výrobci mají komerční a jiné důvody, proč oficiální dokumentaci nezveřejňovat a to dokonce i v době, kdy pro ně konkrétní systém už nemá žádný komerční význam.

Autoři emulátorů jsou tak odkázání na metody reverzního inženýrství[1], přičemž vzniká často používaná neoficiální dokumentace, která, i přes svůj přínos, může být neúplná nebo nepřesná. V případě systému NEC PCEngine tento proces navíc komplikuje malá popularita a tedy i malý počet zainteresovaných vývojářů a fakt, že jeho výrobce šel cestou použití zcela specifického hardware (ostatní systémy téže generace byly téměř bez výhrady postaveny na dobře zdokumentovaných procesorech MOS 6502 či Zilog Z80).

Sekce 2.1, která otevírá tuto kapitolu, stručně představuje videoherní systém NEC PCEngine a stručně pojednává o jeho historii.

Protože pravděpodobně neexistuje žádný dokument, který by shrnoval dostupné technické specifikace základní verze systému NEC PCEngine na jednom místě, ale jen dílčí dokumenty specializující se na technickou specifikaci určité částí systému NEC PCEngine, je v sekci 2.2 předložena co možná nejkompletnější[2] specifikace tohoto systému, která je založena nejen na zmiňovaných dokumentech [10, 8, 12, 4, 34, 7, 15], ale také na studiu zdrojových kódů existujících emulátorů [28, 29] a konečně i vlastních zkušenostech nabytých při implementaci emulátoru.

Poslední sekce 2.3 této kapitoly pak stručně pojednává o software dostupném pro systém NEC PCEngine.

2.1 Historie

NEC PCEngine je hybridní videoherní konzole vyvinutá japonskými společnostmi HudsonSoft a NEC v roce 1987. Hybridní je nazývána proto, že ačkoliv hlavní procesor je 8-mi bi-

[1] Snaha o odkrytí principu na kterém funguje zařízení, např. za účelem jeho chování napodobit.

[2] Z hlediska specifik systému NEC PCEngine, pro přehlednost není uvedena např. kompletní specifikace procesoru WDC 65c02 z nějž vychází procesor HuC6280 použitý v systému NEC PCEngine, ale jen odlišnosti použitého procesoru HuC6280.

tový, jako u většiny herních konzolí 3. generace (Nintendo Famicom, Sega Master System, Atari 7800), grafický systém je plně 16-ti bitový. Vzhledem k tomu, že NEC PCEngine bylo uvedeno do prodeje v roce 1987, NEC předběhl své hlavní rivaly v použití 16-ti bitové technologie a z ní vyplývajících výhod o celou jednu generaci herních konzolí. Pokročilý 6-ti kanálový zvukový subsystém dovoloval kromě přehrávání samplů i jednoduchou zvukovou syntézu a modulaci zvuku. Je jen s podivem, že tak pokročilý systém, jakým bezesporu ve své době NEC PCEngine byl, zdaleka nedosáhl světové obliby na alespoň stejné úrovni jako ostatní videoherní konzole téže generace. [17, 33]

Obrázek 2.1: Systém NEC PCEngine, základní varianta

Systém NEC PCEngine byl firmou NEC postupně inovován. Kromě rozšíření paměti a přidání vstupního portu pro další herní ovladač(e) se v následujících verzích dočkala i rozšíření (jako první videoherní konzole na světě) o CD-ROM jednotku a další grafický koprocesor. Mírně modifikovaný systém byl od roku 1989 v malém nákladu exportován z Japonska do Severní Ameriky a Evropy pod názvem NEC TurboGrafx 16, kde se ale vůbec neprosadil vlivem špatného marketingu a nedostatku vývojářů her. *Pozn.: popsaná rozšíření ani modifikace nejsou v práci uvažovány.* [17, 9]

S příchodem plně 16-ti bitových herních konzolí 4. generace (Sega Genesis, Super Nintendo Entertainment System) v roce 1991 přišel systém NEC PCEngine o své místo na trhu a po neúspěších s dalším systémem (PC FX32) opouští NEC videoherní průmysl. [33]

Zajímavostí pro doplnění je, že svými fyzickými rozměry 14cm x 14cm x 3.8cm se NEC PCEngine zapsalo do Guinnessovy knihy rekordů jako nejmenší nepřenosná videoherní konzole. [1]

2.2 Hardware

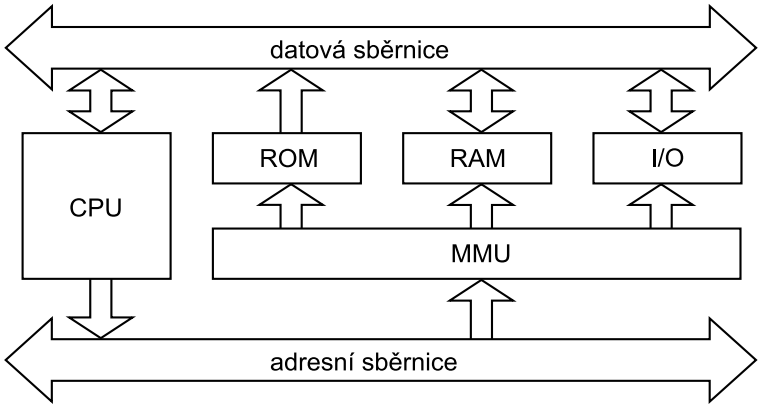
Základní verze NEC PCEngine je vybavena 8 KB operační paměti (RAM) a 64 KB videopaměti (VRAM). Hlavní procesor HuC6280 řídí zpracování programu uloženého v paměti určené pouze pro čtení (ROM) na výměnné čipové kartě HuCard[3]. Dále zajišťuje ozvučení programu pomocí programovatelného generátoru zvuků (Programmable Sound Generator, PSG) o 6ti kanálech a zpracování vstupu z portu herního ovladače.

[3]Čipové karty HuCard měly formát kreditní karty, čímž se zásadně lišily od podstatně větších zásuvných modulů rivalských systémů od společností Nintendo a Sega.

O obrazový výstup se stará dvojice 16-ti bitových grafických pomocných procesorů, konkrétně kontrolér pro zobrazování (Video Display Controller, VDC) HuC6270 a enkodér barev (Video Color Encoder, VCE) HuC6260. Ty společně zajišťují zobrazení v rozlišeních až 512x240 pixelů v 9-ti bitové barevné hloubce (512 barev celkem, na obrazovce zobrazitelných maximálně 482 barev) pomocí RF (anténního) výstupu na televizní set ve standardu NTSC. [33]

2.2.1 CPU

Hlavní procesor celého systému (Central Processing Unit, CPU) nese označení HuC6280 a vychází z mikroprocesoru WDC 65c02 společnosti Western Digital, jehož specifikaci [15] rozšiřuje o řadu instrukcí a adresních módů a přidává některá, pro účely videoherního systému, specifická periferní zařízení. Jedná se o 8-mi bitový procesor schopný operovat na frekvencích 3.58 MHz a 7.16 MHz (lze libovolně měnit za běhu programu vykonáním dvou speciálních instrukcí CSL a CSH). Architektura systému je znázorněna na obr. 2.2.



Obrázek 2.2: Architektura systému NEC PCEngine

Z hlediska endianity je CPU HuC6280 „little-endian“[4]. Tento fakt se pochopitelně týká i obou grafických pomocných procesorů VDC HuC6270 a VCE HuC6260. V následujícím textu se s touto skutečností mlčky počítá.

Jak již bylo zmíněno, oproti WDC 65c02 disponuje CPU HuC6280 řadou periferních zařízení: jednotkou obsluhy přerušení, 8-mi bitovým paralelním I/O portem (pro herní ovladač), jednotkou správy paměti (mapovačem), časovačem a programovatelným generátorem zvuku (PSG).

2.2.1.1 Registry

A Střadač (Accumulator)
: Bezpochyby nejdůležitější registr CPU HuC6280. Slouží jako zdrojový, cílový nebo oba operandy pro většinu instrukcí. Je to jedinný registr, který lze použít s řadou insturkcí pro provádění matematických výpočtů. Délka registru A je 8 bitů.

[4]Při práci s vícebajtovými položkami je nejméně významný bajt je v paměti uložen na nejnižší adrese.

X Indexový registr X (Index X)
: Jedná se o registr obecného použití často využívaný při některých adresních módech s nepřímou adresou. Je též používán jako vnitřní čítač pro cykly. Délka registru X je 8 bitů.

Y Indexový registr Y (Index Y)
: Jedná se opět o registr obecného použití s podobným využitím a vlastnostmi jako má registr X. Délka registru Y je 8 bitů.

PC Čítač programu (Program counter)
: Obsahuje adresu instrukce, která se má vykonat. Ve chvíli kdy dojde k vykonání instrukce je hodnota čítače automaticky zvýšena/změněna[5]. Délka registru PC je 16 bitů.

S Ukazatel na zásobník (Stack pointer)
: Obsahuje adresu prvního volného místa v zásobníku[6]. Délka registru S je 8 bitů.

P Příznakový registr (Program flags)
: Tento velice významný registr obsahuje bitové pole s příznaky indikujícími aktuální stav procesoru po poslední provedené operaci. Některé z těchto příznaků mohou být i nastaveny z programu. Jednotlivé příznaky jsou uvedeny v tabulce 2.1. Význam jednotlivých příznaků následuje.

Bit	7	6	5	4	3	2	1	0
Příznak	N	O	T	B	D	I	Z	C

Tabulka 2.1: Příznaky indikované registrem P CPU HuC6280

N Záporný výsledek (Negative)
: Stav „1“ tohoto příznaku indikuje, že poslední operace nastavila nejvyšší bit výsledku.

O Přetečení (Overflow)
: Stav „1“ tohoto příznaku indikuje přetečení (nesprávný přenos nejvyššího bitu). Používá se hlavně při aritmetických operacích, kde záleží na znaménku.

T Set režim (Set mode)
: Speciální akumulátorový režim, souvisí s instrukcí SET. Více o tomto režimu lze nalézt např. v [8].

B Přerušení (Break)
: Stav „0“ tohoto příznaku indikuje hardwarové přerušení. Slouží k rozlišení hardwarových přerušení od použití instrukce BRK[7].

D Decimální mód (Decimal mode)
: Stav „1“ tohoto příznaku indikuje, že některé matematické instrukce počítají v BCD kódu namísto binárního. Tento příznak lze přímo nastavit.

[5] Ke změnám, které nemusí být inkrementací dochází např. při použití instrukcí skoku nebo volání podprogramu.

[6] Místo v paměti určené k dočasným odložení hodnot programem.

[7] Instrukce BRK způsobí softwarové přerušení.

I Zákaz přerušení (Interrupt disable)
Stav „1“ tohoto příznaku indikuje zákaz všech běžných přerušení (s vyjímkou Non-maskable interrupt (NMI). Tento příznak lze přímo nastavit.

Z Nula (Zero)
Nastavení tohoto příznaku indikuje, že výsledek poslední operace byl „nula“.

C Přenos (Carry)
Stav „1“ tohoto příznaku indikuje přenos bitu z poslední vykonané operace.

Délka registru P je 8 bitů.

MPR0-7 Registr paměťové stránky (Memory page register)
Každý z 8 MPR registrů obsahuje index paměťové stránky, ze kterého se počítá výsledná 21-bitová adresa. Délka registru MPR0-7 je 8x8 bitů.

CS Rychlost hodin (Clock speed)
Skrytý jednobitový registr ovládaný pomocí dvou instrukcí (CSL a CSH). Pokud je hodnota tohoto registru „1“, frekvence procesoru je 7.16 MHz. V případě, že je hodnota registru „0“, pak je frekvence procesoru poloviční (tedy 3.58 Mhz). Délka registru CS je 1 bit. [10]

2.2.1.2 Paměť

Hlavní změnou oproti WDC 65c02 je způsob práce s pamětí. CPU HuC6280 je schopný adresovat 64 KB paměti logickou adresou z fyzického paměťového prostoru 2 MB. Adresa je oproti původním 16-ti bitům dlouhá 21 bitů a pro přístup k paměti se využívá 8 mapovacích registrů MPR0-7, z nichž každý obsahuje index 8 KB velké stránky z fyzického paměťového prostoru. Specifikace příslušného MPR registru je prováděna pomocí tří horních bitů logické adresy. Práce s těmito registry probíha pomocí speciálních instrukcí: TAM a TMA. Rozdělení logické paměti do stránek je znázorněno v tabulce 2.2.

Stránka	Logický adresní prostor
0	$0000 - $1FFF
1	$2000 - $3FFF
2	$4000 - $5FFF
3	$6000 - $7FFF
4	$8000 - $9FFF
5	$A000 - $BFFF
6	$C000 - $DFFF
7	$E000 - $FFFF

Tabulka 2.2: Rozdělení logické paměti CPU HuC6280 do stránek

Pro dokončení představy o způsobu adresace paměti je v tabulce 2.3 znázorněno rozdělení fyzického adresního prostoru (2 MB) na 256 8 KB velkých bank.

Výpočet 21-bitové adresy probíhá tak, že pomocí horních 3 bitů 16-ti bitové logické adresy je zvolen jeden z MPR registrů. Z něj je načten 8-mi bitový index banky, který je

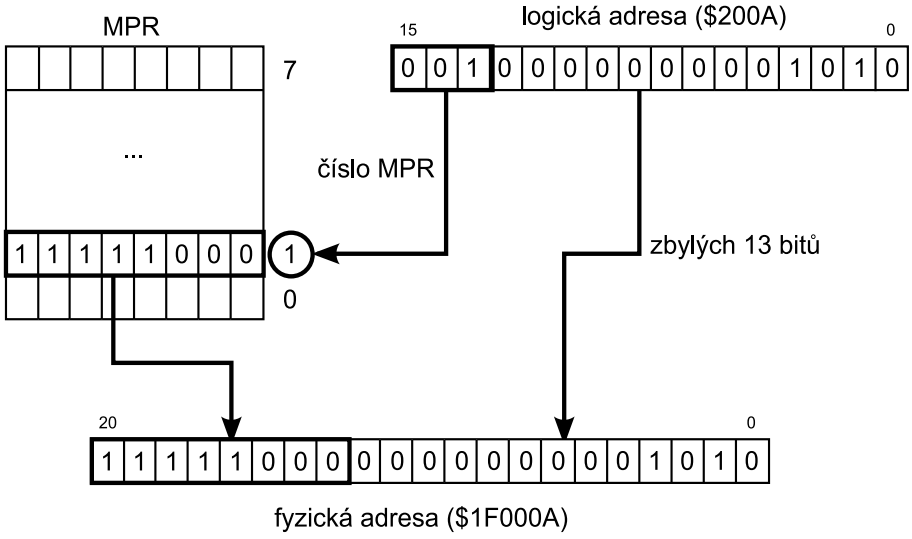
Banky	Význam
$00 - $7F	paměť ROM výměnného modulu HuCard (viz. 2.2.5)
$80 - $F7	*nevyužito*
$F8 - $FB	RAM (pouze 8 KB 4x zrcadlených)
$FC - $FE	*nevyužito*
$FF	vstupně-výstupní banka (tzv. „I/O page“, viz. 2.2.1.6)

Tabulka 2.3: Rozdělení fyzické paměti CPU HuC6280 do bank

posléze posunut o 13 bitů vlevo a sečten se zbývajícími 13ti bity logické adresy (offsetem ve stránce). Výsledkem je fyzická 21-bitová adresa [10]. Postup výpočtu fyzické adresy je znázorněn na obr. 2.3.

Ve výchozím stavu je pomocí MPR registrů banka $FF mapována jako stránka 0, banka $F8 jako stránka 1 a konečně banka $00 jako stránka 7. **Následující text s tímto mapováním počítá při uvádění logických adres.**

Jen pro kompletnost a lepší orientaci v dalším textu (zejména pak v sekci 2.2.1.3) uvádíme fakt, že veškeré operace se zásobníkem a stránkou nula[8] vždy pracují v rozsahu logických adres $2000 - $2FFF, což odpovídá použití mapovacího registru MPR1 a fyzickým adresám $1F0000 pro počátek stránky nula a $1F0100 pro dno zásobníku.



Obrázek 2.3: Výpočet fyzické adresy

[8]První stránka adresního prostoru.

2.2.1.3 Instrukční sada

CPU HuC6280 rozšiřuje instrukční sadu WDC 65c02 (která je podrobně popsána, včetně možnosti adresace operandů a vlivu na příznakový registr pro každou instrukci, např. v [15]) o několik specifických instrukcí a způsobů adresace operandů.

Jedná se o následující výčet instrukcí. Počty cyklů, možnosti adresace operandů a vliv na příznakový registr pro jednotlivé instrukce lze nalézt v [10].

BSR (Branch To Subroutine)
: Uloží absolutní adresu následující instrukce sníženou o jedna na zásobník a skočí do subrutiny.

CLA (Clear Accumulator)
: Nastaví hodnotu střadače na „0“.

CLX (Clear Index X)
: Nastaví hodnotu indexového registru X na „0“.

CLY (Clear Index Y)
: Nastaví hodnotu indexového registru Y na „0“.

CSH (Clock Select High)
: Slouží ke změně obsahu skrytého registru CS na „1“. Procesor se přepne na kmitočet 7.16 MHz.

CSL (Clock Select Low)
: Slouží ke změně obsahu skrytého registru CS na „0“. Procesor se přepne na kmitočet 3.58 MHz.

SAX (Swap Accumulator And Index X)
: Prohodí obsah střadače a indexového registru X.

SAY (Swap Accumulator And Index Y)
: Prohodí obsah střadače a indexového registru Y.

SET (Set Decimal Memory Operation Flag)
: Přepne procesor do akumulátorového módu (decimálního) na jeden instrukční cyklus (nastaví příznak „T“ na „1“).

SXY (Swap Index X And Index Y)
: Prohodí obsah indexových registrů X a Y.

ST0 (Store At VDC Address 1)
: Zapíše data pro VDC HuC6270 do hardwarového registru 0 (fyzická adr. `$1FE000`).

ST1 (Store At VDC Address 2)
: Zapíše data pro VDC HuC6270 do hardwarového registru 1 (fyzická adr. `$1FE002`).

ST2 (Store At VDC Address 3)
: Zapíše data pro VDC HuC6270 do hardwarového registru 2 (fyzická adr. `$1FE003`).

TAI (Transfer 'from' Alternate 'to' Increment)
Přenese blok dat z adresy *from* na adresu *to* tak, že v každém cyklu zamění adresu *from* s obsahem paměti na této adrese a zvýší adresu *to* o „1“.

TAM (Transfer Accumulator To MPR)
Přenese obsah střadače do mapovacího registru.

TDD (Transfer 'from' Decrement 'to' Decrement)
Přenese blok dat z adresy *from* na adresu *to* tak, že v každém cyklu obě adresy sníží o „1“.

TIA (Transfer 'from' Increment 'to' Alternate)
Přenese blok dat z adresy *from* na adresu *to* tak, že v každém cyklu sníží adresu *from* o „1“ a zamění adresu *to* s obsahem paměti na této adrese.

TII (Transfer 'from' Increment 'to' Increment)
Přenese blok dat z adresy *from* na adresu *to* tak, že v každém cyklu obě adresy zvýší o „1“.

TIN (Transfer 'from' Increment 'to' Does Nothing)
Přenese blok dat z adresy *from* na adresu *to* tak, že v každém cyklu zvýší adresu *from* o „1“.

TMA (Transfer MPR To Accumulator)
Přenese obsah mapovacího registru do střadače.

TST (Test Bits)
Otestuje bity mezi dvěma operandy kde jeden je přímý a jeden z paměti. Nastaví příznakový registr tak, že indikuje stav horních dvou bitů hodnoty načtené z paměti pomocí příznaků „N“ a „O“ a shodu alespoň v jednom bitu pomocí příznaku „Z“.

2.2.1.4 Režimy adresace operandů

Režimy adresace operandů instrukcí CPU HuC6280 jsou uvedeny v tabulce 2.4. Většina instrukcí je použitelná jen s několika ze všech vyjmenovaných způsobů adresace operandů, které v jejím případě dávají smysl (např. jedinná instrukce, která může použít nepřímou adresaci nebo nepřímou adresaci s indexem je JMP). Způsoby adresace operandů specifické pro CPU HuC6280 jsou uvedeny ve spodní části tabulky 2.4 oddělené čarou a popsány v následujícím textu. Podrobnosti o jednotlivých způsobech adresace operandů lze nalézt v [15, 10].

Adresa	**Syntaxe**
Implicitní (Implicite)	-
Střadačová (Accumulator)	`INS A`
Přímá (Immediate)	`INS #$10`
Stránka nula (Zero page)	`INS $10`
Stránka nula s indexem (Zero page, X/Y)	`INS $10,X` nebo `INS $10,Y`
Relativní (Relative)	`INS *+1`
Absolutní (Absolute)	`INS $1000`
Absolutní s indexem (Absolute, X/Y)	`INS $1000,X` nebo `INS $1000,Y`
Nepřímá (Indirect)	`INS ($1000)`
Nepřímá stránkou nula (Indirect zero page)	`INS ($10)`
Nepřímá s indexem (Indirect, X/Y)	`INS ($10, X)` nebo `INS ($10),Y`
Absolutní nepřímá s indexem (Absolute Indirect, X)	`INS (TBL, X)`
Přenosová (Transfer)	`INS $1000, $2000, #$3000`
Přímá a stránka nula (Immediate and Zero page)	`INS #$10, $20`
Přímá a stránka nula s indexem X (Immediate and Zero page, X)	`INS #$10, $20, X`
Přímá a absolutní (Immediate and Absolute)	`INS #$10, $1000`
Přímá a absolutní s indexem X (Immediate and Absolute, X)	`INS #$10, $1000, X`

Pozn.: Instrukce INS je smyšlená stejně jako všechny použité adresy a návěstí. Přehled má sloužit pouze pro demonstraci zápisu v assembleru pro CPU HuC6280.

Tabulka 2.4: Možnosti adresace operandů CPU HuC6280

Režimy specifické pro CPU HuC6280 jsou obvykle spjaty s některou konkrétní instrukcí nebo jejich skupinou. Tyto instrukce vyžadují adresaci operandů právě jedním konkrétním způsobem. Jedná se o režimy popsané v následujícím výčtu (včetně uvedení instrukcí, se kterými mohou být použity):

Přenosová adresa (Transfer)
: Používá se pouze ve spojení s instrukcemi blokových přenosů (TAI, TDD, TIA, TII a TIN), které očekávají tři operandy - absolutní adresu zdroje *from*, absolutní adresu cíle *to* a délku bloku.

Přímá adresa a stránka nula (Immediate and Zero page)
Tento způsob adresace se využívá jen s instrukcí TST. Operandem je přímá hodnota a offset ve stránce nula.

Přímá adresa a stránka nula s indexem X (Immediate and Zero page, X)
Tento způsob adresace se opět využívá jen s instrukcí TST. Operandem je přímá hodnota a offset ve stránce nula s indexem X (adresa ve stránce nula zvýšenou o hodnotu indexového registru X).

Přímá adresa a absolutní adresa (Immediate and Absolute)
Tento způsob adresace se opět využívá jen s instrukcí TST. Operandem je přímá hodnota a absolutní adresa.

Přímá adresa a absolutní adresa s indexem X (Immediate and Absolute, X)
Tento způsob adresace se opět využívá jen s instrukcí TST. Operandem je přímá hodnota a absolutní adresa s indexem X (absolutní adresa zvýšená o hodnotu indexového registru X).

2.2.1.5 Přerušení

Přerušení procesoru je stav kdy procesor vyskočí ze sekvenčního zpracování programu na předem určené místo (toto místo určuje speciální odskoková adresa, tzv. „vektor přerušení“) a zahájí zpracování přerušení obslužnou rutinou. Seznam dostupných přerušení a jejich vektorů je uveden v tabulce 2.5.

Přerušení	Vektor
RESET	`$FFFE`
NMI	`$FFFC`
TIMER	`$FFFA`
IRQ1	`$FFF8`
IRQ2	`$FFF6`

Tabulka 2.5: Přerušení CPU HuC6280 a jejich vektory

RESET - Přerušení vyvolané při každém spuštění procesoru.

NMI - Nemaskovatelné přerušení. Toto přerušení nastává i v případě že je pomocí příznakového registru P zakázáno zpracování přerušení (příznak „I“). Procesor HuC6280 přerušení umí obsloužit a má pro něj vyhrazen vektor, ale dle [8] neexistuje způsob (pin přerušení není fyzicky s ničím propojen), jak by bylo možné ho vyvolat.

TIMER - Přerušení vyvolané interním časovačem procesoru HuC6280. K přerušení dochází v okamžiku kdy čítač časovače přeteče na maximální hodnotu čítáním z hodnoty 0.

IRQ1 - Přerušení vyvolané pomocným grafickým procesorem VDC (HuC6270).

IRQ2 - Přerušení vyvolané instrukcí BRK. [8] uvádí, že pin přerušení IRQ2 je vyveden do portu pro čipové karty HuCard a přerušení je sdíleno s některými hardwarovými rozšířeními konzole NEC PCEngine. Zda-li bylo přerušení vyvoláno softwarově instrukcí BRK, nebo hardwarově lze zjistit pomocí příznaku „B“ příznakového registru P.

Adresy určené ke čtení vstupně-výstupní vyrovnávací paměti při obsluze přerušení jsou `$1400` a `$1401`. Přerušení mohou být, s vyjímkou NMI, zakázána (maskována) pomocí příznaku „I“ příznakového registru P nebo pomocí zápisu na spodní tři bity na adrese `$1402`.

Všechna přerušení musí být po zpracování potvrzena libovolným zápisem na adresu `$1403`. Pokud se tak nestane, bude přerušení (v případě že není zakázano) nastávat po provedení každé instrukce.

2.2.1.6 Periferní zařízení

Všechna zařízení připojená k řídící sběrnici procesoru HuC6280 (jak interní periférie typu paralelní I/O port či časovač, tak externí podpůrné grafické procesory VDC HuC6270 a VCE HuC6260) s procesorem komunikují pomocí vstupně-výstupní banky (tzv. „I/O page“) `$FF`, do které mapují regiony své paměti (registry atd.). Mapa této banky je uvedena v tabulce 2.6

Logická adresa	Význam
`$0000 - $03FF`	VDC HuC6270 (viz. 2.2.2)
`$0400 - $07FF`	VCE HuC6260 (viz. 2.2.3)
`$0800 - $0BFF`	PSG (viz. 2.2.4)
`$0C00 - $0FFF`	časovač
`$1000 - $13FF`	paralelní I/O port
`$1400 - $17FF`	jednotka obsluhy přerušení (viz. 2.2.1.5)
`$1800 - $1BFF`	*nevyužito*
`$1C00 - $1FFF`	*nevyužito*

Tabulka 2.6: Mapa vstupně-výstupní banky $FF CPU HuC6280

Časovač - Jedná se o interní časovač implementovaný pomocí 7-mi bitového zachytávacího (latch) registru. Časovač je řízen stejným zdrojem hodinového signálu jako celý procesor v „rychlém“ režimu (7.16 MHz) a NENÍ ovlivněn použitím instrukcí CSL a CSH pro změnu kmitočtu procesoru. Čítač časovače dekrementuje svojí hodnotu každých 1024 cyklů.

Časovač je ovládán pomocí dvojice logických adres kde `$0C00` slouží k nastavení výchozí hodnoty zachytávacího registru časovače (nižších 6 bitů 0-127) a `$0C01` slouží k aktivaci časovače (nejnižší bit). Přerušení TIMER je vyvoláno při přetečení hodnoty zachytávacího registru z hodnoty 0 do hodnoty 127.

Paralelní I/O port - standardní 8-mi bitový vstupně-výstupní port pro herní ovladač. Informace o stavu herního ovladače jsou dostupná pomocí logické adresy `$1000` kde čtením nižších 4 bitů získáme data o stavu herního ovladače podle tabulky 2.7.

V případě zápisu jsou zajímavé nejnižší dva bity. Na nejnižší se zapisuje stav „SEL“. Hodnota „0“ vybírá čtení stavu čtveřice akčních tlačítek, hodnota „1“ pak čtení stavu čtveřice směrovek. Na druhý nejnižší bit se zapisuje stav „CLR“, který v případě nastavení na 1 způsobí ignoraci ovladače.

Bit	Stav SEL	Tlačítko
3	0	tlačítko „Start“
2	0	tlačítko „Select“
1	0	tlačítko „I.“
0	0	tlačítko „II.“
3	1	směrový kříž, doleva
2	1	směrový kříž, dolů
1	1	směrový kříž, doprava
0	1	směrový kříž, nahoru

Tabulka 2.7: Kódování stavu herního ovladače na paralelním I/O portu CPU HuC6280

2.2.2 VDC

Video Display Controller (VDC), neboli kontrolér pro zobrazení nese označení HuC6270. Je to plně 16-ti bitový procesor s 20-ti 16-ti bitovými registry a schopností adresovat až 128 KB video paměti. K VDC HuC6270 se přistupuje prostřednictvím tří speciálních instrukcí ST0, ST1 a ST2, a vstupně-výstupní banky `$FF`. VDC HuC6270 zajišťuje generování výsledného obrazu z dlaždic pozadí a sprajtů[9]. Pro získání informace o barvách a jejich uspořádání do barevných palet využívá druhého z dvojice grafických pomocných procesorů, Video Color Encoder (VCE) - enkodér barev HuC6260 (viz. 2.2.3).

Vzhledem k povaze herního systému NEC PCEngine byl zobrazovací systém navržen pro připojení k televiznímu setu pomocí RF (anténního) výstupu. Veškerý zobrazovací mechanismus je tedy přizpůsoben zobrazování na televizoru a vykreslování na obrazovku tedy probíhá po řádcích[10] (scanlines).

Řádky na stínítku televizní obrazovky jsou tvořeny dopadáním elektronů vyslaných katodou obrazovky na mřížku a předáváním energie luminoforu, který se v místě dopadu rozzáří a vytvoří světelný bod. Paprsek elektronů vysílaný katodou obrazovky pochopitelně vykresluje jednotlivé řádky s prodlevou pro navrácení na začátek dalšího řádku (tato prodleva se nazývá *horizontal blanking*). Stejně tak nastává prodlení při návratu paprsku z pravého dolního rohu obrazovky do levého horního - tedy po vykreslení celého jednoho snímku (tato prodleva se nazývá *vertical blanking*). [14]

Dění na obrazovce je s děním uvnitř logiky programu synchronizováno právě pomocí tohoto údaje. Televizor samozřejmě nijak systému nepředává informaci o každém vykreslení

[9] Grafických symbolů poskládaných zpravidla do popředí větší scény. Více o „sprites“ viz. [32].

[10] V případě současných digitálních televizorů s LCD/TFT displejem samozřejmě probíhá vykreslování jinak, než pomocí katodového paprsku na CRT televizorech. Nicméně i tyto televizory jsou kompatibilní se standardy PAL a NTSC a tudíž po připojení systému k RF (anténního) vstupu televizoru bude vše fungovat jak má z důvodu zpětné kompatibility s analogovým signálem (pokud touto funkčností televizor disponuje). Následující dva odstavce jsou zde uvedeny záměrně pro osvětlení pojmu vertikální synchronizace.

jednoho celého snímku, ale vzhledem k tomu, že je zobrazení na televizoru standardizováno do norem PAL a NTSC, kde je pevně dán počet snímku zobrazených za sekundu a počet řádků tvořících jeden snímek, je možné přesně vypočítat čas synchronizace. Tato synchronizace je zpravidla prováděna vyvoláním přerušení (a nazývá se *vertikální synchronizace*, neboli *vertical synchronization*).

NEC PCEngine pracuje v režimu NTSC, který je definován 60-ti zobrazenými proloženými (vykreslují se jen liché, nebo jen sudé řádky) snímky za sekundu při 263 řádcích na jeden snímek. [14]

2.2.2.1 Parametry zobrazení

VDC HuC6270 dokáže pracovat ve variabilním rozlišení obrazu. To se může měnit za běhu programu modifikací registrů VDC HuC6270. Maximální horizontální rozlišení je 512 pixelů (lze snížit po násobcích 8 pixelů), maximální vertikální rozlišení je 240 pixelů (opět lze snížit po násobcích 8 pixelů). Většina herních programů využívá rozlišení 256x224 pixelů. Ačkoliv je barevná hloubka zobrazení 9 bitů (512 barev rozděleno na dvě skupiny separátních palet pro dlaždice pozadí a sprajty), dokáže systém NEC PCEngine zobrazit najednou maximálně 482 barev. [33, 12]

Jak již bylo naznačeno, VDC HuC6270 generuje výsledný obraz ze dvou skupin grafických elementů. Každá tato skupina představuje jednu z následujících rovin:

dlaždic pozadí velkých 8x8 pixelů. Každá z těchto dlaždic může používat až 16 barev z jedné z 16-ti palet určených pro dlaždice pozadí. Jedna z barev je stejná napříč všemi paletami.

sprajtů ve velikostech od 16x16 pixelů až 32x64 pixelů. Sprajtů může být zároveň zobrazeno maximálně 64 a každý z nich může používat až 15 barev z jedné z 16-ti palet určených pro sprajty. Jedna barva ve všech paletách je vždy transparentní.

Vykreslování roviny sprajtů i roviny dlaždic pozadí může být potlačeno prostřednictvím zápisu do jednoho z registrů VDC HuC6270 (viz. 2.2.2.4).

2.2.2.2 Organizace videopaměti

Protože je videopaměť (VRAM) systému NEC PCEngine velká jen 64 KB, nemůže být výsledný obraz uložen v podobě bitové mapy jako je tomu např. u PC. Nebylo by totiž možné dosáhnout tak vysokých rozlišení při celkovém počtu barev.

Při nejvyšším rozlišení 512x240 pixelů a využití 8-mi bitové barevné hloubky by uložení výsledné bitové mapy do paměti znamenalo: $512*240*8b = 983040b = 120\ KB$, což je zhruba dvojnásobek paměti kterou máme k dispozici. Namísto toho jsou obrazová data rozdělena do dlaždic pozadí a sprajtů, ze kterých je obraz výsledné scény poskládán (u dlaždic pozadí je pevně dána velikost 8x8 pixelů, u sprajtů je velikost odkrokována po 16-ti pixelech v rozsahu 16x16 pixelů až 32x64 pixelů).

Dlaždice pozadí a sprajty jsou definovány pomocí dvou atributových tabulek: Background Attribute Table (*BAT*) a Sprite Attribute Table (*SAT*), které vycházejí z rozdělení výsledného obrazu na rovinu dlaždic pozadí a rovinu sprajtů (viz. výše).

Podstatným činitelem z hlediska snížení nároků na pamět je způsob práce s barvami. Konkrétně rozdělením barev do dvou skupin po 16-ti 16-ti barevných paletách, kde index každé barvy lze zákodovat do 4 bitů (přijatelnou nevýhodou je možnost použít jen 16 barev v rámci jedné dlaždice pozadí nebo sprajtu). Obrazová data jsou v paměti uložena *planárním* způsobem (viz. dále).

BAT - tabulka atributů dlaždic pozadí

Tabulka *BAT* je uložena na začátku videopaměti (adresa `$00`) a její velikost je závislá na rozlišení (tj. na počtu zobrazených dlaždic pozadí). Pokud je tabulka kratší, než je velikost obrazovky, dojde při zpracování k přetečení na začátek a opakování tabulky (čehož lze s výhodou využít). Délka tabulky *BAT* je nastavována pomocí jednoho z registrů VDC HuC6270 (viz. 2.2.2.4). Nejmenší možná velikost tabulky *BAT* je 32x32 dlaždic, největší pak 128x64 dlaždic. Tabulky generující rovinu dlaždic pozadí větší, než je fyzické rozlišení televizní obrazovky jsou využívány pro skrolování pozadí (které výsledné scéně dodává hloubku).

Každým prvkem tabulky *BAT* je ukazatel do videopaměti o délce jednoho slova, které se skládá z indexu použité palety (vybírá jednu z 16-ti palet určených pro dlaždice pozadí) a indexu vzoru dlaždice pozadí. Adresu vzoru dlaždice pozadí ve videopaměti získáme vynásobením indexu vzoru dlaždice pozadí číslem 32 (bitový posun o 5 vlevo).

Ukazatel má tvar popsaný v tabulce 2.8. Vzhledem k velikosti videopaměti by index vzoru dlaždice pozadí neměl překročit číslo 2048 (VDC HuC6270 má v případě NEC PCEngine jen polovinu videopaměti, kterou je schopen adresovat). [12]

Uspořádání ukazatelů v tabulce *BAT* je zleva doprava, shora dolů (tak, jak budou dlaždice vykresleny do roviny dlaždic pozadí).

Bit	**Význam**
15-12	index palety
11-0	index vzoru

Tabulka 2.8: Tvar ukazatele na dlaždici pozadí v tabulce BAT VDC HuC6270 HuC6270

SAT - tabulka atributů sprajtů

Tabulka *SAT* nemá, narozdíl od tabulky *BAT*, pevnou pozici ve videopaměti. Ukazatel na její začátek je uložen v jednom z registrů VDC HuC6270 (viz. 2.2.2.4). Maximalní počet sprajtů (ukazatelů v tabulce *SAT*) je 64.

Princip je stejný jako u tabulky *BAT*. Narozdíl od „ jednoduchých“ dlaždic pozadí mají ale sprajty řadu atributů a lze na ně aplikovat několik transformací (např. zrcadlové otočení v obou osách). Proto je každému prvku tabulky *SAT*, tedy ukazateli na sprajt, vyhrazen prostor 4 slova, kde nejnižší dvě slova udávají souřadnice sprajtu v rámci výsledné scény, následující slovo (odspoda) udává index vzoru sprajtu ve videopaměti a nejvyšší slovo je bitové pole atributů sprajtu. Adresu vzoru sprajtu ve videopaměti získáme vynásobením indexu vzoru sprajtu číslem 64 (bitový posun o 6 vlevo).

Ukazatel má tvar popsaný v tabulce 2.9. Vzhledem k velikosti videopaměti by index

vzoru sprajtu neměl překročit číslo 512 (VDC má v případě NEC PCEngine jen polovinu paměti, kterou je schopen adresovat). [12]

Pozice jednotlivých sprajtů ve scéně je uvedena přímo v tabulce *SAT* hodnotou slov 0 a 1. Vzhledem k povaze sprajtů nemá, narozdíl od dlaždic pozadí, uspořádání tabulky *SAT* vliv na pozici při vykreslování.

Slovo	**Bit**	**Význam**
0	15-10	*nevyužito*
	9-0	pozice v ose Y
1	15-10	*nevyužito*
	9-0	pozice v ose X
2	15-11	*nevyužito*
	10-0	index vzoru
3	15	zrcadlení v ose Y
	14	*nevyužito*
	13-12	výška sprajtu (CGY)
	11	zrcadlení v ose X
	10-9	*nevyužito*
	8	šířka sprajtu (CGX)
	7	priorita (SPBG)
	6-4	*nevyužito*
	3-0	index palety

Tabulka 2.9: Tvar ukazatele na sprajt v tabulce SAT VDC HuC6270

Význam jednotlivých atributů sprajtu uložených v bitovém poli v nejvyšším slově každého prvku tabulky *SAT* je následující:

zrcadlení v ose Y - pixely sprajtu budou vykresleny zrcadlově v ose Y.

výška sprajtu (CGY) - dvoubitový kód udávající výšku sprajtu danou hodnotou tohoto atributu (decimálně): „0“ pro 16 pixelů, „1“ pro 32 pixelů a „3“ pro 64 pixelů).

zrcadlení v ose X - pixely sprajtu budou vykresleny zrcadlově v ose X.

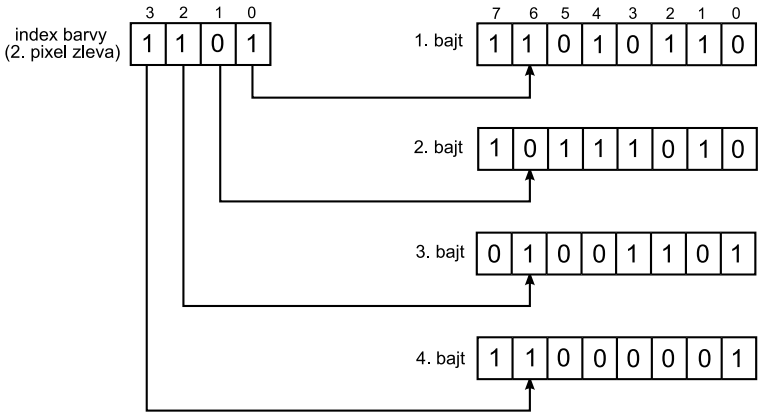
šířka sprajtu (CGX) - jednobitový kód udávající šířku sprajtu. Pro hodnotu „0“ bude mít sprajt šířku 16 pixelů, pro hodnotu „1“ pak 32 pixelů.

priorita (SPBG) - pokud je priorita nastavena na hodnotu 0, pak jsou pixely sprajtu vykresleny jen v místech překryvu s průhlednými pixely dlaždic pozadí v rovině dlaždic pozadí.

index palety - index palety, vybírá jednu ze 16-ti palet určených pro sprajty.

Kromě atributových tabulek videopamět samozřejmě obsahuje i samotná obrazová data dostupná pomocí indexu vzoru (dlaždice pozadí nebo sprajtu). Pomocí tohoto indexu lze snadno spočítat adresu vzoru ve videopaměti. Samotný vzor je pak uložen *planárním* způsobem na této adrese.

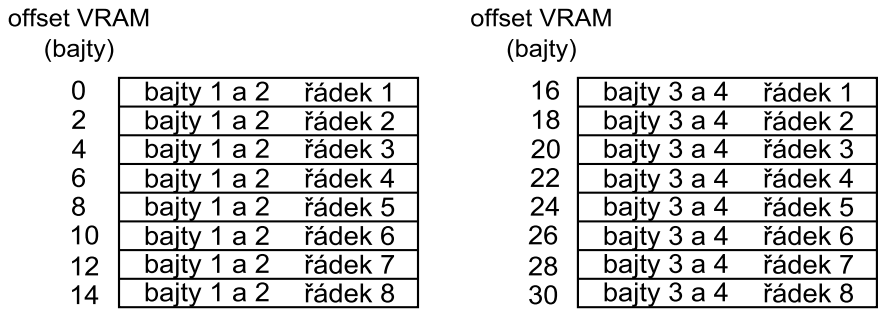
Planární způsob uložení obrazových dat je způsob, při kterém se data výsledného obrazu ukládají po bitových rovinách (bitplanes) jejichž spojením vznikne index barvy v paletě. Namísto toho, aby byl ve videopaměti obraz uložen v podobě sledu indexů barev jednotlivých bodů, je v paměti uložen sled bitových rovin představujících vždy informaci pro celý obraz (viz. obr. 2.4). Spojením informací ze všech rovin v jednom místě obrazu pak dostaneme index barvy v paletě pro toto místo. Z předchozího popisu plyne, že do n bitových rovin zakódujeme 2^n barev.



Obrázek 2.4: Planární způsob ukládání obrazových dat.

V případě VDC HuC6270 systému NEC PCEngine index barvy uvádíme v rámci palety určené ukazatelem v tabulce *BAT* nebo *SAT*. Každá z 16-ti palet (at' už palet pro dlaždice pozadí, nebo palet pro sprajty) může obsahovat maximálně 16 barev, jejichž indexy zakódujeme pomocí 4 bitů. Pro uložení obrazových dat zpracovávanhých VDC HuC6270 budou tedy potřeba 4 bitové roviny.

Vzory dlaždic pozadí jsou ve videopaměti reprezentovány jako 8 čtveřic bajtů pro jednotlivé bitové roviny. Tyto čtveřice reprezentují jednotlivé řádky vzoru. Každý vzor dlaždice pozadí ve videopaměti zabírá $8 * 8 * 4b = 256b/8 = 32B$. VDC HuC6270 neočekává čtveřice bajtů ve videopaměti přímo za sebou, ale proložené mezerou 16 bajtů (viz. obr. 2.5). Více podrobností lze nalézt např. v [8].



Obrázek 2.5: Reprezentace vzoru dlaždice pozadí a vzoru sprajtu ve videopaměti.

Vzory sprajtů jsou ve videopaměti reprezentovány stejným způsobem jako vzory dlaždic pozadí. Je však třeba vzít v úvahu několik drobných odlišností týkajících se velikosti. Minimální velikost sprajtu je 16x16 pixelů, čtveřic tedy není 8, ale 16 a pro jednotlivé řádky vzoru nejsou použity bajty ale slova. Velikost jednoho vzoru sprajtu ve videopaměti je tedy $16 * 16 * 4b = 1024b/8 = 128B$. VDC HuC6270 opět počítá s 16-ti bajtovým proložením ve videopaměti (viz. obr. 2.5).

Sprajty mohou navíc nabývat velikosti až 32x64 pixelů, což se řeší tak, že se použijí sousední vzory sprajtů (ve skutečnosti se jedná přímo o maskování pomocí atributů CGY a CGX). VDC HuC6270 se sprajtem větším než 16x16 pixelů zachází z hlediska *SAT* úplně stejně, jako se sprajtem velkým 16x16 pixelů (např. při použití atributu zrcadlového otočení v libovolné ose budou tedy otočeny i příslušné okolní části sprajtu). Více podrobností lze nalézt např. v [8].

2.2.2.3 Přímý přístup do paměti

VDC HuC6270 podporuje dva druhy přímého přístupu do paměti (Direct Memory Access, DMA). Jedná se o:

- kopírování z videopaměti do videopaměti (dále jen VRAM/VRAM)
- přesun z videopaměti do *SAT* (dále jen VRAM/*SAT*)

Řízení DMA u VDC HuC6270 je prováděno pomocí čtveřice registrů DCR, SOUR, DESR a LENR. Přenos je zahájen zápisem do registru LENR, který určuje délku bloku přenášeného z adresy uložené v registru SOUR na adresu uloženou v registru DESR. Délka bloku může nabývat hodnot 0 pro 1B dlouhý blok, až po 65535 pro 64 KB dlouhý blok. Hodnoty jednotlivých registrů slouží jako čítače a po dokončení se nenulují. Po ukončení libovolného typu DMA přenosu může VDC HuC6270 vygenerovat přerušení IRQ1 (čtením ze stavového registru VDC HuC6270 na logické adrese `$0000` jsme schopni určit o jaký druh DMA se jednalo). Během DMA přenosu je stavový příznak VDC HuC6270 „čekání na CPU pro DMA (BSY)“ nastaven na hodnotu „1“.

2.2.2.4 Registry

VDC HuC6270 disponuje dvaceti registry o délce jednoho slova. K těmto registrům se z CPU HuC6280 přistupuje pomocí tří speciálních instrukcí ST0, ST1 a ST2 a vstupně-výstupní banky `$FF` (viz. 2.2.1.6). V té jsou namapovány tři důležité adresy určené k čtení a zápisu do registrů VDC HuC6270. Jejich výčet a význam je uveden v tabulce 2.10. Detaily o významu jednotlivých bitů stavového registru přístupného čtením log. adresy `$0000` lze nalézt např. v [8].

Zápis a čtení registrů VDC HuC6270 se provádí tak, že je zápisem spodních čtyř bitů na logické adrese `$0000` nebo pomocí instrukce ST0 zvolen konkétní registr VDC HuC6270. Ten lze potom číst a zapisovat pomocí logických adres `$0002` a `$0003`, nebo pomocí instrukcí ST1 a ST2. Dvojice adres pro zápis a čtení je použita, protože CPU HuC6280 je 8-mi bitový, zatímco registry VDC HuC6270 jsou 16-ti bitové.

Logická adresa	Operace	Bit	Význam
$0000	čtení	7	*nevyužito*
		6	čekání na CPU pro DMA (BSY)
		5	přerušení vertikální synchronizace (VD)
		4	přerušení „konec DMA VRAM/VRAM“ (DV)
		3	přerušení „konec DMA VRAM/*SAT*“ (DS)
		2	přerušení „raster compare“ (RR)
		1	přerušení přetečení sprajtů (OR)
		0	přerušení kolize sprajtu 0 (CR)
	zápis	7-5	*nevyužito*
		4-0	offset registru VDC pro zápis/čtení
$0002	čtení/zápis	7-0	méně významný bajt slova v registru VDC
$0003	čtení/zápis	7-0	více významný bajt slova v registru VDC

Tabulka 2.10: Adresy VDC HuC6270 ve vstupně-výstupní bance $FF CPU HuC6280

Význam důležitých registrů je popsán v následujícím výčtu. Číslo uvedené za slovem „Registr“ je offset registru, které se skutečně zapisuje na spodní čtyři bity logické adresy $0000. Podrobné informace o jednotlivých registrech lze nalézt např. v [12].

Registr $00 MAWR (Memory Address Write)
Čítač adresy při zápisu do videopaměti. Je nutné si uvědomit, že ačkoliv je VDC HuC6270 schopný adresovat 128 KB videopaměti, dostupných je v případě systému NEC PCEngine jen 64 KB, proto je poslední platnou adresou $7FFF.

Registr $01 MARR (Memory Address Read)
Čítač adresy při čtení z videopaměti. Platí zde stejné adresní omezení jako v případě registru MAWR.

Registr $02 VRR/VWR (VRAM Read/Write)
Registr čtení/zápisu hodnoty z/do videopaměti na adrese udané hodnotou registru MARR, příp. MAWR. Při čtení více významného bajtu pomocí logické adresy $0003 dochází k automatické inkrementaci registru MARR. příp. MAWR.

Registr $05 CR (Control Register)
Konfigurační registr VDC HuC6270. Obsahuje bitové pole umožňující provádět řadu nastavení VDC HuC6270. Význam jednotlivých bitů (do kterého spadají příznaku) tohoto bitového pole je uveden v tabulce 2.11. Význam jednotlivých příznaků uložených v tomto registru následuje:

Bit	15-13	12-11	10	9-8	7	6	5-4	3	2	1	0
Příznak	?	IW	?	?	BB	SB	?	VB	RC	SO	SC

Tabulka 2.11: Význam bitů registru CR VDC HuC6270

IW Hodnota těchto dvou bitů registru CR určuje skok, o který se bude automaticky inkrementovat hodnota registru MAWR po čtení významějšího slova (tj. čtení z logické adresy $0003). Hodnoty (decimálně):

0 : inkrementace o 1

1 : inkrementace o 32

2 : inkrementace o 64

3 : inkrementace o 128

BB Povolení/zákaz vykreslování roviny dlaždic pozadí.

SB Povolení/zákaz vykreslování roviny sprajtů.

VB Povolení/zákaz generování přerušení IRQ1 při „vertical blanking“.

RC Povolení/zákaz generování přerušení IRQ1 při „raster compare“.

SO Povolení/zákaz generování přerušení IRQ1 při překrytí sprajtů.

SC Povolení/zákaz generování přerušení IRQ1 při kolizi sprajtu 0.

Registr $06 RCR (Raster Compare)
Obsah tohoto registru po součtu s číslem 64 určuje číslo řádku při kterém VDC HuC6270 vyvolá přerušení IRQ1 (pokud je toto povoleno 3 nejnižším bitem registru CR).

Registr $07 BXR (Background Scroll X)
Spodních deset bitů tohoto registru udává posun roviny dlaždic pozadí v ose X (je to offset v pixelech, nikoliv v počtu dlaždic jak mylně uvádí některé dokumenty).

Registr $08 BYR (Background Scroll Y)
Spodních devět bitů tohoto registru udává posun roviny dlaždic pozadí v ose Y (opět jde o offset v pixelech).

Registr $09 MWR (Memory Width)
Tento registr obsahuje bitové pole, které určuje velikost roviny dlaždic pozadí (ta může být větší než rozlišení obrazovky, čehož se často využívá při použití registrů BXR a BYR). Význam jednotlivých bitů v tomto bitovém poli je uveden v tabulce 2.12.

Bit	Význam	Hodnota	Šířka
15-7	*nevyužito*		
6	výška roviny dlaždic pozadí	0	32 dlaždic
		1	64 dlaždic
5-4	šířka roviny dlaždic pozadí	00	32 dlaždic
		01	64 dlaždic
		10	128 dlaždic
		11	128 dlaždic
3-0	*nevyužito*		

Tabulka 2.12: Význam bitů registru MWR VDC HuC6270

Registr `$0F` DCR (DMA Control)
Tento registr slouží k řízení DMA přenosů. Význam jednotlivých bitů v tomto bitovém poli je uveden v tabulce 2.13. Bity 2 a 3 určují operaci na zdrojové/cílové adrese, která se bude vykonávat v každé iteraci blokového přenosu. Podrobnosti o DMA přenosech jsou uvedeny v sekci 2.2.2.3.

Bit	Význam	Hodnota	Operace
15-5	*nevyužito*		
4	DSR DMA (opakování DMA přenosu z VRAM do *SAT*)		
3	Operace aplikovaná na cílovou adresu	0	snížení o 1
		1	zvýšení o 1
2	Operace aplikované na zdrojovou adresu	0	snížení o 1
		1	zvýšení o 1
1	Povolení/zákaz přerušení „konec DMA VRAM/VRAM“		
0	Povolení/zákaz přerušení „konec DMA VRAM/*SAT*“		

Tabulka 2.13: Význam bitů registru DCR VDC HuC6270

Registr `$10` SOUR (DMA Source Address)
Tento registr slouží k nastavení zdrojové adresy DMA přenosu.

Registr `$11` DESR (DMA Destination Address)
Tento registr slouží k nastavení cílové adresy DMA přenosu.

Registr `$12` LENR (DMA Block length Address)
Tento registr slouží k nastavení délky přenášeného bloku při DMA přenosu. Zápis do tohoto registru automaticky zahajuje DMA přenos.

Registr `$13` SATB (Sprite Attribute Table)
Tento registr obsahuje ukazatel na počátek tabulky *SAT* ve videopaměti.

2.2.2.5 Přerušení

VDC HuC6270 může v závislosti na nastavení registrů CR a DCR generovat přerušení IRQ1 v řadě případů. Tyto případy lze snadno odlišit pomocí příznaků přerušení ve stavovém registru VDC HuC6270, který lze číst na logické adrese `$0000`.

Významná přerušení, která může VDC HuC6270 generovat jsou uvedena v následujícím výčtu. Pro přehlednost jsou přerušení uvedena pod názvem příslušných příznaků stavového registru VDC HuC6270.

VD (Vertical Blanking)
Toto přerušení je generováno při *vertikální synchronizaci*. Používá se k jednoduchému synchronizování programového kódu a vykreslování obrazu.

DV (VRAM to VRAM DMA Transfer Completion)
Toto přerušení je generováno při dokončení operace DMA kopírování z videopaměti do videopaměti (VRAM/VRAM).

DS (VRAM to SAT DMA Transfer Completion)
Toto přerušení je generováno při dokončení operace DMA přesun z videopaměti do *SAT*.

RR (Raster Compare)
Toto přerušení je generováno, když interní čítač řádek obrazovky nabyde hodnoty registru RCR zvětšené o číslo 64. [12]

OR (Sprite Overflow)
Toto přerušení nastává pokud je v aktuálně vykreslované řádce více než 16 sprajtů.

CR (Sprite 0 Collision)
Toto přerušení nastává v případě, kdy se netransparentní pixel sprajtu s pořadovým číslem 0 (tj. prvního záznamu v tabulce *SAT*) překryje s netransparentním pixelem kteréhokoliv jiného sprajtu.

2.2.3 VCE

Video Color Encoder (VCE), neboli enkodér barev nese označení HuC6260 a obstarává systému NEC PCEngine správu barev. Pomocí registrů VCE je možné měnit barvy ve všech 32 paletách (16 palet pro dlaždice pozadí a 16 palet pro sprajty).

2.2.3.1 Palety

Jak již bylo uvedeno dříve, VDC HuC6270 a VCE HuC6260 pracují celkem s 512-ti barvami uloženými v barevné tabulce. Prvních 256 záznamů je určeno výhradně pro vykreslování dlaždic pozadí a zbývajících 256 záznamů, pak výhradně pro vykreslování sprajtů. Logicky jsou barvy v tabulce uspořádány do dvou skupin 16-ti palet, kde každá paleta obsahuje 16 barev. Tyto palety jsou odkazovány přímo v ukazatelích z atributových tabulek *BAT* a *SAT* spravovaných VDC HuC6270.

Jednotlivé záznamy tabulky barev jsou slova obsahující barevnou informaci. Tvar jednoho záznamu z barevné tabulky je uveden v tabulce 2.14. Indexace barev probíhá v rámci celé tabulky 9-ti bitovým indexem barvy (logické rozdělení na palety a skupiny palet je vnímáno pouze při práci s tabulkami *BAT* a *SAT* v rámci VDC HuC620.

Bit	**Význam**
15-9	*nevyužito*
8-6	zelená
5-3	červená
2-0	modrá

Tabulka 2.14: Tvar záznamu v barevné tabulce VCE HuC6260

Palety obsahují několik speciálních barev, při jejichž použití musíme očekávat jiné chování vykreslovacích mechanismů. Tyto speciální barvy jsou:

transparentní barva - transparentní barvu mají pixely (dlaždice pozadí nebo sprajtu), které nemají být vykresleny. V případě dlaždice pozadí bude na jejich místě vykreslena *barva pozadí*, v případě sprajtu barva pixelu dlaždice pozadí ležícím v místě transparentního pixelu sprajtu (případně opět *barva pozadí*, pokud je i pixel dlaždice pozadí transparentní). Transparentní barva je umístěna na indexu 0 ve všech paletách dlaždic pozadí i sprajtů.

barva pozadí - barva, která je vykreslena na místě „pod“ transparentním pixelem dlaždice pozadí. Tato barva je umístěna v 0-té paletě pro dlaždice pozadí na indexu 0 (tato barva je vlastně transparentní barvou pro všechny palety dlaždic pozadí). Barvu na indexu 0 všech ostatních palet pro dlaždice pozadí není možné zobrazit.

barva přesahu - barva, která je vykreslována zpravidla mimo plochu aktivního obrazu. Jedinný případ, kdy je tato barva zobrazena na obrazovce je, když je pomocí registru CR VDC HuC6270 zakázáno vykreslování obou rovin (dlaždic pozadí i sprajtů). Jedná se o barvu na indexu 0 v 0-té paletě pro sprajty. Barvu na indexu 0 barvu všech ostatních palet pro sprajty též není možné zobrazit.

2.2.3.2 Registry

VCE mapuje své registry do adresního prostoru CPU HuC6280 pomocí čtveřice adres v rámci stránky `$FF` (viz. 2.2.1.6). Jejich výčet a význam je uveden v tabulce 2.15. Práce s VCE HuC6260 je v mnohém analogická s prací VDC HuC6270.

Logická adresa	Význam
`$0402`	Index barvy v tabulce (méně významný bajt)
`$0403`	Index barvy v tabulce (více významný bajt)
`$0404`	Data (méně významný bajt)
`$0405`	Data (více významný bajt)

Tabulka 2.15: Adresy VCE HuC6260 ve vstupně-výstupní bance $FF CPU HuC6280

2.2.4 PSG

Programmable Sound Generator (PSG), tedy programovatelný generátor zvuku, je jedno z periferních zařízení CPU HuC6280 osazené na stejném čipu. Podrobně se jím zabývá např. [4].

PSG je vybaven šesti kanály, které jsou primárně určeny pro přehrávání 32 bajtových lineárních samplů s rozlišením 5 bitů. Některé kanály mohou být využity k frekvenční modulaci[11] pomocí nízkofrekvenčního oscilátoru (Low Frequency Oscillator, LFO), či generování bílého šumu[12] (White Noise). Každý z kanálů může být také přepnut do módu přímého přístupu k datům (Direct Data Access, DDA), kdy je zvuk programován přímo CPU HuC6280, což

[11] Možnost měnit aktuální nosnou frekvenci přehrávaného signálu pomocí modulační amplitudy.

[12] Náhodný signál se stejným výkonem v pásmech se shodnou šířkou. Využívá se k neumělé syntetizaci zvuku perkusí apod.

dovoluje např. jednoduchou syntézu zvuku. Možnosti jednotlivých kanálů PSG jsou uvedeny v tabulce 2.16.

V případě nastavení módu frekvenční modulace pro kanál 0 bude kanál 1 vždy automaticky ztlumen.

Kanál	Dostupné režimy
0	samply, frekvenční modulace (podle kanálu 1), DDA
1	samply, frekvenční modulace (kanálu 0), DDA
2	samply, DDA
3	samply, DDA
4	samply, bílý šum, DDA
5	samply, bílý šum, DDA

Tabulka 2.16: Možnosti jednotlivých kanálů PSG

Ke zprocesování dat využívá PSG stejný zdroj hodin jako CPU HuC6280. Jedná se o hodiny pevně nastavené na hodnotu 3.58 MHz nezávisle na stavu skrytého registru CS CPU HuC6280.

2.2.4.1 Registry

Podobně jako VDC HuC6270, nebo VCE HuC6260 i PSG mapuje své registry do adresního prostoru CPU HuC6280. Jedná se o deset adres v rámci stránky $FF (viz. 2.2.1.6). Jejich výčet a význam je uveden v tabulce 2.17, kde jsou ve spodní části tabulky oddělené čarou uvedeny registry ovlivněné hodnotou zapsanou na adresu $8000. Všechny registry PSG jsou 8-mi bitové.

Nastavení hodnot registrů ovládajících režim (frekvenční modulace, generování bílého šumu) u kanálu, kde tento režim není dostupný, bude PSG ignorováno.

Logická adresa	Význam
$0800	Výběr kanálu (Channel Select)
$0801	Globální vyvážení hlasitosti (Global Balance)
$0802	Jemné nastavení frekvence (Fine Frequency Adjust)
$0803	Hrubé nastavení frekvence (Rough Frequency Adjust)
$0804	Režim kanálu (Channel Mode)
$0805	Vyvážení hlasitosti (Channel Balance)
$0806	Data
$0807	Povolení a frekvence bílého šumu (Noise Control)
$0808	Frekvence oscilátoru modulace (LFO Frequency)
$0809	Povolení a ovládání oscilátoru modulace (LFO Control)

Tabulka 2.17: Adresy PSG ve vstupně-výstupní bance $FF CPU HuC6280

Registr `$0800` (Channel Select)
Nastavením spodních 2 bitů se určuje, který z kanálů PSG bude použit při zápisech/čteních registrů `$0802` - `$0809`.

Registr `$0801` (Global Balance)
Hodnota tohoto registru určuje globální hlasitost pro levý a pravý kanál v mezích hodnot 0-15. Pro nastavení globální hlasitosti pravého kanálu slouží spodní 4 bity, pro nastavení globální hlasitosti levého kanálu pak následující 4 bity. Tato hlasitost je aplikována až po smíšení všech kanálů v jejich konkrétních hlasitostech.

Registry `$0802` (Fine Frequency Adjust) a `$0803` (Rough Frequency Adjust)
Frekvence je v případě PSG 12-ti bitové číslo. Spodní 4 bity registru `$0803` představují vrchní 4 bity a hodnota registru `$0802` zbývajících 8 bitů tohoto čísla. Pro převod na Hz lze použít následující vztah:

$$f = \frac{3580000}{32 * 12bit} Hz$$

což je dáno způsobem práce s frekvencí v režimu přehrávání samplů. V tomto režimu se jedná o ukazatel, který je dekrementován o 1 celkem 3580000-krát za sekundu a ukazuje přímo do dat samplu.

Registr `$0804` (Channel Control)
Tento registr slouží jako bitové pole určené ke konfiguraci parametrů zvoleného kanálu. Význam jednotlivých bitů je popsán v tabulce 2.18.

Bit	**Význam**
7	Kanál zapnut/vypnut
6	DDA režim zapnut/vypnut
5	*nevyužito*
4-0	Celková hlasitost kanálu

Tabulka 2.18: Význam bitů registru `$0804` PSG

Registr `$0805` (Channel Balance)
Tento registr slouží k nastavení vyvážení hlasitosti zvoleného kanálu. Pracuje analogicky s registrem `$0801`.

Registr `$0806` (Data)
Spodních 5 bitů tohoto registru slouží k zápisu dat na zvolený kanál. Chování registru je rozdílné podle zvoleného režimu kanálu (viz. bity 7 a 6 registru `$0804`).

V běžném případě se do tohoto registru zapisují data samplu. V případě, že je na kanálu povoleno DDA, pak jsou data zapsaná do tohoto registu přímo mixována s ostatními kanály. V případě, že je zvolen kanál 1, je neaktivní a DDA je vypnuto, slouží tento registr k zápisu modulační vlny (více viz. [4]).

Registr `$0807` (Noise Control)
Tento registr slouží k zapnutí a vypnutí generátoru bílého šumu a nastavení jeho frekvence. Použití tohoto registru má význam pouze v případě kanálů 4 a 5. Registr je

uspořádán jako bitové pole s bity významu popsaného v tabulce 2.19. Frekvence bílého šumu nastavovaná pomocí spodních 5-ti bitů je vyjádřena vzorcem (podrobnosti o původu vzorce lze nalézt v [4]):

$$f = \frac{3580000}{64 * 5bit} Hz$$

Bit	Význam
7	Bílý šum zapnut/vypnut
6-5	*nevyužito*
4-0	Frekvence šumu

Tabulka 2.19: Význam bitů registru `$0807` PSG

Registr `$0808` (LFO Frequency)
V případě, že je kanál 0 v režimu frekvenční modulace, slouží tento registr u kanálu 1 k modelování frekvence modelovací vlny (definované registry `$0802` a `$0803` při vybraném kanálu 1. Frekvence daná tam uloženým 12-ti bitovým číslem je vynásobena frekvencí LFO uloženou v tomto registru před dalším zpracováním.

Registr `$0809` (LFO Control)
Tento registr slouží k ovládání parametrů frekvenční modulace kanálu 0 kanálem 1. Jedná se o bitové pole, kde význam jednotlivých bitů je uveden v tabulce 2.20. Více informací o programování oscilátoru frekvenční modulace je uvedeno v [4].

Bit	Význam
7	Oscilátor modulace zapnut/vypnut
6-2	*nevyužito*
1-0	Režim oscilátoru modulace

Tabulka 2.20: Význam bitů registru `$0809` PSG

2.2.5 Obraz ROM

Obraz ROM je binární otisk obsahu paměti v paměťovém modulu na čipové kartě HuCard do souboru. Lze jej pořídit pomocí speciálního zařízení (např. PCE Pro 32M na obr. 2.6), které sekvenčně adresuje a čte jednotlivé bloky paměti připojené čipové karty HuCard a pomocí speciálního software komunikujícího se zařízením pomocí standardního rozhraní (USB, paralelní port) je ukládá do souboru na disku PC.

V nepříliš široké komunitě zabývající se emulací systému NEC PCEngine se ustálil formát souboru s obrazem ROM použitý prvním emulátorem tohoto systému (MagicEngine, [26]). Tento formát se vyznačuje absencí jakýchkoliv metadat, kromě volitelné 512 bajtové hlavičky v úvodu souboru, kterou lze snadno detekovat lichým počtem 512 bajtových bloků v souboru s obrazem ROM.

Obrázek 2.6: PCE Pro 32M

Speciálním případem obrazu ROM jsou poměrně často používané obrazy o velikosti 384 KB. Tyto obrazy musí být rozděleny na dvě části umístěné v paměti ROM podle schématu naznačeného v tabulce 2.21.

Offset v souboru	**Délka**	**Offset v ROM**
$00000	$40000	$00000
$40000	$20000	$80000

Tabulka 2.21: Schéma rozdělení 384 KB obrazů ROM

Maximální velikost paměťového modulu v čipové kartě HuCard je v základní specifikaci omezena na 1 MB, minimální velikost není určena.

2.3 Software

Za dobu působení společnosti NEC v oblasti videoherního průmyslu, od roku 1987 do roku 1998, vznikla pro systém NEC PCEngine řada titulů. Jak samotný účel systému předurčuje, majoritní většina byly herní programy. Kromě původních titulů se společnost NEC snažila získat uživatele videoherních systémů na svou stranu vydáním některých titulů, které se proslavily na systému Nintendo NES, jako např. *Castlevania* nebo *StreetFighter*. Řada titulů vydaných právě pro NEC PCEngine je přehledně katalogizována na [31].

Kapitola 3

Existující implementace

I přes to, že systém NEC PCEngine nikdy nedosáhl podobné popularity a rozšíření ve světe, jako jiné videoherní konzole stejné generace (Nintendo Famicom, Sega Master System, Atari 7800), vzniklo několik jeho emulátorů. Jejich existence značně usnadňuje vývoj a lazení nového emulátoru. Kromě toho, že mohou být použity jako prostředek pro ověření funkčnosti v případě absence skutečného, dnes již nedostupného, hardware systému NEC PCEngine, je řada z nich šířena i ve zdrojové podobě jako svobodný software, což umožňuje studovat jak specifikaci systému, tak i způsob jakým jejich autoři vyřešili vlastní implementaci jeho jednotlivých částí.

V úvodní sekci 3.1 této kapitoly je nastíněna situace v oblasti emulace systému NEC PCEngine a jsou představeny některé existující implementace emulátorů. Sekce 3.2 shrnuje přednosti a nedostatky těchto implementací a na jejich základě nastiňuje směr pro další analýzu, návrh a implementaci vlastního emulátoru.

3.1 Přehled emulátorů NEC PCEngine

Většina emulátorů systému NEC PCEngine vznikla v letech 1995-2001, hlavně pro osobní počítače s operačními systémy Microsoft DOS a Microsoft Windows. V následujícím přehledu jsou uvedeny pouze tři nejpoužívanější, aktivně udržované a vyvíjené emulátory. Další, které většinou nejsou zdaleka tak kompletní z hlediska funkčnosti a kompatibility se zveřejněnými čipovými kartami HuCard, nebo je jejich kód zastaralý a jsou v dnešních podmínkách nepoužitelné, lze nalézt např. na [21].

3.1.1 MagicEngine

MagicEngine [26] je komerční emulátor systému NEC PCEngine. Podporuje emulaci všech verzí systému včetně řady rozšíření (např. CD-ROM mechanika). Emulace je v podání tohoto emulátoru velice přesná, rychlá a bez problému je možné spustit většinu obrazů zveřejněných na čipových kartách HuCard, včetně těch, které využívají nestandardního chování systému.

Podporovány jsou platformy Microsoft DOS (jen starší verze emulátoru), Microsoft Windows a Apple MacOS X. Program disponuje vlastním jednoduchým grafickým uživatelským

rozhraním. Nároky na hardware jsou poměrně vysoké a při použití pod operačním systémem Windows vyžaduje MagicEngine grafickou akceleraci pomocí DirectX nebo OpenGL. V současné době se jedná o nejpopulárnější emulátor NEC PCEngine pro Microsoft Windows.

Přesto, že autoři MagicEngine významným dílem přispěli ke zdokumentování architektury NEC PCEngine a zveřejnili řadu nástrojů pro vývoj programů s tímto systémem kompatibilních, zdrojový kód MagicEngine není otevřený, což znemožňuje zásahy kýmkoliv jiným než autory.

Klíčové vlastnosti

- vysoká přensnost emulace
- podpora všech verzí a rozšíření systému
- kompatibilita s většinou čipových karet HuCard

3.1.2 Ootake

Ootake [29] je emulátor systému NEC PCEngine od japonského autora Nakamura Kitao. Podobně jako MagicEngine podporuje Ootake všechny verze systému včetně řady rozšíření. Autor se speciálně věnuje problematice dosažení stejného pocitu ze hry[1] jako na originálním hardware, takže je emulátor velice přesný i po stránce zpracování vstupu apod. Množina podporovaných titulů je o něco menší než v případě MagicEngine.

Ootake je napsán výhradně pro platformu Microsoft Windows v jejímž duchu se také nese uživatelské rozhraní programu. Hardwarové nároky Ootake jsou poměrně vysoké, emulátor vyžaduje grafickou akceleraci pomocí DirectX a procesor s rychlostí cca 1.6GHz pro plynulý běh herních programů.

Zdrojový kód emulátoru Ootake je veřejně přístupný a šířen pod licencí GNU/GPL. Vzhledem k tomu, že je program vyvíjen bez ohledu na přenositelnost kódu, na řadě míst se prolíná kód zajišťující logiku emulace a specifický kód pro uživatelské rozhraní v Microsoft Windows. Kód je z důvodu přesnosti emulace protkán řadou optimalizací a jeho čitelnost navíc ztěžuje fakt, že veškeré komentáře jsou psány v japonštině.

Klíčové vlastnosti

- vysoká přesnost emulace s ohledem na pocit ze hry
- podpora všech verzí a rozšíření systému
- otevřený zdrojový kód

[1] Např. poměrně složitě implementuje zpracování vstupu z herního ovladače tak, aby bylo kompenzováno zpoždění způsobené obsluhami operačního systému.

3.1.3 Mednafen

Mednafen [28] je emulátor několika populárních videoherních systémů mezi nimiž je i systém NEC PCEngine. Emulace NEC PCEngine v rámci tohoto programu využívá několika společných komponent s emulátorem systému NEC PC FX32, který je nástupcem NEC PCEngine. Stejně jako v předchozích případech je podporována řada rozšíření. I přes některé nepřesnosti časování v emulaci je celková přesnost emulace vysoká. Součástí programu je podpora hry více hráčů na jedné konzoli přes TCP/IP (vzdálený herní ovladač), nebo vestavěný ladicí nástroj, který umožňuje krokovat běžící herní program a sledovat obsah významných oblastí paměti a registrů.

Program je psán přenositelně a je prokazatelně možné ho sestavit a používat na platformách GNU/Linux, FreeBSD, Apple MacOS X a Microsoft Windows. Uživatelské rozhraní programu je zajištěno pomocí přenositelné knihovny pro uživatelská rozhraní libSDL. Hardwarové nároky na plynulou emulaci jsou oproti předchozím řešením nižší.

Mednafem má otevřený zdrojový kód šířený pod licencí GNU/GPL. Je psán v jazyce C++ s důrazem na přenositelnost, hlavně mezi platformami GNU/Linux a Microsoft Windows. Nepočítá se však s přenositelností na platformy, kde není dostupná knihovna libSDL, která je pevnou součástí programového kódu. Stejně tak jednotlivé emulátory podporovaných systémů nepoužívají žádné společné rozhraní, které by umožňovalo program jednoduše rozšířit.

Klíčové vlastnosti

- podpora dalších videoherních systémů
- integrovaný nástroj pro lazení emulovaných programů
- podpora hry více hráčů po síti TCP/IP
- otevřený zdrojový kód
- přenositelnost

3.2 Shrnutí

Většina existujících implementací emulátoru NEC PCEngine (včetně řady těch, které nejsou v předchozím výčtu uvedeny) jsou jednoúčelové programy soustředící se výhradně na emulaci tohoto systému, případně jeho dalších variant a rozšíření.

Jistě pozitivní stránkou tohoto přístupu je v některých případech vysoká přesnost a kvalita emulace, protože se autor nemusí soustředit na řadu dalších podporovaných videoherních systémů. Na druhou stranu je množina opravdu nezapomenutelných a populárních herních titulů pro jednotlivé videoherní systémy poměrně malá[2] a uživatelé často používají více emulátorů několika různých systému. Jsou tak nuceni používat více jednoúčelových programů s

[2]Velice populární japonská série herních titulů Final Fantasy vydávaná společnosti Square Enix má v současné době více než 13 dílů, které vyšly střídavě pro 12 různých typů videoherních systémů s průměrnou bilancí 1-2 díly na jeden typ videoherního systému.

různým způsobem ovládání a chováním i přesto, že jejich úkol je z uživatelského pohledu stejný.

„Slabinou“ existujících implementací je také poměrně malá množina podporovaných platforem. Jak již bylo uvedeno, většina emulátorů systému NEC PCEngine vznikla v letech 1995-2001, tedy v období kdy poměrně vysokým nárokům na výkon při provádění emulace dostačovaly prakticky jen osobní počítače a v oblasti operačních systémů nebyl např. GNU/Linux zdaleka tak populární jako Microsoft DOS nebo Microsoft Windows.

V současné době existuje řada videoherních handheldů[3], chytrých mobilních telefonů[4] a kapesních počítačů, které mohou výkonem směle konkurovat tehdejším osobním počítačům, nikoliv však těm dnešním, což je jeden z důvodů proč právě tato zařízení jsou zajímavou, platformou pro emulátory starších videoherních systémů. Většina existujících emulátorů ale stěží počítá s přenositelností na jiné operační systémy než je Microsoft Windows, natož na tato zařízení.

Kromě slabin existujících implmentací ale existují i jejich přednosti. Nejvíce inspirativní je v této oblasti poslední zmíněný emulátor Mednafen, který implementuje přenositelné uživatelské rozhraní pomocí knihovny libSDL a obsahuje podporu pro lazení emulovaného programu, nebo hru více hráčů po síti TCP/IP.

Při následující analýze a návrhu vlastního emulátoru je kladen důraz na potlačení uvedených nedostatků při zachování předností existujících řešení.

Emulátor bude navržen modulárně, aby bylo v budoucnu snadné ho rozšířit o podporu emulace dalších videoherních systémů při zachování společného ovladání. Stejně tak podpora platforem nebude způsobem implementace omezena pouze na operační systémy osobních počítačů.

[3]Přenosné kapesní videoherní konzole vybavené displejem a ovládacími prvky herního ovladače, často napájené z baterií, např. Sony Playstation Portable, Nintendo DS nebo Gamepark Wiz.

[4]Mobilní telefony vybavené operačním systémem umožňujícím instalaci a spouštění dalších programů, např. Nokia Series60 s operačním systémem Symbian nebo telefony HTC s operačními systémy Google Android nebo Microsoft Windows Mobile

Kapitola 4

Analýza a návrh

Analýza a návrh, jimž se věnuje tato kapitola, jsou obecně nejdůležitější fází vývojového procesu. V sekcích 4.1 a 4.2 jsou popsány nejčastější používané techniky emulace jednotlivých součástí videoherních systémů s ohledem na specifikaci systému NEC PCEngine (viz. kapitola 2).

Na jejich základě a v závislosti na požadavcích vyplývajících z rešerše existujících implementací (viz. kapitola 3), shrnutých v sekci 4.3, byla pro následnou implementaci modulárního, přenositelného emulátoru navržena architektura, kterou popisuje sekce 4.4.

4.1 Emulace

Emulace je v oblasti počítačové vědy proces, kdy program zvaný emulátor co nejpřesněji napodobuje interní chování emulovaného zařízení. Z hlediska uživatele tak navozuje dojem práce s tímto zařízením. Hlavní výhodou emulace je fakt, že v případě skutečného zařízení můžeme očekávat velmi podobné až stejné chování jako v emulátoru.

Ačkoliv Church-Turingova teze[1] říká, že jsme schopni v libovolném výpočetním prostředí emulovat libovolný algoritmus, je nutno brát v potaz výpočetní a paměťovou náročnost tohoto procesu. Jinými slovy, nevýhodou emulace může být výkon emulátoru, a to od stavu, kdy je emulované zařízení pomalejší než jeho skutečná varianta až po stav, kdy emulace postrádá smysl, nebo je z hlediska poměru výkonů neproveditelná. [6]

Emulace je často zaměňována se simulací, což je proces, kdy je prostřednictvím programu zvaného simulátor napodobeno chování simulovaného zařízení z hlediska uživatele, nikoliv však z hlediska interního chování tohoto zařízení. Hlavní výhodou simulace je vysoký výkon a často i jednoduchost z důvodu použití nativních vlastností platformy při implementaci simulátoru. Nevýhodou je pak nízká přesnost až odlišné chování na různých platformách.

4.2 Techniky používané při emulaci

V oblasti emulace existuje řada rozdílných technik a přístupů, jejichž volba má významný dopad nejen na výkon a přesnost, ale také na obtížnost implementace, způsob lazení, přeno-

[1] Hypotéza, která říka, že ke každému algoritmu splňujícímu určité podmínky lze najít ekvivalentní Turingův stroj.

sitelnost, nebo na výkonové a paměťové nároky výsledného emulátoru.

Následující text představuje základní z těchto technik a nastiňuje způsob emulace jednotlivých částí emulovaného systému NEC PCEngine (vzhledem k tomu, že je architektura většiny videoherních systémů podobná, lze je považovat za obecné). Více informací o těchto a dalších technikách lze najít např. v [5, 3].

4.2.1 Zpracování instrukcí

Základním stavebním kamenem emulátoru videoherního systému je emulátor hlavního procesoru, který zpracovává instrukce herního programu. Podle způsobu, jakým emulátor nakládá s načtenými instrukcemi nebo jejich bloky, rozlišujeme dva základní typy emulace zpracování instrukcí:

interpretační - emulátory založené na tomto principu pracují tak, že načítají jednotlivé instrukce kódu emulovaného programu a interpretují je na datové struktuře představující procesor. Tento přístup je velmi jednoduchý na implementaci a lazení, lehce přenositelný, ale náročný na výkon a paměť.

rekompilační - emulátory, které provádějí rekompilaci (nebo také binární translaci) načítají logické celky instrukcí kódu emulovaného programu a překládají je na ekvivalentní celky nativních instrukcí. Tento přístup je sice velmi náročný na implementaci, ale umožňuje dosažení vysokého výkonu díky tomu, že se naplno využije potenciál nativního procesoru (včetně instrukční cache apod.). Pokud dochází k rekompilaci celého programu při načtení do emulátoru, mluvíme o *statické rekompilaci*, v případě že k rekompilaci dochází až při volání kódu, nebo jeho načtení do paměti v průběhu vykonávání programu, jedná se o *rekompilaci dynamickou*.

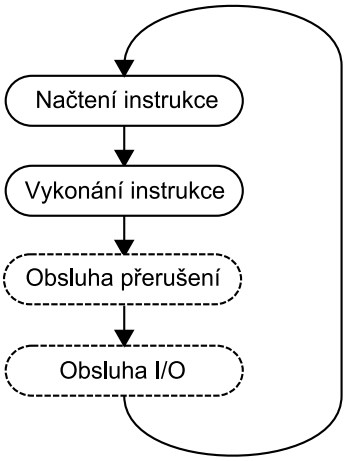
Vzhledem k požadavku na přenositelnost, rychlosti a schopnostem CPU HuC6280 v porovnání s dnešními procesory, bude pro účely návrhu a implementace emulátoru systému NEC PCEngine použita metoda interpretační emulace.

Celý proces emulace pomocí interpretačního emulátoru je tvořen smyčkou, ve které emulátor načte z paměti instrukci, interpretuje ji na datové struktuře představující procesor a provede případné další obsluhy (např. přerušení). Tato smyčka je zhruba znázorněna diagramem na obr. 4.1.

4.2.2 Paměť

Instrukce zpracovávané hlavním procesorem obvykle pracují s daty uloženými v paměti, která je dalším významnou součástí emulovaného systému NEC PCEngine, stejně jako mapovač, který procesoru umožňuje přistupovat k paměti v celém rozsahu.

Emulovaná paměť je většinou realizována jako pre-alokované bloky paměti v rámci platformy, na které emulátor běží. Tyto bloky jsou využívany v situaci, kdy emulátor v emulovaném programu odchytí pokus o přístup do některého z regionů paměti jako je RAM, ROM nebo vstupně-výstupní pamět namapovaná zařízeními. Jednotlivé regiony paměti pak mají nastarost funkce emulující mapovač, které zajišťují přepočet adresy a aplikaci případných restrikcí přístupu do paměti.



Obrázek 4.1: Interpretační emulace

4.2.3 Periferní zařízení

Systém NEC PCEngine je kromě samotného CPU HuC6280 vybaven řadou periferních zařízení. Způsob emulace těchto zařízení je závislý na jejich účelu a druhu činnosti.

Periferní zařízení jsou obvykle představována svým vnitřním stavem (obsah interní paměti, registrů apod.) a případně uživatelským rozhraním. Často jsou stejně jako procesor reprezentována datovou strukturou obsahující informace o vnitřním stavu zařízení, nad kterou operují funkce volané emulačním kódem procesoru. U zařízení, která disponují zmiňovaným uživatelským rozhraním, je třeba funkčně zajistit propojení s uživatelským rozhraním emulátoru a patřičnou konverzi vstupu/výstupu.

V případě systému NEC PCEngine je nutno brát v potaz zejména uživatelské rozhraní následujících periferních zařízení:

- port herního ovladače
- PSG
- VDC HuC6270, VCE HuC6260

U řady zařízení je nutné si uvědomit, že prostředky použité k jejich emulaci fungují diametrálně odlišně a proto nemusí být vždy nutné emulovat zařízení zcela přesně. Např. systém vykreslování obrazu pomocí VDC HuC6270 a VCE HuC6260 na obrazovku je v případě emulátoru zcela odlišný (do bitové mapy) než u původního hardware (na televizní obrazovku) a emulování videovýstupu až po modulaci analogového signálu vhodného pro RF vstup televizního setu by bylo zbytečné.

4.2.4 Časování

Ve skutečném hardware je časování a synchronizace jednotlivých komponent zajišťena generátorem hodinových pulzů, který generuje pulzy o konstantní frekvenci. Hodinový signál

je pak vhodně předdělen a dodán jednotlivým zařízením v systému. Z hlediska emulace existují dva základní přístupy k časování s ohledem na:

přesnost - kód emulátoru zpravidla odpočítává cykly procesoru během provádění emulovaného programu a jednotlivé zdroje časových pulzů jsou přímo emulovány pomocí služeb platformy, na které emulátor běží. To zaručuje dosažení věrohodné rychlosti emulace, pokud to dovolují prostředky této platformy. Tento přístup je vhodný spíše tam, kde je kladen důraz na průběh emulovaného programu a čas získání výsledku není kritický.

rychlost - původní zdroje hodinového signálu jsou zcela ignorovány a veškerá emulace probíhá maximální možnou rychlostí, kterou udává platforma, na které emulátor běží. Tento přístup se často používá při emulaci, kde je kladen důraz spíše na výsledek než průběh emulovaného programu.

V případě videoherních konzolí je rychlost provádění programu důležitým faktorem. Většina herních programů je závislá na postřehu a zručnosti hráče a rychlost provádění programu je častokrát přímo závislá na taktu procesoru, protože narozdíl od různorodých konfigurací osobních počítačů je u videoherní konzole tento parametr pro vývojáře konstantní. Provádění programu určeného pro procesor s taktovací frekvencí 7.16MHz plnou rychlostí na procesoru s taktovací frekvencí 2.0GHz bude, i přes výkonostní srážku způsobenou režií emulace, tak rychlé, že bude program nepoužitelný.

Také je nutné vzít v úvahu fakt, že pro synchronizaci kódu a dění na obrazovce se u videoherních systémů ve většině případů používá „vertikální synchronizace“. Do výpočtu časování tedy vstupuje další element - televizní standard (NTSC nebo PAL), který přesně určuje počet řádků tvořících jeden snímek, a kolikrát za vteřinu dojde k překreslení celého snímku.

4.3 Požadavky na program

Úkolem výsledného programu je emulace videoherního systému NEC PCEngine v rozsahu jeho základní verze, popsané specifikací uvedenou v kapitole 2, za použití výše popsaných technik a závěrů plynoucích z rešerše existujících implementací v kapitole 3.

4.3.1 Funkční požadavky

Výsledný program bude provádět následující činnosti:

- načtení obrazu ROM pro NEC PCEngine
  - načte obraz ROM s programem ze souboru
  - provede potřebné úpravy obrazu ROM (rozdělení, ořez hlavičky)

- emulace zpracování programu
  - zpracuje interpretačním způsobem program v načteném obrazu ROM

- podpora emulace jednotlivých součástí NEC PCEngine
  – bude emulovat CPU HuC6280 a jeho interní periferní zařízení
  – bude emulovat základní funkce VDC HuC6270 a VCE HuC6260
  – bude emulovat základní funkce PSG[2]
- interakce s uživatelem
  – vykreslí aktuální snímek generovaný VDC a VCE
  – přehraje zvuk generovaný PSG
  – zpracuje vstup z klávesnice jako vstup z herního ovladače

4.3.2 Nefunkční požadavky

Výsledný program bude mít následující vlastnosti:

- přenositelnost kódu
  – bude možné ho sestavit a použít na platformách GNU/Linux a Microsoft Windows
  – bude možné ho jednoduše přenést na další platformy
- modularita
  – bude možné ho jednoduše rozšířit o další emulátory videoherních systémů
  – bude možné ho jednoduše rozšířit o další uživatelská rozhraní[3]
- implementační prostředí
  – bude napsán v jazyce C nebo C++

4.4 Architektura programu

Architektura programu vychází z principů návrhu otevřeného software pro operační systém UNIX popsaných v [11]. Její základní myšlenkou je jednoduchost, rozšiřitelnost a oddělení jednotlivých logických částí programu.

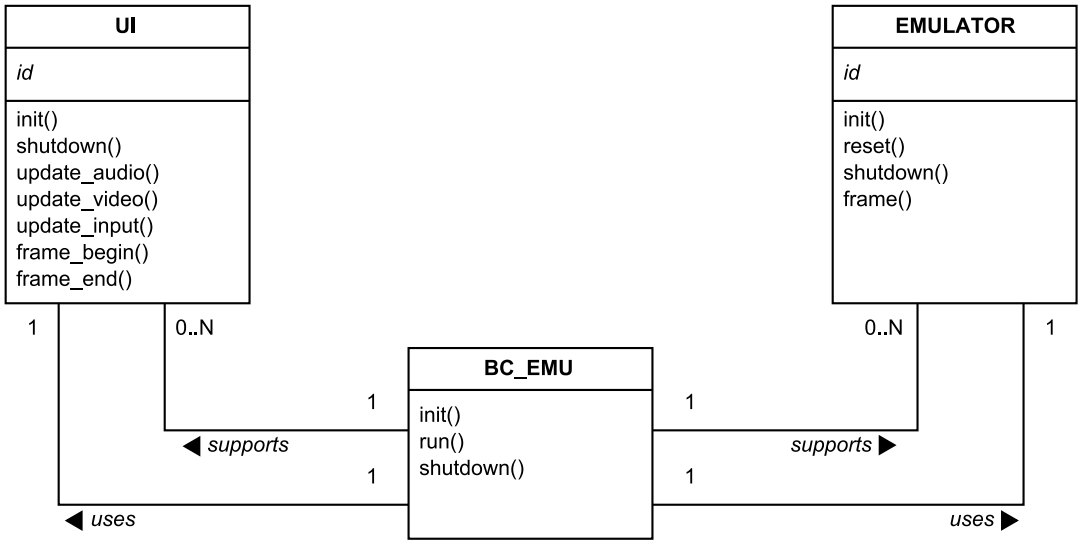
Stěžejními prvky architektury programu jsou, přímo na základě nefunkčního požadavku *modularity*, dva typy modulů:

- moduly emulátorů
  – zajišťují emulaci videoherních systémů
- moduly uživatelských rozhraní
  – zajišťují rozhraní mezi emulátorem a uživatelem (obraz, zvuk, vstup)

Tyto moduly mají přesně specifikované rozhraní jak pro komunikaci mezi sebou, tak pro komunikaci s jádrem programu, které zajišťuje inicializaci správné dvojice modulů (vždy právě jednoho emulátoru a právě jednoho uživatelského rozhraní) při spuštění a dále koordinuje jejich práci z hlavní programové smyčky. Celkový pohled na architekturu programu je naznačen pomocí diagramu analytického modelu tříd na obr. 4.2.

[2]PSG je fakticky součástí CPU HuC6280, ale jedná se o tak rozsáhlou a specifickou součást, že ji narozdíl od časovače nebo vstupně-výstupního paralelního portu uvádíme separátně.

[3]Uživatelským rozhraním je zde myšlena vrstva využívající služeb platformy a zprostředkovávající vstup a výstup emulátoru uživateli.



Obrázek 4.2: Architektura programu

Kromě možnosti volby z více druhů emulátorů či uživatelských rozhraní, přinese takto navržená architektura, podpořená možnostmi sestavovacího systému, také možnost vytvořit personalizovaná sestavení obsahující jen moduly zvolené na základě potřeb uživatele, nebo restrikcí plynoucích z možností platformy, pro kterou bude výsledné sestavení programu určeno.

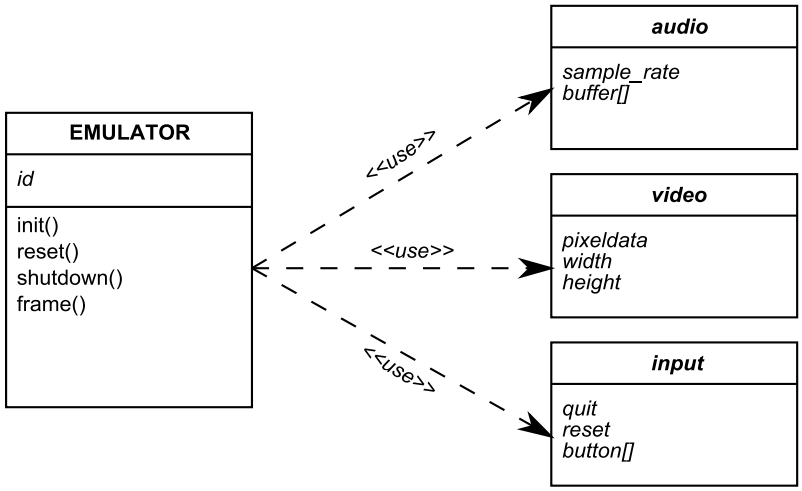
Modularita programu je tak navíc efektivním řešením nefunkčního požadavku na *přenositelnost kódu*. Na platformách, kde nebude možné program sestavit z důvodu např. závislosti na nedostupné knihovně uživatelského rozhraní, stačí doimplementovat specifický modul založený na jiné, dostupné knihovně, který bude „suplovat“ přenositelný kód. Poté stačí zavést příslušná omezení sestavovacího systému pro nucený výběr takového modulu při sestavení pro tuto platformu.

4.4.1 Moduly emulátorů

Emulátory jednotlivých videoherních systémů (v tomto případě NEC PCEngine) jsou oddělenými moduly, komunikující s ostatními částmi programu pomocí pevně specifikovaného rozhraní tvořeného několika funkcemi a datovými strukturami pro výměnu dat. Analytická třída modulu emulátoru je naznačena pomocí diagramu na obr. 4.3.

Veřejné rozhraní modulu emulátoru tvoří, kromě struktur pro výměnu dat s modulem uživatelského rozhraní (viz. podsekce 4.4.3), následující funkce:

- *init()* - inicializace emulátoru (konstruktor)
- *reset()* - re-inicializace emulátoru
- *shutdown()* - ukončení činnosti emulátoru (destruktor)
- *frame()* - posun v emulaci o jeden vykreslený snímek



Obrázek 4.3: Analytická třída modulu emulátoru

Hlavní třídu modulu emulátoru je nutné chápat pouze jako zapouzdření funkčnosti emulátoru. Je pochopitelné, že kód bude rozsáhlý a složitý a bude reflektovat řadu specifik architektury konkrétního systému. Proto je architektura za hranicí tohoto rozhraní zcela ponechána na autorovi konkrétního modulu.

Pro modul emulátoru systému NEC PCEngine je zvolena architektura tvořená čtyřmi třídami, z nichž každá představuje jeden z funkčních bloků systému (CPU, VDC, VCE a PSG), jehož vnitřní stav (registry, namapovaná paměť atd.) reprezentuje a jehož chování emuluje pomocí svých metod.

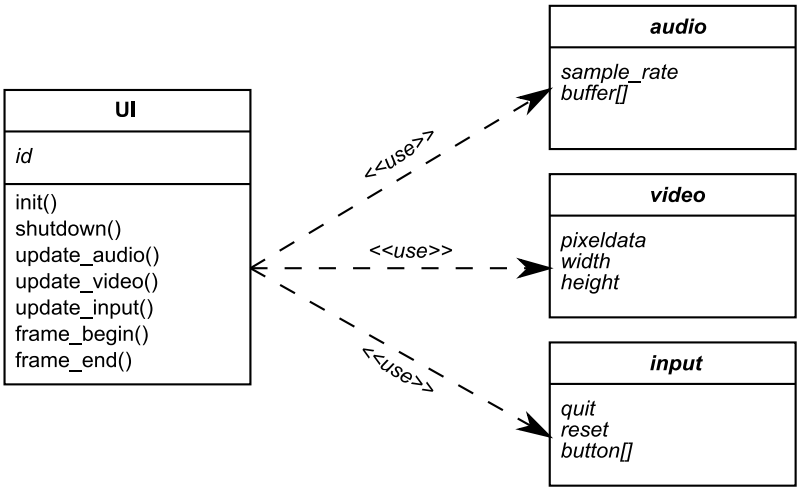
4.4.2 Moduly uživatelských rozhraní

Úkolem modulu uživatelského rozhraní je zajistit interakci mezi uživatelem a modulem emulátoru, resp. v něm běžícího programu. Zpravidla provádí následující tři činnosti:

- zobrazit obrazová data získaná od modulu emulátoru
- přehrát zvuková data získaná od modulu emulátoru
- zpracovat uživatelský vstup a předat jej modulu emulátoru

Požadavek na *přenositelnost kódu* komplikuje potřeba programu používat služby specifické pro určité platformy. Nepochybně největší částí programu závislou na službách platformy je uživatelské rozhraní. I přes to, že existuje řada knihoven a rámců zastřešujících práci s uživatelským rozhraním napříč více platformami, nikdy se nepodaří jednou knihovnou či rámcem pokrýt všechny možnosti. Oddělení kódu uživatelského rozhraní do modulu, jehož analytická třída je naznačena pomocí diagramu na obr. 4.4, je tak logickým krokem, který je v souladu i s použitím zmíněných knihoven a rámců[4].

[4]To je ostatně i případ použité knihovny libSDL (viz. kapitola 5), díky které bude pro obě platformy dané požadavky, tedy GNU/Linux a Microsoft Windows, zapotřebí pouze jeden modul uživatelského rozhraní.



Obrázek 4.4: Analytická třída modulu uživatelského rozhraní

Tímto přístupem navíc docílíme možnosti snadné integrace s různými prostředími. Modul uživatelského rozhraní nemusí být nutně interaktivní, jednou z mnoha možností je např. síťová vrstva, která předává a přijímá informace od vzdáleného klientu. Bez výraznějších úprav tak může výsledný program s tímto modulem operovat v režimu klient/server po síti.

Moduly uživatelského rozhraní budou se zbytkem programu, podobně jako moduly emulátorů, komunikovat pomocí pevně specifikovaného rozhraní, které bude tvořeno, kromě struktur pro výměnu dat s modulem emulátoru (viz. podsekce 4.4.3), následujícími funkcemi:

- *init()* - inicializace uživatelského rozhraní (konstruktor)
- *shutdown()* - ukončení činnosti uživatelského rozhraní (destruktor)
- *update_audio()* - aktualizace zvukového výstupu
- *update_video()* - aktualizace obrazového výstupu
- *update_input()* - aktualizace vstupu
- *frame_begin()* - činnost před vykreslením snímku
- *frame_end()* - činnost po vykreslení snímku

Modul uživatelskéh rozhraní obsahuje synchronizační mechanismus tvořený dvojicí funkcí *frame_begin()* a *frame_end()*. Ten umožňuje provádět kód před zahájením vykreslení snímku a po jeho ukončení.

Protože mají knihovny a rámce použité pro práci s uživatelským rozhraním většinou přímý přístup k platformně závislým službám, jako je např. správa časovačů, je logické umístit mechanismus pro časovou synchronizaci pomocí těchto služeb právě do modulu uživatelského rozhraní. Kód vykonávaný touto dvojicí funkcí může např. měřit délku snímku a kompenzovat počet snímků za sekundu.

4.4.3 Společné struktury pro výměnu dat

Rozhraní obou typů modulů dotváří trojice pomocných struktur určených k výměně dat v pevně daném formátu. To zajišťuje, že si libovolná, správně provedená implementace modulu emulátoru „bude rozumět“ s libovolnou, správně provedenou implementací modulu uživatelského rozhraní a naopak. Jedná se o následující struktury:

- *audio* - pro výměnu informací o zvukovém výstupu (emulátor → uživatelské rozhraní)
- *video* - pro výměnu informací o obrazovém výstupu (emulátor → uživatelské rozhraní)
- *input* - pro výměnu informací o stavu vstupu (uživatelské rozhraní → emulátor)

Kapitola 5

Implementace

Implementace spočívá v převodu navržené architektury do podoby spustitelného kódu. Způsob jakým jsou jednotlivé části programu implementovány je dán volbou programovacího jazyka a knihoven popsanou v sekci 5.1 této kapitoly, ale také použitím specifických implementačních technik a konvencí popsaných v sekci 5.2.

Následující sekce kapitoly rámcově rozebírají základní myšlenky použité při implementaci programového jádra (sekce 5.3), modulu emulátoru systému NEC PCEngine (sekce 5.4) a modulů uživatelských rozhraní libSDL (sekce 5.5) a libSDL s podporou OpenGL (sekce 5.6).

K podrobnějšímu studiu implementace poslouží zdrojový kód programu, který je okomentován a psán tak, aby byl co nejvíce čitelný. Tam kde je to vhodné jsou v kódu (hlavně emulátoru systému NEC PCEngine) uvedeny i části specifikace. Veškeré komentáře v kódu jsou zpracovatelné systémem Doxygen [20] do přehledné programátorské dokumentace.

Kapitolu uzavírá sekce 5.7, která se věnuje použití sestavovacího systému CMake za účelem konfigurace sestavení a sestavení programu.

5.1 Implementační prostředí

Implementace emulátoru je nelehký úkol, který vyžaduje, kromě znalosti architektury emulovaného systému a jeho asembleru, také dobrou znalost programovacího jazyka, ve kterém bude implementace probíhat. Nejvhodnějšími kandidáty z kategorie vyšších programovacích jazyků jsou jazyky C a C++, které disponují datovým typem ukazatel a dovolují přímou práci s pamětí. Zároveň se jedná o nejrozšířenější jazyky kompilované do nativního kódu cílové platformy.

5.1.1 Programovací jazyk

Pro implementaci programu byl z nabízené dvojice programovacích jazyků C a C++ zvolen programovací jazyk C. Toto rozhodnutí je založeno na mých zkušenostech s touto dvojící programovacích jazyků, ze kterých ovládám více právě programovací jazyk C.

Ačkoliv je architektura programu navržena částečně objektově, zdaleka nenachází plné využití možností objektového programování (rozsáhlé abstrakce, složitá hierarchie tříd vyžadující komplexní využití dědičnosti atd.). Možnosti, které z tohoto přístupu k programování

využívá lze bez větších problémů implementovat i v jazyce C při dodržení určitých postupů a konvencí, o kterých podrobně pojednává např. [13].

Programovací jazyk C má navíc několik vlastností, které ulehčí přenositelnost výsledného řešení na případné méně standardní platformy. Jedná se o:

- jazyk C má oproti jazyku C++ méně běhových závislostí, což ho favorizuje v případě programování nízkoúrovňových systémů, vestavěných zařízení apod., na která by program mohl být v budoucnu přenášen.
- jazyk C neprovádí transformaci jmen symbolů, což je velice užitečné při např. při lazení a optimalizacích na úrovni assemblerového kódu programu.
- jazyk C umožňuje, narozdíl od jazyka C++ uložení polí proměnné délky na zásobník, kde je alokace velice rychlá. Na některých platformách by toto mohlo posloužit k uložení kritických míst emulované paměti.

5.1.2 Knihovny a nástroje

K implementaci uživatelského rozhraní je, po vzoru emulátoru Mednafen uvedeného v rešerši existujících řešení (viz. kapitola 3), použita knihovna pro tvorbu uživatelských rozhraní libSDL [24] a knihovna pro akcelerovanou grafiku OpenGL [30]. Kód používající tyto knihovny je zapouzdřen do modulu uživatelského rozhraní, nikoliv použit přímo, a tak může být kdykoliv nahrazen.

Důvodem použití právě těchto knihoven je fakt, že podporují obě, nefunkčními požadavky, vytyčené platformy (tj. GNU/Linux a Microsoft Windows). Nebude tedy nutné vytvářet speciální modul uživatelského rozhraní pro každou z těchto platforem.

Pro konfiguraci sestavení a programu se využívá multiplatformního sestavovacího nástroje CMake [18]. Více o použití tohoto nástroje je uvedeno v sekci 5.7 této kapitoly.

Součástí implementačního prostředí jsou nepochybně i nástroje využité k psaní a lazení testovacích programů pro procesor HuC6280 - tedy emulátor procesoru 6502 - M6502 [25] a assembler pro procesor HuC6280 - MagicKit [27].

5.2 Techniky použité při implementaci

Na základě některých nefunkčních požadavků a použití zvolených nástrojů byly při implementaci použity techniky popsané v následujícím textu.

5.2.1 Objektový přístup a zapouzdření

Objektově orientovaná analýza a návrh software jsou postupy založené na modelování skutečné reality v programu pomocí tzv. „objektů“. Tyto objekty jsou zpravidla tvořeny atributy reprezentujícími stav objektu a metodami, pomocí kterých mohou tento stav ovlivňovat ostatní objekty v modelu. Více o objektově orientovaném programovacím stylu lze nalézt např v. [2].

Analýza a návrh programu byly přirozeně provedeny touto cestou a jednotlivé jeho části jsou tvořeny objekty, které zapouzdřují činnost těchto částí.

Pro implementaci byl zvolen programovací jazyk C, který, jak už bylo uvedeno, nemá přímou podporu metodik objektového programování. Nicméně objektově orientovaná architektura progamu je poměrně jednoduchá a nestaví žádné závážnější překážky implementaci v tomto jazyce. Pro potřeby implementace navržené architektury si vystačíme s následujícími technikami:

třída - je implementována pomocí datové struktury sdružující všechny atributy objektů této třídy a množiny funkcí, které představují metody této třídy.

instance - instance je tvořena ukazatelem na dříve popsanou strukturu třídy, které odpovídá. Vytvoření instance proběhne alokací paměti (pomocí funkce jazyka C *malloc()*) pro tuto strukturu a případně zavoláním funkce, která zinicializuje její prvky (konstruktoru). Zrušení instance pak proběhne jednoduše uvolněním paměti pro strukturu (pomocí funkce jazyka C *free()*), případně jemu předcházejícím zavoláním funkce, která provede např. uložení dat ze struktury na disk před jejím zánikem (destruktoru).

metoda - je funkce, jejímž prvním argumentem je vždy ukazatel na konkrétní instanci objektu. Tato funkce při svém volání ověří existenci (např. pomocí makra jazyka C *assert()*) a vykoná činnost.

atribut - je členská proměnná struktury třídy. Pokud se jedná o ukazatel do paměti, měl by destruktor třídy zajistit uvolnění této paměti, pokud je to žádoucí.

Protože návrh programu nepočítá s tím, že by od některé třídy mělo existovat více instancí, jsou v rámci zdrojového kódu programu instance globální a nepředávají se metodám. Tento přístup může do budoucna znamenat nepřehlednost kódu a proto je považován za chybu, která bude odstraněna.

5.2.2 Přenositelnost kódu

Dalším aspektem úzce spojeným s volbou jazyka C je přenositelnost kódu. Protože se zdrojový kód programů napsaných v jazyce C překládá přímo do strojového kódu, není možné přenášet výsledné binární soubory mezi platformami. Místo toho je nutné pro každou z podporovaných platforem kód přeložit pro ni určeným překladačem jazyka C.

Kromě toho, že je nutné architektonicky oddělit kód závislý na platformě tak, jak je to popsáno v kapitole 4, je také nutné vzít v potaz několik dalších faktorů, které ovlivňují přenositelnost kódu.

5.2.2.1 Standardy jazyka C

I přesto, že je množina základních datových typů, knihovních funkcí apod. v jazyce C předepsaná standardem ANSI C, řada distribucí vývojových nástrojů jazyka C se liší v jeho výkladu, zavádí jiné pojmenování pro datové typy, nebo nedefinuje některá makra.

Z toho důvodu existuje soubor `xtypes.h`, kde jsou definovány jednotlivé datové typy, které by měly být používany v přenositelném kódu. V případě, že nastane potřeba pro specifický překladač nebo platformu definovat typ jinak, bude možné to provést podmíněně na jednom místě.

V některých případech navíc není možné, aby překladač odpovídal standardu[1]. Pro tento případ je i jádro programu rozděleno na dvě části z nichž jedna implementuje na platformě nezávislý kód a druhá kód na platformě závislý.

5.2.2.2 Endianita

Endianita (pořadí bajtů) architektury, na které emulátor běží, má v případě implementace v jazyce C významný dopad na jeho funkčnost. Jak již bylo uvedeno v kapitole 4, paměť emulovaného systému je reprezentována přímo pre-alokovanými bloky paměti, uspořádánými dle endianity této architektury. Emulovaný program ale pracuje s uspořádáním paměti daným endianitou emulovaného procesoru. V případě že se tyto endianity liší, je nutné na kritických místech pořadí bajtů obrátit.

Aby těchto míst v kódu programu vznikalo co nejméně, je v souboru `xtypes.h` zaveden speciální union *pair* představující dvojici slov, jehož definice je následující:

```
typedef union {
#ifdef LSB
    struct { uint8 l,h,h2,h3; } b;
    struct { uint16 l,h; } w;
#else
    struct { uint8 h3,h2,h,l; } b;
    struct { uint16 h,l; } w;
#endif
    uint32 d;
} pair;
```

Pro přístup k celé dvojici slov reprezentované tímto unionem lze použít jeho prvek *d*, pro přístup k hornímu slovu pak prvek *w.h* atd. Interní struktura tohoto unionu je stanovena v okamžiku sestavování programu nastavením makra *LSB*, resp. *MSB*, tak aby bylo v kódu přistupováno k žádanému slovu nebo bajtu.

Jedno z dvojice maker *LSB*, *MSB*, je vždy nastaveno sestavovacím systémem CMake na základě informace o endianitě architektury, pro kterou probíhá sestavení. Podobně jako je použito v případě unionu *pair* ho lze použít na libovolném místě v kódu, kde je nutné zařídit aby data byla správně reprezentována emulovanému programu a nelze tam union *pair* použít (např. kód VDC nebo PSG).

[1] Např. v případě vývojového prostředí devkitPro pro Nintendo DS je nutné aby program místo standardní vstupní funkce *main()* obsahoval dvě vstupní funkce z nichž každá je určena pro jeden z dvojice v systému přítomných procesorů.

5.2.3 Konvence dodržované v kódu

V kódu jsou pro lepší přehlednost a srozumitelnost dodržovány následující konvence:

- názvy identifikátorů jsou tvořeny malými písmeny a podtržítky
- konstruktory tříd jsou nazvány *init*
- destruktory tříd jsou nazvány *shutdown*
- názvy metod jsou prefixovány názvem příslušné třídy
- identifikátory v rámci modulu jsou prefixovány názvem modulu

5.3 Program

Celý program je příznačně nazván *bc_emu*. Skládá se z programového jádra, platformně závislého kódu pro spuštění z příkazové řádky operačního systému osobního počítače a následující trojice modulů:

- emulátoru systému NEC PCEngine (*pce*)
- uživatelského rozhraní libSDL (*sdl*)
- uživatelského rozhraní libSDL s podporou OpenGL (*sdlgl*)

5.3.1 Podpora modulů

Podpora modulů v programu je implementována pomocí ukazatelů na funkce [16]. Při konfiguraci sestavení je systémem CMake vygenerován soubor `bc_emu_modules.c`, který obsahuje dvojici polí struktur modulů (emulátorů a uživatelských rozhraní) obsahujících ukazatele na funkce, které tvoří veřejné rozhraní jednotlivých modulů.

Po sestavení má program tyto dvě pole k dispozici a při spuštění jen provede nastavení globálních ukazatelů podle aktuálně zvolené dvojice modulů. Obecné rozhraní je dotvořeno trojicí globálních společných datových struktur pro výměnu dat mezi moduly, definovaných v souboru `interface.h`.

Tento způsob je plně vyhovující pokud budeme předpokládat statické[2] sestavování výsledného programu a fakt, že nikdy nebude zapotřebí více instancí jednoho typu modulu. Při běžném používaní programu zamýšleným způsobem budou tyto předpoklady vždy splněny. Pokud by nastal důvod sestavit program dynamicky[3], bylo by nutné vhodně rozšířit funkce pro hledání jednotlivých typů modulů v souboru `module.c`. Podpora více instancí modulu bude vyřešena opravením chyby s předáváním ukazatele na instanci popsané v podsekci 5.2.1.

[2] Knihovny jednotlivých modulů jsou přímo připojeny do binárního souboru programu.
[3] Knihovny jednotlivých modulů jsou načítány službou operačního systému při každém spuštění programu.

5.4 Modul emulátoru NEC PCEngine

Modul emulátoru systému NEC PCEngine je nejvýznamnější částí programu. Obsahuje veškerý kód zajišťující emulaci jednotlivých částí tohoto systému. Soubory se zdrojovými kódy modulu jsou umístěny v adresáři `/emu/pce/`.

Nejpodstatnější funkcí celého modulu je bezpochyby funkce *pce_frame()*, která zajišťuje posun o vykreslení jednoho obrazového snímku v emulovaném programu. Standard NTSC má při kmitočtu 60Hz 59.9 prokládaných snímků tvořených 263-mi řádky.

5.4.1 CPU HuC6280

Veškerá emulace činnosti CPU HuC6280 je zapouzdřena do třídy *pce_cpu*. Základ této třídy tvoří soubor `cpu_huc6280.c`, který obsahuje řadu metod pro samotnou práci s procesorem a soubory instrukční sady: `cpu_instr.inc`, `cpu_instr_util.inc` a `cpu_op_tab.inc`. Implementace částečně vychází z emulátoru procesoru M6502 Juergena Buchmuellera.

5.4.1.1 Časování

Emulátor systému NEC PCEngine je naimplementován tak, aby počet procesorových cyklů prováděných při vykreslení jednoho řádku snímku odpovídal skutečnému hardware. Tento počet cyklů lze získat následujícím výpočtem:

$$n_{cykly} = \frac{f_{VDC}}{n_{snimky} * n_{radky}} = \frac{7,16 * 10^6}{59.9 * 263} \approx 455$$

kde f_{VDC} je taktovací frekvence VDC [12], n_{snimky} je počet prokládaných snímků za sekundu dle standardu NTSC a n_{radky} je počet řádků tvořících jeden proložený snímek (čitatel je počet vykreslených řádků za sekundu).

Vzhledem k tomu, že dochází k vykonání přesného počtu procesorových cyklů během vykreslení řádku a tedy i snímku, bude při počtu snímků za sekundu definovaném standardem NTSC rychlost emulace věrohodná. To je stav, který je z hlediska modulu emulátoru vyhovující.

Nicméně pokud není v hlavní programové smyčce, ze které vykreslování jednotlivých snímků probíhá (soubor `bc_emu.c`), provedeno omezení počtu vykreslených snímků na přibližně tuto hodnotu, bude rychlost emulace omezená pouze rychlostí platformy na které emulátor běží, tedy pravděpodobně příliš vysoká.

Odstranění této chyby, která není v kódu emulátoru a nelze jí řešit bez použití služeb knihovny nebo rámce uživatelského rozhraní, případně platformy, je přenecháno dvojici funkcí *frame_begin()* a *frame_end()* modulu uživatelského rozhraní (viz. 5.5.1).

5.4.1.2 Zpracování instrukcí

Stěžejní částí emulace CPU HuC6280 je zpracování instrukcí, které probíhá v rámci metody *pce_cpu_exec()*. Jejím argumentem je počet procesorových cyklů k dispozici (pro jeden snímek 455) a návratovou hodnotou je buď záporné číslo vyjadřující počet chybějících

cyklů (instrukce se vykonávají atomicky, takže byly cykly provedeny navíc), nebo kladné číslo vyjadřující počet cyklů, který přebyl.

Jednotlivé instrukce jsou implementovány jako inline[4] funkce uspořádané do pole, kde indexem je přímo opkód. Definice těchto funkcí lze nalézt v přehledné tabulce v souboru `cpu_op_tab.inc` a mají typicky tvar:

```
OP(016) { int arg1; cycl_n -= 6;    RD_ZPX;         ASL;        WB_EAZ; }
```

Kromě deklarace proměnných představujících operandy instrukce a dekrementace počtu zbývajících cyklů procesoru, tvoří tělo každé funkce implementující instrukci jedno až tři makra.

Makra s dvoupísmenným prefixem (např. `RD_ZPX` nebo `WB_EAZ`) zajišťují přístup do paměti s použitím konkrétního adresního módu. Zdrojový kód těchto maker je umístěn v souboru `cpu_instr_util.inc` a jejich úkolem je spočítat adresu a obsah paměti na této adrese načíst do připravené proměnné jako operand instrukce, případně obsah této proměnné nebo některého z registrů na tuto adresu uložit.

Zbývající makro pak zajišťuje samotnou emulaci instrukce. Zdrojový kód jednotlivých maker instrukcí je umístěn v souboru `cpu_instr.inc`.

Z předchozího popisu je zřejmé, že při implementaci cyklického zpracování instrukcí tímto způsobem stačí v interpretační funkci inkrementovat čítač programu (PC) a zavolat funkci z pole s indexem jeho nové hodnoty.

5.4.1.3 Zpracování přerušení

Implementace emulace CPU HuC6280 bere v potaz zpracování přerušení na všech linkách přerušení. V případě, že během emulace programu nastane přerušení, je na příslušné lince nastaven mód „aserce“. Po zpracování přerušení je linka uvedena zpět do módu „volno“. Stav jednotlivých linek přerušení je uchován pomocí pole *irq_state[]* v rámci struktury třídy *pce_cpu*.

Zpracování přerušení je implementováno pomocí makra *CHECK_IRQ_LINES*, které kontroluje stav jednotlivých linek a v případě, že je některá z nich nastavena na stav „aserce“, zavolá další makro *DO_INTERRUPT* s vektorem příslušného přerušení v argumentu. Zdrojový kód obou maker je umístěn v souboru `cpu_instr_util.inc`.

Kromě běžného ošetření v rámci emulovaného programu (odskok na vektor přerušení a spuštění ošetřující rutiny) je navíc volána funkce zpětného volání, na níž byl nastaven ukazatel pomocí funkce *pce_cpu_set_irq_callback()*. Tato funkce dostává parametrem identifikátor linky, na které došlo k přerušení. Tento mechanismus je výhodný např. pro lazení.

5.4.2 VDC HuC6270 + VCE HuC6260

Emulace zobrazovacího subsystému NEC PCEngine je zajištěna třídami *pce_vdc* a *pce_vce*. Jejich kód nalezneme v souborech `vdc_huc6270.c` a `vce_huc6260.c`. Výstup obrazových dat

[4] Označení pro funkce, které překladač jazyka C překládá tak, že v místě volání funkce nevloží do strojového kódu odskok na rutinu představující kód funkce, ale přímo tento kód

je proveden jejich vykreslením do bitové mapy reprezentované strukturou *t_video*, kterou dále zpracovává modul uživatelského rozhraní.

Vzhledem k tomu, že předmětem práce nebylo studium způsobu zobrazování na televizoru, emulace VDC i VCE postrádá implementaci řady vlastností (barva překryvu v prostoru nevyužité obrazovky, nastavení šířky bodových pulzů atd.), které nijak výrazně neovlivňují funkčnost modulu emulátoru a mohou být doplněny později.

5.4.2.1 VDC HuC6270

Třída *pce_vdc* zajišťuje veškeré vykreslovací operace. Vykreslování výsledného obrazu do bitové mapy je, stejně jako vykreslování na obrazovku televizoru v případě skutečného hardware, prováděno po jednotlivých řádcích. Tento přístup zvyšuje přesnost emulace u programů, které využívají k synchronizaci přerušení vyvolaných před samotnou *vertikální synchronizací* (např. raster compare, nebo kolize sprajtu 0).

Ústřední funkcí třídy *pce_vdc* je *pce_vdc_render_line()*, jejímž argumentem je číslo právě vykreslovaného řádku. Podle jednotlivých povolených rovin deleguje tato funkce činnost vykreslování řádku na funkce *pce_vdc_render_bp()*, která provádí vykreslování dlaždic pozadí, a *pce_vdc_render_sp()*, která provádí vykreslování sprajtů. Obě tyto funkce pracují s emulovanými atributovými tabulkami *BAT* a *SAT*. V případě sprajtů, jejichž vykreslování je ovlivněno řadou atributů v tabulce *SAT* jsou jednotlivé záznamy reprezentovány strukturou *t_pce_vdc_sprite*.

Pro urychlení a zpřehlednění vykreslovacího kódu se namísto vykreslování vzorů dlaždic pozadí a sprajtů přímo z videopaměti používá dvojice vyrovnávacích pamětí, kde jsou jednotlivé vzory předvykresleny a uloženy. Aby nedocházelo ke zbytečnému přegenerovávání obsahu obou vyrovnávacích pamětí při každém vykreslování, jsou záznamy „znečišťovány“ při přístupu do odpovídající části videopaměti a regenerační algoritmus obnovuje jen tyto „znečištěné“ záznamy. Regerenerace obou vyrovnávacích pamětí je zajištěna funkcemi *pce_vdc_cache_bp()* a *pce_vdc_cache_sp()*.

5.4.2.2 VCE HuC6260

Třída *pce_vce* zajišťuje správu barevných informací při vykreslování výsledného obrazu třídou *pce_vdc*.

Kromě emulace jednotlivých registrů skutečného VCE třída spravuje dvě vyhledávací tabulky *pixel_lut* a *bp_lut*. První z těchto tabulek umožňuje dohledávat složkově definovanou barevnou informaci (potřebnou k vykreslení pixelu do bitové mapy) pomocí indexu barvy VCE, a druhá pak slouží k dohledání správné bitové roviny podle indexu barvy v planárním způsobu uložení obrazových dat.

Formát pixelu[5] v tabulce *pixel_lut* je v současné době pevně dán tak, aby se shodoval s tím, který používá knihovna libSDL. Z hlediska použití různých knihoven pro uživatelské rozhraní by bylo výhodnější umožnit nastavení formátu pixelu v rámci třídy *vce* na takový, který použitá knihovna očekává. Vzhledem k současné implementaci může na platformách, sice podporovaných knihovnou libSDL, ale s jiným formátem pixelu, dojít k barevné deformaci výstupního obrazu.

[5]Formát pixelu udává které bity v rámci reprezentace pixelu vyjadřují kterou barevnou složku.

5.4.3 PSG

Vzhledem k možnostem PSG systému NEC PCEngine je emulace zvukového výstupu poměrně složitý problém. Základní implementace v souboru `psg.c` nezohledňuje řadu možností které PSG nabízí.

Emulace zvukového výstupu je implementována funkcí *pce_psg_fill()*, jejímž úkolem je naplnit vyrovnávací paměť pro jednotlivé zvukové kanály daty, která vzniknou na základě smíšení emulovaných kanálů PSG. To je provedeno tak, že pro každý sampl výsledného zvukového jsou iterativně zpracovány všechny emulované kanály PSG v příslušném samplu. Takto naplněná vyrovnávací pamět je předána v rámci struktury *t_audio* modulu uživatelského rozhraní.

Současná implementace PSG nepodporuje přímý přístup k datům (režim DDA) a přespříliš neřeší synchronizaci a časování zvuku. Vyrovnávací paměť zvukového výstupu je plněna vždy po vykreslení snímku (při vertikální synchronizaci), což odpovídá způsobu zpracování zvuku většinou herních programů.

Formát výstupního zvuku[6] je opět pevně nastaven na stejný, jaký používá knihovna libSDL a z hlediska použití jiných knihoven by měla třída *psg* umožňovat jeho nastavení v souladu s očekáváním použité knihovny.

5.4.4 Parser obrazů ROM

Implementace přenositelného parseru obrazů ROM je provedena tak, že namísto práce se souborem, se pracuje s ukazatelem na oblast paměti, kde jsou ve struktuře uložena data, o kterých se celý program domnívá, že jsou obrazem paměti ROM. Jak se taková data do příslušné struktury dostanou je pak problém, který řeší jádro. Toto řešení bere v potaz situaci, kdy by program mohl být portován na architekturu, kde neexistuje souborový systém a data uložená v ROM jsou reprezentována např. pouze ukazatelem do paměti.

Samotný parser obrazů ROM přečte během inicializace modulu emulátoru obsah ROM a na základě velikosti předané ve struktuře *t_rom* společně s daty provede analýzu a nutné kroky jako je oddělení hlavičky nebo rozdělení obrazu. Tyto kroky jsou popsány v kap. 2.

5.5 Modul uživatelského rozhraní libSDL

Modul uživatelského rozhraní libSDL umožňuje uživatelskou interakci s modulem emulátoru. Zpracovává uživatelský vstup, poskytuje obrazový a zvukový výstup a zajišťuje omezení počtu vykreslených snímků. Na jeho vývoj nebyl kladen přílišný důraz, protože se jedná o modul implementující pouze základní funkčnost uživatelského rozhraní pro potřeby demonstrace výsledků dosažených při implementaci modulu emulátoru NEC PCEngine. Soubory se zdrojovými kódy tohoto modulu se nacházejí v adresáři `/ui/sdl`.

Tento modul je tvořen jedinnou třídou *pce_sdl*, která obsahuje funkce pro zpracování dat ve strukturách *t_audio*, *t_video* a *t_input* a dvojici funkcí *sdl_frame_begin()* a *sdl_frame_end()* pro vykonání kódu před a po vykreslení každého snímku modulem emulátoru.

[6] Formát je u digitálního zvuku tvořen zpravidla délkou jednotlivých samplů, pořadím bajtů, frekvencí a přesností a znaménkem čísla reprezentujícího sampl.

5.5.1 Omezení počtu vykreslených snímků

Jak již bylo uvedeno dříve (viz. 5.4.1.1), modul emulátoru systému NEC PCEngine je naimplementován tak, aby dosahoval věrohodné rychlosti při vykreslení cca. 60-ti snímků během jedné sekundy (což přibližně odpovídá počtu proložených snímků za sekundu dle standardu NTSC).

Omezení počtu vykreslených snímků je provedeno pomocí dvojice atributů v rámci třídy *sdl*. Tím prvním je počet „tiků“ před zahájením vykreslování snímku *ticks_odl*, tím druhým pak počet „tiků“ po dokončení vykreslování. Tyto hodnoty jsou nastaveny funkcemi *sdl_frame_begin()* a *sdl_frame_end()* při každém vykreslování snímku. Při volání funkce poslední zmíněné funkce jsou zároveň tyto hodnoty odečteny a pokud v rámci vykresleného snímku zbývá čas do $\frac{1}{60}$ *sekundy*, jednoduše se o tento čas prodlí vykreslení následujícího snímku.

5.5.2 Interakce s uživatelem

Propojení a způsob práce se vstupními a výstupními zařízeními je provedeno pevně ve zdrojovém kódu programu a uživatel nemá možnost ho modifikovat. V současné implementaci je výstupní rozlišení programu 640x480 pixelů v barevné hloubce 16 bitů, což je nastavení pokrývající potřeby většiny herních programů. Obraz není deformován na velikost okna, ale vykreslován od pravého horního rohu do velikosti určené aktuálním rozlišením VDC. Tlačítka herního ovladače a ovládaní běhu programu jsou mapována na tlačítka klávesnice podle tabulky 5.1.

Klávesa	**Význam**
`<nahoru>`	směrový kříž herního ovladače, směr nahoru
`<dolů>`	směrový kříž herního ovladače, směr dolů
`<vlevo>`	směrový kříž herního ovladače, směr vlevo
`<vpravo>`	směrový kříž herního ovladače, směr vpravo
`<enter>`	tlačítko „Start“ herního ovladače
`a`	tlačítko „I.“ herního ovladače
`s`	tlačítko „II.“ herního ovladače
`r`	restart emulace
`q`	ukončení programu

Tabulka 5.1: Mapování kláves v modulu uživatelského rozhraní libSDL

Na obr. 5.1 je zobrazena sejmutá obrazovka s výstupem emulace herního programu *Skweek* v průběhu emulace.

5.6 Modul uživatelského rozhraní libSDL s podporou OpenGL

Modul uživatelského rozhraní libSDL s podporou OpenGL byl naimplementován pro odlazení a ověření funkčnosti programu s více moduly. Jeho parametry i používání jsou naprosto

identické s původním modulem uživatelského rozhraní libSDL. Jediným rozdílem je, v případě tohoto modulu, použití akcelerovaného rozhraní OpenGL pro zobrazování obrazových dat.

Obsah struktury *t_video* je použit jako textura roviny vykreslené přes celé okno programu (640x480 pixelů) s efektem rozmazání. Snímek obrazovky s výstupem emulace herního programu *Pc Genjin 2* je zobrazen na obr. 5.2.

5.7 Sestavovací systém CMake

Jedním z nefunkčních požadavků na program je *přenositelnost kódu*, s čímž je neodmyslitelně spojen i překlad. Tento úkol velice usnadňuje použitý sestavovací systém CMake [18] vyvinutý společností Kitware.

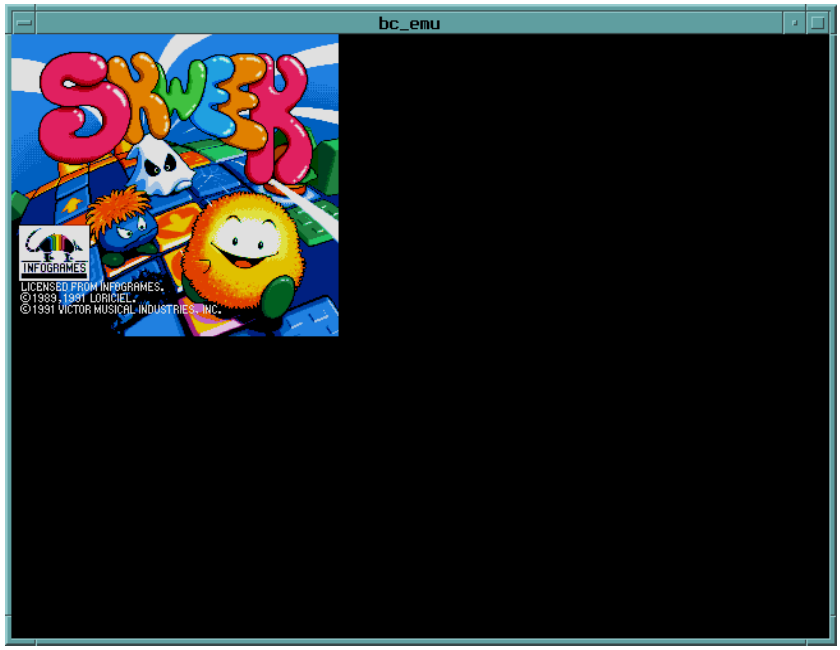
CMake slouží jako prostředník při překladu zdrojového kódu (napsaného hlavně v jazycích C a C++). Na základě platformně nezávislého definičního souboru `CMakeLists.txt` generuje platformně závislou konfiguraci pro překlad a sestavení programu. Program tedy může být přeložen běžně dostupnými nástroji příslušné platformy. CMake podporuje nejběžnější platformy a kromě definičních souborů `Makefile` dokáže generovat i projektové soubory pro nejpoužívanější vývojová prostředí Eclipse, KDevelop nebo Microsoft Visual Studio.

Díky možnosti používání proměnných a testů v rámci zmíněného definičního souboru `CMakeLists.txt` je možné vytvořit sestavovací prostředí umožňující jak zavedení platformních restrikcí (sestavení a připojení některého z modulů jen na určité platformě), tak možnost personifikace sestavení začleněním pouze určitých modulů zvolených uživatelem.

Sestavení programu je ovlivněno řadou proměnných systému CMake, z nichž většinu nastavují testy tímto systémem prováděné. V tabulce 5.2 je pro úplnost uveden seznam nejdůležitějších uživatelských proměnných ovlivňujících sestavení programu včetně jejich významu.

Proměnná	**Význam**	**Hodnota**
`DEBUG`	sestavení vhodné pro lazení	`OFF`
`ARCH`	cílová architektura sestavení (v současné době jen „pc“)	`pc`
`EMU_PCE`	začlenění modulu emulátoru NEC PCEngine	`ON`
`UI_SDL`	začlenění modulu uživ. rozhraní libSDL	`ON`
`UI_SDLGL`	začlenění modulu uživ. rozhraní libSDL s podporou OpenGL	`ON`

Tabulka 5.2: Uživatelské proměnné pro konfigurace sestavení pomocí CMake



Obrázek 5.1: Obrazovka programu, modul uživatelského rozhraní „sdl“

Obrázek 5.2: Obrazovka programu, modul uživatelského rozhraní „sdlgl“

Kapitola 6

Testování a lazení

Emulace videoherního systému je složitý proces sestávající z napodobování řady funkčních součástí tohoto systému, včetně replikace jejich nedokonalostí, nebo dokonce, chyb. Dvojnásobně toto platí o emulátorech videoherních konzolí, kdy ve většině případů není dostupná žádná oficiální specifikace hardwaru a neoficiální se často liší od skutečnosti, je mylná nebo vůbec neexistuje.

Od samotného počátku implementace vede k prvním viditelným výsledkům často dlouhá cesta, na níž se nelze obejít bez důkladného odlazení a otestování jednotlivých částí kódu, které jsou stavebními kameny výsledného programu. Vzhledem ke způsobu práce emulátoru mohou i malé chyby snadno způsobit řetězovou reakci ústící v neočekávané chování na místě, kde chyba vůbec nenastala. Není třeba zmiňovat, že takové chyby se hledají nejhůře.

Tato kapitola v sekci 6.1 stručně popisuje metody jakými byl lazen a testován vyvíjený program a v sekci 6.2 shrnuje výsledky testování současné implementace.

6.1 Prostředky a způsob lazení

Interpretační způsob implementace emulátoru výrazně usnadňuje lazení programu tím, že jsou všechny části systému představovány datovými strukturami reprezentujícími jejich vnitřní stav (obsah interní paměti, registrů apod.) a kód emulovaného programu je zpracováván v iteracích hlavní programové smyčky.

V průběhu vývoje je tak možné „sledovat“ dění uvnitř emulovaného systému pomocí běžných ladících nástrojů umožňujících krokování a sledování obsahu proměnných, jakým je například debugger GDB [23].

Kromě toho je možné využít některých ladících mechanismů implementovaných přímo v programu. Jedním z nich je například funkce zpětného volání implementovaná při obsluze přerušení.

Veškeré důležité informace lze v každém kroku programu vypsat pomocí ladícího makra *debug()*, jehož použití je podmíněno sestavením s aktivovanou proměnnou `DEBUG` sestavovacího systému CMake. Aktivace této proměnné navíc způsobí, že program bude na standardní chybový výstup vypisovat řadu informací z důležitých míst v kódu (např. upozornění na provedení neošetřeného přístupu do paměti, pokus o vykonání ilegální instrukce atd.)

K pohodlnosti lazení také přispívá modulární architektura programu, která dovoluje jednotlivé moduly, nebo jejich části oddělit a ladit zvlášť. Tento způsob byl využit zejména při implementaci emulace CPU HuC6280, která je z hlediska funkčnosti nejdůležitější.

I přes všechny tyto skutečnosti může být lazení programu někdy problematické a hodilo by se mít integrovaný ladící nástroj, který by umožňoval alespoň sledování registrů a obsahu paměti během krokování emulovaného programu. Implementace takového nástroje dostatečně obecného vzhledem k architektuře a vizi dalšího rozšiřování programu o nové moduly emulátorů je rozhodně nad rámec této práce.

6.2 Testování

V rané fázi vývoje probíhalo testování (hlavně implementace CPU HuC6280) pomocí miniaturních assemblerových programů přeložených pomocí software MagicKit [27] a porovnání výsledků proti běhu stejného programu v mírně modifikovaném emulátoru M6502 [25].

Další testy byly provedeny sadou ROM obrazů pořízených z paměti čipových karet HuCard speciálním zařízením PCE Pro 32M. To umožňuje záznam obrazu ROM vloženého modulu HuCard pomocí rozhraní USB do binárního souboru.

Toto testování proběhlo na řadě odlišných obrazů ROM ve velikostech od 257 KB do 1 MB, kdy byl porovnán průběh programu v programu *bc_emu* proti průběhu téhož programu monitorovanému ladícím nástrojem emulátoru Mednafen [28].

V průběhu testování pomocí obrazů ROM původních herních programů byla objevena a opravena řada chyb a nesrovnalostí v emulaci CPU HuC6280. Během provádění oprav byly na řadu citlivých míst (zápisy do paměti, vstupně-výstupní stránky apod.) přidány ladící hlášky, které umožňují sledovat podezřelé zápisy, nebo čtení z neexistujících adres a určit tak příčinu nefunkčnosti programu.

Stále problematickou oblastí je emulace VDC a VCE. I přes to, že byla řada chyb opravena, je spouštění některých herních programů poznamenáno chybami v obrazovém výstupu (artefakty, chybějící sprajty). Řada těchto chyb je způsobena, z hlediska specifikace, nekorektním chováním programu (např. herní program *Pc Genjin 2* se snaží do barevných palet zapisovat hodnoty barev delší než $3 * 3$ bajty, což bylo třeba ošetřit použitím pouze spodních 9-ti bajtů apod.).

I přes uvedené skutečnosti je v případě velké části obrazů ROM, které bylo možné otestovat, emulátor plně funkční a herní programy se dají bez větších problémů používat, např.: *Doraemon Meikyu Daisakusen*, *Makai Prince Dorabo Chan*, *Pc Genjin 2*, *Son Son*, *Xevious* nebo *F1 Circus*.

Za funkční jsou považovány i programy které trpí některou z následujících kosmetických vad, vyplývajích z částečné, nebo zcela zanedbané implementace nekritických částí specifikace. Tyto chyby lze lehce opravit v krátkém čase:

- Řada herních programů (např. *Mesopotamia*) využívá DDA režim PSG. Ten není v této implementaci programu *bc_emu* podporován, takže emulace postrádá zvukový výstup.
- Některé herní programy (např. *Shinobi*) používají efekt paralaxního skrolování obrazu implementovaný pomocí registru BXR pomocného grafického procesoru VDC. Práce

s tímto registrem je naimplementována chybně a proto je obrazový výstup programu nesprávný.

- Některé herní programy využívají kolizi sprajtu 0, která v současné době není implementována (lze pozorovat např. v pokročilejší fázi programu *Doraemon Nobita no Dorabitan Night*).
- Řada herních programů využívá k příznaku priority sprajtu (SPBG), která v současné době není implementována (např. *Soukoban World*).
- Některé herní programy využívají instrukci SET, jejíž implementace je v některých případech nefunkční.

Vzhledem k tomu, že vývoj programu proběhl za dobu uplynulých dvou semestrů a podporovaná množina funkcí videoherního systému NEC PCEngine je oproti existujícím emulátorům velmi malá, nemá smysl porovnávat řešení z hlediska výkonu, nebo právě podpory jednotlivých funkcí a rozšíření systému NEC PCEngine.

6.2.1 Platformy

Překlad a testování programu proběhlo s výše uvedenými výsledky na všech platformách, kde byla požadována funkčnost programu:

- GNU/Linux, distribuce Debian a RedHat Enterprise Linux 5 (x86-64 i i386)
- Microsoft Windows XP SP3 (i386)

Navíc bylo ověřeno, že program je po drobných úpravách možné přeložit ve vývojovém prostředí devkitPro [19] pro videoherní handheld Nintendo DS. Tato platforma není podporována knihovnou SDL, takže bez implementace specifického modulu uživatelské rozhraní nebylo možné skutečně ověřit funkčnost.

K překladu bylo ve všech případech využito překladačů ze sady GCC [22], nebo klonů na nich založených. Vzhledem k tomu, že je program psán tak, aby byla dodržena norma ANSI C, měl by být zaručen i překlad (vyžadující maximálně kosmetické úpravy) jinými překladači implementujícími tuto normu.

Kapitola 7

Závěr

Na základě dostupných informací byla nastudována architektura videoherního systému NEC PCEngine a získané poznatky byly shrnuty do ucelené technické specifikace uvedené v kapitole 2. Ta posloužila jako základ pro analýzu, návrh a následnou implementaci emulátoru tohoto systému.

S ohledem na modularitu ve smyslu možnosti rozšíření o podporu dalších videoherních systémů, nebo uživatelských rozhraní, byl naimplementován emulátor systému NEC PCEngine, jehož funkčnost byla úspěšně ověřena pomocí řady původních herních programů.

Emulátor zahrnuje alespoň základní podporu všech částí základní varianty systému NEC PCEngine a interpretačním způsobem umožňuje spouštění a provádění herních programů uložených v souborech představujících obsah paměti ROM původních čipových karet HuCard.

Zvolená architektura programu odděluje veškerou logiku emulace a interakce s uživatelem do diskrétních modulů s pevně definovaným rozhraním, čímž umožňuje nejen snadné rozšíření o podporu emulace dalších videoherních systému, ale i zjednodušení přenositelnosti zapouzdřením kódu závislého na platformě.

Vzhledem k tomu, že program splňuje všechny zadáním specifikované požadavky a nesrovnalosti, které se projevují při provádění emulovaných programů, jsou buď kosmetické a lze je bez výraznějších zásahů do architektury programu odstranit, nebo jsou způsobeny použitím nestandartních technik v rámci těchto programů a je třeba je řešit individuálně, lze prohlásit, že cíl této práce byl splněn.

7.1 Další vývoj

Díky modulární architektuře umožňující implementaci podpory dalších videoherních systémů a uživatelských rozhraní jsou možnosti rozšiřování téměř nekonečné. Před samotným rozšiřováním tímto směrem by však mělo být zváženo několik zásadnějších zásahů do architektury programu.

Do programu by mělo být přidáno obecné rozhraní umožňující modulům dotazovat se na hodnoty konfiguračních klíčů (např. moduly uživatelského rozhraní by měly mít možnost konfigurce vstupních zařízení, rozlišení obrazovky, formátu zvuku apod.).

Dalším možným rozšířením je implementace nového typu modulu - ladícího nástroje pro emulovaný program, který by uměl na obecné úrovni (samozřejmě by autor modulu emulátoru musel provést nastavení) sledovat registry a paměťové oblasti emulovaného systému, případně krokovat program.

Implementace ladícího nástroje by jistě pomohla při dalším zdokonalování modulu emulátoru systému NEC PCEngine. Kromě odstranění drobných chyb a doimplementace chybějících, v kapitole 6.2 zmíněných, nekritických částí specifikace, by bylo vhodné zdokonalit kód PSG a přidat podporu pro některá populární rozšíření.

Po zásadnějších úpravách a stabilizaci architektury může být zajímavou výzvou přenést program *bc_emu* na videoherní handheld Nintendo DS. S drobnými úpravami je už nyní možné pro tento systém přeložit jádro programu. Dalším krokem je nastudování programového rozhraní knihoven devkitPro a implementace modulu uživatelského rozhraní. Vzhledem k výkonu procesoru tohoto systému mohou nastat potíže s výkonem. Dalším směrem vývoje proto může být optimalizace emulačního kódu při zachování přenositelnosti.

Literatura

[1] *Guinness World Records Gamer's Edition 2008*. Guinness World Records Limited, 2008.

[2] Grady Booch. *Object Oriented Analysis and Design*. The Benjamin Cummings Publishing, 1994.

[3] Daniel Boris. How Do I Write an Emulator? [online], 1999. `http://www.atarihq.com/danb/files/emu_vol1.txt`.

[4] Paul Clifford. PC Engine Programmable Sound Generator. [online]. `http://www.magicengine.com/mkit/doc_hard_psg.html`.

[5] Victor Moya del Barrio. Study of the Techniques for Emulation Programming. Master's thesis, 2001.

[6] Donald Ervin Knuth. *Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison Wesley, 3rd edition, 1997.

[7] Andre LaMothe. *The Black Art of Video Game Console Design*. Sams, 1st edition, 2005.

[8] Charles MacDonald. TurboGrafx-16 Hardware Notes. [online], 2002. `http://cgfm2.emuviews.com/txt/pcetech.txt`.

[9] Bill Loguidice Matt Barton. *Vintage Games: An Insider Look at the History of Grand Theft Auto, Super Mario, and the Most Influential Games of All Time*. Focal Press, 2009.

[10] Mark Ormston. 65xx Processor Data. [online], 2006. `http://www.romhacking.net/docs/65xx-Processor-Data.zip`.

[11] Eric Steven Raymond. *Umění programování v UNIXu*. Computer Press, 2004.

[12] Emanuel Schleussinger. PC-Engine Video Display Controller Documentation. [online], 1998. `http://www.classicgaming.com/aec/`.

[13] Axel-Tobias Schreiner. *Object-Oriented Programming With ANSI-C*. Hanser Fachbuch, Germany, 1994.

[14] Vladimír Vít. *Televizní technika 3A - přenosové barevné soustavy*. BEN - Technická literatura, 2002.

[15] 65c02 processor reference pages. [online].
http://www.obelisk.demon.co.uk/65C02/.

[16] C, C++ Function Pointer. [online].
http://newty.de/fpt/index.html.

[17] ClassicGaming Museum. [online].
http://classicgaming.gamespy.com/View.php?view=ConsoleMuseum.Detail&id=32.

[18] Kitware CMake. [online].
http://www.cmake.org/.

[19] devkitPro. [online].
http://www.devkitpro.org/.

[20] Doxygen - Source code documentation generator tool. [online].
http://www.stack.nl/~dimitri/doxygen/.

[21] The Emulator Zone - PC Engine Emulators. [online].
http://www.emulator-zone.com/doc.php/pcengine/.

[22] GCC - The GNU Compiler Collection. [online].
http://gcc.gnu.org/.

[23] GDB - The GNU Project Debbuger. [online].
http://www.gnu.org/software/gdb/.

[24] SDL library. [online].
http://www.libsdl.org/.

[25] M6502. [online].
http://fms.komkon.org/EMUL8/.

[26] MagicEngine. [online].
http://www.magicengine.com/.

[27] MagicKit assembler. [online].
http://www.magicengine.com/mkit/.

[28] Mednafen. [online].
http://mednafen.sourceforge.net/.

[29] Ootake. [online].
http://www.ouma.jp/ootake/.

[30] OpenGL - The Industry's Foundation for High Performance Graphics. [online].
http://www.opengl.org/.

[31] PCE Catalog Project. [online].
http://pcecp.com/.

[32] Sprite (computer graphics), from Wikipedia, the free encyclopedia. [online]. `http://en.wikipedia.org/wiki/Sprite_%28computer_graphics%29`.

[33] TurboGrafx-16, from Wikipedia, the free encyclopedia. [online]. `http://en.wikipedia.org/wiki/TurboGrafx-16`.

[34] David Woodford. Pce/tg resources mini-faq. [online], 1994. `http://www-personal.umich.edu/~dfw/hackfaq`.

Dodatek A

Seznam použitých zkratek

ANSI American National Standards Institute

BAT Background Attribute Table

CPU Central Processing Unit

DDA Direct Data Access

DMA Direct Memory Access

I/O Input/Output

IRQ Interrupt Request

LFO Low Frequency Oscillator

NEC Nippon Electronic

NMI Non-Maskable Interrupt

NTSC National Television System Committee

PAL Phase Alternating Line

PSG Programmable Sound Generator

RAM Random Access Memory

RF Radio Frequency

ROM Read Only Memory

SAT Sprite Attribute Table

VCE Video Color Encoder

VDC Video Display Controller

VRAM Video Random Access Memory

WDC Western Digital Company

Dodatek B

Uživatelská příručka

Text tohoto dodatku popisuje způsob sestavení, spuštění a ovládání programu *bc_emu*.

B.1 Sestavení programu

Sestavení programu není nutné provádět na platformách GNU/Linux (i386 a x86-64 s aktuální verzí knihovny glibc) a Microsoft Windows. Pro tyto platformy jsou v příslušných adresářích na přiloženém disku DVD (viz. dodatek C) připraveny binární soubory, které stačí spustit.

Pro sestavení je nutné mít v systému nainstalovány následující prerekvizity: *CMake 2.8*, *překladač jazyka C (gcc, mingw-gcc)*, *libSDL1.2-devel*, *make*. Samotný postup sestavení je následující:

1. zkopírování adresáře `src` z přiloženého DVD na pevný disk
2. vytvoření adresáře sestavení `build`
3. konfigurace sestavení souborem `src/CMakeLists.txt` do adresáře `build` pomocí sestavovacího systému CMake (viz. [18])
4. spuštění programu make v adresáři `build`

B.2 Spuštění programu

Program *bc_emu* se spouští pomocí příkazové řádky s přepínači v krátké UNIXové notaci (např. `-h`). Pro úspěšné spuštění emulace musí být programu kromě jména obrazu ROM předána ještě informace o tom jaký modul emulátoru (`-e`) a uživatelského rozhraní (`-u`) má být použit.

Pro demonstraci veškerých schopností programu *bc_emu* stačí program spustit jedním z této dvojice příkazů:

```
bc_emu -e pce -u sdl obraz_rom.pce
bc_emu -e pce -u sdlgl obraz_rom.pce
```

První spustí emulaci se zobrazováním výstupního obrazu pomocí knihovny libSDL, druhý pomocí knihovny OpenGL.

V případě použití dodaného binárního souboru, zkompilovaného překladačem MinGW GCC pro Microsoft Windows, jsou standardní i chybový výstup přesměrovány do souborů `stderr.txt` a `stdout.txt` vytvořených v cestě odkud byl program spuštěn. Proto i ladící verze programu nevypisuje žádné hlášky do konzolového okna.

B.3 Ovládání programu

V rámci obou modulů uživatelského rozhraní (*sdl* i *sdlgl*) lze použít k ovládání programu klávesy uvedené v tabulce B.1 (tabulka je shodná s tabulkou 5.1 a je opakovaně uvedena pro větší přehlednost této příručky).

Klávesa	**Význam**
`<nahoru>`	směrový kříž herního ovladače, směr nahoru
`<dolů>`	směrový kříž herního ovladače, směr dolů
`<vlevo>`	směrový kříž herního ovladače, směr vlevo
`<vpravo>`	směrový kříž herního ovladače, směr vpravo
`<enter>`	tlačítko „Start“ herního ovladače
`a`	tlačítko „I.“ herního ovladače
`s`	tlačítko „II.“ herního ovladače
`r`	restart emulace
`q`	ukončení programu

Tabulka B.1: Mapování kláves

Dodatek C

Obsah přiloženého DVD

Na přiloženém DVD jsou umístěny následující adresáře:

`bin_linux_i386`	binární soubory pro GNU/Linux (i386)
`bin_linux_x86-64`	binární soubory pro GNU/Linux (x86-64)
`bin_win32`	binární soubory pro Microsoft Windows (32bit)
`rom_com`	obrazy ROM (komerční)
`rom_free`	obrazy ROM (volně dostupné)
`src`	zdrojový kód programu *bc_emu*
`src_doc`	dokumentace ke zdrojovému kódu (HTML)
`thesis`	soubory PDF s tímto textem (tisk, online)
`thesis_src`	zdrojový kód tohoto textu pro systém LaTeX