

---

# EMUSTUDIO

POUŽÍVATEĽSKÝ MANUÁL  
**0.37B**

Peter Jakubčo

© 2006-2010 Peter Jakubčo

# Obsah

<b>Obsah</b>	iii
<b>Predhovor</b>	1
<b>Inštalácia programu</b>	3
<b>1 Štruktúra platformy</b>	5
<b>I Hlavný modul</b>	7
<b>2 Prehľad</b>	9
2.1 Panel „Source code“ . . . . .	10
2.2 Panel „Emulator“ . . . . .	10
2.3 Základný postup, ako využívať a emulovať . . . . .	11
<b>3 Konfigurácie a virtuálne architektúry</b>	15
3.1 Voľba konfigurácie . . . . .	15
3.2 Vytváranie a editácia konfigurácií . . . . .	17
3.2.1 Vytváranie nových konfigurácií . . . . .	17
3.2.2 Editovanie konfigurácie . . . . .	21
<b>4 Textový editor</b>	23
4.1 Práca v textovom editore . . . . .	23
4.2 Vyhladávanie, nahradzanie textu . . . . .	24
<b>5 Emulátor</b>	27
5.1 Prehľad aktuálnej konfigurácie . . . . .	27
5.2 Okno debuggera . . . . .	27
5.2.1 Stránkovanie inštrukcií . . . . .	28
5.2.2 Ovládanie emulácie . . . . .	29
5.2.3 Breakpointy na neviditeľnej adrese . . . . .	31
5.3 Stavové okno . . . . .	32
5.4 Okno zariadení . . . . .	33

<b>II Virtuálne architektúry</b>	<b>35</b>
<b>6 Počítač MITS Altair</b>	<b>37</b>
6.1 Schéma pre emulátor . . . . .	38
6.2 Zavádzanie softvéru z obrazov diskiet . . . . .	38
6.2.1 Operačný systém CP/M v2.2 . . . . .	39
6.2.2 Operačný systém CP/M v3 . . . . .	41
6.2.3 Operačný systém Altair DOS v1.0 . . . . .	41
6.2.4 Altair Basic v4.1 . . . . .	42
<b>7 Abstraktný stroj RAM</b>	<b>45</b>
7.1 Schéma pre emulátor . . . . .	45
<b>Literatúra</b>	<b>47</b>

# Predhovor

V tejto príručke nájdete popis emulačnej platformy emuStudio. Ide o softvér, ktorý umožňuje napodobenie správania a štruktúry iného počítača. Hovoríme vlastne o počítačovej emulácii, čiže platforma emuStudio je emulátor.

Existuje mnoho emulátorov, čím je však tento výnimočný? Emulátory sa líšia v prvom rade účelom použitia. Existujú emulátory určené ako podporný softvér pre spustenie hier napísaných pre inú platformu (napr. Sony Playstation). Existujú tiež emulátory, snažiace sa zachovať akúsi "fungujúcu spomienku" na dnes už nedostupné historické počítače (napr. IBM System/360). V súčasnosti sa azda najviac uplatňujú emulátory mobilných telefónov, ktoré používajú vývojári mobilných aplikácií. Namiesto neustáleho prenášania skompilovaného kódu mobilnej aplikácie do telefónu, si ju programátori môžu najprv odskúšať na emulátore. Urýchľuje sa tým čas vývoja, dnes tak dôležitý.

Účel platformy emuStudio sa však mierne lísi od ostatných spomenutých, aj keď ich nevylučuje ako nemožné. Dá sa povedať, že v úplnej všeobecnosti si účel tejto platformy vyberie každý sám. Ale predsa existuje východisko, ktorým bol vývoj platformy sprevádzaný. Účel sa akosi vynoril sám - mala na to vplyv stále rovnako namáhavá dlhorčná snaha učiteľov naučiť niečo, s čím ani oni sami často nemali skúsenosti a tiež ľažké mordovanie študentov, aby pochopili niečo, čo neexistovalo.

Platforma emuStudio je zrealizovanou snahou premeniť niečo abstraktné na niečo jednoduché a ľahko pochopiteľné. Dnes už niekoľko rokov úspešne sprevádzá študentov na ceste k chápaniu základov programovania v assembléri zatiaľ pre staršie 8-bitové počítače, ako je napr. slávny MITS Altair8800.

Na príklade bolo ukázané, že je možné v platforme emulovať aj abstraktné stroje, ktoré boli stredobodom záujmu v počiatočných érach vzniku skutočných počítačov. Dnes sa tieto stroje študujú hlavne v teoretickej informatike za účelom výpočtu zložitosti algoritmov a pochopeniu základných princípov práce skutočných počítačov. Týmto príkladom v platorme emuStudio je abstraktný stroj RAM (Random Access Machine), ktorého Harvardská architektúra je odlišná od základnej myšlienky Johna von Neumanna používať jednu pamäť na uloženie dát ako aj inštrukcií programu.

Vďaka týmto príkladom už nebude také prekvapujúce, ak poviem, že platforma emuStudio umožňuje emulovať teoreticky ľubovoľný počítač. Umožňuje to jej komunikačný model založený na komunikácii a spolupráci zásuvných modulov, ktoré reprezentujú jednotlivé počítačové komponenty.



# Inštalácia programu

Ak chcete používať platformu emuStudio, je potrebné si ju v prvom rade stiahnuť. Platforma emuStudio je vyvíjaná ako slobodný softvér, teda pod licenciou GNU GPL v2 [GPLv2].

Existuje niekoľko možností, ako a kde stiahnuť platformu emuStudio. Prvou a azda najjednoduchšou možnosťou je stiahnuť si celú, dopredu pripravenú distribúciu. Druhou možnosťou je stiahnuť si len to, čo potrebujete. Ak si zvolíte túto možnosť, musíte mať na zreteli kompatibilitu verzií jednotlivých zásuvných modulov a hlavného modulu.

Existujú dva hlavné internetové zdroje, kde sú umiestnené celé distribúcie:

**Sourceforge.net** <https://sourceforge.net/projects/emustudio/>

**Github** <http://github.com/vbmacher/emuStudio/downloads>

Tieto zdroje sú stránkami hlavného modulu (samotnej platformy) a umožňujú tak stiahnutie jeho zdrojového kódu, dokumentácie alebo binárnej verzie ľubovoľnej verzie. Zásuvné moduly (v prípade, že ich autorom som ja) by mali byť umiestnené na rovnakých serveroch (alebo aspoň na serveri Github), ako samostatné projekty.

Na to, aby bolo možné spustiť platformu emuStudio, je potrebné mať nainštalované prostredie *Java Runtime Environment* (JRE) aspoň verziu **1.6**. Dá sa stiahnuť zo stránky:

<http://www.java.com/en/download/manual.jsp>

Na samotné spustenie a používanie programu (v prípade rozbalenia dopredu pripravenej distribúcie) už nie je potrebná žiadna ďalšia inštalačia. Súbory z inštalačného balíka rozbaľte do ľubovoľného adresára. Štruktúra podadresárov musí byť nasledovná:

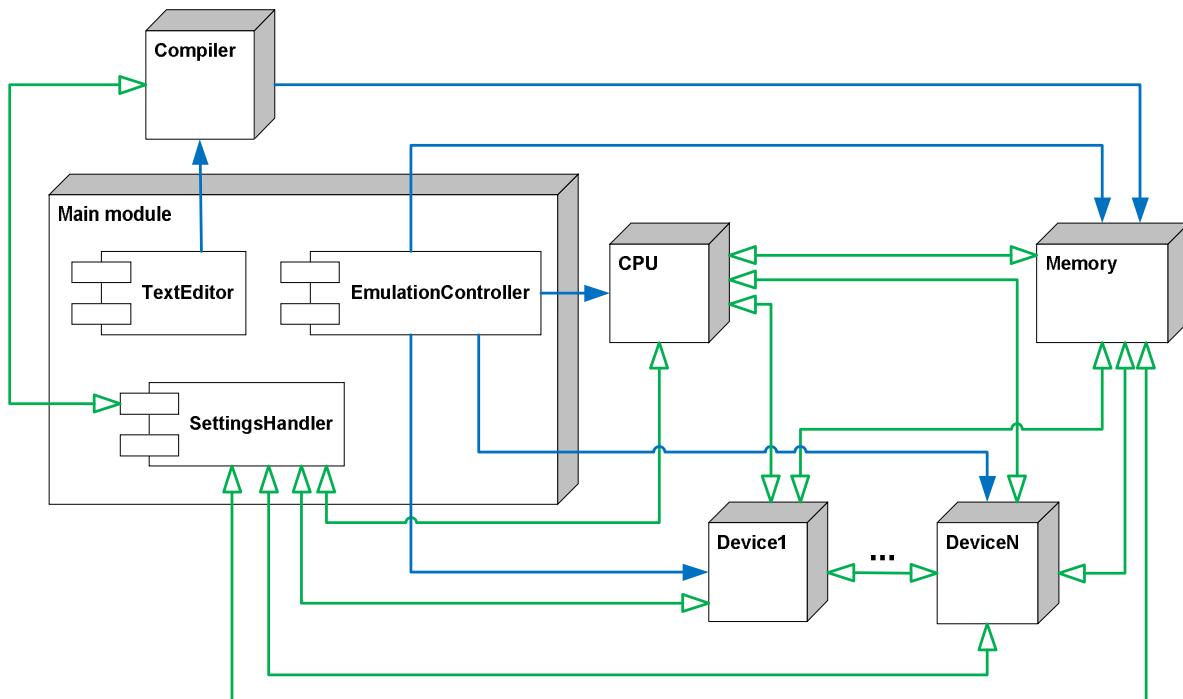
\compilers	... Zásuvné moduly kompilátorov
\config	... Súbory virtuálnych konfigurácií
\cpu	... Zásuvné moduly všetkých CPU
\devices	... Zásuvné moduly zariadení
\lib	... Potrebné knižnice
\mem	... Zásuvné moduly operačných pamäti

Program sa dá spustiť nasledovným príkazom v príkazovom riadku. Bližší popis nájdete v súbore README.TXT.

```
java -jar emuStudio.jar
```

# 1 Štruktúra platformy

Štruktúru platformy je možné vidieť na Obr.1.1. Uzol **Main module** sa skladá z niekoľkých komponentov, pričom na obrázku sú uvedené len tri (*EmulationController*, *TextEditor* a *SettingsHandler*). Komunikačné linky medzi jednotlivými uzlami sú bud' jednosmerné (modrá farba, plné šípky), alebo obojsmerné (zelená farba, prázdne šípky). Keď sa zahľadíme len na uzly **CPU**, **Memory** a **Device<sub>1..n</sub>**, štruktúra pripomína Von-Neumanovskú architektúru (čo bol, aj je úmysel); kde procesor, pamäť a periférne zariadenia sú oddelené, ale prepojené. Snaha o zachovanie podobnosti s touto architektúrou vyústila do riešenia, v ktorom sú všetky spomenuté moduly realizované formou zásuvných modulov.



Obr. 1.1: Štruktúra platformy emuStudio



# Časť I

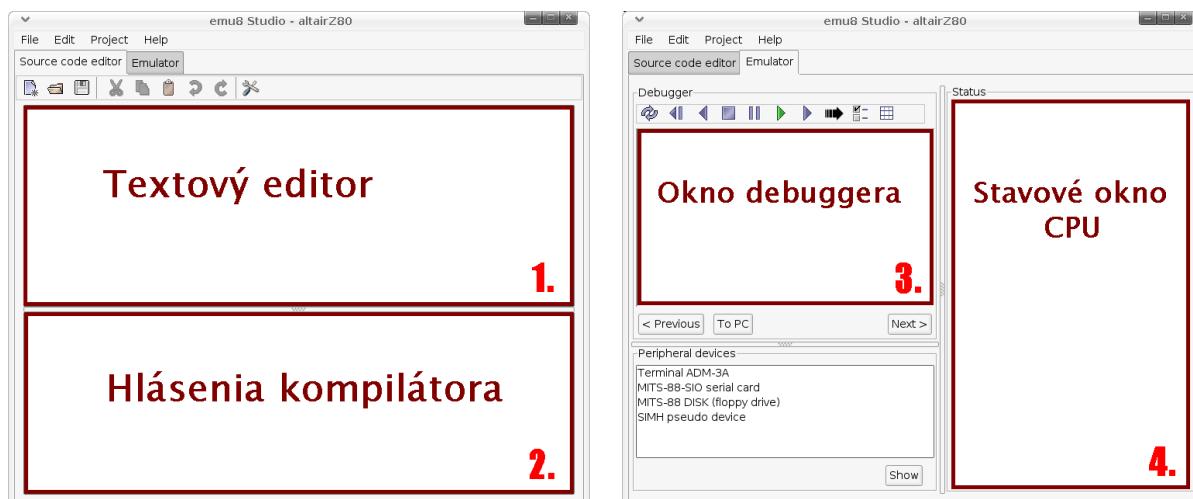
# Hlavný modul



## 2 Prehľad

Hlavný modul je samostatný program vyvíjaný nezávisle od zásuvných modulov. Jeho úloha je komunikovať s používateľom - umožniť používateľovi riadiť emuláciu ľahko a pohodlne, a tiež integrovať zásuvné moduly - aby všetko spolu vyzeralo ako jeden celok. Výhodou nezávislého modulu je v tom, že je univerzálny a používateľ sa nemusí učiť používať stále nový spôsob ovládania emulátora. Základné funkcie pre písanie, komplikovanie zdrojových textov, a ovládanie emulácie sú rovnaké pre všetky zvolené konfigurácie.

Teraz sa budeme zaoberať štruktúrou hlavného okna (Obr. 2.1), ktoré je najdôležitejším prvkom hlavného modulu. Síce trochu predbiehame, ale je dobré základné veci vysvetliť už tu.



Obr. 2.1: Štruktúra hlavného okna

Obrázok hlavného okna obsahuje štyri orámované časti. Tieto časti môžu mať po spustení programu „premenlivý“ obsah — môže sa meniť vzhľadom na zvolenú konfiguráciu počítača.

Celkovo je hlavné okno rozdelené do dvoch panelov: *Source code* (ľavá časť Obr. 2.1) a *Emulator* (pravá časť Obr. 2.1).

## 2.1 Panel „Source code“

Panel *Source code* obsahuje nástroje na manipuláciu so zdrojovým kódom. Skladá sa z dvoch veľmi dôležitých častí - textový editor (na obrázku označený číslom 1) a okno s hlásením komplilátora (číslo 2).

Voľba konfigurácie počítača (popísaná v časti 3) zahŕňa aj voľbu komplilátora (prekladača). Tento dokáže preložiť zdrojový kód (napísaný v textovom editore) do strojového kódu zvolenej CPU. Pre jednu CPU môže existovať aj viacero komplilátorov (ktoré nemusia byť nevyhnutne assemblermi<sup>1</sup>), preto má zmysel si komplilátor voliť.

Textový editor sa správa a vyzerá na prvý pohľad rovnako pre všetky zvolené komplilátory. Používateľ v ňom môže písat zdrojový kód presne podľa syntaxe jazyka zvoleného komplilátora. Má k dispozícii čislovanie riadkov a zvýrazňovanie syntaxe jazyka. Pri snahe zvýrazňovať syntaxu textový editor komunikuje priamo s komplilátorom, ktorý mu poskytuje lexikálne jednotky a preto zvýrazňovanie syntaxe funguje rovnako dobre pre každý jazyk.

O priebehu komplilovania zdrojového kódu informuje komplilátor používateľa zobrazením jeho hlásení, v časti 2 na Obr. 2.1. Hlásenia sa môžu (a spravidla sa aj) pre každý komplilátor líšiť. V každom prípade by sa po komplilovaní malo vypísať hlásenie o úspechu/neúspechu prekladu. Dobrý komplilátor by mal vedieť identifikovať chyby/varovania číslom riadku a stĺpca s popisom chyby.

## 2.2 Panel „Emulator“

Panel *Emulator* napravo na Obr. 2.1 obsahuje tri časti - okno debuggera (ladiace okno, označené číslom 3), stavové okno CPU (časť 4) a okno periférnych zariadení<sup>2</sup>. Obsah označených okien (grafická reprezentácia) je závislý na zvolenej konfigurácii počítača.

*Napríklad:* pri emulácii skutočných CPU ladiace okno vypisuje určitú oblasť operačnej pamäte, do ktorej ukazuje programové počítadlo<sup>3</sup>. Táto oblasť je obvykle reprezentovaná krajšou formou ako iba výpisom čísel - hodnôt buniek v pamäti. Väčšinou je okno rozdelené do niekoľkých stĺpcov a každý riadok predstavuje jednu alebo viac adres operačnej pamäte. Každý stĺpec potom reprezentuje hodnotu na tejto adrese inak - jeden stĺpec môže zobrazovať mnemonický tvar inštrukcie, iný, či je na adrese tzv. breakpoint (bod pozastavenia emulácie), atď. (Obr 5.2). Štruktúra tohto okna však nemusí byť rovnaká pre všetky architektúry, ktoré sú k dispozícii. Pre reálne CPU takáto štruktúru vyhovuje, no pri reprezentácii napr. abstraktného Turingovho stroja (ktorý však zatiaľ nie je implementovaný), by

---

<sup>1</sup>v podstate môžu existovať prekladače rôznych programovacích jazykov, ktorých výstupom je strojový kód zvolenej CPU

<sup>2</sup>nie je označené číslom, jeho obsah má jednotnú štruktúru - zoznam zariadení, čiže sa dá pokiaľ za nie „premenlivý“

<sup>3</sup>programové počítadlo, alebo PC, je v 8 bitových procesoroch názov registra obsahujúceho adresu nasledujúcej inštrukcie

takáto štruktúra vhodná nebola (kedže inštrukcie nie sú reprezentované rovnakým spôsobom ako dátá na tej istej pamäti/páske).

Ak si však na Obr. 2.1 všimneme nástrojový panel umiestnený nad časťou 3 (zväčšený na Obr. 5.3), zistíme, že ovládanie emulácie je jednotné pre ľubovoľnú zvolenú konfiguráciu, čiže ovládanie emulácie nezávisí od štruktúry zobrazenia aktuálnej inštrukcie a jej „okolia“.

Ďalej časť 4, predstavujúca stavové okno CPU, má takisto meniaci sa obsah v závislosti od zvolenej CPU. Pod stavom CPU rozumieme aktuálne hodnoty jej všetkých registrov a vnútorných nastavení/signálov v jednom časovom okamihu ako aj informáciu, či CPU beží, alebo nebeží. Okno nemusí zobrazovať všetky stavy a naopak môže zobraziť aj niektoré informácie navyše.

Abstraktné stroje nemusia mať registre, preto ak predpokladáme, že stroj je dobre formálne popísaný, potom pojem stav takéhoto stroja budeme chápať v zmysle jeho definície pre tento stroj. Pretože samotný stav môže byť reprezentovaný rôznymi informáciami, resp. reálne CPU majú iný počet a typ registrov a nastavení, obsah stavového okna nemôže byť rovnaký pre všetky CPU. Obvykle toto okno pre reálne CPU zobrazuje práve hodnoty jeho registrov, nastavení; prípadne obsahuje možnosti aj pre nastavenie rýchlosťi CPU.

Nakoniec neorámované okno na Obr. 2.1 dole obsahuje zoznam zariadení, ktoré sú načítané v aktuálnej konfigurácii (čiže mení sa iba aktuálny „zoznam“, nie celá štruktúra okna, ako to je v prípade označených okien). Položka zoznamu popisuje zariadenie svojim interným menom, ako je definované vo vnútri zásuvného modulu, teda nie názov súboru. Používateľ môže zobraziť grafické rozhranie zariadenia (ak ho zariadenie má), kliknutím na tlačidlo *Show*.

## 2.3 Základný postup, ako vyvíjať a emulovať

Je pravdou, že emulátor bol vyvíjaný so zreteľom na jednoduchosť a pohodlie pre používateľa. Napriek tomu, možno je to vplyvom aj samotných emulovaných architektúr, sa môže cítiť používateľ na prvý pohľad zmätený a nemusí hneď zo štruktúry emulátora (Obr. 1.1) pochopiť, ako pod emulátorom správne pracovať.

Z toho dôvodu je táto časť venovaná práve základnému postupu práce na emulátore. Pod pojmom „práca na emulátore“ rozumieme jednu z dvoch činností - vývoj programu, alebo samotné emulovanie. Z takejto definície vychádza aj diagram na Obr. 2.2.

Ide o diagram aktivít, ktorý sa podobá vývojovému diagramu. Sú v ňom zahrnuté základné procesy v postupnosti, aká je pre platformu prirodzená. Poradie niektorých krokov v postupnosti sa môže zmeniť, niektoré možno dokonca aj vynechať a okrem uvedených krokov môžu existovať aj iné, ktoré na diagrame uvedené nie sú. Tie vznikajú už z potrieb konkrétnego emulovania (napríklad nastavenie dodatočných parametrov operačnej pamäte).

Práca na emulátore začína formálnym rozhodnutím, čo chceme robiť - či vyvíjať pro-

gram, alebo rovno emulovať. Podľa rozhodnutia sa postupuje do danej vetvy v diagrame. Vývoj programu znamená napísať jeho zdrojový kód. Používateľ samozrejme nemusí vychádzať zo striktného pojmu *nаписа* zdrojový kód, môže otvoriť aj existujúci zo súboru alebo vložiť ho do textového editora iným spôsobom. Aktivita *Písanie zdrojového kódu* znamená vytváranie nového kódu, dopisovanie, modifikáciu, atď. Používateľ však musí kód písat so zreteľom na syntax a sémantiku zvoleného kompilátora vo virtuálnej architektúre.

Nasleduje aktivita *Kompilovanie zdrojového kódu*, ktorej výstupom je buď hotový program pripravený na emuláciu, alebo chybové hlásenia kompilátora. V prípade, že kód neboli preložený z dôvodu chýb, je na používateľovi, aby kód opravil a znova prekompiloval. V opačnom prípade sa prejde na jeho emuláciu<sup>4</sup>.

Emulovať sa môže začať, až keď bude všetko pripravené - hlavne treba emulátoru povedať, čo chceme emulovať. Všetky architektúry na emulátore pracujú podobne ako von-neumannovská architektúra, preto odpoveď na otázku čo sa má emulovať je v operačnej pamäti. Operačná pamäť obsahuje program aj dátu. Procesor z nej číta hodnoty jednotlivých buniek sekvenčne, ktorých význam rozpoznáva ako inštrukcie a tie potom vykonáva. Preto je potrebné, aby prvým krokom v emulácii bolo načítanie vyvíjaného programu (alebo externého(-ých) súboru(-ov), tzv. obrazu) do operačnej pamäte. Takže ak má používateľ k dispozícii už hotový, skompilovaný program vo formáte podporovanom operačnou pamäťou, môže písanie zdrojového kódu a jeho kompilovanie preskočiť.

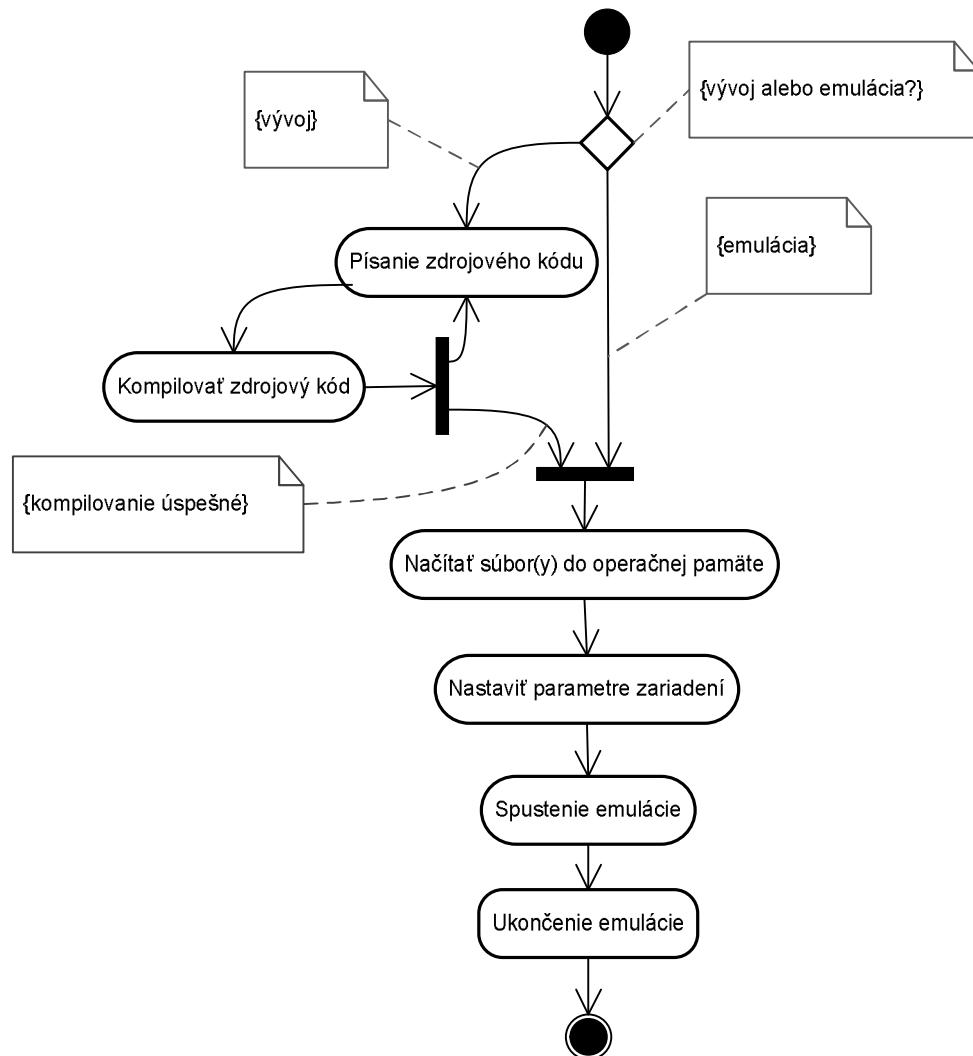
V prípade, že virtuálna architektúra obsahuje aj prídavné zariadenia, teraz je vhodný čas nastaviť im parametre. Parametre môžu mať rôzny charakter, napr. pre terminál môže ísiť o nastavenie parametra „always on top“, pre diskové zariadenie to môže byť namontovanie obrazov diskiet (virtuálne vloženie diskety do zariadenia). Zariadenia môžu podporovať aj uloženie/znovučítanie svojich nastavení a preto niekedy netreba nič nastavovať, pretože si nastavenia zariadenie vie z danej architektúry zistíť samo.

Nezáleží na tom, či sa najskôr pripraví operačná pamäť a až potom parametre zariadení, alebo naopak. Emulácia je teraz pozastavená, preto príprava môže prebiehať v kľúde, „bez stresu“ a bez nároku na poradie prípravných krovov. Výnimkou je však poradie načítavania obrazov do operačnej pamäte. Poradie nie je podstatné, ak sa obrazy v operačnej pamäti neprekryvajú. V opačnom prípade musí mať používateľ na zreteli, že oblasť, v ktorej obrazy kolidujú, bude mať platné hodnoty posledného obrazu, ktorý oblasť prekryl.

Poslednou aktivitou je *Spustenie emulácie*. Znova upozorňujem, že nejde o striktné stlačenie tlačidla *Run*. Používateľ sa môže rozhodnúť, či emuláciu bude krokoval, alebo emuláciu spustí. Pred samotným spustením môže používateľ nastaviť adresu v pamäti, od ktorej sa emulácia spustí (od ktorej začne procesor čítať inštrukcie).

---

<sup>4</sup>ak používateľ vôbec chce kód emulovať, pretože táto aktivita nie je povinná, aj keď podľa diagramu nemá inú možnosť



Obr. 2.2: Základný postup pre vývoj a emulovanie na platforme



# 3 Konfigurácie a virtuálne architektúry

Emulátor vďaka schopnosti akceptovať rôzne zásuvné moduly toho istého typu umožňuje používateľovi emulovať rôzne počítačové architektúry. Vychádzajúc pritom zo známej schémy Von-Neumanna, každá emulovaná architektúra musí obsahovať procesor, operačnú pamäť a voliteľne aj niekoľko prídavných zariadení.

Nová verzia emulátora so sebou prináša schopnosť tvoriť hierarchické prepojenia zariadení medzi sebou, operačnou pamäťou a procesorom. Používateľ prepojenia kreslí v abstraktnej schéme. Každý blok schémy predstavuje určitý zásuvný modul, resp. prvok v architektúre. Viac o vytváraní konfigurácií sa čitateľ dozvie v časti 3.2.

Konfigurácie sú uložené v konfiguračných súboroch, pričom každý súbor odpovedá jednej architektúre. Konfigurácie je samozrejme možné aj vytvárať a preto je možné vytvoriť aj takú konfiguráciu počítača, aká existovala (alebo existuje) v skutočnosti. Z toho vyplýva, že programy napísané pre pôvodný hardvér sa dajú spustiť aj na tejto platforme, ak jej konfigurácia odpovedá pôvodnej (a naopak).

Pojem *virtuálna architektúra* definujem ako inštanciu konkrétnej konfigurácie počítača. Konfigurácie sú uložené v konfiguračných súboroch, pričom každý súbor odpovedá práve jednej konfigurácii.

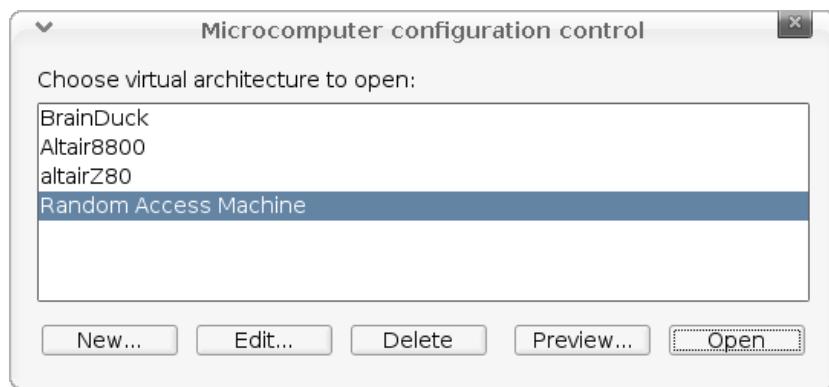
Súčasná verzia emulátora dokáže (počas behu) pracovať len s jednou virtuálnou architektúrou. Z tohto dôvodu je potrebné *zvoliť si*, s ktorou. Používateľ volí konfiguráciu virtuálnej architektúry hned' po spustení programu.

## 3.1 Volba konfigurácie

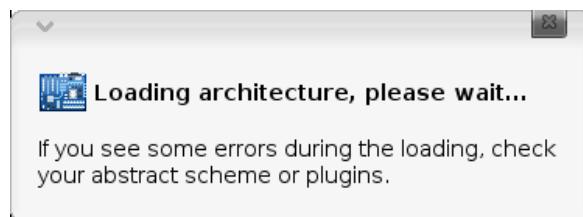
Okno voľby konfigurácie je možné vidieť na Obr. 3.1. Ide o klasický zoznam konfigurácií, pričom každá má svoj názov. Používateľ najprv klikne na názov konfigurácie a potom na tlačidlo *Open*. Ak okno používateľ zruší, zavrie sa aj celý program (voľba architektúry je nutná). V tomto okamihu sa začne načítavať konfigurácia a postupne z nej vytvárať virtuálna architektúra.

Proces sa začne zobrazením informačného okna (Obr. 3.2). Je aktivované po výbere architektúry a zmizne, keď je architektúra úspešne načítaná.

Poznámka:



Obr. 3.1: Voľba konfigurácie počítača



Obr. 3.2: Informačné okno - načítavanie konfigurácie

Prípadné chyby pri vytvaraní architektúry (alebo prepojení) vyskakujú, len keď je toto okno aktívne, pričom je na ňom uvedené upozornenie, čo treba robiť, ak vyskočí chyba (overiť abstraktnu schému alebo samotné zásuvné moduly).

Hlavný modul pri vytváraní architektúry nekontroluje celú sémantiku abstraktnej schémy, teda nevie určiť, či schéma bude fungovať. Počas tohto procesu môžu vzniknúť chyby dvojakého druhu:

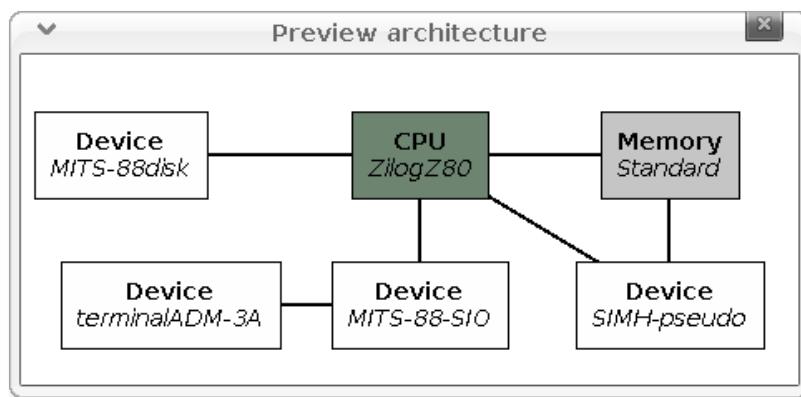
- Chyby pri načítavaní zásuvných modulov (neexistujúci súbor zásuvného modulu, ...)
- Chyby pri realizácii prepojení (ak sa zásuvné moduly medzi sebou nedajú prepojiť v žiadnom smere)

Ak sa objaví ľubovoľná chyba, virtuálna architektúra sa prestane vytvárať a emulátor sa ukončí. Ak všetko prebehne v poriadku, tak sa konfigurácia premení na virtuálnu architektúru, ktorá sa už dá emulovať. Počas behu emulátora sa už nedá meniť architektúra, ani prepínať medzi inými architektúrami.

## 3.2 Vytváranie a editácia konfigurácií

Konfigurácie je samozrejme možné aj vytvárať a preto je možné vytvoriť aj takú konfiguráciu počítača, aká existovala (alebo existuje) v skutočnosti. Z toho vyplýva, že programy napísané pre pôvodný hardvér sa dajú spustiť aj na tejto platforme, ak jej konfigurácia odpovedá pôvodnej (a naopak).

Oproti predošej verzii sa vytváranie konfigurácií zásadne líši. Používateľ namiesto strohého výberu zásuvných modulov teraz musí nakresliť abstraktnú schému celej architektúry. Schéma je v podstate „pekný“ graf, skladajúci sa z *uzlov* - typových zásuvných modulov a z *hrán* - prepojení medzi nimi. Príklad abstraktnej schémy je možné vidieť na Obr. 3.3



Obr. 3.3: Príklad abstraktnej schémy

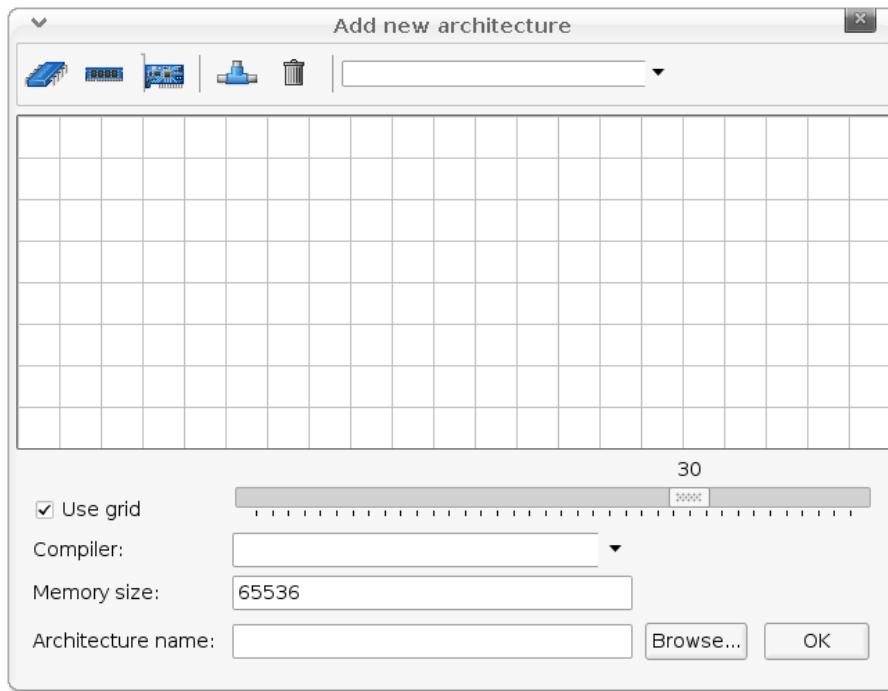
Emulátor rozoznáva štyri typy zásuvných modulov: komplátory, CPU, operačné pamäte a zariadenia. Všetky tieto typy sa v abstraktnej schéme môžu nachádzať. Z toho okrem zariadení, ktorých môže byť 0 a viac, sa každý typ zásuvného modulu v schéme môže nachádzať len raz. To znamená práve jeden komplátor, práve jedna CPU, a práve jedna operačná pamäť. Editor abstraktných schém nedovolí v schéme použiť väčší, ani menší počet týchto zásuvných modulov.

V schémach neexistujú žiadne zbernice. Cieľom bolo vyhnúť sa schémam a prepojeniam zasahujúcich do viac elektrotechnického charakteru. Je lepšie, aby mali schémy viac logický charakter (formu Von-Neumanovskej schémy bez zbernice). Aj takto je stále možné prepájať skoro ľubovoľne medzi sebou CPU, zariadenia a operačnú pamäť. Aj keď to editor schém nevyžaduje, väčšinou vždy existuje prepojenie medzi CPU a operačnou pamäťou, ktoré treba explicitne vytvoriť.

### 3.2.1 Vytváranie nových konfigurácií

Novú architektúru vytvoríme kliknutím na tlačidlo *New* v okne voľby konfigurácie (Obr. 3.1). Zobrazí sa editor abstraktnej schémy, v ktorom používateľ nakreslí schému ar-

chitektúry, vyberie komplítačor, nastaví veľkosť operačnej pamäte a samozrejme tiež názov novej konfigurácie. Okno editora je možné vidieť na Obr. 3.4.



Obr. 3.4: Editor novej konfigurácie

## Mriežka

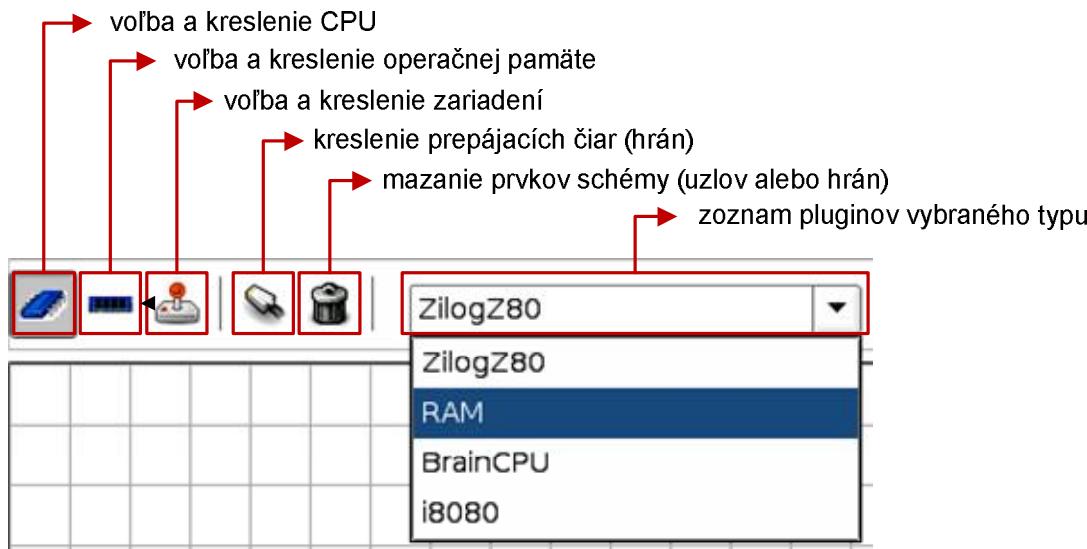
Kreslenie schém je možné trochu uľahčiť použitím mriežky (*Use grid*), čo spôsobí uchytenie kreslených prvkov na pravidelné pozície kresliacej plochy. Mriežka sa zobrazuje vo forme križujúcich sa vodorovných a zvislých čiar na pozadí. Pevné body, v ktorých sa prvky uchýcujú, sú priesečníkmi týchto čiar. Veľkosť mriežky je možné meniť (posúvací panel napravo od checkboxu *Use grid*, Obr. 3.4) a tým sa zvyšuje/znižuje hustota pevných bodov na kresliacej ploche. V prípade, že sa mriežka nepoužije, pozícia prvkov na ploche nie je korigovaná.

## Voľba a kreslenie prvkov schémy

Panel nástrojov editora schém je možné vidieť na Obr. 3.5

Používateľ umiestňuje na kresiacu plochu vybrané prvky schémy - CPU, operačnú pamäť a zariadenia. Postup pri ich kreslení je jednoduchý: najprv je potrebné vybrať, čo ideme kresliť (čiže kliknúť na príslušnú ikonu na nástrojovom paneli, Obr. 3.5). Na to sa aktualizuje zoznam dostupných zásuvných modulov pre vybraný typ prvkova.

POZNÁMKA:



Obr. 3.5: Panel nástrojov editora konfigurácie

Aby sa pri kreslení schémy dali použiť všetky zásuvné moduly vybraného typu, musia byť uložené v príslušných podadresároch adresára, kde je program nainštalovaný (vid' časť Inštalácia programu na str. 3).

Používateľ si zo zoznamu vyberie už konkrétny zásuvný modul, ktorý chce použiť a umiestní ho do schémy jedným kliknutím ľavým tlačidlom. Tým istým postupom umiestní do schémy všetky ostatné prvky.

Pri voľbe operačnej pamäte je tiež potrebné určiť jej veľkosť - túto potom už nie je možné dynamicky meniť (po vytvorení virtuálnej architektúry). Osembitové počítače mají túto veľkosť štandardne od  $64 \div 256 kB$ . Veľkosť operačnej pamäte sa zadáva v spodnej časti okna (*Memory size*, Obr. 3.4).

Názvy zásuvných modulov v zozname dostupných modulov sú vlastne ich názvy súborov bez prípony, a teda neodpovedajú názvom v hlavnom okne (po vytvorení virtuálnej architektúry). Je to tak preto, lebo interné názvy zariadení ešte nie sú známe (zásuvné moduly ešte neboli načítané).

### Presúvanie prvkov

V prípade, že používateľ chce zmeniť pozíciu niektorého prvku na schéme, musí najprv zrušiť funkciu vytvárania prvkov - „odkliknutím“ vybranej ikony z nástrojového panela<sup>1</sup>. Tým sa zruší výber dostupných zásuvných modulov a teraz môže používateľ prvky presúvať. Prvky sa presúvajú metódou „drag & drop“ - klikneme na prvak ľavým tlačidlom myši, a za

<sup>1</sup>v skutočnosti musí byť zrušená každá funkcia - vytváranie prvkov, mazanie, kreslenie prepájacích čiar

súčasného držania tlačidla presunieme prvok na požadované miesto. Potom tlačidlo pustíme a prvok už ostane na novom mieste (v prípade, že je použitá mriežka, sa prvok umiestní na najbližší pevný bod).

### Prepájacie čiary

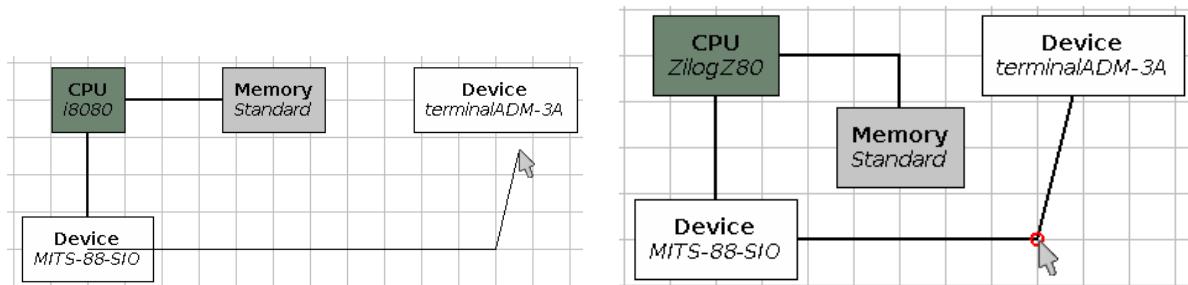
Po umiestnení prvkov do schémy nasleduje ich vzájomné prepájanie prepájacími čiarami (hranami). Najprv je potrebné kliknúť na správnu ikonu nástrojového panela (Obr 3.5). Prepájacia čiara sa skladá nutne z dvoch koncov, ktorými sú prepájané uzly a môže sa skladať aj z ďalších pomocných bodov, ktoré lomia čiaru na viac častí.

Čiary sa nekreslia metódou „drag & drop“. „Rovné“ prepojenia vznikajú jedným kliknutím ľavým tlačidlom myši na prvý prvek a potom ďalším kliknutím na druhý prvek. Pozícia čiary je korigovaná automaticky tak, že jej konce sú umiestnené v strede pod prepojenými prvkami. Pritom nezáleží na tom, od ktorého prvku sa začne čiara kresliť, pretože používateľ v schéme (ani nikde inde) nedefinuje smer prepojenia.

POZNÁMKA:

Emulátor na začiatku predpokladá obojsmerné prepojenie a pri vytváraní virtuálnej architektúry ho skúša realizovať. Ak sa mu to nepodarí, tak skúša realizovať aspoň jednosmerné prepojenie (najprv zapája prvý prvek (prvý koniec čiary) do druhého, ak sa to nepodarí, tak opačne). Ak sa nepodarí prepojiť zariadenia vôbec (ani obojsmerne, ani jednosmerne), vyskočí chybové hlásenie a emulátor sa ukončí.

Počas vytvárania prepájacej čiary môže používateľ definovať pomocné body čiary, od ktorých jedna čiara môže pokračovať iným smerom. Vznikne tak lomená čiara. Vytváranie lomených čiar je zobrazené na Obr. 3.6. Pomocné body pri vytváraní prepájacej čiary vytvárame jedným kliknutím ľavým tlačidlom myši na miesto, kde bod má ležať.



Obr. 3.6: Ukážka kreslenia lomenej prepájacej  
Obr. 3.7: Ukážka presúvania pomocného  
čiary bodu

Ak nie je v nástrojovom paneli vybratá žiadna funkcia, je možné pomocné body presúvať metódou „drag & drop“ - klikneme na bod ľavým tlačidlom myši, a počas súčasného

držania tlačidla presunieme bod inam. Bod, ktorý presúvame je zvýraznený - okolo neho je nakreslený červený krúžok (Obr. 3.7).

Pomocné body sa dajú vytvárať aj keď je už prepájacia čiara vytvorená. Stačí kliknúť ľavým tlačidlom myši na ľubovoľné miesto na čiare<sup>2</sup> a počas súčasného držaním tlačidla pohnúť myšou. Na kliknutom mieste sa vytvorí nový pomocný bod, ktorý súčasným pohybom presúvame na požadované miesto.

Rušenie pomocných bodov je jednoduché - na pomocný bod, ktorý chceme zrušiť klikneme raz pravým tlačidlom myši. Po zrušení pomocného bodu sa čiara „narovná“ - priamou úsečkou sa spoja body ležiace bezprostredne pred a za rušeným bodom.

### Mazanie prvkov

V prípade, že chce používateľ odstrániť nejaký prvok alebo prepájaciu čiaru zo schémy, musí z panelu nástrojov vybrať funkciu mazania. Táto funkcia sa však nepoužíva na mazanie pomocných bodov lomenej čiary. Jedným kliknutím ľavým tlačidlom myši na prvok sa tento zo schémy odstráni. Ak odstraňujeme uzol (nie prepájaciu čiaru), odstránia sa aj všetky prepojenia s odstraňovaným prvkom.

#### 3.2.2 Editovanie konfigurácie

Editovanie, alebo úprava existujúcej konfigurácie znamená úpravu alebo zmenu abstraknej schémy, výberu komplátora a veľkosti operačnej pamäte bez zmeny jej názvu. Editor použitý na úpravu existujúcej konfigurácie však neumožňuje zmenu nastavení pre jednotlivé zásuvné moduly. Ide totiž o rovnaký editor, aký sa používa pre vytváranie novej konfigurácie. Preto pre jeho bližší opis nech čitateľ pozrie časť 3.2.1.

Editovanie existujúcej konfigurácie aktivujeme v okne voľby konfigurácie (Obr. 3.1) kliknutím na tlačidlo *Edit*.

---

<sup>2</sup>všetky funkcie nástrojového panela musia byť neaktívne



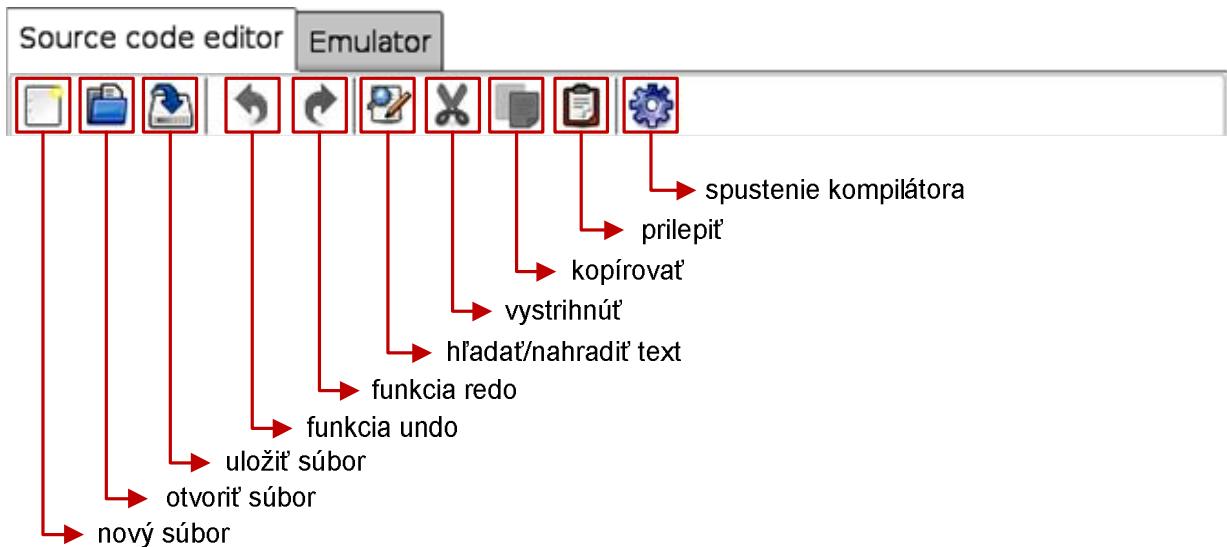
# 4 Textový editor

## 4.1 Práca v textovom editore

Ako už bolo spomenuté, textový editor slúži na písanie zdrojových kódov. V konečnom dôsledku použitý byť vôbec nemusí — môžeme emulovať kód, ktorý už je skompilovaný.

Písanie textu je veľmi jednoduché. Textový editor automaticky podporuje zvýrazňovanie syntaxe jazyka zvoleného kompilátora, vyznačuje čísla riadkov. Hlásenia kompilátora sú tak prehľadnejšie (riadok a stĺpec sa dá rýchlejšie nájsť v zdrojovom kóde). Funkcie undo (vráti späť jeden napísaný znak) a redo (znova napiše znak, ktorý bol predtým vrátený) sú samozrejmosťou.

Napísaný kód nie je možné skompilovať, pokiaľ neboli predtým uložený. Nástrojový panel textového editora aj s popisom jednotlivých ikon je možné vidieť na Obr. 4.1.



Obr. 4.1: Nástrojový panel textového editora

Nástrojový panel na Obr. 4.1 obsahuje ikony reprezentujúce iba tie základné, najpoužívanejšie funkcie textového editora a spoluprácu s kompilátorom. Všetky funkcie programu

zahŕňajúce aj funkcie spomínaného nástrojového panela, sú k dispozícii v ponuke programu — textové menu umiestnené celkom navrchu okna.

Niektoré funkcie (najčastejšie používané) je možno spustiť aj klávesovými skratkami, ako to ukazuje Tab. 4.1.

Názov funkcie	Umiestnenie v ponuke	Skratka
Nový súbor	File → New	CTRL+N
Otvoriť súbor	File → Open	CTRL+O
Uložiť súbor	File → Save	CTRL+S
Späť	Edit → Undo	CTRL+Z
Dopredu	Edit → Redo	CTRL+Y
Vystrihnúť výber textu	Edit → Cut selection	CTRL+X
Kopírovať výber textu	Edit → Copy selection	CTRL+C
Prilepiť do výberu textu	Edit → Paste selection	CTRL+V
Hľadať/nahradiť text	Edit → Find/replace text	CTRL+F
Hľadať ďalej	Edit → Find next	F3
Nahradiť ďalej	Edit → Replace next	F4

Tabuľka 4.1: Klávesové skratky v hlavnom module

Význam väčšiny spomenutých funkcií je intuitívne jasné, preto sa budeme ďalej zaoberať menej jasným funkciám alebo funkciám, ktorých význam je širší.

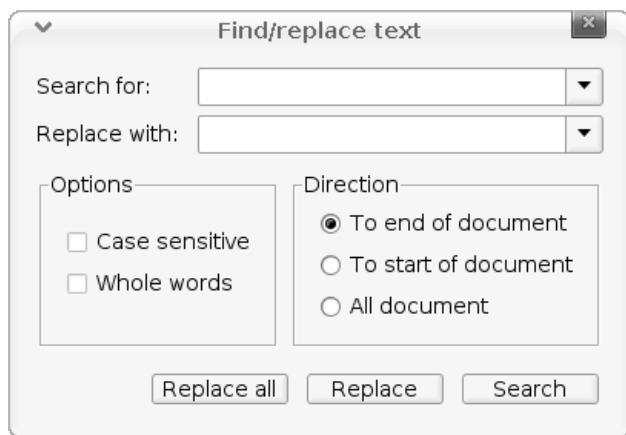
## 4.2 Vyhľadávanie, nahradzanie textu

Vyhľadávanie/nahrádzanie textu vie pracovať v dvoch režimoch: klasický a rýchly.

V *klasickom* režime po aktivácii funkcie vyhľadávania/nahrádzania sa zobrazí dialógové okno (Obr. 4.2).

V tomto okne používateľ zadáva/vyberá:

- povinne text, ktorý chce nájsť/zameniť (textové okno *Search for*)
- v prípade nahradzania textu povinne text, ktorý sa použije ako náhrada (textové okno *Replace with*)
- v sekcií *Options* sa vyberajú detaile vyhľadávania — podpora rozlišovania veľkosti znakov (*Case sensitive*) a či hľadaný text predstavuje samostatné slovo (*Whole words*)
- sekcia *Direction* umožňuje vybrať jeden z troch typov smerov vyhľadávania:
  - smerom ku koncu dokumentu (*To end of document*)



Obr. 4.2: Dialógové okno vyhľadávania/nahrádzania textu

- smerom k začiatku dokumentu (*To start of document*)
- prehľadá sa celý dokument, začne sa smerom ku koncu dokumentu; ak sa hľadaný text nenájde, pokračuje hľadaním od začiatku (*All document*)

Výber samotnej funkcie je realizovaný kliknutím na jedno z troch tlačidiel: Vyhľať (*Search*), Nahradíť (*Replace*) alebo Nahradíť všetko, čo sa nájde vo vybranom smere vyhľadávania (*Replace All*). Vykonanie vybranej funkcie je jednorázové; dialógové okno sa následne zavrie.

Hľadaný/nahrádzajúci text môže mať aj tvar regulárneho výrazu (teda akéhosi vzoru, podľa ktorého by mal vyzeráť hľadaný/nahrádzajúci text). V implementácii je použitý javaovský formát regulárnych výrazov. Podrobnejšie sa štruktúra regulárnych výrazov a práca s nimi rozoberá tu: <http://java.sun.com/docs/books/tutorial/essential/regex/index.html>

Ako príklad vyhľadávania ľubovoľného reťazca uzavretého v jednoduchých úvodzovkách ' ', by sme napísali napr. takýto vyhľadávací text:

'([^\']\*)'

Preto ak chceme vyhľadať text, ktorý obsahuje špeciálne riadiace znaky regulárnych výrazov, je potrebné dať pred nich spätné lomítko \.

Špeciálne znaky sú tieto:

\$ ^ . \* + ? [ ] ( ) " \

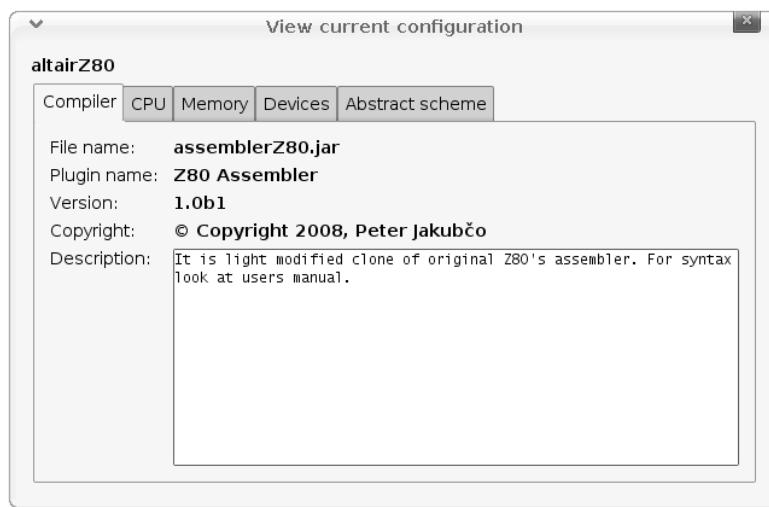
Práca vyhľadávania/nahrádzania textu v *rýchлом* režime predstavuje rýchle vyvolanie funkcie vyhľadávania alebo nahradzania textu, pričom sa nezobrazí žiadne dialógové okno

a parametre vyhľadávania/nahrádzania sa použijú presne také isté ako pri poslednej takejto funkcií. Rýchly režim sa vyvoláva položkou z menu *Edit* alebo klávesovými skratkami, ako je uvedené v Tab. 4.1 v položkách *Edit* → *Find Next* resp. *Edit* → *Replace Next*. Klasický režim však vždy predchádza rýchlemu režimu, v prípade ak bol rýchly režim vyvolaný pred použitím klasického režimu, klasický režim sa automaticky vyvolá sám.

# 5 Emulátor

## 5.1 Prehľad aktuálnej konfigurácie

Konfigurácia zvolená na začiatku programu sa už počas jeho behu meniť nedá. Je však možnosť si ju podrobnejšie prezrieť, zmysel to má pri pozorání detailov jednotlivých zásuvných modulov, ako je napr. verzia zásuvného modulu alebo abstraktná schéma virtuálnej architektúry. Okno zobrazujúce vybranú konfiguráciu sa aktivuje položkou v menu *Project* → *View configuration*. Okno je zobrazené na Obr. 5.1



Obr. 5.1: Dialógové okno zobrazujúce aktuálnu konfiguráciu

## 5.2 Okno debuggera

Panel emulátora obsahuje tri základné časti, ako boli popísané na začiatku kapitoly 1. V tejto časti sa budeme zaoberať oknom debuggera. Pri štúdiu práce emulovaného procesora je to azda najdôležitejšie okno, ktoré poskytuje hlavne informáciu o tom, ktorá inštrukcia pri vykonávaní práve nasleduje. Rôzne implementácie rôznych procesorov môžu v tomto

okne zobrazovať aj iné informácie, určite vždy však pôjde o akýsi zoznam zložený z riadkov a stĺpcov.

Na Obr. 5.2 je zobrazené okno procesora Intel 8080 zobrazujúce inštrukcie programu.

breakpoint	address	mnemonics	opcode
<input type="checkbox"/>	004Fh	daa	27
<input type="checkbox"/>	0050h	nop	00
<input type="checkbox"/>	0051h	out 11h	D3 11
<input type="checkbox"/>	0053h	stax DE	12
<input type="checkbox"/>	0054h	inx DE	13
<input type="checkbox"/>	0055h	inx C	0C
<input type="checkbox"/>	0056h	jmp 0027h	C3 27 00
<input checked="" type="checkbox"/>	0059h	mvi A,0Ah	3E 0A
<input type="checkbox"/>	005Bh	stax DE	12
<input type="checkbox"/>	005Ch	inx DE	13
<input type="checkbox"/>	005Dh	mvi A,0Dh	3E 0D
<input type="checkbox"/>	005Fh	stax DE	12

Obr. 5.2: Okno debuggera „v akcii“

Všeobecne nejde o nič iné, ako o zobrazenie obsahu operačnej pamäte v určitej oblasti, v určitom rozsahu (25 riadkov = 25 inštrukcií). Každá inštrukcia je uložená, rovnako ako aj dátá, v operačnej pamäti. Inštrukcie spomínaného procesora Intel 8080 zaberajú v operačnej pamäti od jedného po tri bajty. Z tohto dôvodu na Obr. 5.2 adresy sa nezväčšujú stále o rovnakú hodnotu. V prípade, že nasledujúca adresa je o napr. dve čísla väčšia ako aktuálna, znamená to, že inštrukcia na aktuálnej adrese zaberá dva bajty. Všimnite si na Obr. 5.2 napr. riadok s adresou 0022h. Na tejto adrese v operačnej pamäti je uložená inštrukcia in 08h. Táto inštrukcia zaberá dva bajty: DB 08. V takomto tvare je inštrukcia zobrazená v stĺpci *opcode*, ako aj v operačnej pamäti.

Implementovaná CPU procesora Intel 8080 umožňuje pozastaviť emuláciu, ak programové počítadlo (PC) dosiahne určitú adresu. Takýto bod pozastavenia sa nazýva *breakpoint* a v okne ho možno jednoducho nastaviť kliknutím na takto označený stĺpec v príslušnom riadku (viď riadok s adresou 0022h na Obr. 5.2). Potom v prípade, že spustíme emuláciu (bez krokovania), v tomto bode sa emulácia pozastaví a používateľ tak môže sledovať stav CPU pred spuštením inštrukcie nachádzajúcej sa na adrese pozastavenia.

### 5.2.1 Stránkovanie inštrukcií

Nová verzia emulátora prináša možnosť stránkovania zoznamu inštrukcií v okne debuggera. Ide o schopnosť, pomocou ktorej sa používateľ vie pozrieť na ďalšie inštrukcie, nezobrazené v aktuálnom zozname inštrukcií. Prvá stránka je zoznam inštrukcií, v ktorom

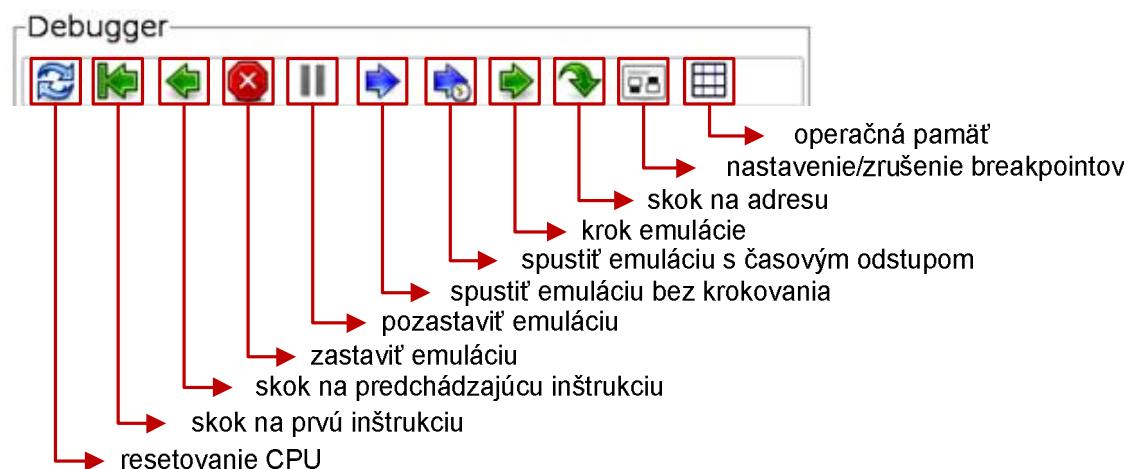
sa nachádza aktuálna inštrukcia. Prepínanie stránok dopredu resp. dozadu spôsobí posun zoznamu inštrukcií dopredu resp. dozadu.

K dispozícii má používateľ tlačidlá:

- *Next* - Zobrazí nasledujúcu stránku (ak sme na poslednej stránke, nič sa nestane). Pár inštrukcií na konci zoznamu na aktuálnej stránke sa budú zhodovať zo začiatčnými inštrukciami na nasledujúcej stránke, teda aktuálna stránka nekončí na mieste, kde potom začína nasledujúca. Má to výhodu v tom, že ak používateľ sleduje inštrukcie v nejakej postupnosti, časť tejto postupnosti sa zachová aj na ďalšej stránke („nestratí nit“, tj. sa nemusí vracať o stránku späť a pozerat na poslednú inštrukciu).
- *Previous* - Zobrazí predchádzajúcu stránku (ak sme na prvej stránke, nič sa nestane). Pár inštrukcií na konci zoznamu na predchádzajúcej stránke sa budú zhodovať zo začiatčnými inštrukciami na aktuálnej stránke, teda predchádzajúca stránka nekončí na mieste, kde potom začína aktuálna. Je to z podobného dôvodu, ako pri zobrazovaní inštrukcií nasledujúcich stránok.
- *To PC* - Vráti sa na prvú stránku (tam, kde sa nachádza aktuálna inštrukcia)

### 5.2.2 Ovládanie emulácie

Nad samotným oknom zobrazujúcim inštrukcie programu sa nachádza panel na ovládanie emulácie. Tento panel, aj s krátkym popisom, je zobrazený na Obr. 5.3.



Obr. 5.3: Panel ovládania emulácie

Funkcia **Reset CPU** môže mať pre každú CPU iný význam. Vo všeobecnosti je to funkcia, ktorá vráti CPU do počiatočného stavu, teda do stavu, v akom sa nachádzala pred prvým spustením emulácie. Pre reálne 8 bitové CPU to väčšinou znamená inicializácia registrov, príznakov a nastavenie programového počítadla PC na nejakú počiatočnú hodnotu, väčšinou

0. Táto funkcia má v reálnych počítačoch skôr širší význam. Napríklad stlačením tlačidla reset na počítači sa neinicializuje iba CPU, ale aj všetky ostatné zariadenia a iné obvody v priamom kontakte s CPU. Obsah operačnej pamäte však ostáva zachovaný.

Ďalšia sada funkcií zobrazených na paneli na Obr. 5.3 vyzerá ako ovládanie nejakého prehrávača. Používateľ už intuitívne môže zistiť, že sa jedná o priame ovládanie emulácie.

Funkcia **nastavenia PC na začiatok** znova inicializuje programové počítadlo na počatočnú hodnotu. Niektoré CPU môžu PC inicializovať na hodnotu adresy výskytu prvej inštrukcie programu (pokiaľ je táto adresa dobre známa).

Nasledujúca funkcia **dekrementácie PC** odpočíta od programového počítadla jednotku. Nemení registre, ani výsledok predchádzajúcej inštrukcie. Dekrementácia PC môže, ale nemusí znamenať posun o inštrukciu späť. Ak bola predchádzajúca inštrukcia na adrese  $a$  veľkosti napr. 3 bajty, PC sa dekrementuje a bude ukazovať na inštrukciu (možno známu, možno nie) na adrese  $a - 1$ . To spôsobí zmenu v chápaní aj nasledujúcich inštrukcií, pretože známa inštrukcia začínajúca na adrese  $a - 1$  nemusí nutne zaberať 2 bajty (aby sa posun vydružil). Situácia je zobrazená v Tab. 5.1. Ak chceme posunúť PC späť o celú inštrukciu, musíme zavolať funkciu dekrementácie PC toľkokrát, aká je inštrukcia veľká v bajtoch.

### [ ] Interpretácia inštrukcií

Tabuľka 5.1: Ukážka rozdielnej interpretácie inštrukcií pri nepatrnom posunutí PC

Ak nastane takáto zmena v interpretácii, prejaví aj v okne debuggera, to znamená, že inštrukcie sa začnú interpretovať od aktuálnej adresy dozadu, aj dopredu.

Funkcia **zastavenia emulácie** zastaví prácu CPU bez možnosti pokračovania. Stav CPU ostane zachovaný. Jediným možným východiskom z tejto situácie (ako je možné vidieť na Obr. 2.2) je resetovanie CPU, čo spôsobí jej re-inicializáciu. Do tohto stavu príde CPU veľakrát aj automaticky, dôvodov môže byť niekoľko:

- inštrukciou programu (`halt`)
- výpadkom inštrukcie (ak CPU narazí na neznámu alebo neúplnú inštrukciu)
- výpadkom pamäte (ak sa program pokúša pristúpiť k pamäti mimo jej adresnej oblasti)

**Pozastavením** emulácie pozastavíme prácu CPU, s možnosťou ďalšieho pokračovania. Situácia je podobná, ako keby sme emuláciu zastavili úplne, len s tým rozdielom, že je ešte možné v nej pokračovať. Ak CPU narazí na bod pozastavenia (*breakpoint*), dostane sa

presne do tohto stavu. Zavolanie tejto funkcie je ako vyvolanie umelého bodu pozastavenia.

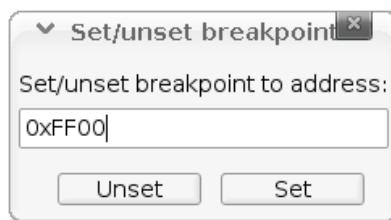
**Spustením** emulácie ju spustíme od miesta, na ktoré ukazuje programové počítadlo. Zo stavom CPU sa nič nerobí. Emuláciu nie je možné spustiť, ak je CPU zastavená úplne. V prípade, že emuláciu spustíme týmto spôsobom, prestanú sa automaticky zobrazovať inštrukcie v okne debuggera. Je to z dôvodu urýchlenia emulácie — implementácia samotného spustenia emulácie môže byť iná (výkonovo rádovo rýchlejšia, ale bez možnosti podrobnej interakcie), ako implementácia krokovania. Procesy tu prebiehajú veľmi rýchlo, plynule - tak, ako keby sme spustili skutočný počítač. Niektoré CPU môžu implementovať presnú časovú synchronizáciu, aby bolo možné regulovať rýchlosť (frekvenciu) CPU.

Ďalšou funkciou je **krok** emulácie. Po vyvolaní tejto funkcie CPU dostane priestor na vykonanie jediného kroku (inštrukcie) programu. Zmeny stavu CPU spôsobené vykonanou inštrukciou sa samozrejme zobrazia. Po vykonaní kroku sa štandardne CPU uvedie do stavu pozastavenia (teda ak vykonalý krok nespôsobil úplné zastavenie emulácie).

Ak je emulácia pozastavená, má zmysel uvažovať o „ručnom skoku“ na nejakú adresu. Tento skok znamená v podstate prepísanie hodnoty programového počítadla na hodnotu zadanú používateľom. Po kliknutí na funkciu **skoku** sa zobrazí dialógové okno, kde je používateľ vyzvaný k zadaniu adresy, na ktorú sa má „skočiť“. Okrem programového počítadla PC sa nič iné v CPU nemení.

### 5.2.3 Breakpointy na neviditeľnej adrese

Body pozastavenia, alebo *breakpointy* sa dajú nastaviť aj na „neviditeľnú“ adresu. Kliknutím na príslušnú ikonu na Obr. 5.3 sa zobrazí dialógové okno (Obr. 5.4) s výzvou na zadanie adresy operačnej pamäte, na ktorej sa nastaví/zruší breakpoint.

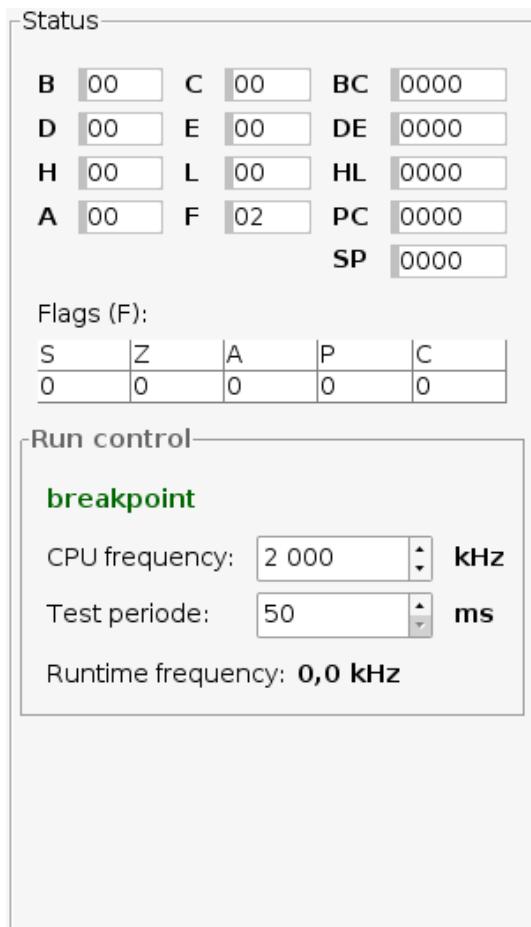


Obr. 5.4: Dialógové okno na nastavenie/zrušenie breakpointu

Čísla adres sa dajú zapisovať vo formátoch: dekadický (klasické čísla od 0 – 9), hexadeximálny (tvar 0xčíslo, kde číslo je od 0 – F), osmičkový (tvar 0číslo, kde číslo je od 0 – 7). Takýto zápis je možný vo všetkých výzvach v celom hlavnom module.

### 5.3 Stavové okno

Stavové okno zobrazuje stav CPU. Podľa implementácie CPU sa toto okno aktualizuje, keď nie je emulácia spustená permanentne. Príklad zobrazenia stavu CPU procesora Intel 8080 ukazuje Obr. 5.5.



Obr. 5.5: Stavové okno procesora Intel 8080

Okno na Obr. 5.5 je rozdelené na dve časti. Horná časť zobrazuje obsah všetkých regisťrov a príznakov procesora. Spodná časť umožňuje používateľovi manipulovať s frekvenciou CPU a umožňuje tak koordinovať jej rýchlosť.

Bližšie informácie o význame jednotlivých regisťrov a príznakov si môže čitateľ nájsť v nejakej príručke k procesoru Intel 8080.

Teraz trochu popíšem spôsob, akým sa nastavuje rýchlosť CPU. Keďže práca CPU je diskretizovaná, každý jej krok sa udeje v určitom diskrétnom čase. Počet krokov za určitú periódu sa nazýva frekvencia CPU. Čím je počet krokov za jednu periódu väčší, tým CPU pracuje rýchlejšie. Tieto kroky sú elementárne, každá inštrukcia sa skladá z niekoľkých

takýchto krokov. Na nastavenie frekvencie CPU stačí, ak používateľ zadá hodnotu od 100 do 99999 kHz do vyznačeného textového polička. Nastaviť frekvenciu je možné iba keď je emulácia pozastavená, alebo zastavená. Pri krokovaní emulácie nastavovanie frekvencie nemá význam, pretože je náročné a zbytočné presne simulať rýchlosť vykonania jedinej inštrukcie. Ak je emulácia spustená, zobrazuje sa frekvencia, s akou CPU pracuje v reálnom čase (*Runtime frequency*).

Má to význam preto, lebo zabezpečenie frekvencie sa deje na základe vzorkovania s konštantnou vzorkovacou periódou a teda presnosť zachovania frekvencie je dané touto vzorkovacou periódou. Čím je perióda menšia, tým častejšie sa koná kontrola, resp. usmerňovanie frekvencie na požadovanú hodnotu. Táto vzorkovacia perióda sa tiež dá nastaviť, v textovom poli označenom ako *Test periode*. Čím je jej hodnota menšia, tým častejšie prebieha vzorkovanie a tým je frekvencia presnejšia. Pre nižšie frekvencie sa odporúča, aby táto hodnota bola nastavená na nižšie hodnoty, a naopak pre vyššie frekvencie na vyššie hodnoty (vzorkovanie takisto stojí určitý výkon).

## 5.4 Okno zariadení

Okno zariadení, ktoré je možné vidieť na Obr. 2.1 vpravo, je iba akýmsi zoznamom obsahujúcim všetky načítané zariadenia v danej konfigurácii. Zariadenia môžu podporovať svoje vlastné, individuálne funkcie. Tieto funkcie sa aktivujú interaktívne pomocou grafického rozhrania. Na zobrazenie tohto GUI stačí kliknúť na tlačidlo *Show*. Ak ho zariadenie nepodporuje, mala by vyskočiť chybová hláška.



## Časť II

# Virtuálne architektúry



## 6 Počítač MITS Altair

Počítač Altair 8800, pomenovaný podľa planéty v jednom z prvých dielov seriálu Star Trek, bol počítač, ktorý si mohol záujemca o elektroniku postaviť za len 397 dolárov. S procesorom Intel 8080 a 256 bajtmi pamäte bez obrazovky či klávesnice to bolo podľa dnešných štandardov úplne nič. Jeho autor, Ed Roberts, pomenoval svoj vynález „personal computer“ (osobný počítač). V dnešnej dobe sa termínom PC označuje prakticky každý počítač, ktorý môžete odniesť v rukách.

Obrázok 6.1 ukazuje počítač Altair 8800 spolu s terminálom a disketovou mechanikou.



Obr. 6.1: Počítač MITS Altair 8800, spolu s terminálom LSI ADM-3A a disketovou mechanikou

Altair 8800 je jedným z najstarších komerčne dostupných osobných počítačov vôbec. Ed Roberts (zakladateľ a prezident spoločnosti MITS) predával tieto stroje poštou priamo z továrne.

Rôzni nadšenci pochopili silu Altairu a začali pre neho vytvárať programy a hardware. Pre týchto priekopníkov predstavoval Altair slobodu — akéosi uvoľnenie sa od dávkových úloh pre mainframové systémy, spravované elitou. Na fenoméne počítača, ktorý môžete mať doma na kuchynskom stole, zarobili závratné bohatstvo dvaja zbehli vysokoškolskí študenti

— v roku 1975, Paul Allen a Bill Gates (vtedy študent na Harvarde), napísali orezanú verziu jazyka BASIC, čo ich priamo posunulo k založeniu spoločnosti Microsoft.

Objavila sa rada rôzneho hardwaru s rôznymi rozdielmi a softwaroví nadšenci radostne vytvárali nové programy pre nové systémy. Táto kapitola uvádza návody, ako spustiť obrazy softvéru určeného pre počítač Altair. Väčšina obrazov dodávaných k emulátoru *emuStudio* sú prevzaté z emulátora SIMH [6]. Niektoré boli modifikované pre beh na emulátore SIMH, čím trochu strácajú na originalite svojou štruktúrou, ale rozhodne nie funkcia.

Základná konfigurácia počítača MITS Altair 8800 bola [10]:

<b>Procesor</b>	Intel 8080 alebo 8080a
<b>Rýchlosť</b>	2 MHz
<b>RAM</b>	od 256 bytov, do 64 KB
<b>ROM</b>	voliteľná. Obyčajne pamäte typu EPROM značky Intel 1702 s veľkosťou 256 bytov na každej, na ktorých boli uložené rôzne bootloader-y.
<b>Pevná pamäť</b>	voliteľne: papierové pásky, kazetové pásky, 5.25 alebo 8" diskety.
<b>Rozšírenia</b>	najprv 16 slotov, neskôr matičné dosky mali 18 slotov.
<b>Zbernice</b>	S-100
<b>Video</b>	žiadne
<b>I/O</b>	voliteľne sériová alebo paralelná karta
<b>Možnosti OS</b>	MITS DOS, CP/M, Altair Disk BASIC

Neskôr firma MITS vyvinula ďalšie verzie tohto počítača: *Altair 8800a*, *Altair 8800b* a nakoniec *Altair 680b*. Všetky tieto počítače mali podobné konfigurácie, takže ich nebudem uvádzat.

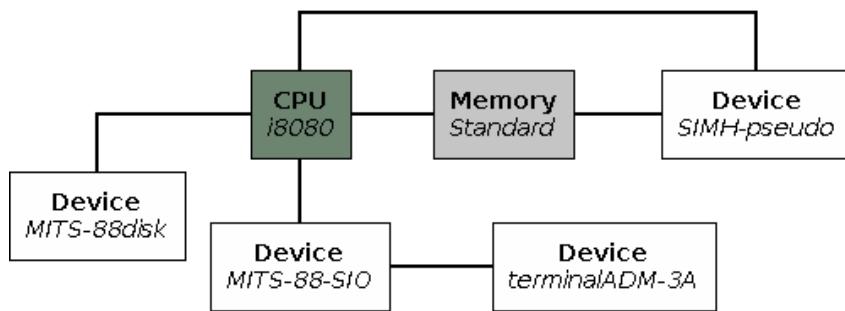
## 6.1 Schéma pre emulátor

Na Obr. 6.2 je možné vidieť abstraktnú schému, ktorá je použitá pre emulovanie tohto počítača. Niektorý softvér (ako bude uvedené ďalej) vyžaduje pre svoj beh procesor Z80 (napr. operačný systém CPM v3). V takom prípade stačí zmeniť procesor abstraktnej schémy na Z80. Väčšina programov bežiacich na 8080 bude bežať aj na procesore Z80.

## 6.2 Zavádzanie softvéru z obrazov diskiet

Ked'že je emulátor navrhnutý so zreteľom na realitu, je možné na ňom spustiť aj pôvodné verzie operačných systémov a iných programov určených pre počítač Altair 8800. Autori emulátora *simh* [6] na svojej stránke uverejňujú množstvo obrazov niektorých starších operačných systémov a programov, určených nielen pre Altair.

Nie všetky obrazy pre Altair boli odskúšané. Veľa softvéru potrebuje pre svoju funkčnosť inú konfiguráciu Altairu, ako je implementovaná v mojom emulátore. Ide o iné zariadenia,



Obr. 6.2: Abstraktné schéma počítača MITS Altair 8800 použiteľná v emulátore

ktoré zatiaľ nie sú implementované.

Odskúšané a plne funkčné sú obrazy operačných systémov CP/M v2.2 a v3, Altair DOS v1.0 a programovacieho jazyka BASIC.

### 6.2.1 Operačný systém CP/M v2.2

V období rozkvetu počítača Altair 8800 sa vyvinulo množstvo operačných systémov, programov a programovacích jazykov. Jeden z najznámejších je iste operačný systém CP/M.

CP/M (Control Program for Microcomputers) je operačný systém pôvodne navrhnutý Gary-m Kildall-om z firmy Digital Research, Inc. Najskôr išlo o jednoúlohový a jednopoužívateľský operačný systém, ktorý nepotreboval viac ako 64 KB pamäte, neskôršie verzie doplnili viacpoužívateľské varianty, a boli portované na 16-bitové procesory.

Kombinácia operačného systému CP/M a počítačov so zbernicou S-100 (8-bitové počítače podobné Altair-u) bola veľkým „priemyselným štandardom“, široko rozšíreným v 70-tych až do polovice 80-tých rokov minulého storočia. Tým, že tento operačný systém odbremenoval používateľa od potreby veľkého programovania pre nainštalovanie aplikácie na počítač, CP/M rapidne zvýšil dopyt po hardvéri, aj po softvéri.

Obraz s týmto operačným systémom obsahujúcim aj iné doplnkové programy, je v súbore `altpcm.dsk`. Postup pre jeho spustenie je nasledovný (predpokladá sa, že je spustená platforma *emuStudio* konfigurácií Altair 8800):

1. Načítanie štandardného bootloadera, ktorého úlohou je načítať operačný systém z disku
  - a) v GUI operačnej pamäte (Obr. ??) kliknite na ikonu otvorenia (importu) obrazu, je to druhá ikona v poradí na Obr. ??.
  - b) v otvorenom okne *Load an image* vyberte súbor s ROM obrazom bootloadera s názvom `boot.bin` a kliknite na tlačidlo *Load*

- c) zobrazí sa dialógové okno žiadajúce adresu operačnej pamäte, na ktorú sa má obraz načítať. Zadajte adresu 0xFF00<sup>1</sup>.

2. Namontovanie obrazu operačného systému

- a) v GUI diskového radiča 88-DISK (Obr. ??), v paneli *Configuration* overte nastavanie disku č. 0 (*Drive 0 (A)*) a kliknite na tlačidlo *Browse*
- b) v otvorenom okne *Open an image* vyberte súbor s obrazom diskety, pre operačný systém CP/M je to *altpcm.dsk* a kliknite na tlačidlo *Open*
- c) kliknite na tlačidlo *Mount*

3. Skok na adresu bootloadera

- a) v GUI emulátora, okno debuggera (Obr. 5.2) kliknite na ikonu skoku na adresu (na Obr. 5.3 tretia ikona sprava)
- b) v zobrazenom dialógovom okne *Jump* napíšte adresu 0xFF00, čo je adresa načítaného bootloadera, a kliknite na tlačidlo *OK*

4. Nastavenie parametrov CPU — nepovinný krok.

- a) V stavovom okne CPU (Obr. 5.5) nastavte požadovanú frekvenciu, odporúča sa 2000 kHz (pôvodná frekvencia CPU na počítači Altair)

5. Zobrazenie terminálu

- a) otvorte GUI terminálu (Obr. ?? vpravo)
- b) v okne terminálu kliknite na tlačidlo *Config*
- c) v zobrazenom okne konfigurácie terminálu (Obr. ??) označte checkbox *Always on top*
- d) zavrite okno konfigurácie

6. Spustenie emulácie — v okne emulátora kliknite na ikonu spustenia emulácie (piata ikona sprava na Obr. 5.3)

Po úspešnom absolvovaní všetkých týchto krokov by sa na termináli mala objaviť hláška o štarte operačného systému CP/M a potom výzva na zadanie príkazu (príkazový riadok), ako ukazuje Obr. 6.3.

Príkaz *dir* funguje, *ls* je lepšie *dir*. Podrobnejší popis tohto operačného systému a jeho príkazov môžete nájsť v [7].

---

<sup>1</sup>Tvar zápisu čísla *0xčíslo* určuje, že ide o hexadecimálne číslo. Okrem tohto zápisu existuje aj osmičkový zápis (tvar *Očíslo*) a klasický dekadický zápis



Obr. 6.3: Štart operačného systému CP/M

### 6.2.2 Operačný systém CP/M v3

Postup spustenia operačného systému CP/M verzie 3 sa veľmi nelíši od postupu uvedeného v predchádzajúcej časti. Obraz systému má názov *cpm3.dsk* a je veľký asi 1,1 MB. Súčasná verzia zásuvného modulu disketovej jednotky (popísaný v časti ??) vie zvládnuť skoro ľubovoľne veľké súbory.

Pre svoj beh systém potrebuje mať podporu bankovania operačnej pamäte. Odporúčaný je počet 8 báň, so spoločnou pamäťou od adresy *C000h*. Pre výpočet veľkosti pamäte a popis techniky bankovania nech čitateľ pozrie časť ??.

Pre samotný beh operačného systému postačí aj procesor 8080, no väčšina programov na tomto obraze potrebuje procesor Z80. Preto sa odporúča zmeniť použiť konfiguráciu s procesorom Z80.

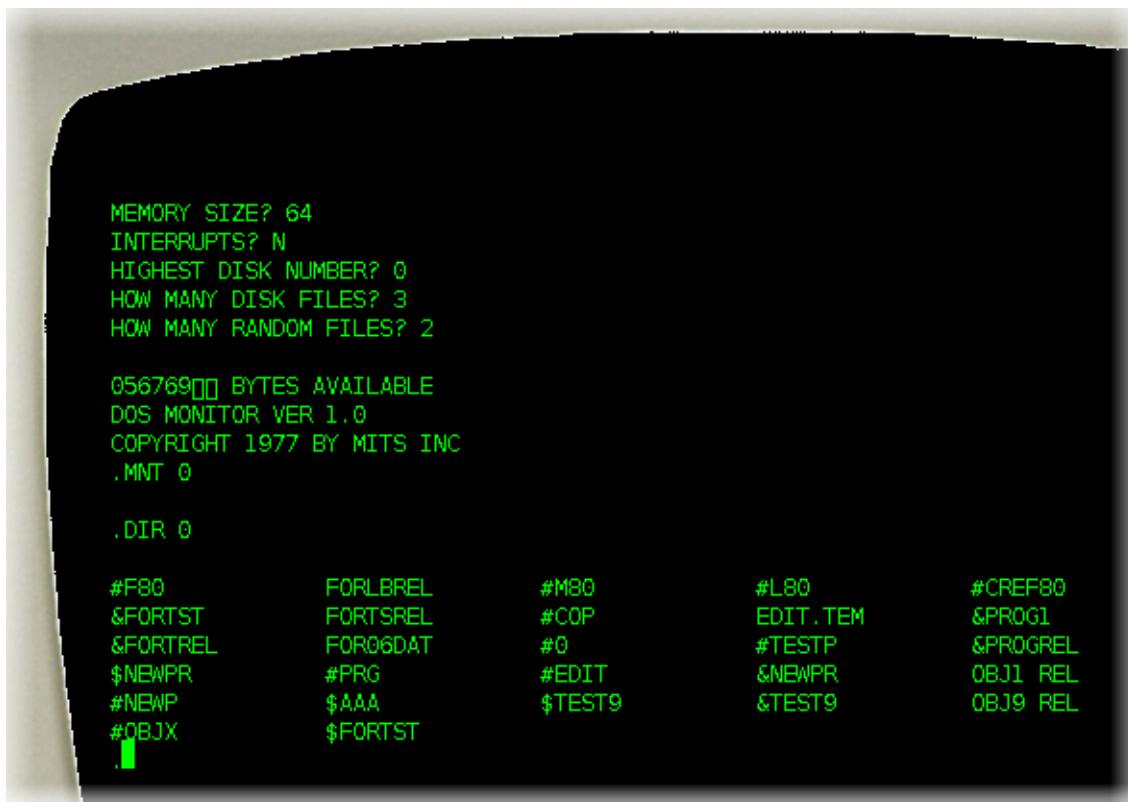
Ostatný postup spustenia systému sa nelíši od popísaného v časti 6.2.1. Snáď bolo by dobré ešte upozorniť čitateľa, že montovanie diskov je možné vykonávať aj počas behu emulácie. Pri odmontovávaní treba dávať pozor na to, aby bežiaci program disk nepoužíval. Súčasné diskové zariadenie umožňuje pripojenie až 16-tich diskov súčasne, preto ak má čitateľ k dispozícii aj iné obrazy diskov s programami spustiteľnými na tomto operačnom systéme, môže ich namontovať (pred spustením emulácie, ale aj počas jej behu). Disky v operačnom systéme sa sprístupnia napísaním písmena disku a dvojbodky v konzole, napr.: *B:* namontuje disk „*B*“, teda č. 1. Príkazom *dir* sa zobrazí jeho obsah.

### 6.2.3 Operačný systém Altair DOS v1.0

Tento operačný systém bol dlho sľubovaný už od roku 1975. Niektorí ľudia si ho v tejto dobe dopredu objednali, ale svetlo sveta uzrel až roku 1977, kedy to už skoro prestalo byť dôležité. Aj preto tento systém nie je až tak známy a neboli ani veľmi používaný. Okrem toho po krátkej „obchôdzke“ po systéme zistíte, že je o dosť slabší ako operačný systém CP/M.

Tento systém sa spúšťa rovnakým postupom, ako bol popísaný v časti 6.2.1. Rozdiel je v tom, že pri montovaní obrazu diskety použijete súbor *altdos.dsk*. Bootloader je ten istý.

Ukážka spustenia systému je na Obr. 6.4.



```

MEMORY SIZE? 64
INTERRUPTS? N
HIGHEST DISK NUMBER? 0
HOW MANY DISK FILES? 3
HOW MANY RANDOM FILES? 2

05676900 BYTES AVAILABLE
DOS MONITOR VER 1.0
COPYRIGHT 1977 BY MITS INC
.MNT 0

.DIR 0

#F80      FORLBREL    #M80        #L80        #CREF80
&FORTST   FORTSREL    #COP        EDIT.TEM   &PROG1
&FORTREL   FORT06DAT  #0          #TESTP     &PROGREL
$NEWPR     #PRG         #EDIT       &NEWPR    OBJ1 REL
$NEWPP    $AAA         $TEST9     &TEST9    OBJ9 REL
#OBJX     $FORTST

```

Obr. 6.4: Štart operačného systému Altair DOS

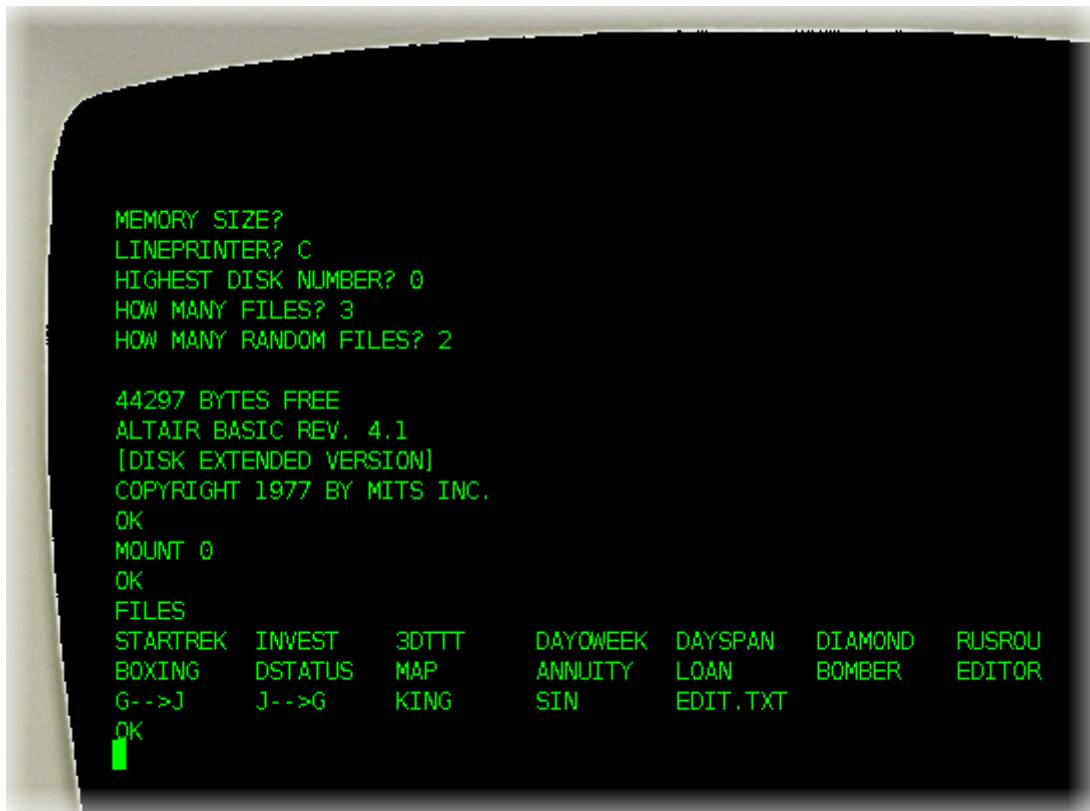
Po spustení sa operačný systém opýta pári nasledujúcich otázok. Odpovedzte mu na tieto otázky podľa Obr. 6.4. Viac o tomto systéme sa môžete dočítať v manuáli [8].

#### 6.2.4 Altair Basic v4.1

BASIC (*Beginner's All Purpose Symbolic Instruction Code*) je programovací jazyk vyvinutý na Dartmouth College v roku 1964 pod vedením J. Kemeny-ho a T. Kurtz-a. Najprv bol implementovaný pre mainframeový počítač G.E.225 (vyrobený firmou General Electric). Pôvodná myšlienka bola, aby sa dal jazyk veľmi ľahko naučiť a tiež ľahko kompilovať. Neskôr sa vývojári tohto jazyka rozhodli, aby sa stal akýmsi medzikromkom pre študentov, ktorí sa chceli učiť výkonnejšie jazyky, ako napr. FORTRAN či ALGOL.

Bill Gates a Paul Allen však mali iný úmysel. V 70-tych rokoch minulého storočia, keď bol predstavený osobný počítač MITS Altair, Allen presvedčil Gatesa, aby mu naňho pomohol portovať jazyk BASIC. Keďže firma MITS odpovedala so záujmom, začala sa tak budúcnosť jazyka BASIC aj na počítačoch PC.

Gates chodil na školu v Harvarde, Allen bol v tom čase zamestnancom firmy Honeywell. Allen a Gates predali firme MITS licenciu na ich BASIC. Táto verzia zaberala celkovo 4 KB pamäte vrátane kódu aj dát potrebných pre uchovanie interpretovaných zdrojových kódov.



Obr. 6.5: Štart jazyka Altair BASIC

Na mojom emulátore bol odskúšaný Altair Basic v4.1 (Disk Extended Version), ide o súbor obrazu diskety s názvom mbasic.dsk. Postup pre jeho zavedenie je rovnaký, ako je uvedený v časti 6.2.1.

Na Obr. 6.5 je zobrazená ukážka spustenia spomínamej verzie jazyka Basic. Po jeho spustení sa systém opýta na pár otázok, podobne ako systém Altair DOS. V tomto prípade sú však niektoré otázky a odpovede iné, postupujte podľa Obr. 6.5.

Manuál k tejto verzii jazyka sa mi nepodarilo zohnať, iba článok v časopise *Computer Notes* [9].



# 7 Abstraktný stroj RAM

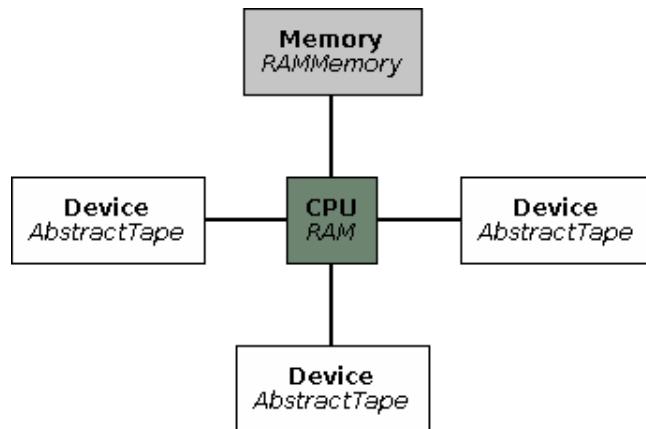
Abstraktné stroje sa používajú hlavne pre analýzu zložitosti programov. **RAM** (*Random Access Machine*) je abstraktný stroj, ktorý používa štyri pásky - vstupnú, výstupnú, pamäť registrov (dátová pamäť) a pamäť programu.

Skutočnosť, že má oddelenú pamäť dát od pamäti programu naznačuje, že stroj nie je reprezentantom Von-Neumannovej architektúry, ale má bližšie k tzv. Hardvardskej architektúre. Aj keď je emulátor stavaný na prácu s počítačmi reprezentovanými Von-Neumannovou architektúrou, úspešne sa podarilo vytvoriť emulátor aj tohto stroja.

Ekvivalent RAM stroja k univerzálnemu Turingovmu stroju (ktorý má dátá a program na jednej páske spolu), sa nazýva **RASP** (*Random Access Stored Program machine*). Tento už je príkladom Von-Neumannovej architektúry a má veľmi blízko k reálnym počítačom.

## 7.1 Schéma pre emulátor

Na Obr. 7.1 je možné vidieť abstraktnú schému, ktorá je použitá pre emulovanie tohto stroja.



Obr. 7.1: Abstraktná schéma stroja RAM



# Literatúra

- [AHK90] Paul W. Abrahams, Kathryn Hargreaves and Karl Berry. *TeX for the Impatient*. Addison-Wesley, 1990. (Available at <ftp://tug.org/tex/impatient>)
- [GPLv2] GNU General Public License, version 2  
<http://www.gnu.org/licenses/gpl-2.0.html>
- [1] MITS, Inc.: Altair 8080 Operators Manual, 1975  
<http://www.classiccmp.org/dunfield/altair/d/88opman.pdf>
- [2] Intel Corp.: 8080 Assembly Language Programming Manual, 1975  
<http://www.classiccmp.org/dunfield/r/8080asm.pdf>
- [3] Lear Siegler, Inc. (LSI): ADM-3A Operators Manual, 1979  
<http://www.classiccmp.org/dunfield/altair/d/adm3a.pdf>
- [4] MITS, Inc.: Serial I/O Board documentation, 1975  
[http://www.classiccmp.org/dunfield/s100c/mits/88sio\\_1.pdf](http://www.classiccmp.org/dunfield/s100c/mits/88sio_1.pdf)
- [5] MITS, Inc.: Disk Operators Manual, 1975  
<http://www.altair32.com/pdf/88dsk\%20manual\%20v2.pdf>
- [6] The Computer History Simulation Project  
<http://simh.trailing-edge.com/>
- [7] Digital Research: CP/M Operating System Manual  
<http://public.planetmirror.com/pub/cpm/manuals/cpm22-m.pdf>
- [8] MITS, Inc.: Altair Disk Operating System Documentation, 1977  
<http://www.classiccmp.org/dunfield/altair/d/altdos.pdf>
- [9] Altair Basic: Up and Running, Computer Notes, Vol 1, 1975  
[http://www.startupgallery.org/gallery/notesViewer.php?ii=75\\_4](http://www.startupgallery.org/gallery/notesViewer.php?ii=75_4)
- [10] The Vintage Computer My Collection of Vintage Machines  
<http://www.vintage-computer.com/altair8800.shtml>
- [11] HUDÁK, Š.: Strojovo orientované jazyky *FEI, Košice*, 2003, 218 strán, ISBN 80-969071-3-1