UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

PRINCÍPY PRÁCE FPU SÚČASNÝCH OSOBNÝCH POČÍTAČOV

2010 **Marek Mrva**

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

PRINCÍPY PRÁCE FPU SÚČASNÝCH OSOBNÝCH POČÍTAČOV

Bakalárska práca

Študijný program: Informatika
Študijný odbor: 9.2.1 Informatika
Školiace pracovisko: Katedra Informatiky
Školiteľ: RNDr. Peter Tomcsányi

Bratislava, 2010 **Marek Mrva**

Čestne prehlasujem, že som túto bakalársku prácu vypracoval samostatne len s použitím uvedenej literatúry.

. . . . . .

Abstrakt

Autor:	Marek Mrva
Názov práce:	Princípy práce FPU súčasných osobných počítačov
Univerzita:	Univerzita Komenského v Bratislave
Fakulta:	Fakulta matematiky, fyziky a informatiky
Katedra:	Katedra informatiky
Školiteľ:	RNDr. Peter Tomcsányi
Rozsah práce:	39 strán
Bratislava, 2010	

Táto bakalárska práca si kladie za cieľ popísať prostredie a fungovanie jednotky FPU, ktorá sa nachádza v dnešných bežne dostupných procesoroch. Aby sa dalo fungovanie popísaných vlastností experimentálne overiť, obsahuje práca aj aplikáciu určenú na vizualizáciu tejto jednotky, ktorá umožní používateľovi vytvárať programy a overiť si tak svoje znalosti.

Kľúčové slová: FPU, x87, IEEE 754, Simulácia

Obsah

Kapitola 1

Úvod

„Errors using inadequate data are much less than those using no data at all.“

– Charles Babbage

V dnešnej dobe berieme výpočtovú silu počítačov za samozrejmosť. Počítače zvládajúce vykonať milióny operácií za sekundu už nie sú výsadou veľkých výpočtových laboratórií a stredísk. Napísanie programu dnes už neznamená prípravu najprv na papier a následný zdĺhavý a zložitý proces prenosu na dierne štítky. Človek sa dnes už nemusí spoľahnúť na pracovníka obsluhujúceho počítač, aby doň správne vložil náš drahocenný a dlho pripravovaný program. Doba sa zmenila.

Naprogramovanie jedného matematického vzorca dnes trvá niekoľko sekúnd. Nemusíme čakať, kým nám bude pridelený čas na halovom počítači - môžeme si predsa spustiť program aj na počítači v našej obývačke. Množstvo výpočtov, ktoré môžeme v dnešnej dobe vykonať na bežne dostupnom hardvéri, stlačilo cenu výpočtov veľmi, veľmi nízko.

Aby sme dokázali plne využiť programátorské možnosti, ktoré nám súčasný[1] hardvér ponúka, potrebujeme sa často pozrieť až na najnižšiu úroveň, nakoľko podpora v programovacích jazykoch je často stále nedostatočná. V tejto práci si práve túto úroveň ukážeme, pričom sa budeme zaoberať FPU[2] od spoločnosti Intel, ako reprezentanta najrozšírenejších súčasných procesorov.

[1] Pod pojmom „súčasný“ sa v tejto práci myslí aktuálny v roku 2010.

[2] Pojmy FPU, x87 a matematický koprocesor budeme v tomto kontexte chápať ako ekvivalentné.

Kapitola 2

História rodiny x87

Aby sme boli schopní správne pochopiť príčiny návrhu architektúry koprocesorov x87, musíme sa najprv pozrieť do histórie, až do časov prvých modelov z tejto rodiny. Práve táto kapitola by mala stručne zhrnúť historický priebeh udalostí v tejto oblasti informačných technológií.

- **Rok 1978**: Spoločnosť Intel uviedla na trh procesor 8086, prvý procesor z rodiny x86. Procesor dokáže vykonávať iba základné matematické a logické operácie s celými číslami.
- **Rok 1980** priniesol uverejnenie prvého koprocesora pre rodinu x86. Jednalo sa o koprocesor 8087, ktorý mal zaplniť prázdne miesto na trhu hardvérových výpočtov s reálnymi číslami. Výrobná technológia ešte nebola v tejto dobe natoľko vyspelá, aby mohol byť koprocesor integrovaný priamo do procesora. Práve preto sa zvolil variant, pri ktorom sa umiestnil koprocesor mimo procesora, na osobitný čip.
- **Rok 1982**: Uvedenie procesora 80286 s prepracovanou správou pamäte.
- **Rok 1983**. V tomto roku bol oficiálne uvedený koprocesor 80287. Nakoľko sa zmenila štruktúra adresácie pamäte v procesore 80286, s ktorým sa mal tento čip párovať, musela sa zmeniť aj štruktúra koprocesora. Komunikácia medzi koprocesorom a procesorom tu po prvýkrát začala spôsobovať problémy s dosiahnuteľným výkonom. Ostatné funkcie zostali prakticky bez zmeny.

- **Rok 1985**: Konzorcium IEEE schválilo štandard 754. V tom istom roku uviedol Intel procesor 80386, ktorý bol kompatibilný s vtedy už populárnym koprocesorom 80287.
- **Rok 1987** znamenal uvedenie koprocesora 80387, ktorý už plne spĺňal kritériá definované štandardom IEEE 754. Kvôli tejto kompatibilite boli pridané nové inštrukcie do existujúcej inštrukčnej sady. Algoritmy operácií boli prepracované. Niektoré sa vykonali rýchlejšie ako dokázal koprocesor prijímať inštrukcie cez vonkajšie rozhranie, ktoré sa tak stalo definitívne úzkym hrdlom.
- **Rok 1989**: Uvedenie procesora 80486DX. Tento procesor, ako prvý z rodiny x86, obsahoval koprocesor integrovaný priamo v čipe procesora.
- **Rok 1991**: Intel uviedol procesor 486SX. Išlo o defektný procesor 486DX, v ktorom bol vypnutý integrovaný koprocesor. Procesor sa dal rozšíriť pomocou koprocesora 80487, čo bol v skutočnosti plne funkčný model 486DX.
- **Rok 1993**. Procesor Pentium ukončuje éru matematických koprocesorov. Od tohto dátumu je koprocesor trvalou časťou procesorov.

Kapitola 3

Dátové typy

FPU dokáže pracovať s rôznymi dátovými typmi, v závislosti od požadovanej operácie. Sú to hlavne celé a reálne čísla, a čísla vo formáte BCD. Nasledujúce podkapitoly slúžia práve na zhrnutie týchto typov, nakoľko je ich znalosť kľúčová pre správne pochopenie operácii procesora.

3.1 Celé čísla

Ide o v praxi najbežnejší číselný typ, ktorý je vo FPU reprezentovaný celými číslami v dvojkovom doplnkovom kóde. Tento formát umožňuje presne vyjadriť všetky celé čísla začínajúc -2^{p-1} a končiac $2^{p-1}-1$, pričom p je počet bitov použitých na vyjadrenie daného čísla. FPU dokáže pracovať s tromi rôznymi veľkosťami, menovite 16, 32 a 64 bitov. V tabuľke 3.1 sa nachádza zhrnutie možných rozsahov čísel pre jednotlivé veľkosti.

Každý z týchto typov môže po niektorých operáciách v jednotke FPU nadobúdať ešte aj špeciálnu hodnotu, zvanú „Integer indefinite“. Táto hodnota je použitá pri maskovaných výnimkách, a jej hodnota je rovná najmenšiemu vyjadriteľnému číslu v danom formáte.[1]

[1]Toto môže spôsobovať chyby v prípade neošetrených výnimiek. Napríklad konverzia hodnoty ∞ na celočíselný typ vloží do tohto čísla hodnotu -2^{p-1}. Pri ďalšej konverzii sa už ale použije táto hodnota, ktorá je plne platná. Toto môže mať pri behu programu neočakávateľné následky.

Tabuľka 3.1: Rozsahy celých čísiel

Názov	p	Vyjadriteľné čísla
WORD	16	$-2^{15} = -32768 \cdots 2^{15} - 1 = 32767$
DWORD	32	$-2^{31} = -2147483648 \cdots 2^{31} - 1 = 2147483647$
QWORD	64	$-2^{63} \approx -9.22 \times 10^{18} \cdots 2^{63} - 1 \approx 9.22 \times 10^{18}$

3.2 Reálne čísla

V tejto sekcii najprv prezentujeme možnosť prepisu reálneho čísla do formátu, ktorý sa dá ľahko implementovať v binárnych počítačoch, a ukážeme mapovanie týchto čísel do existujúcich premenných. Nezabudneme na opis štandardov v danej oblasti, ani ich reálnej implementácii na platforme x87 v praxi.

3.2.1 Vedecká notácia čísla

Nato, aby sme reprezentovali reálne číslo, potrebujeme nejakú rozumnú metódu jeho zápisu. Pod pojmom rozumná rozumieme v tomto prípade takú metódu, aby sa dala jednoducho a uniformne reprezentovať na binárnych počítačoch. Jedna z najrozumnejších takýchto metód sa nazýva vedecká notácia čísla. Ak by sme chceli vyjadriť číslo $x, x \neq 0$ (Špeciálny prípad pre $x = 0$ vysvetlíme v časti 3.2.1) pomocou tohto zápisu, rozdelili by sme ho na tri časti nasledovne:

znamienko S ktoré je definované nasledujúcou funkciou:
$$S = \begin{cases} 0 & \text{ak } x \geq 0 \\ 1 & \text{ak } x < 0 \end{cases}$$

exponent E vyjadruje „binárny rozsah“ čísla, alebo formálnejšie, $E = \lfloor \log_2 |x| \rfloor$

mantisa f vyjadruje frakciu binárneho čísla x, vyjadrenú matematicky ako $f = \frac{|x|}{2^E}$

Samotná hodnota čísla x je potom vyjadrená použitím vzťahu 3.1:

$$x = (-1)^S 2^E f \tag{3.1}$$

Nula so znamienkom

Pri vedeckej notácii čísla vzniká jeden problém, ktorý je potrebné adresovať. Ide o vyjadrenie hodnoty čísla 0. Aby sme boli schopní uniformne vyjadriť aj túto hodnotu, dodefinujeme funkcie potrebné pre výpočet jednotlivých častí čísla nasledovne: $\lfloor \log_2 (0) \rfloor = -\infty$, $2^{-\infty} = 0$, $\frac{0}{0} = 0$.

Pre číslo nula sú teda jednotlivé zložky jeho vedeckej notácie nasledovné: $E = -\infty$, $f = 0$. Tento fakt znamená, že budeme musieť vyhradiť pre exponent čísla 0 jednu špeciálnu hodnotu. Aj keď sa môže zdať tento zápis teraz nie veľmi vhodný, ako neskôr uvidíme v sekcii 3.2.3, má pomerne jednoduché a elegantné riešenie.

Jeden zo zaujímavých dôsledkov práve prezentovaného kódovania čísla nula je aj fakt, že sa dá táto hodnota zakódovať ako $+0$, ale aj ako -0. Táto vlastnosť sa dá použiť pri niektorých operáciách, pri ktorých záleží aj na znamienku tohoto čísla.[2]

3.2.2 Binárny formát vedeckej notácie

Aby sme boli schopní zapísať číslo x binárne, potrebujeme zapísať jednotlivé jeho zložky. So znamienkom S problém nie je, postačí nám naň jeden bit. Nakoľko je exponent E celé číslo, môžeme ho zapísať napríklad v dvojkovom doplnkovom kóde,[3] pričom pre hodnotu $-\infty$ vyhradíme najmenšie možné vyjadriteľné číslo.

Zostáva nám iba binárne reprezentovať mantisu. Z jej definície $f = \frac{|x|}{2^E}$ dostaneme po dosadení E vzťah $f = \frac{|x|}{2^{\lfloor \log_2 |x| \rfloor}}$. Ak by sme si na základe tohto vzorca spravili rozbor možných hodnôt f, ktoré môže mantisa nadobudnúť, zistili by sme, že

$$\begin{aligned} f &= 0 && \text{ak } x = 0 \\ f &= 1 && \text{ak } (\exists k)\, x = 2^k \\ f &\in (1, 2) && \text{inak} \end{aligned}$$

Ak $x = 0$, tak nám stačí nastaviť exponent na hodnotu pre toto číslo vyhradenú, takže sa nemusíme zaťažovať špeciálnym kódovaním mantisy.

[2] Typickým príkladom je delenie, pri ktorom platí: $1 \div 0 = \infty$, $1 \div (-0) = -\infty$.

[3] Pri existujúcich premenných je exponent vyjadrený vo formáte známom ako Excess-N kódovanie. Viac sa zápisu exponentu venuje kapitola 3.2.3.

To znamená, že nám postačuje zakódovať čísla z intervalu $\langle 1, 2)$. Toto kódovanie vytvoríme v dvojkovej sústave, pričom použijeme exponenty menšie rovné 0. Nasledujúca rovnosť demonštruje zápis čísla f v tomto formáte:

$$f = \sum_{i=0}^{\infty} 2^{-i} b_i = 2^0 b_0 + 2^{-1} b_1 + \cdots = b_0 + 0.5 b_1 + 0.25 b_2 + \cdots$$

pričom hodnoty $b_i, i \leq 0$ sú vyjadrené jedným bitom.
Skúsme si teraz demonštrovať tento formát zápisu čísiel na príklade s použitím čísla -13.6875.

$$-13.6875 = (-1)^1 (8 + 4 + 1 + 0.5 + 0.125 + 0.0625) = \tag{3.2a}$$
$$= (-1)^1 \left(2^3 + 2^2 + 2^0 + 2^{-1} + 2^{-3} + 2^{-4}\right) = \tag{3.2b}$$
$$= (-1)^1 2^3 \left(2^0 + 2^{-1} + 2^{-3} + 2^{-4} + 2^{-6} + 2^{-7}\right) = \tag{3.2c}$$
$$= (-1)^1 2^3 (1.1011011) \tag{3.2d}$$

Pre číslo -13.6875 by teda platilo
$S = 1$,
$E = 3$,
$b_0 = 1, b_1 = 1, b_2 = 0, b_3 = 1, b_4 = 1, b_5 = 0, b_6 = 1, b_7 = 1, (\forall i > 7)\, b_i = 0$.

Poznámka: V rovnici 3.2d je použitý skrátený formát zápisu. Tento formát sa skladá z hodnoty b_0 pred desatinnou bodkou,[4] nasledovaný hodnotami $b_1, b_2, \cdots$ za desatinnou bodkou. Ďalej už budeme používať tento skrátený formát.

3.2.3 Štandard IEEE 754

Štandard IEEE 754 [5] je rozsiahle dielo, ktoré definuje rôzne aspekty reprezentácie reálnych čísiel. My sa v tejto práci sústredíme hlavne na tie aspekty, ktoré sa týkajú platformy x87.[5] Ale ešte predtým, než si bližšie rozpíšeme vlastnosti a špecifiká tohto štandardu, opíšeme

[4] V niektorých textoch sa stretneme skôr s pojmom desatinná čiarka.

[5] Medzi vynechané časti patrí napríklad aj implementácia reálnych čísiel v desiatkovej sústave, ktorá je štandardom presne definovaná, ale nevyskytuje sa na fyzickom FPU. Záujemcov o túto problematiku môžeme odkázať priamo na tento štandard.

trošku bližšie niektoré vlastnosti formátu vedeckej notácie čísla, ktoré sme nechtiac v predchádzajúcich sekciách zanedbali. S vedeckou notáciou totižto vznikajú niektoré problémy, ak by sme ju chceli implementovať pomocou hardvérových (alebo aj softvérových) prostriedkov na reprezentáciu reálnych čísiel. Práve tieto problémy má tento štandard adresovať, pričom sa sústreďuje hlavne na:

- Problém spojený s konečnou veľkosťou
- Problémy spojené s definovateľnosťou

Teraz si ich skúsime bližšie popísať.

Problém spojený s konečnou veľkosťou

Tento praktický problém vzniká pri pokuse o reprezentáciu množiny reálnych čísel, o ktorej vieme, že jej mohutnosť je nekonečná, pomocou konečného počtu hodnôt. Podobný problém vzniká aj pri celých číslach.[6] Tam sa problém vyriešil tak, že sa zakódovala len podmnožina všetkých možných čísiel, a v prípade reálnych čísiel tomu nebude inak. Preto ani my nezakódujeme všetky reálne čísla, ale iba ich časť. Presnejšie, zakódujeme iba podmnožinu všetkých racionálnych čísiel.[7]

Problémy spojené s definovateľnosťou

Tento problém vzniká vtedy, keď má funkcia vrátiť ako výsledok hodnotu, pričom táto nie je definovaná. Napríklad, čo by malo byť výsledkom operácie $\sqrt{-4}$, alebo $\log_{10}(-4)$? Práve preto treba do nášho formátu pridať ešte aj možnosti na definovanie návratových hodnôt takýchto operácií.

Implementácia štandardu IEEE 754

Štandard IEEE definuje pre každý formát čísla zapísaný pomocou vedeckej notácie nasledujúce parametre:

[6] Problém pri reálnych číslach je nanešťastie omnoho horší, nakoľko je mohutnosť reálnych čísel väčšia ako mohutnosť celých čísiel.

[7] Čo je množina tých čísiel $\frac{p}{q}$, kde $p, q \in \mathbb{N}, q \neq 0$.

p = počet bitov použitých na vyjadrenie presnosti (mantisy)

$emax$ = maximálna hodnota exponentu

$emin$ = minimálna hodnota exponentu (tak aby platilo $emin = 1 - emax$)

Samotný vzorec na výpočet čísla x je potom iba mierne upravený s ohľadom na tieto parametre:

$$x = (-1)^S \, 2^E \, (b_0.b_1 b_2 \cdots b_{p-1}) \, , E \in \langle emin, emax \rangle$$

Poznámka: Môže sa zdať zvláštne, že $|emin| < emax$. Vysvetlenie pre tento problém podľa [3] spočíva vo fakte, že je dôležitejšie, aby sa prevrátená hodnota najmenšieho čísla zobrazila správne a nepretiekla. Potom síce platí, že prevrátená hodnota najväčšieho čísla podtečie, ale podtečenie nie je v tomto prípade také dôležité ako pretečenie.

Ďalej štandard definuje nasledovné symboly, a aj ich kódovanie:[8]

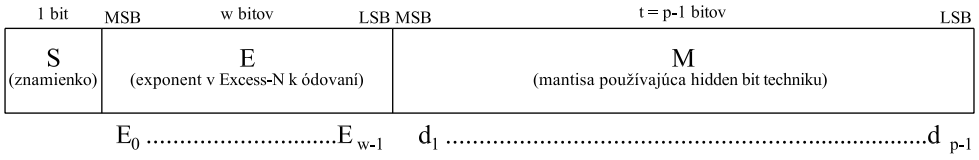
$+\infty$ = symbol pre reprezentovanie kladného nekonečna

$-\infty$ = symbol pre reprezentovanie záporného nekonečna

qNaN = symbol pre reprezentovanie NaN nevyvolávajúci výnimky

sNaN = symbol pre reprezentovanie NaN vyvolávajúci výnimky

Prejdime teraz k rozloženiu jednotlivých častí čísla. Na obrázku 3.1 je znázornené ich rozloženie vrámci registra.



Obr. 3.1: IEEE 754 formát binárneho čísla

Na **zápis exponentu** je použité Excess-N kódovanie. Pri tomto kódovaní sa exponent E zapíše pomocou binárneho kódovania, ale jeho reálna hodnota, ktorú toto číslo predstavuje, je rovná $E - N$, kde N je závislé od použitého kódovania. Napríklad pri použití Excess-127 kódu sa skutočná hodnota 45 zakóduje ako $45 + 127 = 172$. Hodnota N sa dá vyjadriť pomocou rovnice $N = 2^{w-1} - 1$, kde w je počet bitov použitých na reprezentáciu exponentu.

[8] Kódovanie týchto symbolov je bližšie popísané v tabuľke 3.2.

Na **zápis mantisy** sa použije iba konečný počet bitov, takže sú čísla ktoré dokáže vyjadriť obmedzené. Ak by sme si pozorne preštudovali formát čísla, ktorý sme použili v sekcii 3.2.2 pri definovaní binárneho formátu zistili by sme, že pre všetky čísla, až na číslo 0 je bit pred desatinnou bodkou, b_0, nastavený, teda má hodnotu 1. Ak by sme teda boli schopní zapísať číslo 0 inak, mohli by sme predpokladať že tento bit má vždy hodnotu 1 a nekódovať tak jeho hodnotu v zápise čísla. Tento spôsob sa nazýva hidden bit technika, a štandard IEEE ju používa.

V tabuľke 3.2 sa nachádza prehľad všetkých možných kódovaní symbolov definovaných štandardom IEEE 754.

Tabuľka 3.2: Kódovania reálnych čísel

Exponent	Mantisa	Reprezentovaný symbol
$-2^{w-1}+1$	$.0000\cdots$	± 0 (v závislosti od znamienka)
$-2^{w-1}+1$	$.XXXX\cdots$ [1]	denormalizované čísla
$emin = -2^{w-1}+2$ $\vdots$ $emax = 2^{w-1}-1$	$.0000\cdots$ $\vdots$ $.1111\cdots$	normálne čísla
2^{w-1}	$.0000\cdots$	$\pm\infty$ (v závislosti od znamienka)
2^{w-1}	$.0XXX\cdots$ [1]	sNaN
2^{w-1}	$.1XXX\cdots$ [1]	qNaN

[1] aspoň jeden z bitov X je nenulový

Nula, Denormalizované čísla

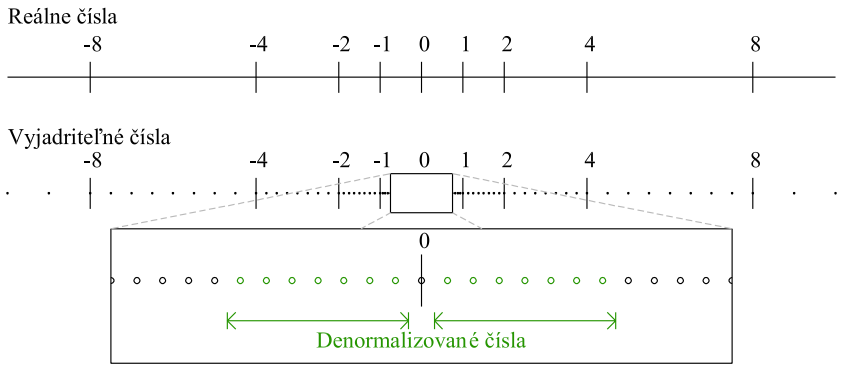
V časti 3.2.1 sme načrtli problém s definovaním čísla 0. V tejto časti si bližšie popíšeme, ako sa tento problém podarilo vyriešiť v štandarde IEEE 754.

Ako už bolo povedané, na zápis čísla nula si musíme vyhradiť aspoň jednu hodnotu exponentu. V štandarde sa zvolila najmenšia možná hodnota, teda $-2^{w-1}+1$. Naviac sa pri čísle 0 zadefinovala hodnota mantisy na $.0000\cdots$. Za týmito rozhodnutiami stoja logické argumenty.

Jedným z nich je fakt, že táto hodnota najbližšie zodpovedá hodnote $-\infty$, ktorú chceme

vyjadriť. Ďalším je fakt, že takto zapísaná nula by sa dala hardvérovo veľmi jednoducho a rýchlo overiť, nakoľko sú všetky jej bity, až na znamienko, povinne nastavené na hodnotu 0.

Pri tomto zápise čísla 0 sa ale plytvá možnými hodnotami exponentu. Aby sa tento problém aspoň čiastočne vyriešil, použijú sa ostatné hodnoty na zápis čísiel, ktoré nazveme **Denormalizované čísla**.



Obr. 3.2: Rozsahy reprezentovateľných čísiel vo vedeckej notácii

Na obrázku 3.2 sa nachádza príklad množiny čísiel, ktoré by sme dokázali v našom formáte reprezentovať. Na spodnej osi sú bodkami znázornené čísla, ktoré sa dajú v danom formáte vyjadriť. Všimnime si, že v okolí čisla 0 vznikla pomerne veľká trhlina, v ktorej nevieme zakódovať žiadne čísla. Práve túto oblasť pokryjeme denormalizovanými číslami.

Pretože používame hidden bit techniku na zakódovanie normálnych čísiel, musíme kvôli zmenšeniu hodnoty čísla zmenšiť samotný exponent. Preto pri denormalizovaných číslach nepoužijeme tútotechniku. Zároveň to ale znamená, že budeme musieť upraviť skutočnú hodnotu, ktorú predstavuje pole vyjadrujúce exponent, nakoľko už nemáme implicitne nultý bit nastavený na hodnotu 1. Táto úprava je veľmi priamočiara, najmenší exponent bude jednoducho reprezentovať číslo o 1 väčšie.

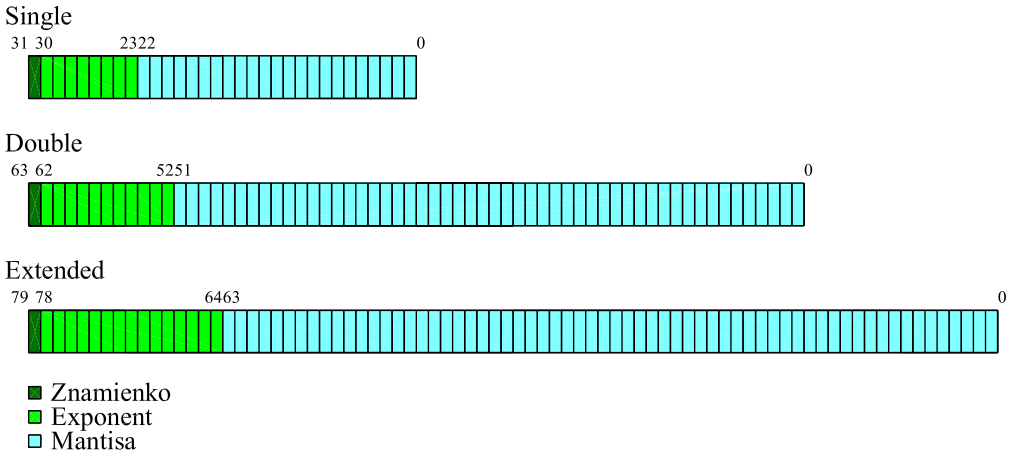
Týmto spôsobom umožníme reprezentovať veľmi malé čísla, aj keď budeme musieť počítať so stratou presnosti mantisy, nakoľko z nej musí byť vrchných k bitov nastavených na hodnotu 0, čím sa efektívne zmenší hodnota exponentu o k.

3.2.4 Existujúce formáty v jednotke FPU

V predchádzajúcej sekcii sme si predstavili štandard IEEE 754 a spôsob, akým zapisuje tento štandard reálne čísla do registrov. V tejto sekcii sa pozrieme na implementáciu tohto štandardu na platforme x87, a popíšeme si jednotlivé typy premenných, ktoré sú tu dostupné.

SINGLE, DOUBLE a EXTENDED

Na FPU sú podporované práve tieto typy premenných. Ich bitové veľkosti sú 32, 64 a 80 bitov. Obrázok 3.3 reprezentuje rozloženie a veľkosť týchto typov.



Obr. 3.3: Formáty reálnych čísiel v jednotke FPU

Formáty SINGLE a DOUBLE presne dodržujú rozloženie prvkov definované štandardom IEEE. Formát EXTENDED je špecifický pre platformu Intel x87, pričom je to variácia na formát DOUBLE EXTENDED definovaný štandardom. Tento formát bol upravený do tej miery, že jeho mantisa nepoužíva hidden bit techniku, a teda zobrazuje aj nultý bit. Táto zmena sa vykonala hlavne z dôvodu urýchlenia rozlíšenia normalizovaných a denormalizovaných čísiel.

Tabuľka 3.3 prezentuje definované rozsahy jednotlivých typov, ako aj ich približnú presnosť vyjadrenú v platných desatinných miestach (označenú ako p_{dec}).

Tabuľka 3.3: Rozsahy reálnych čísiel

Názov	p	p_{dec}[1]	Vyjadriteľné čísla
SINGLE	24	8	$2^{-126} \approx 1.17 \times 10^{-38} \cdots 2^{128} \approx 3.40 \times 10^{38}$
DOUBLE	53	16	$2^{-1022} \approx 2.22 \times 10^{-308} \cdots 2^{1024} \approx 1.79 \times 10^{308}$
EXTENDED	64	20	$2^{-16382} \approx 3.36 \times 10^{-4932} \cdots 2^{16384} \approx 1.19 \times 10^{4932}$

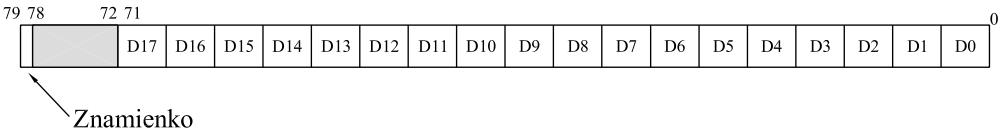
[1] platí: $p_{dec} = \lceil p \log_2 (10) \rceil$

3.3 Čísla vo formáte BCD

Predstavujú reprezentáciu celých čísiel v desiatkovej sústave. Pri tomto formáte sa jedna cifra v desiatkovej sústave zakóduje pomocou 4 bitov, pričom hodnoty ktoré môže táto štvorica nadobudnúť sú $0 - 9$. Ostatné hodnoty $10 - 15$ sú neplatné a nepoužívajú sa.

Premenná tohoto typu má veľkosť 10 bajtov, a je v nej zapísaných 18 cifier v desiatkovej sústave, $D0 \cdots D17$. Najvyšší bajt premennej sa nepoužíva na zapísanie cifier, a obsahuje iba znamienko v najvyššom bite.

Aj keď by sa do tohto bajtu dala zapísať ešte jedna cifra, je v tomto prípade rozumnejšie ju vynechať, nakoľko by potom dokázal tento formát zakódovať viac čísiel ako najväčší typ na reprezentáciu reálnych čísiel, čo by mohlo spôsobovať straty presnosti pri konverzii.



Obr. 3.4: Formát BCD čísiel v jednotke FPU

Poslednou časťou tejto sekcie je tabuľka 3.4, ktorá už obligátne prezentuje rozsah čísiel vyjadriteľných vo formáte BCD.

Tabuľka 3.4: Rozsahy BCD čísiel

Názov	p_{dec}	Vyjadriteľné čísla
BCD	18	$-10^{18} + 1 \approx -9.99 \times 10^{17} \cdots 10^{18} - 1 \approx 9.99 \times 10^{17}$

Kapitola 4

Architektúra FPU

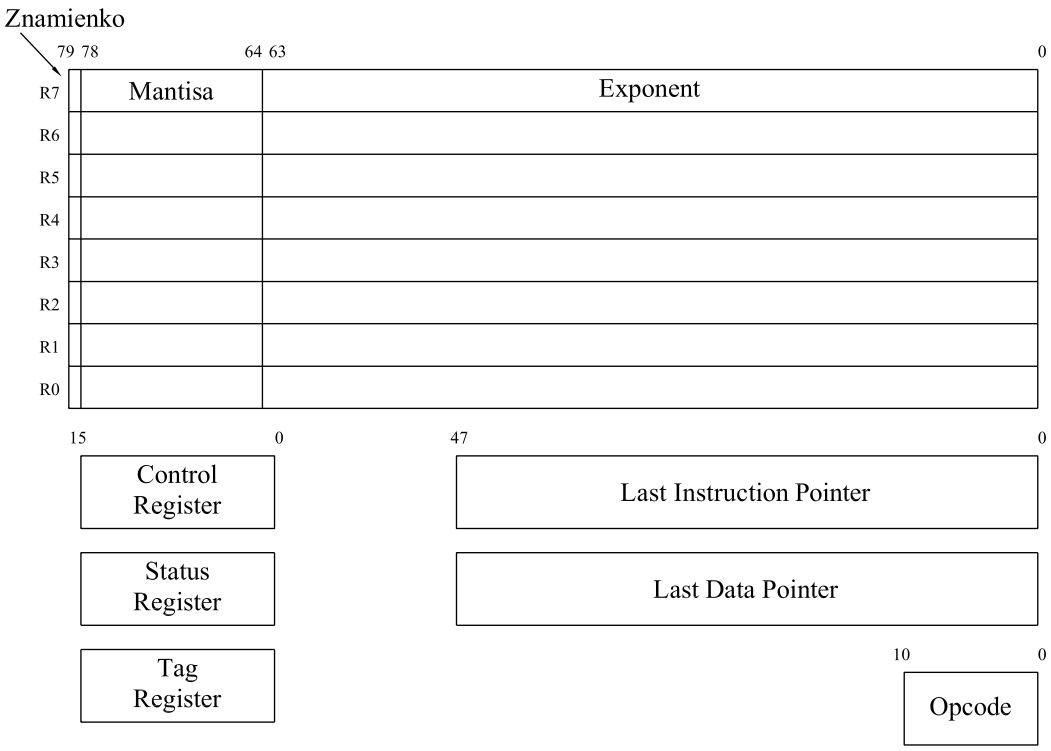
Vzhľadom na historický vývoj FPU predstavuje prostredie na prácu s reálnymi číslami samostatný celok, ktorý môže pracovať paralelne s bežnými aritmetickými operáciami procesora. Vnútorná reprezentácia sa delí na nasledujúce časti:

- Osem dátových registrov usporiadaných do zásobníka
- Control Word
- Status Word
- Tag Word
- Last instruction pointer
- Last data pointer
- Opcode register

Stav FPU je nezávislý od stavu základného prostredia procesora alebo prostredia pre rozšírenia SSE. Jediné výnimky tvoria inštrukcie ktoré pristupujú priamo k stavovému slovu (EFLAGS) procesora, do ktorého ukladajú výsledky porovnania. Tieto inštrukcie sú použité hlavne z dôvodu urýchlenia spracovania inštrukcií skoku, a sú bližšie popísané v sekcii 5.

Ďalšou výnimkou je rozšírenie MMX, nakoľko sú registre z tohto rozšírenia namapované do prostredia FPU registrov, musí byť na túto skutočnosť braný ohľad a vzájomná súčinnosť týchto rozšírení musí byť náležite ošetrená.

Na obrázku 4.1 sa nachádza prehľad architektúry FPU.



Obr. 4.1: Architektúra FPU

4.1 Dátové registre

Jednotka FPU obsahuje osem dátových registrov typu EXTENDED,[1] ktoré sú vnútorne usporiadané do podoby zásobníku. Všetky operácie, ktoré sa na matematickom koprocesore vykonávajú sú spracovávané práve v týchto registroch. Napríklad, aj operácie s číslami typu BCD musia byť najprv konvertované do tohoto typu, a až následne sa s nimi dajú vykonávať matematické operácie.

4.1.1 Dátový zásobník

Zásobník je reprezentovaný pomocou registrov R0-R7 a 3-bitovým číslom TOP, ktoré sa nachádza v registri Status Word.[2] Toto číslo udáva dátový register, ktorý sa aktuálne nachádza na vrchu zásobníka. Pri vkladaní hodnoty do zásobníka sa najprv toto číslo zmenší o 1, a následne sa do nového vrchu zásobníka vloží táto hodnota. Pri vyberaní hodnoty zo zásob-

[1] Podrobnejší opis formátu EXTENDED sa nachádza v sekcii 3.2.4.

[2] Tomuto registru sa bližšie venujeme v sekcii 4.3.

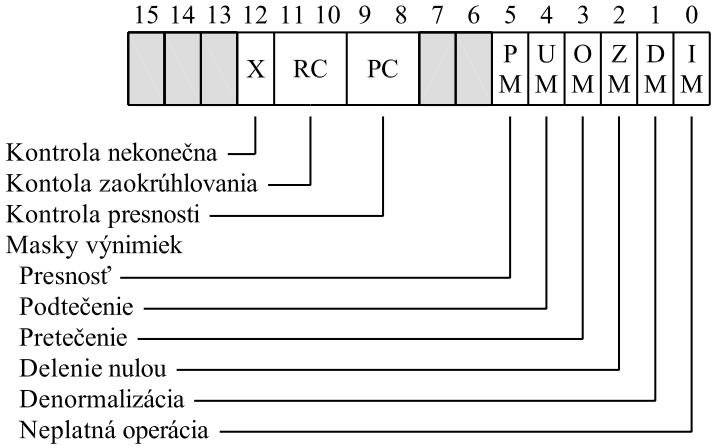
níka sa uloží číslo z vrchu zásobníka, a položka TOP sa naopak zväčší o 1.[3]

Pri zväčšení alebo zmenšení hodnoty vrcholu zásobníka sa pracuje v modulárnej aritmetike s modulom rovným 8. Teda napríklad ak platí TOP $= 7$ a zo zásobníka uložíme číslo do pamäte, tak sa bude nová hodnota vrcholu zásobníka rovnať TOP $= (7+1) = 0 \pmod 8$.

Ak je operandom instrukcie položka zo zásobníka, budeme ju označovať symbolom ST, pričom ST(0) reprezentuje vrch zásobníka, ST(1) druhý najvyšší prvok, ST(2) tretí, atď. Vo zvyšku tejto práce budeme používať rovnaké označenie.

4.2 Control Word

Tento register kontroluje, akým spôsobom sa jednotka FPU správa pri vykonávaní jednotlivých inštrukcií. Štruktúra registra je zobrazená na obrázku 4.2, zvyšok tejto sekcie sa venuje opisu jednotlivých položiek.



Obr. 4.2: Štruktúra Contol Word jednotky FPU

4.2.1 Kontrola nekonečna - X

Podľa špecifikácie [1] je tento bit zachovaný iba z dôvodu kompatibility so starším koprocesorom Intel 287, a na novších verziách x87 nemá žiaden význam.

[3] Operácie vkladania do zásobníka a vyberania zo zásobníka sa označujú aj ako PUSH a POP.

4.2.2 Kontrola zaokrúhľovania - RC

Používa sa na upresnenie módu zaokrúhľovania pri FPU operáciách. Pri vykonaní niektorej operácie vygeneruje procesor najprv presný výsledok, ktorý ale musí následne uložiť do registra konečnej veľkosti. Pri tomto ukladaní môže dojsť k strate presnosti. Práve tieto príznaky majú definovať, akým spôsobom sa má výsledok zaokrúhliť. Možné nastavenia tohoto poľa sa nachádzajú v tabuľke 4.1.

Tabuľka 4.1: Hodnoty RC - kontroly zaokrúhľovania

Zaokrúhlenie	Hodnota (binárna)
Smerom k najbližšiemu číslu[1]	00
Smerom k $-\infty$	01
Smerom k ∞	10
Smerom k 0	11

[1] Prednastavená hodnota

Nasleduje popis jednotlivých módov zaokrúhľovania:

- **Zaokrúhlenie smerom k najbližšiemu číslu** zaokrúhli na číslo najbližšie k presnému výsledku. Ak sú dve čísla rovnako blízko, zaokrúhli k párnemu z nich (k tomu s najmenej významným bitom nastaveným na 0).
- **Zaokrúhlenie smerom k** $-\infty$ zaokrúhli k najbližšiemu číslu, ktoré ale nie je väčšie ako presný výsledok.
- **Zaokrúhlenie smerom k** ∞ zaokrúhli k najbližšiemu číslu, ktoré ale nie je menšie ako presný výsledok.
- **Zaokrúhlenie smerom k 0** zaokrúhli na číslo najbližšie k presnému výsledku, pričom zaokrúhlené číslo je v absolútnej hodnote menšie ako absolútna hodnota presného čísla.

4.2.3 Kontrola presnosti - PC

Tieto príznaky kontrolujú presnosť operácií FPU, pričom umožňujú vybrať z 3 možností: 64, 53 a 24 bitov. Prednastavená hodnota je najvyššia presnosť, 64 bitov, ktorá odpovedá plnej

presnosti EXTENDED. Ostatné dve odpovedajú presnosti použitej pri typoch DOUBLE a SINGLE. Presné nastavenia jednotlivých presností sú zhrnuté v tabuľke 4.2.

Poznámka: Jedným z problémov pri špecifikovaní menšej presnosti operácií je fakt, že sa v skutočnosti zmenší iba počet používaných bitov mantisy, ale už nie exponentu, čo znamená že niektoré operácie (väčšinou týkajúce sa pretečenia a podtečenia) môžu vrátiť iný výsledok ako keby sa použil natívny formát danej presnosti.[4]

Tabuľka 4.2: Hodnoty PC - kontroly presnosti

Presnosť	Hodnota (binárna)
SINGLE (24 bitov)	00
Rezervované	01
DOUBLE (53 bitov)	10
EXTENDED (64 bitov)[1]	11

[1] Prednastavená hodnota

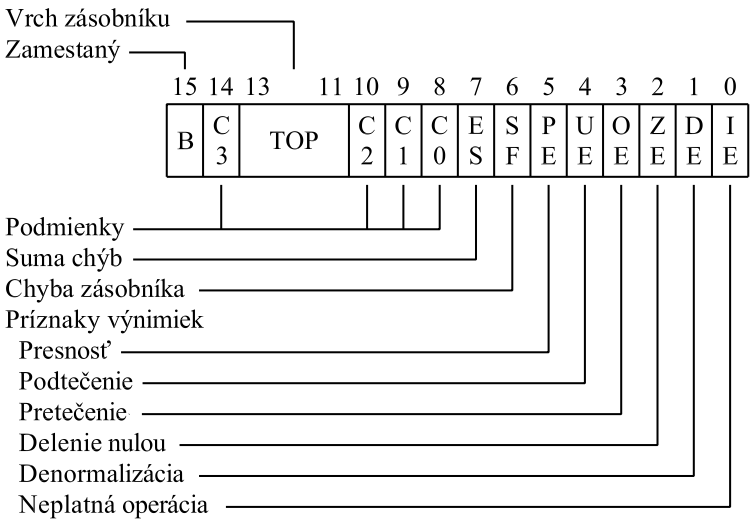
4.2.4 Masky výnimiek - *M

Slúžia na potlačenie generovania výnimiek pri práci FPU. Každý z týchto príznakov má v Status Word odpovedajúci príznak, ktorý potláča, maskuje. V sekcii 4.3.4 sú bližšie popísané jednotlivé výnimky, aj správanie pri ich maskovaní.

4.3 Status Word

Tento register obsahuje aktuálny stav jednotky FPU. Štruktúra registra je zobrazená na obrázku 4.3. Medzi najdôležitejšie časti patrí signalizácia výnimiek. Tieto bity, ktoré ich signalizujú, ostanú nastavené pokiaľ sa explicitne nevynulujú. Zvyšok tejto sekcie opisu jednotlivé položky registra.

[4] Záujemcov o túto problematiku môžeme odkázať na [6].



Obr. 4.3: Štruktúra Status Word jednotky FPU

4.3.1 Zamestnaný - B

Tento bit je zachovaný iba kvôli kompatibilite so starším koprocesorom 8087 a odzrkadľuje obsah príznaku ES.

4.3.2 Podmienky - C0-C3

Tieto 4 bity určujú výsledok aritmetických operácii a operácii porovnania. Používajú sa hlavne na rozhodovanie pri podmienených skokoch a na uchovanie informácií potrebných pri spracovaní výnimiek. Rôzne funkcie FPU nastavujú rôzne bity týchto príznakov. Prehľad nastavenia týchto bitov pri operáciách porovnania sa nachádza v tabuľke 5.2.

4.3.3 Vrch zásobníka - TOP

Ukazuje na vrchol zásobníka. Viď sekciu 4.1.1.

4.3.4 Suma chýb - ES

Tento príznak je nastavený vtedy, keď je ľubovolný z príznakov výnimiek nemaskovaný. Keď sa tento príznak nastaví, vygeneruje sa výnimka, ktorá sa následne musí ošetriť softvérovo.

4.3.5 Chyba zásobníka - SF

Tento príznak indikuje pretečenie alebo podtečenie zásobníka, ktoré nastávajú v momente, keď sa inštrukcia snaží vložiť dáta do neprázdneho registra alebo zapísať dáta z prázdneho registra. Medzi týmito stavmi sa rozlišuje pomocou príznaku C1, ktorý nadobúda hodnoty 1 pri pretečení a 0 pri podtečení zásobníka.

Keď nastane chyba zásobníka, nastaví sa príznak IE na hodnotu 1. Ak je tento príznak maskovaný, operácia ktorá spôsobila chybu zásobníka uloží symbol nedefinovanej hodnoty. Ak nie je maskovaný, vyvolá sa softvérové ošetrenie výnimky, ale operandy inštrukcie zostanú nezmenené.

4.3.6 Presnosť - PE

Nastaví sa vtedy, keď sa výsledok operácie nedá presne reprezentovať vo výslednom formáte.

Ak nastane chyba presnosti a tento príznak je maskovaný, tak sa do výsledného formátu uloží zaokrúhlená hodnota podľa hodnoty RC. Ak nie je maskovaný, tak sa naviac vyvolá softvérové ošetrenie výnimky.[5]

4.3.7 Podtečenie - UE

Nastaví sa vtedy, keď je zaokrúhlený výsledok operácie menší ako najmenšie prípustné normalizované číslo daného typu.

Ak nastane chyba podtečenia a tento príznak je maskovaný, tak sa do výsledného formátu uloží denormalizované číslo. Ak nie je maskovaný, tak sa do výsledku uloží exponent vynásobený konštantou 2^{24576} a bit C1 sa nastaví, ak sa výsledok zaokrúhlil nahor. Následne sa vyvolá softvérové ošetrenie výnimky.[6]

4.3.8 Pretečenie - OE

Nastaví sa vtedy, keď je zaokrúhlený výsledok operácie väčší ako najväčšie prípustné číslo daného typu.

[5] Ak ale nastane táto chyba v súvislosti s ukladaním do pamäte, tak sa tento príznak nenastaví ale vynuluje sa príznak C1.

[6] Ak ale nastane chyba v súvislosti s ukladaním do pamäte, tak sa ukladanie neuskutoční, ošetrenie výnimky sa stále vyvolá.

Ak nastane chyba pretečenia a tento príznak je maskovaný, tak sa do výsledného formátu uloží číslo podľa aktuálnej hodnoty RC. Ak nie je maskovaný, tak sa do výsledku uloží exponent vynásobený konštantou 2^{-24576} a bit C1 sa nastaví, ak sa výsledok zaokrúhlil nahor. Následne sa vyvolá softvérové ošetrenie výnimky.[7]

4.3.9 Delenie nulou - ZE

Tento príznak sa nastaví, keď sa inštrukcia pokúsi vydeliť konečné nenulové číslo nulou.

Ak nastane delenie nulou a tento príznak je maskovaný, tak sa do výsledného formátu uloží hodnota podľa inštrukcie, ktorá výnimku spôsobila. Ak nie je výnimka maskovaná, tak sa zavolá softvérové ošetrenie výnimky.

4.3.10 Denormalizácia - DE

Nastaví sa, keď aritmetická operácia pracuje s denormalizovanými číslami, alebo sa denormalizované hodnoty vkladajú do dátových registrov.

Ak nastane výnimka, pričom je maskovaná, operácia pokračuje bez zmeny. Ak nie je maskovaná, tak sa spracovanie preruší a zavolá sa softvérové ošetrenie výnimky.

4.3.11 Neplatná operácia - IE

Nastaví sa pri rôznych neplatných operáciách, najmä pri použití symbolu NaN ako operandu. Takisto pri pretečení registrového zásobníka.

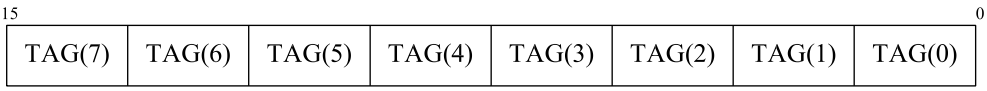
Ak je príznak maskovaný, vráti operácia ktorá spôsobila výnimku hodnotu qNaN. Ak príznak nie je maskovaný, operandy sa nezmenia a vyvolá sa softvérové ošetrenie výnimky.

4.4 Tag Word

Tento register slúži na zapamätanie stavu jednotlivých dátových registrov jednotky FPU. Jeho popis sa nachádza na obrázku 4.4.

Každá z ôsmych štruktúr môže nadobudnúť hodnoty, ktoré reprezentujú aktuálnu hodnotu daného registra. Umožňuje špecifikovať, či daný register obsahuje valídne číslo, nulu,

[7] Ošetrenie výnimky pri ukladaní do pamäte sa správa rovnako ako pri podtečení.



Obr. 4.4: Štruktúra Tag Word jednotky FPU

špeciálne číslo alebo či je prázdny. V tabuľke 4.3 sa nachádza kódovanie týchto hodnôt.

Tabuľka 4.3: Kódovanie obsahu Tag poľa

Obsah	Hodnota (binárna)
Valídne číslo	00
Nula	01
Špeciálne číslo (NaN, $\pm\infty$, denormalizované)	10
Prázdny register[1]	11

[1] Prednastavená hodnota

Aby sa dal zistiť Tag asociovaný s niektorým registrom, musí sa vypočítať jeho poloha pomocou TOP poľa, ktoré sa nachádza v registri Status Word.

Aplikácie môžu zmeniť register Tag Word iba pri obnovovaní stavu celého prostredia z pamäte pomocou príslušnej inštrukcie. V tomto prípade ale FPU použije hodnotu tagu iba na zistenie, či je daný register prázdny, a ak nie je, tak hodnotu tagu vypočíta na základe obsahu registra.

4.5 Last instruction pointer

Tento register obsahuje adresu poslednej spracovanej neriadiacej inštrukcie. Táto hodnota sa používa hlavne pri softvérovom spracovaní výnimiek, a umožňuje zistiť príkaz, v ktorom výnimka nastala.

4.6 Last data pointer

Obsahuje adresu pamäťového operandu, ktorý použila posledná spracovaná neriadiaca inštrukcia. Ak inštrukcia nepoužívala pamäťový operand, obsah tohoto registra je nedefinovaný.

4.7 Opcode register

Register obsahuje spodných 11 bitov operačného kódu poslednej spracovanej neriadiacej inštrukcie, ktorý ju jednoznačne identifikuje.[8]

[8] Inštrukcie FPU majú totiž vždy veľkosť 2 bajty, pričom prvý bajt začína sekvenciou, ktorá dostala názov ESC [Escape]. Táto sekvencia je tvaru 11011 (je to bitový reťazec dĺžky 5) a je pre každú inštrukciu zhodná.

Kapitola 5

Inštrukcie FPU

V tejto kapitole sa bližšie pozrieme na inštrukcie podporované na jednotke FPU a stručne si opíšeme ich správanie. Podrobnejšie informácie o jednotlivých inštrukciách nájde čitateľ v [2].

Budeme tu používať pojmy PUSH a POP, ktoré sme zaviedli v súvislosti s FPU registrovým zásobníkom. Tieto operácie sú bližšie opísané v sekcii 4.1.1. Symbolom ST(i) budeme označovať register daný operandom danej inštrukcie.

5.1 Inštrukcie presunu dát

5.1.1 FLD, FILD, FBLD

Vykonajú operáciu PUSH na registrovom zásobníku, pričom vložia reálne číslo (FLD), Integer (FILD), alebo BCD číslo (FBLD) z pamäte na nový vrchol zásobníka ST(0). Ak je to potrebné, vykonajú typovú konverziu.

5.1.2 FST, FIST

Uložia reálne číslo z vrcholu zásobníka ST(0) do pamäte v podobe reálneho čísla (FST) alebo Integer-u (FIST) zaokrúhleného podľa aktuálneho nastavenia poľa RC. Následne vykonajú typovú konverziu, ak je to potrebné.

5.1.3 FSTP, FISTP, FBSTP

Inštrukcie FSTP a FISTP sú ekvivalentom inštrukcií FST a FIST, inštrukcia FBSTP uloží vrch zásobníka v podobe BCD čísla do pamäte. Všetky inštrukcie následne vykonajú operáciu POP na registrovom zásobníku.

5.1.4 FISTTP

Táto inštrukcia vykoná rovnakú činnosť ako inštrukcia FISTP, ale s tým rozdielom, že zaokrúhli číslo vždy smerom k nule.

Táto inštrukcia bola pridaná spolu s rozšírením SSE3.

5.1.5 FXCH

Inštrukcia vymení obsah registra ST(0) so zvoleným registrom ST(i).

5.1.6 FCMOVcc

Inštrukcia vloží obsah zvoleného registra ST(i) do registra ST(0), ak platí zvolená podmienka. Podmienka sa vyhodnotí podľa obsahu registra EFLAGS. Možné kódovania inštrukcie sú dané tabuľkou 5.1. Možnosti nastavenia registru EFLAGS sú popísané v tabuľke 5.2.

Tabuľka 5.1: Možné podmienky inštrukcie FCMOVcc

Názov inštrukcie	Hodnota EFLAGS	Vysvetlenie
FCMOVB	CF=1	Je menšie
FCMOVNB	CF=0	Nie je menšie
FCMOVE	ZF=1	Je rovné
FCMOVNE	ZF=0	Nie je rovné
FCMOVBE	CF=1 alebo ZF=1	Je menšie alebo rovné
FCMOVNBE	CF=0 alebo ZF=0	Nie je menšie alebo rovné
FCMOVU	PF=1	Je NaN
FCMOVNU	PF=0	Nie je NaN

5.2 Základné aritmetické inštrukcie

5.2.1 FADD, FIADD

Tieto inštrukcie vykonajú matematickú operáciu sčítania, pričom pracujú s registrami a reálnymi číslami (FADD), alebo s Integer-mi (FIADD) po náležitej typovej konverzii.

5.2.2 FSUB, FISUB

Formát je zhodný s inštrukciami FADD, FIADD, až na fakt že tieto inštrukcie vykonávajú operáciu odčítania.

5.2.3 FMUL, FIMUL

Formátom sú tieto inštrukcie zhodné s FADD, FIADD, rozdiel je len vo vykonávanej aritmetickej operácii, ktorou je v tomto prípade násobenie.

5.2.4 FDIV, FIDIV

Tieto inštrukcie sú formátom zhodné s inštrukciami FADD, FIADD. Narozdiel od tých inštrukcií ale tieto vykonávajú aritmetickú operáciu delenia.

5.2.5 FADDP, FSUBP, FMULP, FDIVP

Tieto inštrukcie vykonajú rovnakú operáciu ako inštrukcie FADD, FSUB, FMUL a FDIV, s tým rozdielom že po ukončení operácie tieto inštrukcie vykonajú operáciu POP na registrovom zásobníku.

5.2.6 FSUBR, FSUBRP, FDIVR, FDIVRP

Tieto inštrukcie vykonajú rovnaký typ operácie ako ich protiklady, inštrukcie FSUB, FSUBP, FDIV, FDIVP s tým rozdielom, že tieto inštrukcie vykonajú danú aritmetickú operáciu v opačnom poradí.[1]

[1]Napríklad, ak by inštrukcia FSUB vykonala operáciu ST(0) $\leftarrow$ ST(0) $-$ ST(i), tak inštrukcia FSUBR s rovnakým operandom by vykonala operáciu ST(0) $\leftarrow$ ST(i) $-$ ST(0).

5.2.7 FABS

Táto inštrukcia vloží do registra ST(0) absolútnu hodnotu tohoto registra.

5.2.8 FCHS

Táto inštrukcia zmení znamienko čísla uloženého vo vrchole zásobníka, v registri ST(0).

5.2.9 FSQRT

Táto inštrukcia nahradí aktuálny vrchol zásobníka, teda register ST(0), jeho odmocninou.

5.2.10 FPREM, FPREM1

Tieto inštrukcie vypočítajú čiastočný zvyšok p_{rem}, pričom platí: $p_{rem} = \text{ST(0)} - (n \times \text{ST(1)})$. Hodnota n je Integer, ktorý sa získa z výrazu $n = \frac{\text{ST(0)}}{\text{ST(1)}}$ buď zaokrúhlením na dolnú celú časť (FPREM) alebo k najbližšiemu celému číslu (FPREM1).

Rozdiel medzi týmito operáciami vznikol hlavne z faktu, že spoločnosť Intel navrhla inštrukciu FPREM ešte pred príchodom štandardu IEEE 754 [4]. Jej definícia sa líšila iba mierne, ale Intel, kvôli zachovaniu kompatibility s existujúcim softvérom, vytvoril novú inštrukciu ktorá už spĺňala nový štandard, FPREM1.

5.2.11 FRNDINT

Táto inštrukcia zaokrúhli obsah registra ST(0) na Integer, pričom berie do úvahy aktuálne nastavenie poľa RC.

5.2.12 FXTRACT

Rozloží obsah registra ST(0) na exponent a mantisu, uloží exponent do registra ST(0) a vykoná operáciu PUSH, pričom na vrch zásobníka vloží mantisu.

5.3 Inštrukcie porovnania

5.3.1 FCOM, FUCOM, FICOM

Tieto inštrukcie porovnajú obsah registra ST(0) s registrom, reálnym číslom v pamäti (FCOM, FUCOM) alebo Integer-om (FICOM). Výsledok porovnania zapíšu do príznakov C0, C2 a C3 v Status Word. Ak je aspoň jeden zo vstupných operandov symbol NaN, tak inštrukcia FCOM vygeneruje výnimku neplatnej operácie.

Tabuľka 5.2 obsahuje prehľad nastavenia výsledných príznakov, vzhľadom na vzťah porovnávaných operandov.

Tabuľka 5.2: Nastavenie príznakov pri inštrukciách porovnania

Vzájomný vzťah čísiel	C3 [ZF[1]]	C2 [PF[1]]	C0 [CF[1]]
ST(0) $>$ operand	0	0	0
ST(0) $<$ operand	0	0	1
ST(0) $=$ operand	1	0	0
Aspoň jeden z operandov je NaN	1	1	1

[1] Platí pre operácie FCOMI a FUCOMI.

5.3.2 FCOMI, FUCOMI

Tieto inštrukcie sa správajú obdobne ako FCOM a FUCOM s tým rozdielom, že vedia porovnať navzájom iba registre v zásobníku a výsledok operácie vložia do registra EFLAGS.

5.3.3 FCOMP, FUCOMP, FICOMP, FCOMIP, FUCOMIP

Tieto inštrukcie vykonajú presne tú istú operáciu ako inštrukcie FCOM, FUCOM, FICOM, FCOMI a FUCOMI s tým rozdieľom, že tieto inštrukcie dodatočne vykonajú operáciu POP na registrovom zásobníku.

5.3.4 FCOMPP, FUCOMPP

Tieto inštrukcie sa správajú rovnako ako inštrukcie FCOM a FUCOM, ale navyše vykonajú dvakrát operáciu POP na registrovom zásobníku.

5.3.5 FTST

Táto inštrukcia sa správa rovnako ako inštrukcia FCOM, až na fakt, že porovná obsah registra ST(0) s hodnotou 0.0.

5.4 Logaritmické inštrukcie

5.4.1 FYL2X

Táto inštrukcia vloží do registra ST(1) hodnotu ST(1) $\times \log_2$ (ST(0)) a vykoná operáciu POP na registrovom zásobníku.

5.4.2 FYL2XP1

Táto inštrukcia vloží do registra ST(1) hodnotu ST(1) $\times \log_2$ (ST(0) + 1.0) a vykoná operáciu POP na registrovom zásobníku. Táto inštrukcia je určená primárne pre hodnoty ST(0) blízke nule.

5.4.3 F2XM1

Táto inštrukcia vloží do registra ST(0) hodnotu $2^{\text{ST(0)}-1}$. Vstup pre túto inštrukciu môže byť iba z intervalu $\langle -1, 1 \rangle$.

5.4.4 FSCALE

Uloží hodnotu ST(0) $\times\ 2^{\lfloor \text{ST(1)} \rfloor}$ do registra ST(1) a vykoná operáciu POP na registrovom zásobníku.

5.5 Trigonometrické inštrukcie

5.5.1 FSIN, FCOS

Tieto inštrukcie nahradia obsah registra ST(0) hodnotou sínus (FSIN) alebo kosínus (FCOS), ktoré sú vypočítane z hodnoty registra ST(0).

5.5.2 FSINCOS

Táto inštrukcia vypočíta hodnoty sínus a kosínus registra ST(0). Hodnotu tohoto registra nahradí sínusom, vykoná operáciu PUSH a kosínus uloží sa vrch zásobníka.

5.5.3 FPTAN

Nahradí obsah registra ST(0) jeho tangensom, vykoná operáciu PUSH a uloží hodnotu 1 na nový vrch zásobníka.

5.5.4 FPATAN

Vypočíta arkus-tangens z hodnoty $\frac{\text{ST(1)}}{\text{ST(0)}}$, vloží túto hodnotu do registra ST(1) a vykoná operáciu POP na registrovom zásobníku.

5.6 Inštrukcie pracujúce s konštantami

5.6.1 FLDZ, FLD1

Tieto inštrukcie vykonajú operáciu PUSH a vložia hodnotu $+0.0$ (FLDZ) alebo $+1.0$ (FLD1) na vrch zásobníka.

5.6.2 FLDPI

Táto inštrukcia vykoná operáciu PUSH a uloží hodnotu π na vrch zásobníka.

5.6.3 FLDL2T, FLDL2E, FLDLG2, FLDLN2

Tieto inštrukcie vykonajú operáciu PUSH a uložia hodnotu $\log_2(10)$ (FLDL2T), $\log_2(e)$ (FLDL2E), $\log_{10}(2)$ (FLDLG2) alebo $\log_e(2)$ (FLDLN2) na vrch zásobníka.

5.7 Riadiace inštrukcie

5.7.1 FINIT, FNINIT

Tieto inštrukcie inicializujú stav FPU. Inštrukcia FNINIT ignoruje prebiehajúce nemaskované výnimky.

5.7.2 FLDCW, FSTCW, FNSTCW, FSTSW, FNSTSW

Tieto inštrukcie načítajú z pamäte (FLDCW) alebo uložia do pamäte (FSTCW, FNSTCW) stav Control Word alebo uložia stav Status Word (FSTSW, FNSTSW) do pamäte alebo registra AX v procesore. Inštrukcie FNSTCW a FNSTSW ignorujú prebiehajúce nemaskované výnimky.

5.7.3 FCLEX, FNCLEX

Inštrukcie nastavia všetky príznaky výnimiek v Status Word na hodnotu 0. Inštrukcia FNCLEX ignoruje prebiehajúce nemaskované výnimky.

5.7.4 FLDENV, FSTENV, FNSTENV

Tieto inštrukcie načítajú z pamäte (FLDENV) alebo uložia (FSTENV, FNSTENV) stav FPU Prostredia do pamäte.[2] Inštrukcia FNSTENV ignoruje prebiehajúce nemaskované výnimky.

5.7.5 FRSTOR, FSAVE, FNSAVE

Tieto inštrukcie načítajú z pamäte (FRSTOR) alebo uložia do pamäte (FSAVE, FNSAVE) stav celého FPU. Inštrukcia FNSAVE ignoruje prebiehajúce nemaskované výnimky.

5.7.6 FINCSTP, FDECSTP, FFREE

Inštrukcia FINCSTP zvýši a inštrukcia FDECSTP zníži hodnotu poľa TOP v Status Word. Inštrukcia FFREE nastaví Tag pole registra ST(i) na hodnotu 11 (prázdny).

[2]Toto prostredie pozostáva z nasledovných položiek: Control Word, Status Word, Tag Word, Last instruction pointer, Last data pointer a Opcode register.

5.7.7 FNOP

Nevykoná žiadnu operáciu.

5.7.8 WAIT, FWAIT

Tieto inštrukcie skontrolujú prebiehajúce nemaskované výnimky. Ak sú niektoré výnimky neošetrené, zavolá sa príslušné softvérové ošetrenie.

5.8 Nepodporované inštrukcie

5.8.1 FENI, FDISI, FSETPM

Tieto inštrukcie slúžili pri starších koprocesoroch (pred modelom 387) na synchronizáciu s procesorom. Na novších modeloch už neplnia žiadnu úlohu, a pri ich spustení sa nevykoná žiadna operácia.

Kapitola 6

Aplikácia ViewFPU

Od uvedenia prvého koprocesora z rodiny x87 na trh prešlo k dnešnému dňu už takmer 30 rokov, no napriek tomu je možnosť užívateľa odskúšať si jeho prácu v praxi, na najnižšej úrovni, veľmi obmedzená. Práve aplikácia ViewFPU by mala zaplniť túto medzeru, pričom ponúka užívateľovi interaktívne rozhranie, pomocou ktorého si môže rýchlo a jednoducho overiť zmeny, ktoré nastanú pri jednotlivých inštrukciách.

6.1 Špecifikácia aplikácie

Táto aplikácia bola naprogramovaná s cieľom umožniť užívateľovi interaktívne pracovať v prostredí FPU a sledovať zmeny, ktoré sa vykonávajú pri práci v tomto prostredí. Medzi základné operácie, ktoré aplikácia zvláda, patrí:

- Vytváranie programov zložených z FPU inštrukcií
- Ukladanie a načítavanie programov zo súboru
- Interaktívna zmena vnútorných registrov
- Krokovanie programov
- Zvýrazňovanie zmien, ktoré sú spôsobené inštrukciami
- Vytváranie typovo overených operandov
- Automatické dopĺňanie inštrukcií a operandov

- Vyhľadávanie v zozname inštrukcií

6.1.1 Systémové požiadavky

Aplikácia bola naprogramovaná v prostredí Delphi, pričom cieľový operačný systém aplikácie je **Microsoft Windows XP**, pod ktorým bola aplikácia aj testovaná.

Čo sa týka hardvérových požiadaviek, aplikácia používa k simulácii priamo prostriedky dostupné na danej platforme. Preto potrebuje ku svojej korektnej činnosti procesor **Pentium Pro**, alebo novší.

6.1.2 Bezpečnosť

Aplikácia obsahuje vlastný assembler, ktorý generuje inštrukcie za behu a tieto následne aj spúšťa. Tento fakt by mohol predstavovať bezpečnostné riziko za predpokladu, že by útočník použil inštrukcie schopné narušiť beh systému.

Aby sa predišlo podobným problémom, aplikácia používa striktnú bezpečnostnú politiku. Táto politika umožňuje vytvárať a spúšťať iba známe inštrukcie, čím znemožňuje podobné útoky.

6.1.3 Podporované inštrukcie

Medzi inštrukcie, ktoré môže užívateľ použiť v aplikácii ViewFPU, patria všetky podporované inštrukcie koprocesorov rodiny x87. Tieto inštrukcie sú uložené vo vlastnej databáze, ktorá umožňuje rýchlo a jednoducho pridávať nové inštrukcie, prípadne odoberať existujúce.[1]

Do sady inštrukcií boli ešte pridané niektoré inštrukcie z procesora, menovite inštrukcie skoku, a inštrukcia SAHF, ktorá vloží obsah registra AH do príznakov procesora, a umožňuje tak simulovať starší spôsob vykonávania skokov.

Poznámka: Nakoľko bola inštrukcia FISTTP pridaná až s rozšírením SSE3, je na začiatku zistená podpora tohoto rozšírenia, a v prípade jeho neprítomnosti nevykoná inštrukcia žiadnu

[1] Táto vlastnosť nám môže prísť vhod vtedy, ak by sme chceli simulovať pôvodný koprocesor Intel 8087. V tomto prípade by nám stačilo ubrať inštrukcie ktoré sa zaviedli až na neskorších koprocesoroch.

operáciu. Pri ostatných inštrukciách sa takáto kontrola už nevykonáva, z čoho plynie obmedzenie na procesor spomenuté v časti 6.1.1.

6.1.4 Príklady programov

Súčasťou ViewFPU sú aj príklady naprogramované v tejto aplikácii. Tieto príklady slúžia ako ukážka schopností FPU a ukazujú typické úkony, ktoré by sa mohli vykonávať v danom prostredí.

6.2 Užívateľské rozhranie

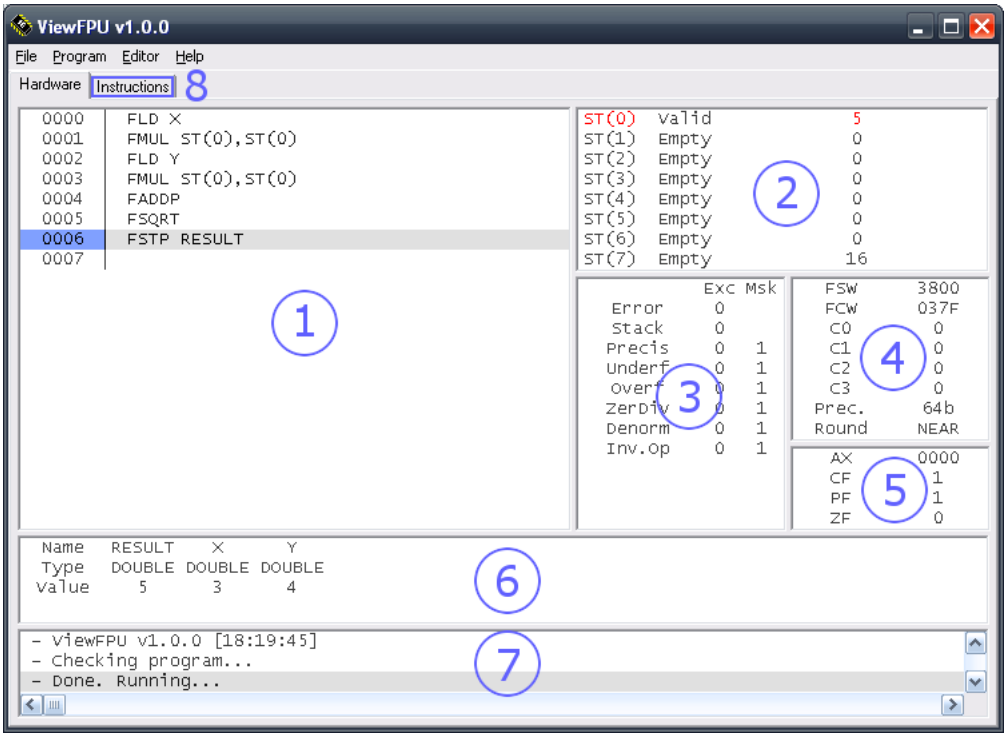
Ukážka grafického rozhrania aplikácie ViewFPU sa nachádza na obrázku 6.1. Pre všetky časti rozhrania platí:

- Na ovládanie sa dá použiť klávesnica aj myš.
- Ak sa dá časť vyznačiť, dá sa aj zmeniť.
- Zmena časti je farebne zvýraznená.[2]

Zvyšok tejto sekcie sa venuje jednotlivým častiam rozhrania.

1. **Editor inštrukcií** zobrazuje jednotlivé inštrukcie programu. Inštrukcie skoku sú farebne odlíšené. Užívateľ môže inštrukcie pridávať, meniť alebo odoberať. Zároveň môže zmeniť aktuálne spracovávanú inštrukciu počas behu programu. Pri zmene inštrukcie sa zobrazí dialóg zmeny inštrukcie, ktorého funkcie sú bližšie popísané v sekcii 6.2.2.
2. **Editor dátových registrov** zobrazuje aktuálny stav registrov a asociovaných Tag-ov. Užívateľ má možnosť zmeniť jednotlivé registre, ako aj Tag-y.
3. **Editor výnimiek** zobrazuje aktuálny stav výnimiek, a k nim asociovaných masiek. Tento editor umožňuje užívateľovi ľahko zistiť, aké výnimky generujú jednotlivé inštrukcie.

[2]Neplatí pre editory inštrukcií a operandov, kde je táto funkcionalita prakticky zbytočná.



Obr. 6.1: Grafické rozhranie aplikácie ViewFPU

4. **Editor rôznych atribútov** zobrazuje hodnoty Control Word, Status Word, podmienok C0-C3, poľa kontroly zaokrúhľovania - RC, a poľa kontroly presnosti - PC. Užívateľ môže zmeniť každú z týchto položiek.

5. **Editor stavu CPU** zobrazuje tie atribúty procesora, ktoré sa dajú zmeniť pomocou FPU inštrukcií. Jedná sa o register AX, ktorý sa dá zmeniť inštrukciou FSTSW AX a príznaky CF, PF a ZF, ktoré sa dajú zmeniť rôznymi inštrukciami, napríklad inštrukciami porovnania alebo inštrukciou SAHF.

6. **Editor operandov** zobrazuje aktuálne dostupné operandy. Okrem toho umožňuje meniť hodnotu samotných operandov, pridávať nové a odoberať staré operandy. Medzi podporované typy operandov aktuálne patria WORD, DWORD, QWORD, SINGLE, DOUBLE a EXTENDED.

7. **Log** aplikácie slúži ako grafický textový výstup niektorých operácií, pričom farebne zvýrazňuje chybové hlášky. Užívateľ môže Log kedykoľvek vymazať.

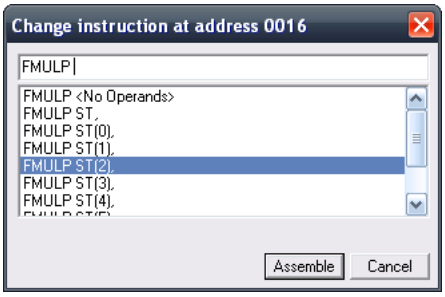
8. **Zoznam inštrukcií** umožňuje užívateľovi rýchlo nájsť inštrukcie, ich možné operandy a stručný opis. Používa rovnaký algoritmus vyhľadávania ako dialóg zmeny inštrukcie.

6.2.1 Generovanie výnimiek

Aplikácia pri zistení nemaskovanej výnimky zobrazí dialógové okno, ktoré informuje užívateľa o vzniknutej situácii. Užívateľ môže v tomto bode výnimku jednoducho odignorovať, alebo ju náležite ošetriť podľa vlastného uváženia.

6.2.2 Dialóg zmeny inštrukcie

Tento dialóg umožňuje užívateľovi nájsť hľadanú inštrukciu na základe začiatočných písmen. Keď sa nájde začiatok existujúcej inštrukcie, dialóg ponúkne užívateľovi vyhľadávanie v dostupných operandoch. Týmto spôsobom môže užívateľ zadať ako vstup iba platné inštrukcie, takže správnosť ich syntaxe je zaručená. Ukážka dialógu je na obrázku 6.2.



Obr. 6.2: Grafické rozhranie aplikácie ViewFPU

Po potvrdení zadania inštrukcie zostane dialóg otvorený a užívateľ môže zmeniť alebo vložiť ďalšiu inštrukciu. Týmto spôsobom má k dispozícii možnosť vyhľadávania inštrukcií keď je to potrebné a zároveň nepotrebuje opakovane otvárať dialógové okno.

6.2.3 Krokovanie programu

Táto operácia umožní užívateľovi spúšťať jednotlivé inštrukcie tak, ako by ich spustil samotný procesor. Aktuálne spracovávaná inštrukcia sa zobrazí v editore inštrukcií, pričom má od ostatných inštrukcií farebne odlíšenú adresu. Editor inštrukcií sa automaticky nastaví tak, aby bola spracovávaná inštrukcia vždy viditeľná.

Klávesová skratka pre túto operáciu je F8. Táto akcia sa dá spustiť z kontextového menu aplikácie alebo editora inštrukcií.

Kapitola 7

Záver

V tejto práci sme čitateľa zoznámili s históriou vývoja FPU, prezentovali sme štandardy používané v danej oblasti a nakoniec sme sa oboznámili s architektúrou a programovacími možnosťami, ktoré nám súčasné procesory ponúkajú. Zároveň sme umožnili overiť si nadobudnuté vedomosti v praxi pomocou aplikácie ViewFPU, ktorá vytvára vizualizáciu prezentovanej architektúry.

Napriek tomu sme v tejto práci odkryli iba malú časť celej problematiky matematických výpočtov pomocou počítačov. Oblasti verifikácie algoritmov, techniky zvyšovania presnosti výpočtov, podpora programovacích jazykov, algoritmy použité na hardvérovú implementáciu alebo formáty používajúce ako základ desiatkovú sústavu, to všetko sú časti ktoré sa už do tejto práce nedostali.

Čo sa týka hardvéru, aj tu sa doba neustále posúva napred, a na trh sa dostávajú stále novšie a novšie technológie. Zo všetkých možností môžeme spomenúť napríklad rozšírenie SSE, navrhnuté hlavne na urýchlenie multimediálnych aplikácií, alebo technológiu CUDA spoločnosti NVIDIA, ktorá je určená pre masívne paralelizované výpočty.

Práve implementácia týchto technológií do našej aplikácie by mohla byť predmetom budúcich prác.

Slovník pojmov

Bit Základná jednotka informácie, ktorá dokáže rozlíšiť medzi dvomi stavmi: Pravdivý(1) alebo Nepravdivý(0). 6, 7, 39

FPU [Floating Point Unit] Jednotka na prácu s číslami zapísanými vo vedeckej notácii. 1, 4, 7, 12, 14–19, 21–24, 31, 33, 35, 36, 39

Hidden bit Bit, ktorý je implicitne nastavený na hodnotu Pravdivý(1). 10–12, 39

IEEE [Institute of Electrical and Electronics Engineers] Medzinárodná nezisková organizácia, ktorá si kladie za cieľ rozvoj technológie súvisiacej s elektronikou. 3, 7, 8, 10, 12, 27, 39

Integer Binárne číslo zapísané v dvojkovom doplnkovom kóde. 24, 26–28, 39

NaN [Not a Number] Symbol reprezentujúci neplatné číslo. 9, 21, 22, 25, 28, 39

Literatúra

[1] Intel Corporation. *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 1: Basic Architecture*, March 2010.

[2] Intel Corporation. *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 2: Instruction Set Reference*, March 2010.

[3] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, March 1991. ISSN 0360-0300.

[4] Randall Hyde. *The Art of Assembly Language*, chapter 6, pages 418–419. No Starch Press, San Francisco, CA, USA, 2003. ISBN 1886411972.

[5] IEEE. IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2008*, pages 1–58, August 2008.

[6] William Kahan. Lecture notes on the status of IEEE Standard 754 for binary floating-point arithmetic. Manuscript, October 1997.