

FPU - základy, aritmetika

ISU cv 11

www.fit.vut.cz/person/isakin/public/isu



Desetinná čísla

Desetinná čísla

Base 10

... 100 10 1 1/10 1/100 ...

$$1/3 + 1/3 + 1/3 = 1$$

$$0.3333 + 0.3333 + 0.3333 = 0.999999999999 \quad ??$$

Base 2

... 8 4 2 1 1/2 1/4 1/8 ...

$$1/10 + 2/10 \neq 3/10 \quad ??$$

Floating point number

Celá čísla

- ukládána v doplňkovém kódu

Desetinná čísla

- ukládána s plovoucí řádovou čárkou (standard IEEE 754)
- Tři části (pro float):
 - znaménko 1b S 0 / 1 $\Rightarrow$ kladné / záporné
 - exponent 8b E mocnina čísla, základní hodnota 127
 - mantisa 23b M desetinná část čísla

$$X = -1^S * 2^{E-127} * (1.M)$$

Floating point number

[illegible]

- Sign=0 (positive)
- Mantissa= $1.11_2 = 1.75_{10}$
- Exponent= $130 - 127 = 3$

$$Value = +1.11_2 \times 2^3 = 1.75_{10} \times 8 = 14_{10}$$



Floating Point Unit [FPU]

FPU - Floating Point Unit

- Matematický koprocessor pro počítání s desetinnými čísly
- Pro práci s plovoucí řádovou čárkou (nikoliv reálná čísla)
- Jiné zařízení než procesor!! **FPU ≠ CPU**

<i>C</i>	<i>ASM</i>
----------	------------

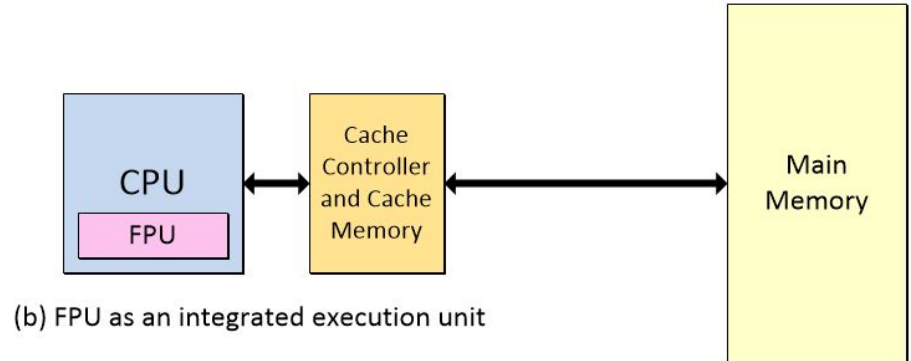
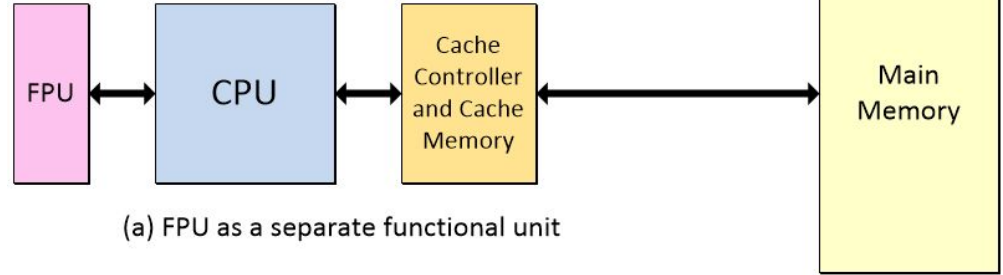
float	⇒ dword - 32bit
-------	-----------------

double	⇒ qword - 64bit
--------	-----------------

long double	⇒ tword - 80bit
-------------	-----------------

FPU - Floating Point Unit

- Dříve zvlášť, dnes součástí procesoru
- společný zdrojový kód
- **ale** každá jednotka má vlastní **instrukční sadu** a **registry**
 - CPU Intel x86
 - FPU Intel x87



FPU - Floating Point Unit

- 8 obecných registrů: R0 - R7
- Každý registr o velikosti **80b**
- Registry pracují jako zásobník (**ST0** - **ST7**)
- **ST0** - vrchol zásobníku (registrů), **ST1**, **ST2**, ... **ST7** - dokud nejsou naplněny, neexistují
- Podobné (stejně) názvy instrukcí + **f** na začátku názvu

FPU - Floating Point Unit

FINIT

- nastavení FPU do výchozího stavu
- nastavení výchozích hodnot, příznaků, promazání registrů FPU
- promaže i R0-R7, tedy **ST0-ST7** neexistuje
- stačí jednou, doporučeno použít vždy

`fld N` - načtení hodnoty z *N* do **ST0** (== push na vrchol zásobníku)

`fst N` - uložení hodnoty z **ST0** do *N*

`fstp N` - stejné jako `fst` + pop ze zásobníku

`call ReadDouble` - do **ST0** načte 64b desetinné číslo od uživatele

`call WriteDouble` - z **ST0** vypíše 64b desetinné číslo od uživatele

Ukázka

- cv11_A.asm

⊖ X87 Registers (FPU)

H	b	[R0] ST0 = NaN	3 (EMPTY)
H	b	[R1] ST1 = NaN	3 (EMPTY)
H	b	[R2] ST2 = NaN	3 (EMPTY)
H	b	[R3] ST3 = NaN	3 (EMPTY)
H	b	[R4] ST4 = NaN	3 (EMPTY)
H	b	[R5] ST5 = NaN	3 (EMPTY)
H	b	[R6] ST6 = NaN	3 (EMPTY)
H	b	[R7] ST7 = NaN	3 (EMPTY)



Datové typy a instrukce

Datové typy

- FPU pracuje se 3 datovými typy: float, double, long double
- FPU umí zpracovat i celá čísla, ale musí si je převést
 - instrukce s předponou **FI**, např: fild, fist, fistp

Velikost		Segment			Jméno	
bity	Bajty	.bss	.data	.text		
32	4	resd	dd	dword	float	double-word
64	8	resq	dq	qword	double	quad-word
80	10	rest	dt	tword	long double	ten-word

FPU aritmetické instrukce

FADD, FSUB, FMUL, FDIV

FADD	~	st0 = st1 + st0	; fadd == faddp
FADDP	~	st0 = st1 + st0	; + pop ⇒ st0
FADD mem	~	st0 = st0 + mem	
FADD stX	~	st0 = st0 + stX	; stX = <st1;st7>
FADD st0, stX	~	st0 = st0 + stX	
FADD stX, st0	~	stX = stX + st0	

FIADD, FISUB, FIMUL, FIDIV

FIADD mem	~	st0 = st0 + mem
FISUB mem	~	st0 = st0 - mem
FIMUL mem	~	st0 = st0 * mem
FIDIV mem	~	st0 = st0 / mem

FPU aritmetické instrukce

FSQRT, FSIN, FCOS

- Počítá odmocninu/sinus/kosinus registru **st0** a výsledek uloží do **st0**

FLD1, FLDZ, FLDPI

- Nahraje do FPU registru **st0** konstantu **1.0** (FLD1), **0.0** (FLDZ) nebo π (FLDPI)

FXCH - záměna FPU registru **st0** s jiným registrem FPU

FCHS - změna znaménka hodnoty registru **st0**

FABS - absolutní hodnota registru **st0**

Ukázky

- cv11_B.asm
- cv11_C.asm

Úkoly

- cv11_1.asm
- cv11_2.asm
- cv11_3.asm
- cv11_4.asm