Porovnávání desetinných čísel, parametry typu float a double

ISU-cv12

Ing. Jakub Husa

Vysoké Učení Technické v Brně, Fakulta informačních technologií
Božetěchova 1/2. 612 66 Brno - Královo Pole

ihusa@fit.vutbr.cz

25. dubna 2024

Porovnávání desetinných čísel

Porovnávání desetinných čísel

Desetinná čísla porovnáváme instrukcí FCOMI:

- Instrukce nastaví příznaky C0, C2 a C3 v registru FSTAT (FPU), a namapuje je na příznaky registru EFLAGS (CPU), podle kterých pak můžeme skákat.
- Porovnává obsah registru ST0 s registrem z operandu.

```
1 fcomi st1          ; ST0 porovnej s ST1 a nastav EFLAGS
```

Příznaky jsou do registru EFLAGS namapovány tímto způsobem:

- `st0 > st1` => ZF = 0, PF = 0, CF = 0
  `st0 = st1` => ZF = 1, PF = 0, CF = 0
  `st0 < st1` => ZF = 0, PF = 0, CF = 1
  `nedefinováno` => ZF = 1, PF = 1, CF = 1

Mapování odpovídá podmíněným skokům pro bez-znaménková čísla JA, JB, JE, ... :

```
2 fcomi st1          ; ST0 porovnej s ST1
3 ja    vetsi        ; skoc pokud ST0  > ST1
4 jb    mensi        ; skoc pokud ST0  < ST1
5 je    stejne       ; skoc pokud ST0 == ST1
6 jp    chyba        ; skoc pokud ST0  a ST1 jsou neporovnatelne (NaN > 0?)
```

Program ze vstupu načte dvě desetinná čísla, porovná je, a vypíše to větší z nich:

```
1  %include 'rw32.inc'          ;knihovna pro vstup a vystup
2
3  section .text                ;kodovy segment
4  main:
5      call ReadDoubleNewLine   ; nacti prvni vstup
6      call ReadDoubleNewLine   ; nacti druhy vstup
7
8      fcomi st1                ; ST0 porovnej s ST1 a nastav EFLAGS
9      ja   skip                ; pokud je ST0 > ST1 tak skoc na navesti SKIP
10     fxch st0, st1            ; vymen hodnoty v registreh ST0 a ST1
11   skip:
12     call WriteDoubleNewLine ; vypis obsah registru ST0
13
14     fstp st0                 ; odstran ST0
15     fstp st0                 ; odstran ST0
16     ret
```

Na procvičení

Vyzkoušejte si:

- Vytvořte si inicializované pole několika 32b desetinných čísel (arr), a dvě 32b desetinná čísla s hodnotami mínus nekonečno (ninf) a Not-a-Number (qnan).
- Napište funkci Maximum volanou dle konvence PASCAL.
- Pokud jsou parametry funkce neplatné (`delka <= 0` nebo `pole == 0`), její návratovou hodnotou (ST0) je konstanta Not-a-Number (viz. cv11s4).
- Jako počáteční hodnotu maxima použijte konstantu mínus nekonečno.

```
1  %include 'rw32.inc'          ;knihovna pro vstup a vystup
2  section .data                ;inicializovany datovy segment
3      arr  dd 1.0, 4.0, 2.0, 5.0, 3.0 ; pole peti 32b desetinnych cisel
4      ninf dd ;???                    ; konstanta minus nekonecno
5      qnan dd ;???                    ; konstanta Not-a-Number
6  section .text                ;kodovy segment
7  main:
8      push arr                 ; predej adresu pole (prvni parametr)
9      push dword 5             ; predej delku  pole (druhy parametr)
10     call Maximum             ; zavolej funkci Maximum
11     call WriteDoubleNewLine  ; vypis obsah ST0 (navratova hodnota)
12     fstp st0                 ; odstran obsah ST0
13     ret
```

Parametry typu float

Předávání desetinných čísel

Desetinná čísla typu float (32b) načítáme a vypisujeme z registru EAX:

```
1  call ReadFloat              ; do registru EAX nacti 32b float
2  call WriteFloat             ; z  registru EAX vypis 32b float
```

Abychom hodnotu z dostali EAX do ST0 použijeme paměťový segment zásobník:

```
3  call ReadFloatNewLine       ; EAX = vstup
4  push eax                    ; EAX vloz na zasobnik
5  fld  dword [esp]            ; hodnotu z vrcholu zasobniku nacti do ST0
6  add  esp, 4                 ; odstran hodnotu z vrcholu zasobniku
7  call WriteDoubleNewLine     ; vypis hodnotu ST0
```

Konstanty typu float na paměťový segment zásobník, nebo do registrů CPU, ukládáme použitím makra __float32__() z překladače NASM:

```
8  mov  eax, __float32__(0.5) ; do registru EAX uloz  32b float (0.5)
9  call WriteFloatNewLine     ; z  registru EAX vypis 32b float
10
11 push __float32__(0.5)      ; na zasobnik vloz 32b float (0.5)
12 push dword 0.5             ; CHYBA - bez pouziti makra float nejde ulozit
```

Vracení desetinných čísel

Obsah registrů FPU není možné vložit na zásobník:

```
1     push ST0                  ; CHYBA - instrukce CPU, registr FPU
```

Problém obejdeme použitím lokálních proměnných:

```
2  %include 'rw32.inc'          ;knihovna pro vstup a vystup
3
4  section .text                ;kodovy segment
5  main:
6      push ebp                 ; vytvor nove dno
7      mov  ebp, esp            ; zalohuj stare dno
8      sub  esp, 4              ; alokuj misto pro jednu 32b lokalni promennou
9
10     call ReadDoubleNewLine   ; ze vstupu nacti 64b desetinne cislo
11     fst  dword [ebp-4]       ; do lokalni promenne uloz 32b cislo z ST0
12     mov  eax,  [ebp-4]       ; do EAX nahraj hodnotu z lokalni promenne
13     call WriteFloatNewLine   ; z  EAX vypis 32b float
14
15     mov  esp, ebp            ; uvolni vsechny lokalni promenne (add esp, 4)
16     pop  ebp                 ; obnov stare dno
17     ret
```

Na vyzkoušení

Funkce sečte dvě desetinná čísla (X a Y) typu float, předaná hodnotou přes zásobník:

```
1  %include 'rw32.inc'          ;knihovna pro vstup a vystup
2  section .text                ;kodovy segment
3  main:
4      push __float32__(20.0) ; predej druhy parametr (Y)
5      push __float32__(10.0) ; predej prvni parametr (X)
6      call soucet            ; zavolej funkci soucet
7      add  esp, 8            ; odstran predane parametry (CDECL)
8      call WriteFloat        ; vypis vystup z EAX (navratova hodnota)
9      ret
10
11 soucet:
12     push ebp               ; zalohuj stare dno
13     mov  ebp, esp          ; vytvor nove dno
14     sub  esp, 4            ; alokuj jednu 32b lokalni promennou
15
16     fld  dword [ebp +12]   ; ST0 = Y
17     fadd dword [ebp + 8]   ; ST0 = X+Y
18     fst  dword [ebp - 4]   ; ST0 uloz do lokalni promenne
19     mov  eax,  [ebp - 4]   ; lokalni promennou uloz do EAX
20     fstp st0               ; odstran obsah registru ST0
21
22     mov  esp, ebp          ; uvolni vsechny lokalni promenne
23     pop  ebp               ; obnov stare dno
24     ret
```

Vyzkoušejte si:

- Napište funkci Odmocnina volanou dle konvence STDCALL.
- Funkce spočítá odmocninu parametru X, a to **BEZ** použití instrukce FSQRT, provedením deseti iterací babylonské metody definované iterační rovnicí:

$$Y_{i+1} = \frac{1}{2}\left(\frac{X}{Y_i} + Y_i\right)$$

- Pro jednoduchost můžete předpokládat že X vždy bude kladné číslo.
- Jako počáteční hodnotu odmocniny Y_0 použijte vstupní hodnotu X.

Například:

- 2.0 => 1.41421
- 4.0 => 2.00000
- 10.0 => 3.16228
- 100.0 => 10.00000

```
1  %include 'rw32.inc'          ;knihovna
2  section .text                ;kodovy segment
3  main:
4      push __float32__(4.0)    ; predej parametr
5      call Odmocnina           ; zavolej funkci
6      call WriteFloatNewLine   ; vypis vystup z EAX
7      ret
```

Parametry typu double

Datový typ double

Desetinná čísla typu double (64b) na zásobníku předáváme po polovinách:

- Počítač proměnné v paměti ukládá ve formátu little-endian a méně významné **BAJTY** (ne bity) jsou tedy na nižší adrese.

```
1 section .data
2     X db 0x01,0x23,0x45,0x67,0x89,0xAB,0xCD,0xEF ;pole osmi bytu
3     Y dq 0x0123456789ABCDEF                      ;jeden qword
```

adresa	[X+0]	[X+1]	[X+2]	[X+3]	[X+4]	[X+5]	[X+6]	[X+7]
hodnota	0x01	0x23	0x45	0x67	0x89	0xAB	0xCD	0xEF

adresa	[Y+0]	[Y+1]	[Y+2]	[Y+3]	[Y+4]	[Y+5]	[Y+6]	[Y+7]
hodnota	0xEF	0xCD	0xAB	0x89	0x67	0x45	0x23	0x01

Instrukcí PUSH na zásobník proto nejprve předáváme horní polovinu proměnné:

- Návratové hodnoty typu double stále vracíme v registru ST0 (konvence CDECL).

```
4 push dword [Y + 4]        ;predej  hornich 32b promenne typu double
5 push dword [Y + 0]        ;predej spodnich 32b promenne typu double
```

Funkce sečte dvě desetinná čísla (X a Y) typu double, předaná hodnotou přes zásobník:

```
1  %include 'rw32.inc'          ;knihovna pro vstup a vystup
2  section .data                ;inicializovany datovy segment
3      X dq 10.0                ; 64b desetinne cislo s hodnotou 10.0
4      Y dq 20.0                ; 64b desetinne cislo s hodnotou 20.0
5  section .text                ;kodovy segment
6  main:
7      push dword [Y + 4]       ; predej  hornich 32b promenne Y
8      push dword [Y + 0]       ; predej spodnich 32b promenne Y
9      push dword [X + 4]       ; predej  hornich 32b promenne X
10     push dword [X + 0]       ; predej spodnich 32b promenne X
11     call Soucet              ; zavolej funkci Soucet
12     add  esp, 16             ; odstran predane parametry (CDECL)
13     call WriteDoubleNewLine  ; vypis vystup z ST0 (navratova hodnota)
14     fstp st0                 ; odstran obsah registru ST0
15     ret
16
17 Soucet:                      ; double Soucet(double X, double Y)
18     push ebp                 ; zalohuj stare dno
19     mov  ebp, esp            ; vytvor nove dno
20     fld  qword [ebp +16]     ; ST0 = Y
21     fadd qword [ebp + 8]     ; ST0 = X+Y
22     pop  ebp                 ; obnov stare dno
23     ret
```

Funkce sečte dvě desetinná čísla (X a Y) typu double, předaná odkazem přes zásobník:

```
1  %include 'rw32.inc'          ;knihovna pro vstup a vystup
2  section .data                ;inicializovany datovy segment
3      X dq 10.0                ; 64b desetinne cislo s hodnotou 10.0
4      Y dq 20.0                ; 64b desetinne cislo s hodnotou 20.0
5  section .text                ;kodovy segment
6  main:
7      push dword Y             ; predej adresu promenne Y
8      push dword X             ; predej adresu promenne X
9      call Soucet              ; zavolej funkci Soucet
10     add  esp, 8              ; odstran predane parametry (CDECL)
11     call WriteDoubleNewLine  ; vypis vystup z ST0 (navratova hodnota)
12     fstp st0                 ; odstran obsah registru ST0
13     ret
14
15 Soucet:                      ; double Soucet(double* X, double* Y)
16     push ebp                 ; zalohuj stare dno
17     mov  ebp, esp            ; vytvor nove dno
18     mov  esi,  [ebp +12]     ; do ESI nahraj adresu promenne Y
19     fld  qword [esi]         ; ST0 = Y
20     mov  esi,  [ebp + 8]     ; do ESI nahraj adresu promenne X
21     fadd qword [esi]         ; ST0 = X+Y
22     pop  ebp                 ; obnov stare dno
23     ret
```

Na procvičení

Vyzkoušejte si:

- Napište a zavolejte funkci PrintEquation, volanou dle konvence CDECL, která pomocí externí funkce printf vypíše rovnici pro součet dvou čísel (X a Y).
- Vstupem funkce jsou řetězec (msg) a dvě čísla typu double předaná odkazem.

Například:

- msg, x, y => 10.000000 + 20.000000 = 30.000000

```
1  %include 'rw32.inc'          ;knihovna pro vstup a vystup
2      extern printf            ; externi funkce printf
3  section .data                ;inicializovany datovy segment
4      y    dq 20.0                    ; 64b desetinne cislo Y
5      x    dq 10.0                    ; 64b desetinne cislo X
6      msg db "%f + %f = %f", 10, 0 ; vypisovany retezec
7  section .text                ;kodovy segment
8  main:
9      push dword y             ; predej treti parametr (Y)
10     push dword x             ; predej druhy parametr (X)
11     push msg                 ; predej prvni parametr (retezec)
12     call PrintEquation       ; zavolej funkci PrintEquation
13     add  esp, 12             ; smaz predane parametry (CDECL)
14     ret
```