Семинар 10: Intel x86 assembly. Часть 3

27 января, 2020

В прошлых сериях...

- Registers
- Instructions
- Memory dereferencing
- Stack
- Calling conventions: x86 & x86-64

План на сегодня

```
float hypot_x86(float x, float y);
float hypot_x86_64(float x, float y);
```

x87: немного истории

- Intel 80387 (1987 год) — первый сопроцессор Intel, поддерживающий IEEE754
- Имелся отдельный набор команд (instruction set) для процессоров x86
- С 1991 года расположен на кристалле процессора

x87: технические детали

- x87 имел **восемь** 64-битных регистров: `st(0)` - `st(1)`
- Регистры образуют стек, `st(0)` – «вершина»
- Инструкции могли оперировать с любыми регистрами
- Но x87 позволял загружать числа из памяти на только в вершину стека и извлекать только из неё в память

x87: пример

```
A: .float 4195835
B: .float 3145727
    # ...
    fld B   # st(0) = B
    fld A   # st(0) = A, st(1) = B
    fdivp   # st(0) ≈ 1.33382
```

Pentium FDIV bug

Зачем нужно знать про x87?

Соглашения о вызовах под x86

- **float** и **double** аргументы передаются **по стеку**
- Возвращаемое значение (**float** или **double**) лежит в `st(0)`
- `st(1)` - `st(7)` **должны быть пустыми** в момент входа функцию и в момент выхода из неё

SSE

- SSE – *Streaming SIMD Extensions*
- SIMD – *Single Instruction Multiple Data*
- Набор команд для процессоров Intel
- Впервые появился в Pentium III (1999 год)
- Расширения: SSE2, SSE3, SSE4

SSE

- Добавляет в процессор 8 регистров: `xmm0` - `xmm7`
- Размер каждого – 128 битов
- Два типа данных – float (single) и double
- Два режима – scalar и packed

SSE: скалярные инструкции

- Последние две буквы отвечают за тип инструкции
- xxxSS = scalar single, то есть операция над одним float'ом
- xxxSD = scalar double, то есть операция над одним double'ом
- Например: `addss`, `subsd`, `divss`

SSE: packed инструкции

- xxxPS = packed single
- xxxPD = packed double
- Оперируют сразу несколькими числами в одном регистре: 4 float или 2 double
- Отсюда и название – *single instruction multiple data*

SSE: чтение данных

- `movupX` читает невыровненные данные
- `movapX` – выровненные по 16 байт
- Для скалярных типов используются `movsX`, которые не требуют выравнивания

Соглашения о вызовах под x86-64

- **float** и **double** аргументы передаются в `xmm0` - `xmm7`
- Если больше 8 аргументов, то остальные – по стеку
- Возвращаемое значение (**float** или **double**) лежит в `xmm0`
- Все xmm-регистры являются caller-saved

Если очень хочется использовать SSE под x86

```
sub $8, %esp            # немного места на стеке
movsd %xmm0, (%esp)     # сохраняем результат на стек
fldl (%esp)             # сохраняем рехультат в st(0)
add $8, %esp
```

Intrinsics

- Иногда писать на голом ассемблере очень трудно
- Intel разработал специальные расширения для компиляторов — intrinsics
- В каждом компиляторе реализованы по-разному
- Можно писать код на C, оптимизируя медленные места с помощью C-подобных функций

Intel Intrinsics Guide

Как сравнивать вещественные числа?

С помощью comisX для SSE:

```
comiss %xmm0, %xmm1   # сравнивает %xmm0 с %xmm1
ja .greater
# ...
```

И fcomi для x87:

```
fcomi
ja .greater   # сравнивает st(0) с st(1)
# ...
```

Дякую!