Машинно-зависимые языки программирования, лекция 6

Каф. ИУ7 МГТУ им. Н. Э. Баумана, 2023 г.

Сопроцессор (FPU – Floating Point Unit)

Изначально - отдельное опциональное устройство на материнской плате, с 80486DX встроен в процессор
Операции над 7-ю типами данных

- целое слово (16 бит)
- короткое целое (32 бита)
- длинное слово (64 бита)
- упакованное десятичное (80 бит)
- короткое вещественное (32 бита)
- длинное вещественное (64 бита)
- расширенное вещественное (80 бит)

Форма представления числа с плавающей запятой в FPU

Нормализованная форма представления числа $(1,...*2^{exp})$

Экспонента увеличена на константу для хранения в положительном виде

Пример представления 0,625 в коротком вещественном типе:

- 1/4+1/8 = 0,101b
- $1,01b*2^{-1}$
- Бит 31 - знак мантиссы, 30-23 - экспонента, увеличенная на 127, 22-0 - мантисса без первой цифры
- 00111111001000000000000000000000

Все вычисления FPU - в расширенном 80-битном формате

Особые числа FPU

Положительная бесконечность: знаковый - 0, мантисса - нули, экспонента - единицы
Отрицательная бесконечность: знаковый - 1, мантисса - нули, экспонента - единицы
NaN (Not a Number):

- qNAN (quiet) - при приведении типов/отдельных сравнениях
- sNAN (signal) - переполнение в большую/меньшую сторону, прочие ошибочные ситуации

Денормализованные числа (экспонента = 0): находятся ближе к нулю, чем наименьшее представимое нормальное число

Регистры FPU

- R0..R7, адресуются не по именам, а рассматриваются в качестве стека ST. ST соответствует регистру - текущей вершине стека, ST(1)..ST(7) - прочие регистры
- SR - регистр состояний, содержит слово состояния FPU. Сигнализирует о различных ошибках, переполнениях
- CR - регистр управления. Контроль округления, точности
- TW – 8 пар битов, описывающих состояния регистров: число, ноль, не-число, пусто
- FIP, FDP - адрес последней выполненной команды и её операнда для обработки исключений

Исключения FPU

- Неточный результат - произошло округление по правилам, заданным в CR. Бит в SR хранит направление округления
- Антипереполнение - переход в денормализованное число
- Переполнение - переход в "бесконечность" соответствующего знака
- Деление на ноль - переход в "бесконечность" соответствующего знака
- Денормализованный операнд
- Недействительная операция

Команды пересылки данных FPU

- FLD - загрузить вещественное число из источника (переменная или ST(n)) в стек. Номер вершины в SR увеличивается
- FST/FSTP - скопировать/считать число с вершины стека в приёмик
- FILD - преобразовать целое число из источника в вещественное и загрузить в стек
- FIST/FISTP - преобразовать вершину в целое и скопировать/считать в приёмник
- FBLD, FBSTP - загрузить/считать десятичное BCD-число
- FXCH - обменять местами два регистра (вершину и источник) стека

Базовая арифметика FPU

- FADD, FADDP, FIADD - сложение, сложение с выталкиванием из стека, сложение целых. Один из операндов - вершина стека
- FSUB, FSUBP, FISUB - вычитание
- FSUBR, FSUBRP, FISUBR - обратное вычитание (приёмника из источника)
- FMUL, FMULP, FIMUL - умножение
- FDIV, FDIVP, FIDIV - деление
- FDIVR, FDIVRP, FIDIVR - обратное деление (источника на приёмник)
- FPREM - найти частичный остаток от деления (делится ST(0) на ST(1)). Остаток ищется цепочкой вычитаний, до 64 раз

Базовая арифметика FPU (продолжение)

- FABS - взять модуль числа
- FCHS - изменить знак
- FRNDINT - округлить до целого
- FSCALE - масштабировать по степеням двойки (ST(0) умножается на 2ST(1))
- FXTRACT - извлечь мантиссу и экспоненту. ST(0) разделяется на мантиссу и экспоненту, мантисса дописывается на вершину стека
- FSQRT - вычисляет квадратный корень ST(0)

Команды сравнения FPU

- FCOM, FCOMP, FCOMPP - сравнить и вытолкнуть из стека
- FUCOM, FUCOMP, FUCOMPP - сравнить без учёта порядков и вытолкнуть
- FICOM, FICOMP, FICOMP - сравнить целые
- FCOMI, FCOMIP, FUCOMI, FUCOMIP (P6)
- FTST - сравнивает с нулём
- FXAM - выставляет флаги в соответствии с типом числа

Трансцендентные операции FPU

- FSIN
- FCOS
- FSINCOS
- FPTAN
- FPATAN
- F2XM1 – 2^x-1
- FYL2X, FYL2XP1 – $y*\log_2 x$, $y*\log_2(x+1)$

Константы FPU

- FLD1 – 1,0
- FLDZ - +0,0
- FLDPI - число Пи
- FLDL2E - $\log_2 e$
- FLDL2T - $\log_2 10$
- FLDLN2 – ln(2)
- FLDLG2 – lg(2)

Команды управления FPU

- FINCSTP, FDECSTP - увеличить/уменьшить указатель вершины стека
- FFREE - освободить регистр
- FINIT, FNINIT - инициализировать сопроцессор / инициализировать без ожидания (очистка данных, инициализация CR и SR по умолчанию)
- FCLEX, FNCLEX - обнулить флаги исключений / обнулить без ожидания
- FSTCW, FNSTCW - сохранить CR в переменную / сохранить без ожидания
- FLDCW - загрузить CR
- FSTENV, FNSTENV – сохранить вспомогательные регистры (14/28 байт) / сохранить без ожидания
- FLDENV - загрузить вспомогательные регистры
- FSAVE, FNSAVE, FXSAVE - сохранить состояние (94/108 байт) и инициализировать, аналогично FINIT
- FRSTOR, FXRSTOR - восстановить состояние FPU
- FSTSW, FNSTSW - сохранение CR
- WAIT, FWAIT - обработка исключений
- FNOP - отсутствие операции

Команда CPUID (с 80486)

Идентификация процессора

- Если EAX = 0, то в EAX - максимальное допустимое значение (1 или 2), а EBX:ECX:EDX – 12-байтный идентификатор производителя (ASCII-строка).
- Если EAX = 1, то в EAX - версия, в EDX - информация о расширениях
  - EAX - модификация, модель, семейство
  - EDX: наличие FPU, поддержка V86, поддержка точек останова, CR4, PAE, APIC, быстрые системные вызовы, PGE, машинно-специфичный регистр, CMOVcc, **MMX, FXSR (MMX2), SSE**
- Если EAX = 2, то в EAX, EBX, ECX, EDX возвращается информация о кэшах и TLB

MMX (Multimedia Extensions - 1997, Pentium MMX)

Увеличение эффективности обработки больших потоков данных (изображения, звук, видео...) - выполнение простых операций над массивами однотипных чисел.

- 8 64-битных регистров MM0..MM7 - **мантиссы регистров FPU**. При записи в MMn экспонента и знаковый бит заполняются единицами
- Пользоваться одновременно и FPU, и MMX не получится, требуется FSAVE+FRSTOR
- Типы данных MMX:
  - учетверённое слово (64 бита);
  - упакованные двойные слова (2);
  - упакованные слова (4);
  - упакованные байты (8).
- Команды MMX перемещают упакованные данные в память или обычные регистры целиком, но арифметические и логические операции выполняют поэлементно.
- *Насыщение* - замена переполнения/антипереполнения превращением в максимальное/минимальное значение

Команды пересылки данных MMX

- MOVD, MOVQ - пересылка двойных/учетверённых слов
- PACKSSWB, PACKSSDW - упаковка со знаковым насыщением слов в байты/двойных слов в слова. *Приёмник -> младшая половина приёмника*, *источник -> старшая половина приёмника*
- PACKUSWB - упаковка слов в байты с беззнаковым насыщением
- PUNPCKH**BW**, PUNPCKH**WD**, PUNPCKH**DQ** - распаковка и объединение старших элементов источника и приёмника через 1

Арифметические операции MMX

- PADDB, PADDW, PADDD - поэлементное сложение, перенос игнорируется
- PADDSB, PADDSW - сложение с насыщением
- PADDUSB, PADDUSW - беззнаковое сложение с насыщением
- PSUBB, PSUBW, PDUBD - вычитание, заём игнорируется
- PSUBSB, PSUBSW - вычитание с насыщением
- PSUBUSB, PSUBUSW - беззнаковое вычитание с насыщением
- PMILHW, PMULLW - старшее/младшее умножение (сохраняет старшую или младшую части результата в приёмник)
- PMADDWD - умножение и сложение. Перемножает 4 слова, затем попарно складывает произведения двух старших и двух младших

Команды сравнения MMX

- PCMPEQB, PCMPEQW, PCMPEQD - проверка на равенство. Если пара равна - соответствующий элемент приёмника заполняется единицами, иначе - нулями
- PCMPGTB, PCMPGTW, PCMPGTD - сравнение. Если элемент приёмника больше, то заполняется единицами, иначе - нулями

Логические операции MMX

- PAND - логическое И
- PANDN - логическое НЕ-И (штрих Шеффера) (источник*НЕ(приёмник))
- POR - логическое ИЛИ
- PXOR - исключающее ИЛИ

Сдвиговые операции MMX

- PSLLW, PSLLD, PSLLQ - логический влево
- PSRLW, PSRLD, PSRLQ - логический вправо
- PSRAW, PSRAD - арифметический вправо

Расширение SSE (Streaming SIMD Extensions - Pentium III, 1999)

- Single Instruction, Multiple Data: одна инструкция — множество данных
- Решение проблемы параллельной работы с FPU
- 8 128-разрядных регистров
- Свой регистр флагов
- Основной тип - вещественные одинарной точности (32 бита)
- Целочисленные команды работают с регистрами MMX
- Команды:
  - Пересылки
  - Арифметические
  - Сравнения
  - Преобразования типов
  - Логические
  - Целочисленные
  - Упаковки
  - Управления состоянием
  - Управления кэшированием
- Развитие: SSE2, SSE3...

SSE2 (2000 г., Pentium 4), SSE3 (2004 г.)

SSE2:

- Развитие и SSE, и MMX, окончательная замена MMX
- Числа двойной точности (2 64-битных в одном регистре)
- 144 новых команды в дополнение к 70 из первой версии SSE
- необходим для Google Chrome с версии 32

SSE3:

- 13 новых инструкций
- горизонтальная работа с регистрами (сложение и вычитание значений в одном регистре)
- необходим для Google Chrome с версии 89

SSE4 (2007 г.)

- 54 новых инструкции (47 SSE4.1 + 7 SSE 4.2)
- опции gcc, начиная с версии 4.3: -msse4.1, -msse4.2, -msse4 (оба набора)
- ускорение видеокодеков
- вычисление CRC32
- обработка строк

Расширение AVX (Advanced Vector Extensions), 2008 г.

- регистры увеличены со 128 (XMM) до 256 (YMM0-YMM15) бит;
- SSE-инструкции используют младшую половину YMM-регистров, не меняя старшую часть;
- “неразрушающие” (трёхоперандные) инструкции: c = a + b вместо a = a + b;
- AVX3 (2013 г.) - 512-битное расширение (регистры ZMM0-ZMM31).

Расширение AES (Intel Advanced Encryption Standard New Instructions; AES-NI, 2008)

Цель - ускорение шифрования по алгоритму AES

Команды:

- раунда шифрования;
- раунда расшифровывания;
- способствования генерации ключа

Виды трансляторов ассемблера

- MASM
- TASM
- NASM
- FASM
- YASM
- as
- ...

AT&T-синтаксис

Синтаксис стандартного ассемблера для UNIX - **as**

Основные отличия от Intel-синтаксиса:

1. Имена регистров предваряются префиксом %.
2. Обратный порядок операндов: вначале источник, затем приёмник.
3. Размер операнда задается суффиксом, замыкающим инструкцию.
4. Числовые константы записываются в Си-соглашении.
5. Для получения смещения метки используется префикс $.

Создание оконных приложений на ассемблере под x86

Системный вызов — обращение прикладной программы к ядру операционной системы для выполнения какой-либо операции.

Для реализации оконных приложений необходима линковка с соответствующими библиотеками и использование как их функций, так и системных вызовов.

Дизассемблирование. Реверс-инжиниринг

Дизассемблер - транслятор, преобразующий машинный код, объектный файл или библиотечные модули в текст программы на языке ассемблера.

Дизассемблирование - процесс получения текста программы на ассемблере из программы в машинных кодах.

Реверс-инжиниринг (обратная разработка) — исследование готовой программы с целью понять принцип работы, поиска недокументированных возможностей или внесения изменений.