**Другие темы раздела**

**FASM Уроки Iczelion'a на FASM** https://www.cyberforum.ru/ fasm/thread1240590.html
Уроки Iczelion'a на FASM Урок первый. MessageBox на FASM format PE GUI include 'win32ax.inc' ; import data in the same section invoke MessageBox,NULL,msgBoxText,msgBoxCaption,MB_OK ...

**Побайтовый вывод файла FASM**
Пытаюсь ввести в консоль файл в шестнадцатеричном виде, но происходит ошибка при выполнении. format PE console 4.0 include 'win32a.inc' xor ebx, ebx ; invoke CreateFile,\ ...

**FASM Вывод адреса на консоль**
Пытаюсь на консоль вывести адрес fin: invoke printf, не робит - как правильно надо? format PE console 4.0 entry start include 'win32a.inc' section '.data' data readable fin ...

**FASM ГСЧ на макросах** Всем привет. Понадобилось заюзать ГСЧ посредством макросов, чтобы каждый раз на стадии компиляции, использовалось уникальное значение. Учитывая семантику препроцессора (там чёрт ногу сломит),... https://www.cyberforum.ru/ fasm/thread1213146.html

**FASM Создание окна на fasm** https://www.cyberforum.ru/ fasm/thread1209394.html
Всем привет. Только что начал изучать ассемблер fasm. Возник первый вопрос: как создать окно? Прошу не просто дать мне код, а ещё объяснить что значит. Заранее благодарен

**FASM Вызываем функции из clib (библиотека Си) в DOS**
Вобщем, сбылась мечта идиота. Теперь, нежели писать свой ввод/вывод(особенно всегда напрягал ввод/вывод вещественных чисел на экран), можно воспользоваться стандартными ф-циями из библиотеки языка...

**Организовать вычисления по формуле FASM**
привет, всем активным участникам этого чудесного форума!!! помогите, пожалуйста, написать программу на Fasm Assembler. задание: Создать программу на языке Ассемблер, что позволяет организовать...

**FASM Как сделать выход по ESC** org 100h old dw 0 jmp start number dw 0 c dw 0 start: xor ax,ax mov es,ax cli https://www.cyberforum.ru/ fasm/thread1161834.html

**FASM Получение CLSID image/png** https://www.cyberforum.ru/ fasm/ thread1160365.html
Всем ку! int GetEncoderClsid(const WCHAR* format, CLSID* pClsid) { UINT num = 0; // number of image encoders UINT size = 0; // size of the image encoder array in bytes...

**FASM Вывод трех строк в один MessageBox** Здравствуйте, помогите, пожалуйста, с такой проблемой: не могу вывести 3 строки (Год+Месяц+День) в один MessageBox Вот такой код: format PE GUI 4.0 entry start include 'win32ax.inc' include... https://www.cyberforum.ru/ fasm/thread1142589.html

Mikl
Ушел с форума
**13987** / 7000 / **813**
Регистрация: 11.11.2010
Сообщений: 12,592

22.08.2014, 08:44 **[ТС]** **0**

# Мануал по flat assembler

22.08.2014, 08:44. **Просмотров** 99777. **Ответов** 50
**Метки** (Все метки)

**Ответ**

## 2.1.21 Инструкции AVX

*Advanced Vector Extensions* (AVX) — расширение системы команд. AVX предоставляет различные улучшения, новые инструкции и новую схему кодирования машинных кодов.

**Новая схема кодирования инструкций VEX**
Ширина векторных регистров SIMD увеличивается со 128 (XMM) до 256 бит (регистры YMM0 — YMM15). Существующие 128-битные SSE инструкции будут использовать младшую половину новых YMM регистров, не изменяя старшую часть. Для работы с YMM регистрами добавлены новые 256-битные AVX инструкции. В будущем возможно расширение векторных регистров SIMD до 512 или 1024 бит. Например, процессоры с архитектурой Larrabee уже имеют векторные регистры (ZMM) шириной в 512 бит, и используют для работы с ними SIMD команды с MVEX и VEX префиксами, но при этом они не поддерживают AVX.

**Неразрушающие операции**
Набор AVX инструкций использует трехоперандный синтаксис. Например, вместо $a = a + b$ можно использовать $c = a + b$, при этом регистр a остается неизмененным. В случаях, когда значение $a$ используется дальше в вычислениях, это повышает производительность, так как избавляет от необходимости сохранять перед вычислением и восстанавливать после вычисления регистр, содержавший $a$, из другого регистра или памяти.

Для большинства новых инструкций отсутствуют требования к выравниванию операндов в памяти. Однако рекомендуется следить за выравниванием на размер операнда, во избежание значительного снижения производительности.

Набор инструкций AVX содержит в себе аналоги 128-битных SSE инструкций для вещественных чисел. При этом, в отличие от оригиналов, сохранение 128-битного результата будет обнулять старшую половину YMM регистра. 128-битные AVX инструкции сохраняют прочие преимущества AVX, такие как новая схема кодирования, трехоперандный синтаксис и невыровненный доступ к памяти. Рекомендуется отказаться от старых SSE инструкций в пользу новых 128-битных AVX инструкций, даже если достаточно двух операндов

**Новая схема кодирования**
Новая схема кодирования инструкций VEX использует VEX префикс. В настоящий момент существуют два VEX префикса, длиной 2 и 3 байта. Для 2-х байтного VEX префикса первый байт равен 0xC5, для 3-х байтного 0xC4. В 64-битном режиме первый байт VEX префикса уникален. В 32-битном режиме возникает конфликт с инструкциями LES и LDS, который разрешается старшим битом второго байта, он имеет значение только в 64-битном режиме, через неподдерживаемые формы инструкций LES и LDS. Длина существующих AVX инструкций, вместе с VEX префиксом, не превышает 11 байт. В следующих версиях ожидается появление более длинных инструкций.

..

**Новые инструкции**

| Инструкция | Описание |
|---|---|
| VBROADCASTSS, VBROADCASTSD, VBROADCASTF128 | Копирует 32-х, 64-х или 128-ми битный операнд из памяти во все элементы векторного регистра XMM или YMM. |
| VINSERTF128 | Замещает младшую или старшую половину 256-ти битного регистра YMM значением 128-ми битного операнда. Другая часть регистра-получателя не изменяется. |
| VEXTRACTF128 | Извлекает младшую или старшую половину 256-ти битного регистра YMM и копирует в 128-ми битный операнд-назначение. |
| VMASKMOVPS, VMASKMOVPD | Условно считывает любое количество элементов из векторного операнда из памяти в регистр-получатель, оставляя остальные элементы несчитанными и обнуляя соответствующие им элементы регистра-получателя. Также может условно записывать любое количество элементов из векторного регистра в векторный операнд в памяти, оставляя остальные элементы операнда памяти неизменёнными |
| VPERMILPS, VPERMILPD | Переставляет 32-х или 64-х битные элементы вектора согласно операнду-селектору (из памяти или из регистра). |
| VPERM2F128 | Переставляет 4 128-ми битных элемента двух 256-ти битных регистров в 256-ти битный операнд-назначение с использованием непосредственной константы (imm) в качестве селектора. |
| VZEROALL | Обнуляет все YMM регистры и помечает их как неиспользуемые. Используется при переключении между 128-ми битным режимом и 256-ти битным. |
| VZEROUPPER | Обнуляет старшие половины всех регистров YMM. Используется при переключении между 128-ми битным режимом и 256-ти битным. |

Также в спецификации AVX описана группа инструкций **PCLMUL** (Parallel Carry-Less Multiplication, Parallel CLMUL)

Assembler Выделить код

```
1  PCLMULLQLQDQ xmmreg,xmmrm [rm: 66 0f 3a 44 /r 00]
2  PCLMULHQLQDQ xmmreg,xmmrm [rm: 66 0f 3a 44 /r 01]
3  PCLMULLQHQDQ xmmreg,xmmrm [rm: 66 0f 3a 44 /r 02]
4  PCLMULHQHQDQ xmmreg,xmmrm [rm: 66 0f 3a 44 /r 03]
5  PCLMULQDQ xmmreg,xmmrm,imm [rmi: 66 0f 3a 44 /r ib]
```

**Применение**
Подходит для интенсивных вычислений с плавающей точкой в мультимедиа программах и научных задачах. Там, где возможна более высокая степень параллелизма, увеличивает производительность с вещественными числами.

The Advanced Vector Extensions introduce instructions that are new variants of SSE instructions, with new scheme of encoding that allows extended syntax having a destination operand separate from all the source operands. It также introduces 256-bit AVX registers, which extend up the old 128-bit SSE registers. Any AVX instruction that puts some result into SSE register, puts zero bits into high portion of the AVX register containing it.
The AVX version of SSE instruction has the mnemonic obtained by prepending SSE instruction name with **v**. For any SSE arithmetic instruction which had a destination operand also being used as one of the source values, the AVX variant has a new syntax with three operands - the destination and two sources. The destination and first source can be SSE registers, and second source can be SSE register or memory. If the operation is performed on single pair of values, the remaining bits of first source SSE register are copied into the the destination register.

Assembler Выделить код

```
1  vsubss xmm0,xmm2,xmm3 ; substract two 32-bit floats
2  vmulsd xmm0,xmm7,qword [esi] ; multiply two 64-bit floats
```

In case of packed operations, each instruction can also operate on the 256-bit data size when the AVX registers are specified instead of SSE registers, and the size of memory operand is also doubled then.

Assembler Выделить код

```
1  vaddps ymm1,ymm5,yword [esi] ; eight sums of 32-bit float pairs
```

The instructions that operate on packed integer types (in particular the ones that earlier had been promoted from MMX to SSE) also acquired the new syntax with three operands, however they are only allowed to operate on 128-bit packed types and thus cannot use the whole AVX registers.

Assembler Выделить код

```
1  vpavgw xmm3,xmm0,xmm2 ; average of 16-bit integers
2  vpslld xmm1,xmm0,1 ; shift double words left
```

If the SSE version of instruction had a syntax with three operands, the third one being an immediate value, the AVX version of such instruction takes four operands, with immediate remaining the last one.

Assembler Выделить код

Assembler Выделить код

```
1  vshufpd ymm0,ymm1,ymm2,10010011b ; shuffle 64-bit floats
2  vpalignr xmm0,xmm4,xmm2,3 ; extract byte aligned value
```

The promotion to new syntax according to the rules described above has been applied to all the instructions from SSE extensions up to SSE4, with the exceptions described below.
**vdppd** instruction has syntax extended to four operans, but it does not have a 256-bit version.

The are a few instructions, namely vsqrtpd, vsqrtps, vrcpps and vrsqrtps, which can operate on 256-bit data size, but retained the syntax with only two operands, because they use data from only one source:

Assembler Выделить код

```
1  vsqrtpd ymm1,ymm0 ; put square roots into other register
```

In a similar way vroundpd and vroundps retained the syntax with three operands, the last one being immediate value.

Assembler Выделить код

```
1  vroundps ymm0,ymm1,0011b ; round toward zero
```

Also some of the operations on packed integers kept their two-operand or three- operand syntax while being promoted to AVX version. In such case these instruc- tions follow exactly the same rules for operands as their SSE counterparts (since operations on packed integers do not have 256-bit variants in AVX extension). These include vpcmpestri, vpcmpestrm, vpcmpistri, vpcmpistrm, vphminposuw, vpshufd, vpshufhw, vpshuflw. And there are more instructions that in AVX versions keep exactly the same syntax for operands as the one from SSE, without any additional options: **vcomiss**, **vcomisd**, **vcvtss2si**, **vcvtsd2si**, **vcvttss2si**, **vcvttsd2si**, **vextractps**, **vpextrb**, **vpextrw**, **vpextrd**, **vpextrq**, **vmovd**, **vmovq**, **vmovntdqa**, **vmaskmovdqu**, **vpmovmskb**, **vpmovsxbw**, **vpmovsxbd**, **vpmovsxbq**, **vpmovsxwd**, **vpmovsxwq**, **vpmovsxdq**, **vpmovzxbw**, **vpmovzxbd**, **vpmovzxbq**, **vpmovzxwd**, **vpmovzxwq** and **vpmovzxdq**.

Вернуться к обсуждению:
Мануал по flat assembler

**Следующий ответ**

**0**

Programming
Эксперт
**94731** / 64177 / **26122**
Регистрация: 12.04.2006
Сообщений: 116,782

22.08.2014, 08:44

Готовые ответы и решения:

**Неофициальная разработка Flat assembler версии 2.0.0**
Разработчик Flat assembler'a Tomasz Grysztar в одном из блогов сообщил о разработке новой...

**Flat assembler ругается на PROC**
Доброго времени суток. Есть программа, собственно вот что она делает: "На экране инициализировать...

**Как подключить include к flat компилятору**
Здравствуйте,как подключить include к flat компилятору?Требуется подключить include 'win32a.inc' к...

**Flat Assembler**
Со временем задачи стали нерешаемыми из-за ужасно медленной скорости. Уже давно хочу перейти на...

**50**