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

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

21.08.2014, 10:18 **[TC]** **0**

Мануал по flat assembler

21.08.2014, 10:18. **Просмотров** 99777. **Ответов** 50
Метки (Все метки)

Ответ

2.1.17 Инструкции SSE3

SSE3 (PNI — *Prescott New Instruction*) — третья версия SIMD-расширения Intel, потомок SSE, SSE2 и MMX. Впервые представлено в ядре Prescott процессора Pentium 4. AMD предложила свою реализацию SSE3 для процессоров Athlon 64 (ядра Venice, San Diego и Newark).

Набор SSE3 содержит 13 инструкций:

1. FISTTP (x87) — преобразование вещественного числа в целое с сохранением целочисленного значения и округлением в сторону нуля
2. MOVSLDUP (SSE)
3. MOVSHDUP (SSE)
4. MOVDDUP (SSE2)
5. LDDQU (SSE/SSE2) — загрузка 128bit невыровненных данных из памяти в регистр xmm, с предотвращением пересечения границы строки кеша
6. ADDSUBPD (SSE) Add Subtract Packed Double
7. ADDSUBPS (SSE2) Add Subtract Packed Single
8. HADDPS (SSE) Horizontal Add Packed Single
9. HSUBPS (SSE) Horizontal Subtract Packed Single
10. HADDPD (SSE2) Horizontal Add Packed Double
11. HSUBPD (SSE2) Horizontal Subtract Packed Double
12. MONITOR (нет аналога в SSE3 для AMD)
13. MWAIT (нет аналога в SSE3 для AMD)

Наиболее заметное изменение - возможность горизонтальной работы с регистрами. Если говорить более конкретно, добавлены команды сложения и вычитания нескольких значений, хранящихся в одном регистре. Эти команды упростили ряд DSP и 3D-операций. Существует также новая команда для преобразования значений с плавающей точкой в целые без необходимости вносить изменения в глобальном режиме округления.
Prescott technology introduced some new instructions to improve the performance of SSE and SSE2 - this extension is called SSE3.
fisttp behaves like the fistp instruction and accepts the same operands, the only dierence is that it always used truncation, irrespective of the rounding mode.
movshdup loads into destination operand the 128-bit value obtained from the source value of the same size by lling the each quad word with the two duplicates of the value in its high double word.
movsldup performs the same action, except it duplicates the values of low double words. The destination operand should be SSE register, the source operand can be SSE register or 128-bit memory location.
movddup loads the 64-bit source value and duplicates it into high and low quad word of the destination operand. The destination operand should be SSE register, the source operand can be SSE register or 64-bit memory location.

lddqu is functionally equivalent to movdqu with memory as source operand, but it may improve performance when the source operand crosses a cacheline boundary. The destination operand has to be SSE register, the source operand must be 128-bit memory location.
addsubps performs single precision addition of second and fourth pairs and single precision substracion of the rst and third pairs of floating point values in the operands.
addsubpd performs double precision addition of the second pair and double precision substraction of the rst pair of floating point values in the operand.
haddps performs the addition of two single precision values within the each quad word of source and destination operands, and stores the results of such horizontal addition of values from destination operand into low quad word of destination operand, and the results from the source operand into high quad word of destination operand.
haddpd performs the addition of two double precision values within each operand, and stores the result from destination operand into low quad word of destination operand, and the result from source operand into high quad word of destination operand. All these instructions need the destination operand to be SSE register, source operand can be SSE register or 128-bit memory location.
monitor sets up an address range for monitoring of write-back stores. It need its three operands to be EAX, ECX and EDX register in that order.
mwait waits for a write-back store to the address range set up by the monitor instruction. It uses two operands with additional parameters, rst being the EAX and second the ECX register.
The functionality of SSE3 is further extended by the set of Supplemental SSE3 instructions (SSSE3). They generally follow the same rules for operands as all the MMX operations extended by SSE.
phaddw and **phaddd** perform the horizontal additional of the pairs of adjacent values from both the source and destination operand, and stores the sums into the destination (sums from the source operand go into lower part of destination register). They operate on 16-bit or 32-bit chunks, respectively. phaddsw performs the same operation on signed 16-bit packed values, but the result of each addition is saturated. phsubw and phsubd analogously perform the horizontal substraction of 16-bit or 32-bit packed value, and phsubsw performs the horizontal substraction of signed 16-bit packed values with saturation.
pabsb, **pabsw** and **pabsd** calculate the absolute value of each signed packed signed value in source operand and stores them into the destination register. They operator on 8-bit, 16-bit and 32-bit elements respectively.
pmaddubsw multiplies signed 8-bit values from the source operand with the corresponding unsigned 8-bit values from the destination operand to produce intermediate 16-bit values, and every adjacent pair of those intermediate values is then added horizontally and those 16-bit sums are stored into the destination operand.
pmulhrsw multiplies corresponding 16-bit integers from the source and destination operand to produce intermediate 32-bit values, and the 16 bits next to the highest bit of each of those values are then rounded and packed into the destination operand.
pshufb shues the bytes in the destination operand according to the mask provided by source operand - each of the bytes in source operand is an index of the target position for the corresponding byte in the destination.
psignb, **psignw** and **psignd** perform the operation on 8-bit, 16-bit or 32-bit integers in destination operand, depending on the signs of the values in the source. If the value in source is negative, the corresponding value in the destination register is negated, if the value in source is positive, no operation is performed on the corresponding value is performed, and if the value in source is zero, the value in destination is zeroed, too.
palignr appends the source operand to the destination operand to form the intermediate value of twice the size, and then extracts into the destination register the 64 or 128 bits that are right-aligned to the byte oset specied by the third operand, which should be an 8-bit immediate value. This is the only SSSE3 instruction that takes three arguments.

Вернуться к обсуждению:
Мануал по flat assembler

Следующий ответ

0

IT_Exp
Эксперт
87844 / 49110 / **22898**
Регистрация: 17.06.2006
Сообщений: 92,604

21.08.2014, 10:18

Ошибка в flat assembler
начал изучать ассемблер сталкнулся с такой проблемой: перепечатываю пример из книги: org...

flat assembler массив
У меня есть задание "Упорядочить по убыванию элементы каждого столбцу матрицы" Числа произвольные...

Массив в Flat Assembler
Всем добрый день! Подскажите, почему не работает массив в Flat Assembler? org 100h jmp start...

Как использовать Flat Assembler в Free Pascal?
Я недавно хотел разработать так ради прикола мини ОС с использованием в связке Free Pasal и Flat...

0