Учебный курс. Часть 9. Сложение и вычитание

Автор: xrnd | Рубрика: Учебный курс | 28-03-2010 | Распечатать запись

Теперь мы уже знаем, как представляются числа в компьютере, и можем перейти к изучению команд процессора. Начнём с самых простых арифметических операций: сложения и вычитания.

Сложение

Для сложения двух чисел предназначена команда ADD. Она работает как с числами со знаком, так и с числами без знака (это особенность дополнительного кода).

Операнды должны иметь одинаковый размер (нельзя складывать 16- и 8-битное значение). Результат помещается на место первого операнда. В общем, эти правила справедливы для большинства команд.

После выполнения команды изменяются флаги, по которым можно определить характеристики результата:

- Флаг *CF* устанавливается, если при сложении произошёл перенос из старшего разряда. Для беззнаковых чисел это будет означать, что произошло переполнение и результат получился некорректным.
- Флаг *OF* обозначает переполнение для чисел со знаком.
- Флаг *SF* равен знаковому биту результата (естественно, для чисел со знаком, а для беззнаковых он равен старшему биту и особо смысла не имеет).
- Флаг *ZF* устанавливается, если результат равен 0.
- Флаг *PF* — признак чётности, равен 1, если результат содержит нечётное число единиц.

Примеры:

```
    add ax,5        ;AX = AX + 5
    add dx,cx       ;DX = DX + CX
    add dx,cl       ;Ошибка: разный размер операндов.
```

Вычитание

Вычитание выполняется с помощью команды SUB. Результат также помещается на место первого операнда и опять же выставляются флаги. Единственная разница в том, что происходит вычитание, а не сложение.

На самом деле вычитание в процессоре реализовано с помощью сложения. Процессор меняет знак второго операнда на противоположный, а затем складывает два числа. Если вам необходимо в программе поменять знак числа на противоположный, можно использовать команду NEG. У этой команды всего один операнд.

Примеры:

```
    sub si,dx       ;SI = SI - DX
    neg ax          ;AX = -AX
```

Инкремент и декремент

Очень часто в программах используется операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание — декрементом. Для этих операций существуют специальные команды процессора: INC и DEC. Обратите внимание, что эти команды не изменяют значение флага *CF*.

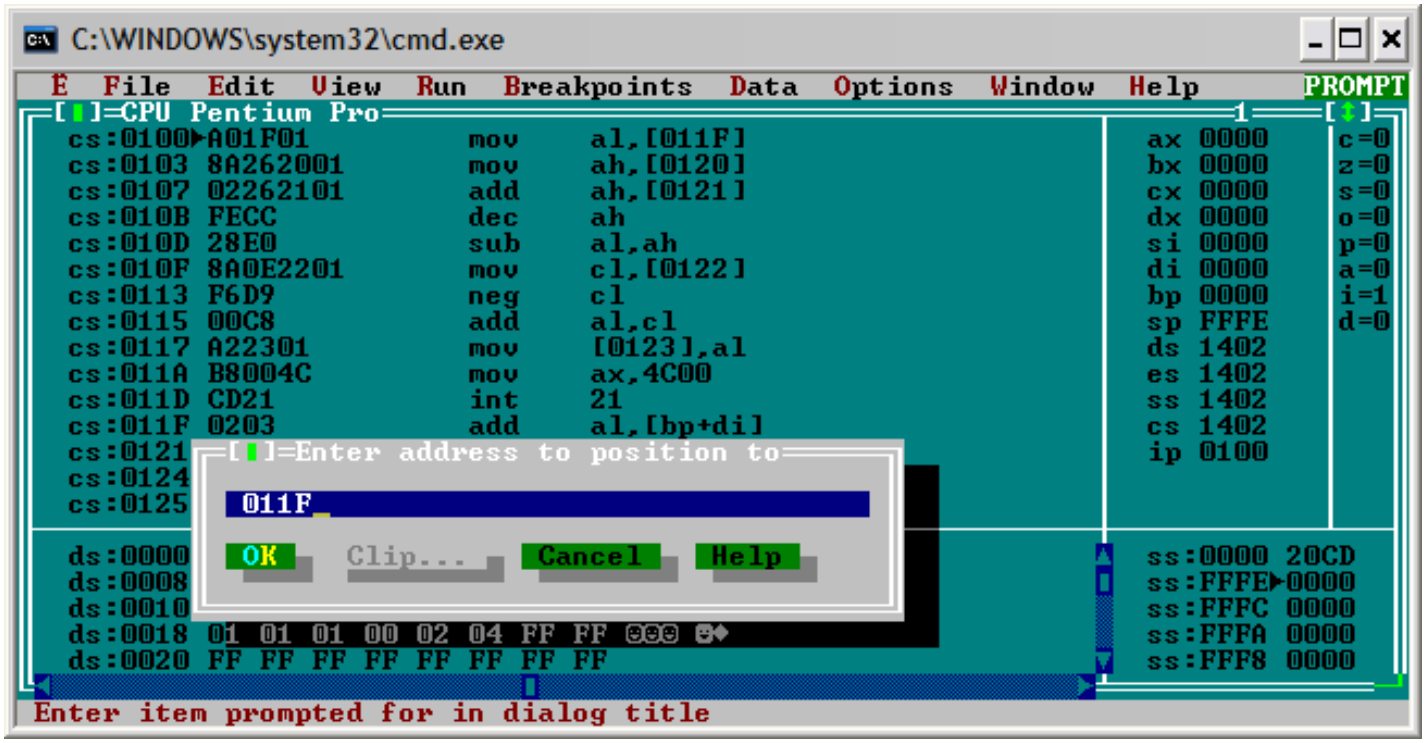
Пример программы

Чтобы всё стало совсем понятно, напишем небольшую программу. Требуется вычислить значение формулы: *e=a-(b+c-1)+(-d)*. Все числа являются 8-битными целыми со знаком. Объявим их после кода и придумаем какие-нибудь значения. Вот что у меня получилось:

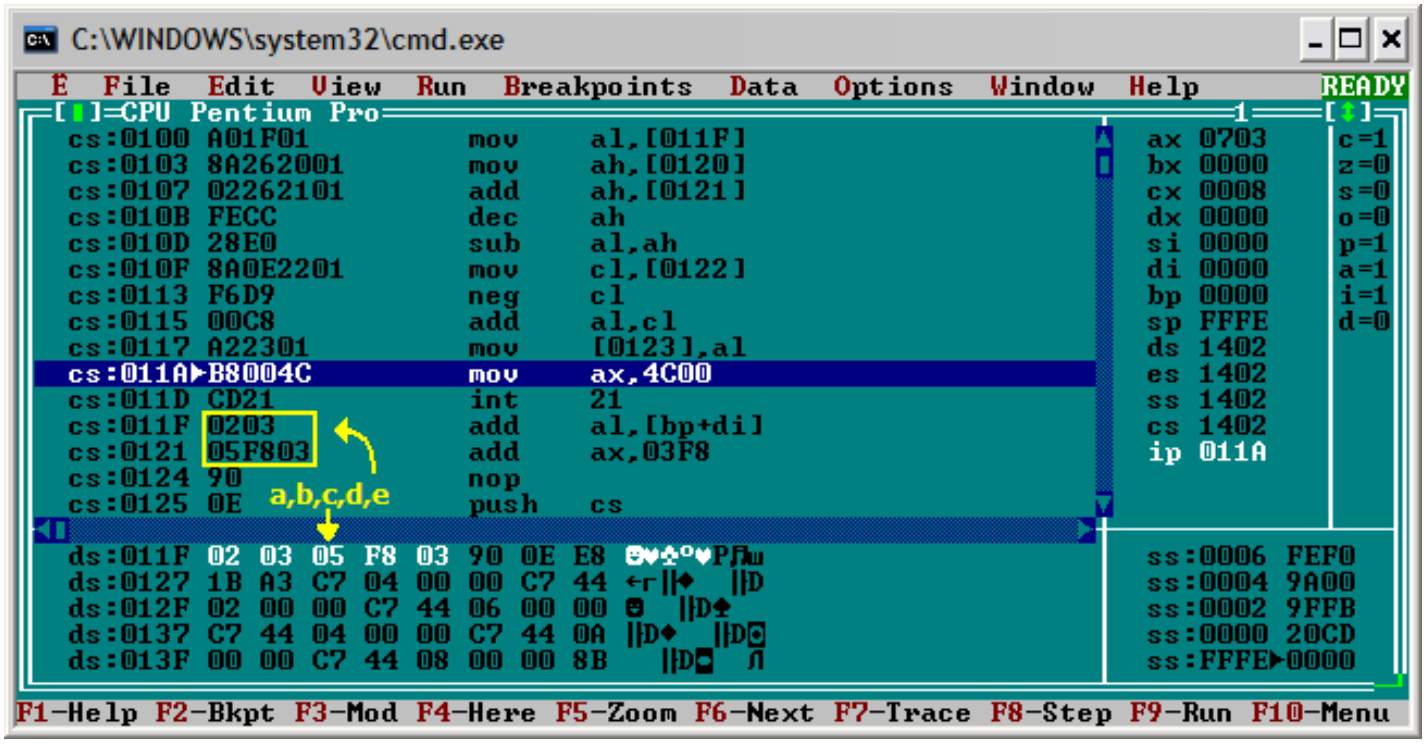
```
 1 use16               ;Генерировать 16-битный код
 2 org 100h            ;Программа начинается с адреса 100h
 3
 4     mov al,[a]      ;Загружаем значение a в AL
 5     mov ah,[b]      ;Загружаем значение b в AH
 6     add ah,[c]      ;AH = AH + c = b+c
 7     dec ah          ;AH = AH - 1 = b+c-1
 8     sub al,ah       ;AL = AL - AH = a-(b+c-1)
 9     mov cl,[d]      ;CL = d
10     neg cl          ;CL = -CL = -d
11     add al,cl       ;AL = AL + CL = a-(b+c-1)+(-d)
12     mov [e],al      ;Сохраняем результат в e
13
14     mov ax,4C00h    ;\
15     int 21h         ;/ Завершение программы
16 ;-------------------------------------------------------
17 a db 2
18 b db 3
19 c db 5
20 d db -8
21 e db ?
```

Квадратные скобки означают, что операнд находится по адресу, указанному внутри этих скобок. Так как вместо имени переменной FASM подставляет её адрес, то такая запись позволяет прочитать или записать значение переменной.

Запустив программу в Turbo Debugger, можно посмотреть её выполнение по шагам. Значения переменных можно увидеть в окне дампа памяти. Для этого нужно кликнуть правой кнопкой в этом окне и выбрать в меню пункт *Goto*…. Переменные начинаются в памяти с адреса 011Fh (этот адрес в первой команде).



В этих байтах легко угадываются наши переменные:



Упражнение

Напишите программу для вычисления формулы *k=m+1-(n-1-r)*. Все числа 16-битные целые со знаком. Запустите в отладчике и проверьте правильность вычисления. Результаты можете выкладывать в комментариях 🙂

Следующая часть »

Комментарии:

ekslamer
29-03-2010 23:33

;Ну както так

```
use16
org 100h

mov al,[m]
inc al
mov ah,[n]
dec ah
mov cl,[r]
sub ah,cl
sub al,ah
mov [k],al

mov ax,4c00h
int 21h

m db 5
n db 4
r db 3
k db ?
```

[Ответить]

xrnd
30-03-2010 19:12

Замечательно. Всё верно написано 🙂

[Ответить]

xrnd
30-03-2010 19:14

Единственное, что можно вместо

```
mov cl,[r]
sub ah,cl
```

вычитать сразу:

```
sub ah,[r]
```

[Ответить]

RoverWWWorm
30-03-2010 11:22

```
use16
org 100h
```

```
mov ah,[m]
inc ah

mov al,[n]
dec al
sub al,[r]

sub ah,al

mov [k],ah

mov ax,4c00h
int 21h

;_____________
m db 5
n db 4
r db 3
k db ?
```

[Ответить]

xrnd
30-03-2010 19:15

Да, всё правильно 🙂

[Ответить]

ymn
30-03-2010 15:17

я сделал вот так. не знаю правильно или нет, в отладчике ещё не смотрел =)

```
use16 ;генерируем 16-битный код
org 100h ;начинаем с адреса 100h

mov al,[n] ;загружаем значение n
dec al ;al=n-1
mov ah,[r] ;загружаем значение r
sub al,ah ;al=n-1-r
xor ah,ah ;очищаем ah (может и не надо)
mov ah,[m] ;загружаем значение m
inc ah ;ah=m+1
sub ah,al ;ah=m+1-(n-1-r)
mov [k],ah ;сохраняем результат

;__________завершение программы__________-

mov ax, 4C00h
int 21h

;__________инициализация переменных_________
```

```
m db -1
n db 2
r db 5
k db ?
```

[Ответить]

xrnd
30-03-2010 19:19

В общем, работать программа будет правильно, но её можно сократить.

xor ah,ah делать не надо, потому что новое значение просто заменит в регистре старое.
И ещё не обязательно помещать r в регистр для вычитания.
Можно заменить:

```
mov ah,[r]
sub al,ah
```

на одну команду:

```
sub al,[r]
```

[Ответить]

ymn
30-03-2010 19:26

спасибо, учту на будущее =)

[Ответить]

IgorKing
26-08-2010 13:44

Я сделал, но по-моему работает неправильно…

```
use16
org 100h

mov al,[n]
dec al
sub al,[r]
mov dh,[m]
inc dh
sub dh,al
mov [k],dh

mov ax,4c00h
int 21h
;_____
n db -5
r db 2
m db 3
k db ?
```

[Ответить]

xrnd
26-08-2010 21:47

Вроде всё правильно 🙂
Единственное что числа тут 8-битные. Почему-то все делали это упражнение с 8-битными целыми, хотя в задании 16-битные целые.

[Ответить]

IgorKing
27-08-2010 07:03

Извиняюсь, это я в отладчике не мог найти свои переменные, т.к. упустил одну деталь с отрицательными числами.

[Ответить]

VanOk
24-10-2010 18:09

Блин жаль я коменты не читал
«mov ah,[r]
sub al,ah

на одну команду:

sub al,[r]»
Но так проверьте пожалуйста
```
;k=m+1-(n-1-r)
use16
org 100h
mov ah,[m]
inc ah
mov ch,[n]
mov cl,[r]
dec ch
sub ch,cl
sub ah,ch
mov [k],ah
mov ax,4C00h
int 21h
;————————————————-
m db 6
n db -7
r db 7
k db ?
```

[Ответить]

xrnd
24-10-2010 18:39

Всё правильно!
Навык оптимально использовать команды приходит с опытом, а для начала и так хорошо.

[Ответить]

oleg
15-11-2010 00:34

В комментах ошибки, а автор аплодирует 🙂

Читаем условие задачи: «Напишите программу для вычисления формулы k=m+1-(n-1-r). Все числа 16-битные целые со знаком.»

1. Числа 16-битные, а это значит объявление должно быть dw.

2. Повсеместно используется ah, al для чисел в 16 бит, должен использоваться весь регистр. То есть ax, bx ,cx и т.д.

Мой код:

```
use16
org 100h

mov ax,[m] ; ax = m
inc ax ; ax = m + 1
mov bx,[n] ; bx = n
dec bx ; bx = n — 1
mov cx,[r] ; cx = r
sub bx,cx ; bx = bx — cx = n-1 — r
sub ax,bx ; ax = ax — bx = m+1 — n-1-r
mov [k],ax ; k = ax = m+1 — n-1-r

mov ax,4c00h
int 21h

;ff

m dw 213
n dw 442
r dw -33
k dw ?
```

[Ответить]

xrnd
15-11-2010 17:03

Видимо, вдохновлённые моим примером, все упорно делали это упражнение с 8-битными регистрами. Я сначала не заметил, а потом решил, что и так неплохо 🙂

Твой код больше соответствует заданию и написано всё правильно.
Единственное что — можно не помещать r в регистр перед вычитанием.
Вместо двух команд:

```
    mov cx,[r] ; cx = r
```

```
    sub bx,cx ; bx = bx - cx = n-1 - r
```

будет одна:

```
    sub bx,[r] ;bx = bx - r = n-1 - r
```

[Ответить]

Хороший Человек
23-11-2010 20:51

Спасибо всем, кто выложил задачи, они мне помогли по Архитектуре эвм. Я кое-что не догонял, теперь всё ясно. На примере как делать:)

[Ответить]

Борис
12-12-2010 17:23

А я вот так сделал, и с трудом прикрутил вывод а терминал вроде считает

use16 ;Ãåíåðèðîâàòü 16-áèòíûé êîä
org 100h ;Ïðîãðàììà íà÷èíàåòñÿ ñ àäðåñà 100h

mov ax,[m] ;Çàãðóæàåì çíà÷åíèå m â AX
inc ax ;Èíêðåìåíò AX = m + 1
mov bx,[n] ;Çàãðóæàåì çíà÷åíèå n â BX
dec bx ;Äåêðåìåíò BX = n — 1
sub bx,[r] ;=(n-1-r)
neg bx ;=-(n-1-r)
add ax, bx ;=(m-1)-(n-1-r)
mov [k],ax ;Ñîõðàíÿåì ðåçóëüòàò â e
add ax,030h
mov [print], al
mov [print+1], ah

mov dx,print ;Â DX àäðåñ ñòðîêè.
mov ah,9 ;Íîìåð ôóíêöèè DOS.
int 21h ;Îáðàùåíèå ê ôóíêöèè DOS.

mov ax,4C00h ;\
int 21h ;/ Çàâåðøåíèå ïðîãðàììû
;———————————————————————————-
m dw 2
n dw 3
r dw 5
k dw ?
print db «oo$»

[Ответить]

xrnd
12-12-2010 22:16

Вычисления написаны правильно. Вывод в терминал — не совсем. но этого и не требовалось.

Сложение здесь можно заменить вычитанием, сократив код на одну команду:

```
    neg bx
    add ax,bx
```

То же самое:

```
    sub ax,bx
```

[Ответить]

Борис
13-12-2010 20:09

Спасиб. Пока все понятно!

[Ответить]

anton
13-12-2010 12:09

Привет ! Спасибо за отличные уроки, все доступно, понятно, без воды.
Вот мой вариант:

use16
org 100h

mov ax,[m]
add ax,[r]
add ax,2
sub ax,[n]
mov [k],ax

mov ax,4c00h
int 21h
;——————————
m dw 30000
n dw 567
r dw 5678
k dw ?

[Ответить]

xrnd
13-12-2010 20:00

Схитрил, упростил формулу 🙂 Можно и так. Код написан правильно.

[Ответить]

Гость
05-01-2011 00:12

Здравствуйте , спасибо за статью.
У меня вопрос возник
из примера автора e=a-(b+c-1)+(-d)
a db 2
b db 3
c db 5
d db -8
e db ?
А сколько там должно получится ? ,)

[Ответить]

xrnd
10-01-2011 23:03

Можно подставить числа и посчитать:
e = 2-(3+5-1)+(-(-8)) = 2-7+8 = 3

Код должен работать правильно и с другими числами.

[Ответить]

Philin
11-01-2011 20:21

Привет, с Новым годом…)
перепроверял, но все же…?)
use16
org 100h

mov ax, [m]
inc ax
mov bx, [n]
dec bx
sub bx, [r]
sub ax,bx
mov [k],ax

mov ax,4C00h
int 21h
;_____
m dw 5
n dw 7
r dw 9
k dw ?

[Ответить]

xrnd
11-01-2011 20:44

Привет. Поздравляю, всё правильно 🙂

[Ответить]

Георгий
18-01-2011 14:45

; Учитель, проверьте пожалуйста код на правильность :). И спасибо за ваши статьи.

```
use16
org 100h

mov al,[m]
inc al
mov ah,[n]
dec ah
sub ah,[r]
sub al,bh
mov [k],al

mov ax, 4C00h
int 21h
; —— init data————
m db 2
n db -3
r db 5
k db ?
```

[Ответить]

Георгий
18-01-2011 15:01

; ну это было немного нерационально, вот так 🙂

```
use16
org 100h

mov al,[m]
sub al,[n]
add al,[r]
add al,2
mov [k],al

mov ax, 4C00h
int 21h
; —— init data————
m db 2
n db 3
r db 4
k db ?
```

[Ответить]

xrnd
18-01-2011 21:53

😀 пожалуйста.

Оба варианта правильные.

[Ответить]

Knight212
03-02-2011 01:02

use16
org 100h

mov ax, [m]
inc ax
mov cx, [n]
dec cx
sub cx, [r]
sub ax, cx
mov [k], ax

mov ax, 4C00h
int 21h

m dw 24
n dw -18
r dw 9
k dw ?

ax = 0035 (16) = 53 (10)

[Ответить]

xrnd
09-02-2011 15:15

Да, это хорошая программа 🙂

[Ответить]

mustdie
24-03-2011 14:57

use16
org 100h

mov al,[n]
dec al
sub al,[r]
mov ah,[m]
inc ah
sub ah, al
mov [k],ah

mov ax,4C00h
int 21h
;————————————————————————————-
m db ?
n db ?

r db ?
k db ?

[Ответить]

xrnd
01-04-2011 13:21

В принципе всё верно, только в задании предлагалось сделать с 16-битными целыми 🙂

[Ответить]

annihilator
14-04-2011 19:55

знаю что поздно пишу, но только нашёл этот сайт,
а если так?:

use16 ;генерировать 16-ти битный код
org 100h ;программа начинается с адреса 100h

mov ax, [m] ;загружаем значение m в AX
inc ax ;AX=AX+1=m+1
mov bx, [n] ;загружаем значение n в BX
dec bx ;BX=BX-1=n-1
sub bx, [r] ;BX=BX-r=n-1-r
sub ax, bx ;AX=AX-BX=k=m+1-(n-1-r)

mov ax, 4C00h ;\
int 21h ;/завершение программы

;———————————————————————

m dw 2
n dw 3
r dw 5

[Ответить]

xrnd
15-04-2011 00:24

Никогда не поздно! Тут многие комментаторы доказали, что даже простое упражнение можно выполнить множеством способов 🙂

Код правильный, только результат остается в AX и нигде не сохраняется.

[Ответить]

annihilator
14-04-2011 20:09

или так?:

k=m+1-(n-1-r)=m-n+r+2

```
use16 ;генерировать 16-ти битный код
org 100h ;программа начинается с адреса 100h

mov ax, [m] ;загружаем значение m в AX
sub ax, [n] ;AX=AX-n=m-n
add ax, [r] ;AX=AX+r=m-n+r
add ax, 2 ;AX=AX+2=m-n+r+2=k

mov ax, 4C00h ;\
int 21h ;/завершение программы

;————————————————————————————

m dw 2
n dw 3
r dw 5
```

[Ответить]

xrnd
15-04-2011 00:26

И тоже правильно. Если упростить формулу, получается меньше кода.

[Ответить]

annihilator
15-04-2011 09:01

Я имел ввиду, что может и не нужно использовать для хранения результата отдельный регистр, ведь по сути, нам важен лишь конечный результат. Если не сохранять результат в отдельном регистре, то при более сложных программах можно экономить эти самые регистры, ведь их количество конечно и сильно ограничено.
Значение регистра AX уже содержит в себе ответ, оно «потеряется» если его не сохранить в другой регистр или нет?
Мне кажется, ответ можно сохранить в оперативку или на жёсткий диск или даже на флешку… при желании наверное можно отправить и по сети.

[Ответить]

xrnd
15-04-2011 14:43

Результат потеряется, если записать в регистр другое число или вызвать функцию DOS. Я предполагал в задании, что k тоже будет переменной в памяти. В регистрах обычно хранятся какие-то промежуточные результаты.

[Ответить]

annihilator
15-04-2011 10:21

Раз AX используется DOS-ом (если я правильно понял), то можно ли вот так?:

```
use16 ;генерировать 16-ти битный код
org 100h ;программа начинается с адреса 100h

mov bx, [m] ;загружаем значение m в BX
sub bx, [n] ;BX=BX-n=m-n
add bx, [r] ;BX=BX+r=m-n+r
add bx, 2 ;BX=BX+2=m-n+r+2=k

mov ax, 4C00h ;\
int 21h ;/завершение программы

;————————————————————————————

m dw 2
n dw 3
r dw 5
```

[Ответить]

xrnd
15-04-2011 14:47

Можно и так — это тоже правильный вариант.
AX не используется DOS-ом, пока его используешь ты.
Но в него записывается номер функции DOS перед int 21h.
В данном случае можно использовать AX и не париться.

Если бы код часто вызывал какую-то функцию DOS — тогда имело бы смысл в AH записать её номер, а вычисления делать в других регистрах.

[Ответить]

annihilator
15-04-2011 12:20

Ещё вопрос 🙂

```
m dw 2
n dw 10
```
— я так понимаю это хранится в оперативке или в кеш памяти процессора?

Я так думаю, что кеш память и регистры процессора это разные вещи (кеш бывает 64Кб-4Мб а то и больше, а суммарный «объём» всех регистров будет гораздо меньше и измеряется в байтах), это так?

Какая разница между кеш памятью, регистрами и оперативкой?
Как с ними работает ассемблер, есть ли разница?

Судя по тому что я «нарыл» в других источниках регистры работают гораздо быстрее оперативки (поскольку это сам процессор), но имеет ли смысл считывать данные оз озу, выполнять действия в регистрах, а потом снова записывать в озу (или это кеш…)

Может в данном случае проще сделать вот так?:

```
add m, n
```

add [m], n

add m, [n]

add [m], [n]

Как правильно и правильно ли вообще, путаюсь со скобками

[Ответить]

plan4ik
15-04-2011 13:30

http://www.wasm.ru/article.php?article=1022001 — Низкоуровневое программирование для дZенствующих … ознакомся с чем имеешь дело (важно)

1) m dw 2 — хранится в оперативке размером в 2 байта

?: Я так думаю, что кеш память и регистры процессора это разные вещи (кеш бывает 64Кб-4Мб а то и больше, а суммарный «объём» всех регистров будет гораздо меньше и измеряется в байтах), это так?
2) Кеш память это внутрепроцессорная память она содержыт часто-используемые опкоды, а регистры (это штото вроде твоих рук ты должен взять (из общей памяти RAM) в руки (в регистры) сделать чтото и положить (сохранить результат в RAM) чтобы можно было еще чтото подержать наночь глядя 😉) — в общих чертах это штото похожее

а вообще те надо больше читать книги Юрова например или Абеля на худой конец

[Ответить]

annihilator
15-04-2011 13:51

Короче без регистров низя, всё делается только через них…
А кеш программисту вообще не доступен.
спс.
Если я не так понял поправте.

[Ответить]

xrnd
15-04-2011 15:01

Кэш недоступен, но его нужно учитывать, если требуется оптимизировать программу по скорости выполнения. Например, располагать код и данные в памяти компактно, чтобы они лучше кэшировались.

[Ответить]

annihilator
15-04-2011 13:57

P.S.: Юров и Абель есть, я только начал их читать (обе сразу, по очереди) только не всё там так просто и понятно как тута, некоторые вещи на китайском… Да и до практики там ещё далеко, не дошёл…

[Ответить]

annihilator
15-04-2011 14:04

Эх… Тяжело на свете октябрёнку Пете… 🙂

[Ответить]

plan4ik
15-04-2011 14:14

337610360 ICQ стучись и я те помогу

[Ответить]

xrnd
15-04-2011 14:57

Данные храняться в оперативной памяти.
При выполнении кода, скорее всего, процессор их помещает в кэш. (Но кэш можно отключить в BIOSe и тогда процессор будет работать напрямую с оперативной памятью).

Быстрее всего происходит обращение к регистрам.
Медленнее кэш. Оперативная память ещё на порядок медленнее.

имеет ли смысл считывать данные оз озу, выполнять действия в регистрах, а потом снова записывать в озу
Да, имеет смысл. Работа с регистрами происходит быстрее. К тому же команды, работающие с регистрами, короче так как не содержат адресов памяти.

add m, n
add [m], n
add m, [n]
add [m], [n]

Нет, так нельзя. Оба операнда не могут находиться в памяти. Хотя бы один должен быть регистром, либо непосредственным значением.
Можно так:
mov ax,[m]
add ax,[n]

[Ответить]

annihilator
15-04-2011 17:54

Спасибо за помощь

[Ответить]

xamelon
07-05-2011 22:46

```
use16
org 100h
mov ah,[n]
mov al,[r]
dec ah
sub ah,al
mov cl,[m]
inc cl
sub cl,ah

mov ax,4c00h
int 21h
;————————————-
m db 2
n db 3
r db 1
k db ?
```

Вот так вот правильно? И кстати. Есть ещё один вопрос. Можно вначале все операции сделать с переменными, а потом загрузить их в регистры?

[Ответить]

Ваш комментарий

Имя *

Почта (скрыта) *

Сайт

Добавить

Уведомить меня о новых комментариях по email.

Уведомлять меня о новых записях почтой.