

Forum:[Hack Forums](#)

Thread: [Assembly\(NASM\) Add/Divide/Multiply](#)

friiz1337 - January 26, 2019, 9:29 am

I'm a beginner in Assembly and I'm trying to make my program execute the following formula using 3 already specified numbers for a,b and h: $((a+b)/2)*h$

The result, I'm trying to store in a variable called result. However I keep getting wrong output (0004). I have commented the way I believe the code works. Could I get some help?

Here is my code:

Code section .data

a: dw 3

b: dw 7

h: dw 4

result: dw 0

section .text

global main

main:

mov ecx, [a] ;ecx = 3

mov edx, [b] ;edx = 7

add ecx, edx ;ecx + edx = 10

mov eax, ecx ;eax = 10

mov edx, 0 ;edx = 0

mov ebx, eax ;ebx = 10

mov eax, 2 ;eax = 2

div eax ;ebx/eax = 10/2 = 5

mov eax, eax ;eax = 5

mov ebx, [h] ;ebx = 4

mul ebx ;5x4 = 20

mov [result], ebx ;result = 20

mov eax, 1

mov ebx, 0

int 80h

Thank you

SIM-3364521 - January 26, 2019, 6:56 pm

with the [] you dereference the variable.

lets say you have something like:

a db 5

if you want to move the value of a in eax you would do:

mov eax, a

if you would do something like this:

mov eax, [a]

you would actually move the value at the address 0x5 in to the register.

That was the first thing i noticed i hope that was the only one and that you get it to work!

I would suggest you to have a look at the different addressing modes you can use in assembler:

https://www.tutorialspoint.com/assembly_programming/assembly_addressing_modes.htm

(edit: just read through the tutorial on that site and its very basic but you should get the idea. On the wikipedia page about assembler there is a more in depth explanation of different addressing modes also some that arent listed on tutorialspoint)

hope that was helpful

friiz1337 - January 27, 2019, 8:11 am

(Jan 26, 2019 07:56 PM)Erarnitox Wrote: [\[CITE\]](#) with the [] you dereference the variable.

lets say you have something like:

a db 5

if you want to move the value of a in eax you would do:

mov eax, a

if you would do something like this:

mov eax, [a]

you would actually move the value at the address 0x5 in to the register.

That was the first thing i noticed i hope that was the only one and that you get it to work!

I would suggest you to have a look at the different addressing modes you can use in assembler:

https://www.tutorialspoint.com/assembly_programming/assembly_addressing_modes.htm

(edit: just read through the tutorial on that site and its very basic but you should get the idea.

On the wikipedia page about assembler there is a more in depth explanation of different addressing modes also some that arent listed on tutorialspoint)

hope that was helpful

I don't think it will work, I tried both ways, none of them got me anything. I thought the [] are to specify the value that is hold in the variable (abh in this case)

HFDone - January 27, 2019, 8:27 am

your problem is the registers you using! read the description

<https://www.aldeid.com/wiki/X86-assembly/Instructions/div>

<https://www.aldeid.com/wiki/X86-assembly/Instructions/mul>

friiz1337 - January 27, 2019, 8:34 am

(Jan 27, 2019 09:27 AM)HFDone Wrote: [\[CITE\]](#) your problem is the registers you using! read the description

<https://www.aldeid.com/wiki/X86-assembly/Instructions/div>

<https://www.aldeid.com/wiki/X86-assembly/Instructions/mul>

Are you talking about multiplying and division only? Or adding too?

Thanks

I managed to edit the code a bit, I get a value(30) close to the intended one(20). However I can't seem to understand where miscalculation is happening.. Any ideas?

Code section .data

a: dw 3

b: dw 7

h: dw 4

result: dw 0

divide: dw 2

section .text

global main

main:

mov ebx, [a] ;ebx = 3

```

mov edx, [b] ;edx = 7
add ebx, edx ;ebx = 10
mov edx, 0 ;edx = 0
mov eax, ebx ;eax = 10
mov ecx, [divide] ;ecx = 2
div ecx ;eax= 5 edx=0
mov edx, 0 ;edx = 0
mov eax, eax ;eax =5
mul eax ;result stored in edx after multiplication
mov [result], edx
int 80h

```

HFDone - January 27, 2019, 10:29 am

```

a: dd 3
b: dd 7
h: dd 4
result: dd 0

```

```

Code mov ebx, [a] ;ecx = 3
mov edx, [b] ;edx = 7
add ebx, edx ;ebx = 10
mov edx, 0 ;edx = 0
mov eax, ebx ;eax = 10
mov ecx, 2 ;ecx = 2
div ecx ;eax = 5
mov edx, 0 ;edx = 0
mul eax ;eax = 25
mov [result], eax
you dont use the h var for the mul
change
mov edx, 0 ;edx = 0
mul eax ;eax = 25
to
mov edx, [h] ;edx = 4
mul edx ;eax = 20

```

friiz1337 - January 27, 2019, 10:37 am

(Jan 27, 2019 11:29 AM)HFDone Wrote: [\[CITE\]](#) a: dd 3
b: dd 7
h: dd 4
result: dd 0

```

Code mov ebx, [a] ;ecx = 3
mov edx, [b] ;edx = 7
add ebx, edx ;ebx = 10
mov edx, 0 ;edx = 0
mov eax, ebx ;eax = 10
mov ecx, 2 ;ecx = 2
div ecx ;eax = 5
mov edx, 0 ;edx = 0
mul eax ;eax = 25
mov [result], eax
you dont use the h var for the mul

```

I understand, check my new code. The answer should be 20, not 25. I got to 30.
Thank you

HFDone - January 27, 2019, 10:38 am

(Jan 27, 2019 11:37 AM)friiz1337 Wrote: [\[CITE\]](#) I understand, check my new code. The answer should be 20, not 25. I got to 30.

Thank you
i edit my reply

friiz1337 - January 27, 2019, 10:46 am

(Jan 27, 2019 11:38 AM)HFDone Wrote: [\[CITE\]](#) i edit my reply
Thank you dude.

(Jan 27, 2019 11:38 AM)HFDone Wrote: [\[CITE\]](#) i edit my reply
Is it possible to explain why do you used DD instead of DW?

HFDone - January 27, 2019, 11:00 am

cuz you are using 32 bit (dword) registers and your vars are 16 bit (word)

friiz1337 - January 27, 2019, 2:27 pm

(Jan 27, 2019 12:00 PM)HFDone Wrote: [\[CITE\]](#) cuz you are using 32 bit (dword) registers and your vars are 16 bit (word)

Thank you, I got it ?