# While, Do While, For loops in Assembly Language (emu8086)

Asked 7 years, 7 months ago Modified 7 months ago Viewed 192k times



I want to convert simple loops in high-level languages into assembly language (for emu8086) say, I have this code:



#### Your privacy

By clicking "Accept all cookies", you agree Stack Exchange can store cookies on your device and disclose information in accordance with our Cookie Policy.

Accept all cookies

Customize settings

edited Aug 20, 2019 at 3:16



asked Feb 23, 2015 at 1:07

Glynn Bacanto **399** 1 5 12

ns and conditional jumps in assembly already, rewrite the code using if and goto first and/or create a flowchart.

3, 2015 at 1:19 🧪

Siegler Feb 23, 2015 at 1:27

- 1 Just lose the e from the register name. e stands for extended (I believe) it indicates 32 bit-wide registers instead of 16 bits. 500 Internal Server Error Feb 23, 2015 at 2:44
- 1 Yes, apart from whatever else you need in the body. For example a do-while is a single conditional jump start: { body }; if (condition) goto start; Jester Jul 10, 2021 at 11:48

2 Answers Sorted by: Highest score (default) 

♦

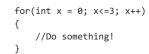


### **For-loops:**

66

For-loop in C:







The same loop in 8086 assembler:

```
xor cx,cx ; cx-register is the counter, set to 0

loop1 nop ; Whatever you wanna do goes here, should not change cx inc cx ; Increment cmp cx,3 ; Compare cx to the limit jle loop1 ; Loop while less or equal
```

That is the loop if you need to access your index (cx). If you just wanna to something 0-3=4 times but you do not need the index, this would be easier:



bes here, should not change cx ents cx and jumps to label if not 0  $\,$ 

ction a constant amount of times, you could also use an assembler-directive which will just hardcore that

#### Your privacy

By clicking "Accept all cookies", you agree Stack Exchange can store cookies on your device and disclose information in accordance with our Cookie Policy.

Accept all cookies

Customize settings

```
do{
    //Do something!
}
while(x==1)
```

The same loop in assembler:

```
mov ax,1
loop1 nop ; Whatever you wanna do goes here
cmp ax,1 ; Check wether cx is 1
je loop1 ; And loop if equal
```

## While-loops

While-loop in C:

```
while(x==1){
    //Do something
}
```

The same loop in assembler:



Your privacy

By clicking "Accept all cookies", you agree Stack Exchange can store cookies on your device and disclose information in accordance with our Cookie Policy.

Accept all cookies

Customize settings

er because it is pretty much standard. For the other loop conditions you can take a register of your liking. with all the instructions you wanna perform in the loop.

answered Mar 21, 2015 at 17:47



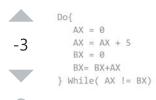
Maximilian Schier **1,489** 14 18

rn CPUs.. - Peter Cordes Nov 8, 2017 at 1:37

whenever possible, <u>for the same reason compilers do</u>: code runs faster with fewer instructions inside the loop. (Usually nping to the bottom of the loop like you're doing here in your while loop.) – <u>Peter Cordes</u> Jan 14, 2018 at 4:41

to the loop condition check before the loop's first iteration. - ecm Aug 28, 2019 at 19:33

mization that x<=3 is known to be true on the first iteration, allowing that check to be skipped. That's 100% standard 1 you know they will run at least once, because of fixed loop bounds. Other kinds of for loops may need to sometimes 3 ps that follow the idiomatic pattern for looping x from 0 .. n. – Peter Cordes Apr 9, 2021 at 1:23



Do while loop always checks the loop the condition at the end of each iteration.

Share Improve this answer Follow

edited Apr 15, 2021 at 7:21



2,453 7 21 31

answered Apr 9, 2021 at 0:06



DHS2020

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.



#### Your privacy

By clicking "Accept all cookies", you agree Stack Exchange can store cookies on your device and disclose information in accordance with our Cookie Policy.

Accept all cookies

Customize settings