

# Re: FC4 and Assembly Language Program

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Fedora/2007-06/msg00811.html>

---

- *From:* Les <hlhowell@xxxxxxxxxxxx>
  - *Date:* Sun, 03 Jun 2007 10:25:38 -0700
- 

On Fri, 2007-06-01 at 20:46 +0500, azeem ahmad wrote:

hi list  
i am about to make a bootable floppy for test  
but i am being unable to get it done  
please review the code below and tell me if there is any problem with it

```
-----  
.code16 #assembler directive to start 16-bit real mode for execution  
.text /*assembler directive to tell the start of 'read-only'  
code segment*/  
.org 0x00 /*assembler directive to set the origon to sector 0  
needed to copy the program to the very first sector  
of the floppy disk*/
```

```
.global _start /*assembler directive to export the start section to  
all other programs, i.e. to make it visible to programs  
like linker or other user programs*/
```

```
_start: #label of the start routine  
mov 0x07C0, %ax /*move immidiate operand 07C0 to the  
accumulator register ax, so that it can be  
transferred to data segment register*/  
mov %ax, %ds /*move contents of register ax to register ds*/  
call _boot #call the boot section  
ret #return the control to the caller routine
```

```
_boot: #label of the boot section  
mov $msg, %si /*move the address of the character string  
constant 'msg' to the source index register*/  
call _disp #call subroutine disp  
ret #return the control to the caller routine
```

```
_disp: #label of the disp routine  
cld /*clear direction flag, to permit string  
instructions to increment index registers  
by their own*/  
lodsbyte /*load the string pointed by ds:si in a  
byte by byte manner into the accumulator*/
```

## Re: FC4 and Assembly Language Program

```
or %al, %al /*check if the entire string has been loaded
byte-wise by oring al to al, if the result is
zero, it shows there are no more byte to load*/
jz ret #jump if al zero to return
mov $0xE, %ah /*else put code for 'write a character on the
screen and move forward' into the ah*/
mov $7, %bh /*enable normal attribute for all the blank
lines on the screen*/
int $0x10 /*call interrupt 0x10, which is responsible
for the video display, it will take codes
from ah and bh*/
jmp _disp
```

```
msg: #label of the string definition
.ascii "My Boot System" #definition of the string
```

```
.org 510 #set origin to 510
.word 0xAA55 #
```

---

saved the file with the name myos.s  
then

```
#as myos.s -o myos.o
#objcopy -O binary myos.o BOOT
#dd if=./BOOT of=/dev/fd0 bs=512 count=1
```

---

the system tries to boot from the floppy and it boots (as it doesn't give an error or it doesn't go to the next boot device. but it doesn't display the string that I had to show from the string

please check it and tell me about any possible errors in the code

Regards  
Azeem

---

Express yourself instantly with MSN Messenger! Download today it's FREE!  
<http://messenger.msn.click-url.com/go/onm00200471ave/direct/01/>

Well, Does the BIOS actually contain the software interrupt code you are attempting to access. That is question one.

Question 2 is what would your bootstrap return to? At boot time, the system invokes a short reader that will read the first sector. This program is built into the bios. There is no IO available for this program, thus the system bio typically uses the system beep to give error messages which you have to interpret by counting the beeps and looking them up in the documentation on that particular bios.

## Re: FC4 and Assembly Language Program

You cannot just debug a boot or bios routine, because the debugger already has been loaded with all the required tools, including all the IO routines. You have set yourself quite a task to boot a system. A good way to learn is to find an older CP/M reference manual. In the early days, one sort of had to write their own bootstraps, BIOS package and then use patches to CP/M to get a working system. Today the drivers are either in ROM on the boards and read during the boot process, or they are downloaded from a CD or the internet. If you are doing this for a class project, you will need to look more closely at the references supplied for the class. If this is something you are doing to learn, this is not the appropriate list. You should google for developer mailing list. Check some of the guys working on robotics or some such, where they are writing simple OS's (some not so simple) for embedded controllers. These days, most people skip this part and go with a developer package which has a boot and bios package built in, like the Basic Stamp or the 8051 kits that are available. Writing a bootable system for a modern PC would be difficult I think, and to fit it on a floppy would be very hard, unless you restricted it to something like DOS or CP/M. In the early days, we had memory mapped displays, and serial I/O. That was it. the chips had 16 interrupts and most systems used about 12. That is no longer true today.

Good luck and I hope you can find some one to guide you on your quest.

Regards,  
Les H

--

fedora-list mailing list

fedora-list@xxxxxxxxxxx

To unsubscribe: <https://www.redhat.com/mailman/listinfo/fedora-list>