University of Derby
Department of Electronics, Computing and Mathematics

Assessment Specification

Module Code and Title: Systems Programming (6CC514)	
Assignment No. and Title: Assessed Component 1	
Assessment Tutor: Wayne Rippin	**Weighting Towards Module Grade:** 35%
Date Set: 5th October 2017	**Hand-In Deadline Date:** Friday 20th October 2017 at 5pm

Penalty for Late Submission

Any work submitted late will not be marked and will be treated as a non-submission.

The only exception to this rule is if you have been given permission for the work to be handed in later due to an appropriate support plan.

If you have been ill or there have been other exceptional extenuating circumstances, you may request a later submission. In this case, you should follow the EEC procedures. Please note that you must still submit something for this assignment by the deadline date.

Level of Collaboration

All assessments in this portfolio are individual pieces of work. No collaboration with other students is allowed. In addition, you should not ask for help on any Internet site such as Stack Overflow, etc.

Learning Outcomes covered in this Assignment:

On successful completion of the module, students will be able to:

1. *Demonstrate critical awareness of the concepts and issues related to systems programming.*
2. *Design, implement, and evaluate programs employing aspects of systems programming.*

Task

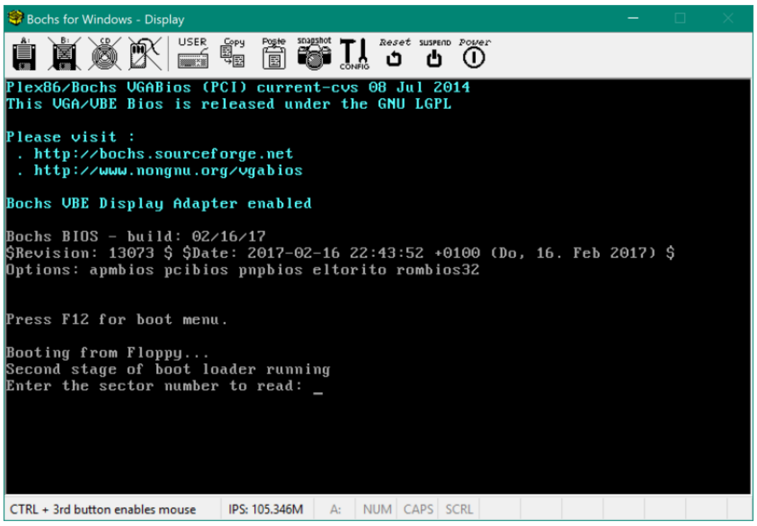
The objective of this assessed component is to assess your ability to read and develop programs using Intel x86 assembler language.

You are to write code to display the contents of sectors on the disk image you are creating in this module.

You have been given a starting point for this assignment in Step3.zip. You should add your code to boot2.asm. You may add code to any include file or create additional include files as appropriate. Note that boot2.asm includes floppy16.asm. This includes routines that you will find useful for reading a sector from the disk.

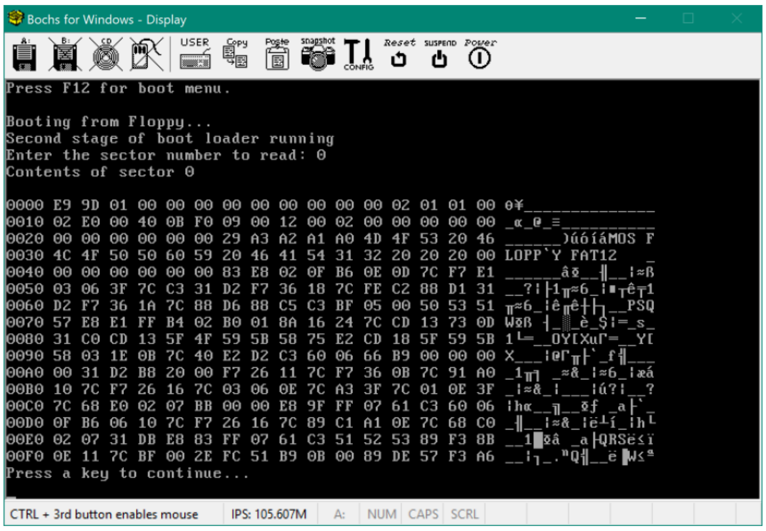
A complete solution will do the following:

a) Ask the user to enter the number of the sector they want to read. The following is an example of what this might look like:



```
Plex86/Bochs VGABios (PCI) current-cvs 08 Jul 2014
This VGA/VBE Bios is released under the GNU LGPL

Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

Bochs BIOS - build: 02/16/17
$Revision: 13073 $ $Date: 2017-02-16 22:43:52 +0100 (Do, 16. Feb 2017) $
Options: apmbios pcibios pnpbios eltorito rombios32


Press F12 for boot menu.

Booting from Floppy...
Second stage of boot loader running
Enter the sector number to read: _
```

b) The user should be able to enter in a sector number and then the code should display the contents of that sector, ideally in both hexadecimal and characters. Each line should begin with the offset into the sector that the line starts at (displayed as a 4-digit hex number). For example, the display might look like the following:



```
Press F12 for boot menu.

Booting from Floppy...
Second stage of boot loader running
Enter the sector number to read: 0
Contents of sector 0

0000 E9 9D 01 00 00 00 00 00 00 00 00 00 02 01 01 00 θ¥______________
0010 02 E0 00 40 0B F0 09 00 12 00 02 00 00 00 00 00 _α_@_≡__________
0020 00 00 00 00 00 00 29 A3 A2 A1 A0 4D 4F 53 20 46 ______)úóíáMOS F
0030 4C 4F 50 50 60 59 20 46 41 54 31 32 20 20 20 00 LOPP`Y FAT12   _
0040 00 00 00 00 00 00 83 E8 02 0F B6 0E 0D 7C F7 E1 ______âΦ__╢__¦≈ß
0050 03 06 3F 7C C3 31 D2 F7 36 18 7C FE C2 88 D1 31 __?¦├1╥≈6_¦■┬ê╤1
0060 D2 F7 36 1A 7C 88 D6 88 C5 C3 BF 05 00 50 53 51 ╥≈6_¦ê╓ê┼├┐__PSQ
0070 57 E8 E1 FF B4 02 B0 01 8A 16 24 7C CD 13 73 0D WΦß ┤_░_è_$¦═_s_
0080 31 C0 CD 13 5F 4F 59 5B 58 75 E2 CD 18 5F 59 5B 1└═__OY[XuΓ═__Y[
0090 58 03 1E 0B 7C 40 E2 D2 C3 60 06 66 B9 00 00 00 X___¦@Γ╥├`_f╣___
00A0 00 31 D2 B8 20 00 F7 26 11 7C F7 36 0B 7C 91 A0 _1╥╕ _≈&_¦≈6_¦æá
00B0 10 7C F7 26 16 7C 03 06 0E 7C A3 3F 7C 01 0E 3F _¦≈&_¦___¦ú?¦__?
00C0 7C 68 E0 02 07 BB 00 00 E8 9F FF 07 61 C3 60 06 ¦hα__╗__Φƒ _a├`_
00D0 0F B6 06 10 7C F7 26 16 7C 89 C1 A1 0E 7C 68 C0 _╢__¦≈&_¦ë┴í_¦h└
00E0 02 07 31 DB E8 83 FF 07 61 C3 51 52 53 89 F3 8B __1█Φâ _a├QRSë≤ï
00F0 0E 11 7C BF 00 2E FC 51 B9 0B 00 89 DE 57 F3 A6 __¦┐_.ⁿQ╣__ë▐W≤ª
Press a key to continue...
```

c) A sector contains 512 bytes, so displaying all of the sector at a time at 16 bytes per line will scroll off the screen. A complete solution should display a smaller number of lines (say 16) and then pause until the user presses a key.

d) Once a sector has been displayed, a complete solution should ask the user for another sector number to display.

e) An enhancement would be to ask the user to enter how many sectors they wish to display, as well as the starting sector number and display the contents of all of those sector.s

Notes.

1. If you are just reading one sector at a time into memory, a suitable address to read it to would be 0D000h. If you are going to read more than one sector, think about how many sectors you will read at once and where you will read them to, so as to avoid interfering with the stack or expanding into the next 64K segment.
2. Code has been provided in floppy16.asm (included in boot2.asm) to read sectors off the disk. The code is commented and you are expected to read the code to understand which routines you should use and how to call them.
3. To read a character from the keyboard, you should read about INT 16h.
4. When displaying the contents of a sector as characters, I would recommend that you display any character with an ASCII code of less than 32 as an underscore. This is because the BIOS will treat some characters (such as 0Dh, 0Ah, 09h, etc.) as special characters and perform different actions for these characters, rather than just displaying a character.

Submission

You need to submit a zip file containing your code to the submission point on Course Resources by the due date and time. The name of the zip file must be Assessment1_xxxxxxxxx.zip where xxxxxxxxx is your student number. For example, if your student number is 051234567, yout submission should be called Assessment1_051234567.zip.

Please do NOT use any other name for your submission or use any compression other than .zip. The tools used to download submissions from Course Resources for marking rely on a consistent naming scheme and compression mechanism.

The zip file must contain the complete folder containing your code, including the code given to you and the associated makefiles. You must run 'make clean' before creating your zip file to ensure that only the source files are submitted. Your zip file should not include the binary files or disk images.

Failure to follow these submission requirements will result in a reduction of the grade awarded to your assignment by 20%.

Assessment Criteria

You do not need to implement all of the functionality described above to obtain a pass grade (40% or more) on this assessment. Part of the grading will depend on how much has been implemented.

If your code can only display the contents of a sector as hexadecimal and the sector number to read is hardcoded (i.e. it is not read from the keyboard), the maximum grade you can achieve is 45%.

If your code can display the contents of a sector as both hexadecimal and characters, but the sector number to read is hardcoded (i.e. it is not read from the keyboard), the maximum grade you can achieve is 55%.

To achieve more than 55%, you must provide code that enables the user to enter the sector number they want to read using the keyboard.

The final grade for this assessment will be determined based on:

- The level of functionality implemented
- The level of error handling included
- The readability of the code
- The efficiency of the code.

Remember that failure to follow the submission requirements will result in a reduction in the grade awarded of 20%.