

OPERATING SYSTEM

System Introduction OS development with PC Simulator 'Bochs'



By :

FADHLIH HASAN SETIAWAN

L200194209

X

INFORMATION TECHNOLOGY

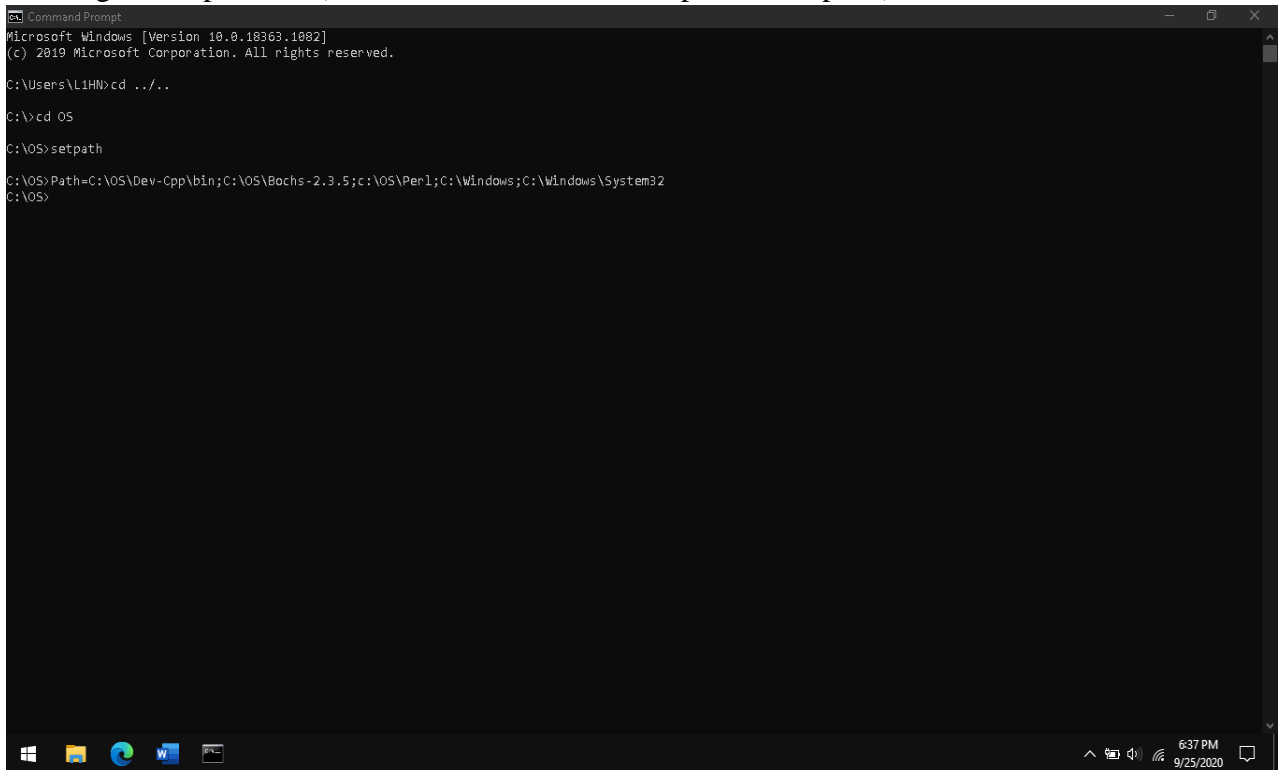
FACULTY OF COMMUNICATION AND INFORMATICS

UNIVERSITY OF MUHAMMADIYAH SURAKARTA

2019/2020 SCHOOL YEAR

Work steps:

1. Run the command prompt program.
2. Reset directory and enter directory 'C: \ OS' using the 'cd ../..' commands and 'cd OS'.
3. Running the setpath file (used to set the curvature of practicum 'path').



```
Command Prompt
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\L1HN>cd ../../..
C:\>cd OS
C:\OS>setpath
C:\OS>Path=C:\OS\Dev-Cpp\bin;C:\OS\Bochs-2.3.5;c:\OS\Perl;c:\Windows;c:\Windows\System32
C:\OS>
```

The screenshot shows a Windows Command Prompt window with a dark background. The title bar reads 'Command Prompt'. The window content shows the following sequence of commands and their outputs: 'cd ../../..' (moving from the user's home directory to the root of the C: drive), 'cd OS' (moving into the OS directory), and 'setpath' (displaying the updated PATH environment variable). The PATH variable is set to include the Dev-Cpp bin directory, the Bochs directory, the Perl directory, and the Windows system directories. The prompt is currently at 'C:\OS>'. The Windows taskbar is visible at the bottom, showing the Start button and several open applications (File Explorer, Edge, Word, and a terminal). The system tray on the right shows the time as 6:37 PM on 9/25/2020.

4. Run the makefile program in the fp.disk file using the 'make fp.disk' command (running this command means that we will compile the source code for the program 'boot.asm', as output file 'boot.bin' and its contents copied to in the bootsector floppy image file 'floppya.img').

5. Check the directory using the 'dir' command.

```
Command Prompt
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\L1HN>cd ../../

C:\>cd OS

C:\OS>setpath

C:\OS>Path=C:\OS\Dev-Cpp\bin;C:\OS\Bochs-2.3.5;c:\OS\Perl;c:\Windows;c:\Windows\System32
C:\OS>cd Lab/Lab1

C:\OS\LAB\LAB1>make fp.disk
nasm boot.asm -o boot.bin -f bin
dd if=boot.bin of=floppya.img
rawwrite dd for windows version 0.5.
Written by John Newbigin <jn@it.swin.edu.au>
This program is covered by the GPL. See copying.txt for details
1+0 records in
1+0 records out

C:\OS\LAB\LAB1>dir
Volume in drive C has no label.
Volume Serial Number is 6082-DFA8

Directory of C:\OS\LAB\LAB1

09/21/2020  01:43 PM  <DIR>          .
09/21/2020  01:43 PM  <DIR>          ..
09/10/2019  04:10 PM             10,235 bochsout.txt
12/15/2008  04:17 PM             1,628 bochsrc.bxrc
12/14/2008  12:02 PM            14,365 boot.asm
09/25/2020  06:30 PM              512 boot.bin
09/16/2015  07:51 AM              512 boots.bin
12/15/2008  12:47 AM              78 dosfp.bat
09/25/2020  06:30 PM          1,474,560 floppya.img
12/14/2008  11:45 AM             7,966 kernel.asm
12/15/2008  04:21 PM              227 Makefile
12/15/2008  12:20 PM              44 s.bat
               10 File(s)      1,510,127 bytes
               2 Dir(s)      145,978,400,768 bytes free

C:\OS\LAB\LAB1>
```

6. Delete the floppies image using the command 'del floppya.img'.
7. Call 'bximage', then select type fd, and the file is named 'floppya.img' with the default capacity option (bximage is useful for creating floppy disk based image files).

```
Command Prompt

C:\OS\LAB\LAB1>del floppya.img

C:\OS\LAB\LAB1>bximage

=====
                bximage
        Disk Image Creation Tool for Bochs
        $Id: bxImage.c,v 1.32 2006/06/16 07:29:33 vruppert Exp $
=====

Do you want to create a floppy disk image or a hard disk image?
Please type hd or fd. [hd] fd

Choose the size of floppy disk image to create, in megabytes.
Please type 0.16, 0.18, 0.32, 0.36, 0.72, 1.2, 1.44, 1.68, 1.72, or 2.88.
[1.44]
I will create a floppy image with
  cyl=80
  heads=2
  sectors per track=18
  total sectors=2880
  total bytes=1474560

What should I name the image?
[a.img] floppya.img

Writing: [] Done.

I wrote 1474560 bytes to floppya.img.

The following line should appear in your bochsrc:
  floppya: image="floppya.img", status=inserted
(The line is stored in your windows clipboard, use CTRL-V to paste)

Press any key to continue

C:\OS\LAB\LAB1>dir
Volume in drive C has no label.
Volume Serial Number is 6082-DFA8

Directory of C:\OS\LAB\LAB1

09/25/2020  06:45 PM  <DIR>          .
09/25/2020  06:45 PM  <DIR>          ..
```

```
Command Prompt
C:\OS\LAB\LAB1>dir
Volume in drive C has no label.
Volume Serial Number is 6002-DFA8

Directory of C:\OS\LAB\LAB1

09/25/2020  06:45 PM  <DIR>          .
09/25/2020  06:45 PM  <DIR>          ..
09/10/2019  04:19 PM           10,235 bochsout.txt
12/15/2008  04:17 PM           1,628 bochssrc.bxrc
12/14/2008  12:02 PM           14,365 boot.asm
09/25/2020  06:39 PM              512 boot.bin
09/16/2015  07:51 AM              512 boots.bin
12/15/2008  12:47 AM              78 dosfp.bat
09/25/2020  06:45 PM       1,474,560 floppy.aimg
12/14/2008  11:45 AM           7,966 kernel.asm
12/15/2008  04:21 PM             227 Makefile
12/15/2008  12:20 PM              44 s.bat
                10 File(s)      1,510,127 bytes
                2 Dir(s)      145,978,249,216 bytes free

C:\OS\LAB\LAB1>
```

8. Run the dosfp command (used to format the image file that was created earlier).
9. Type 'format B:' then enter 2x, then close the pc simulator by clicking the power button.

```
Bochs for Windows - Console
C:\OS\LAB\LAB1>dir
Volume in drive C has no label.
Volume Serial Number is 6002-DFA8

Directory of C:\OS\LAB\LAB1

09/25/2020  06:45 PM  <DIR>          .
09/25/2020  06:45 PM  <DIR>          ..
09/10/2019  04:19 PM           10,235 bochsout.txt
12/15/2008  04:17 PM           1,628 bochssrc.bxrc
12/14/2008  12:02 PM           14,365 boot.asm
09/25/2020  06:39 PM              512 boot.bin
09/16/2015  07:51 AM              512 boots.bin
12/15/2008  12:47 AM              78 dosfp.bat
09/25/2020  06:45 PM       1,474,560 floppy.aimg
12/14/2008  11:45 AM           7,966 kernel.asm
12/15/2008  04:21 PM             227 Makefile
12/15/2008  12:20 PM              44 s.bat
                10 File(s)      1,510,127 bytes
                2 Dir(s)      145,978,249,216 bytes free

C:\OS\LAB\LAB1>dosfp
C:\OS\LAB\LAB1>cd "...\Bochs-2.3.5\dos"
C:\OS\Bochs-2.3.5>dos>..\bochs -q -f bochssrc2.txt
000000000001[APIC?] local apic in  initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
000000000001[      ] reading configuration from bochssrc2.txt
000000000001[      ] installing win32 module as the Bochs GUI
000000000001[      ] using log file bochsout.txt

Bochs for Windows - Display
MSDCDEX Version 2.23
Copyright (C) Microsoft Corp. 1986-1993. All rights reserved.
Drive B: = Driver OSLAB unit 0
A:\>format B:
Insert new diskette for drive B:
and press ENTER when ready...

Checking existing disk format.
Formatting 1.44M
Format complete.

Volume label (11 characters, ENTER for none)?

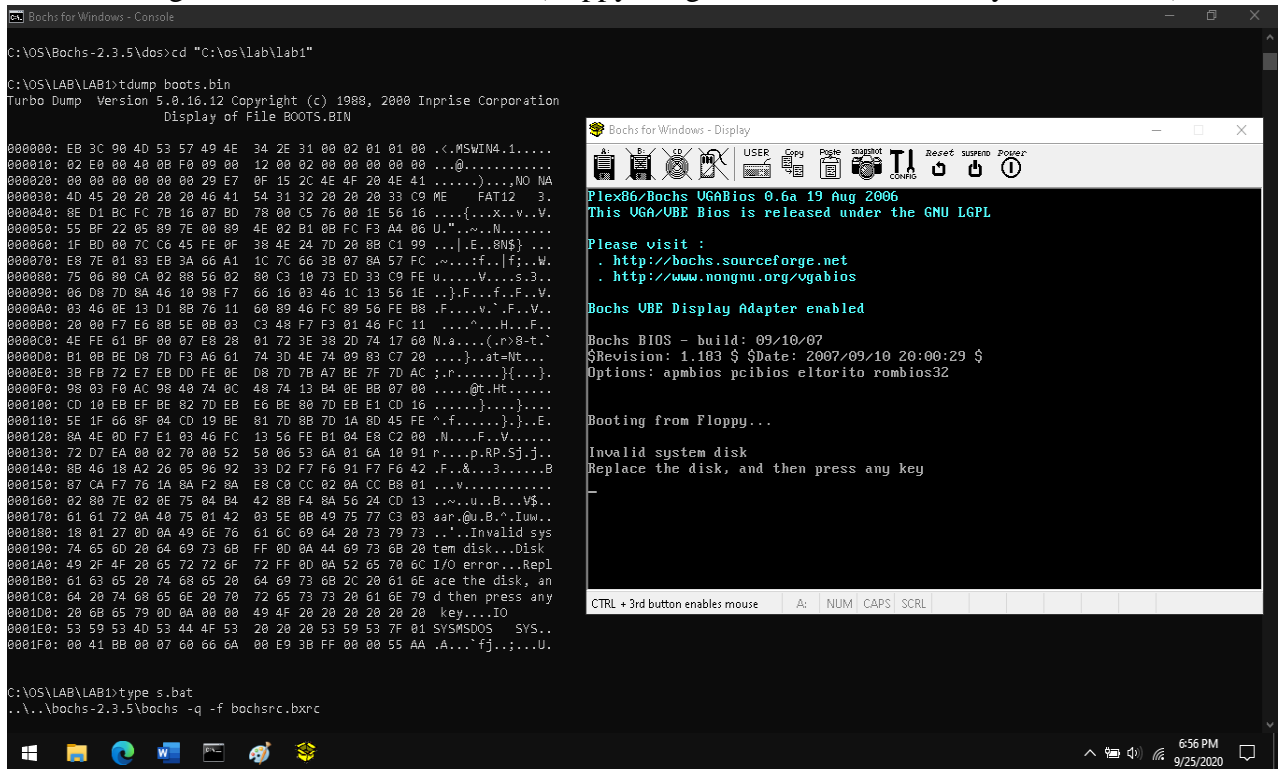
1,457,664 bytes total disk space
1,457,664 bytes available on disk

512 bytes in each allocation unit.
2,047 allocation units available on disk.

Volume Serial Number is 4064-1AF2
Format another (Y/N)?n
A:\>
```

10. To view the bootsector data, use the command 'tdump boots.bin'.
11. View the file 's.bat' with the command 'type s.bat'.

12. Boot the image file with the 's' command (floppya.img does not contain file system / kernel).



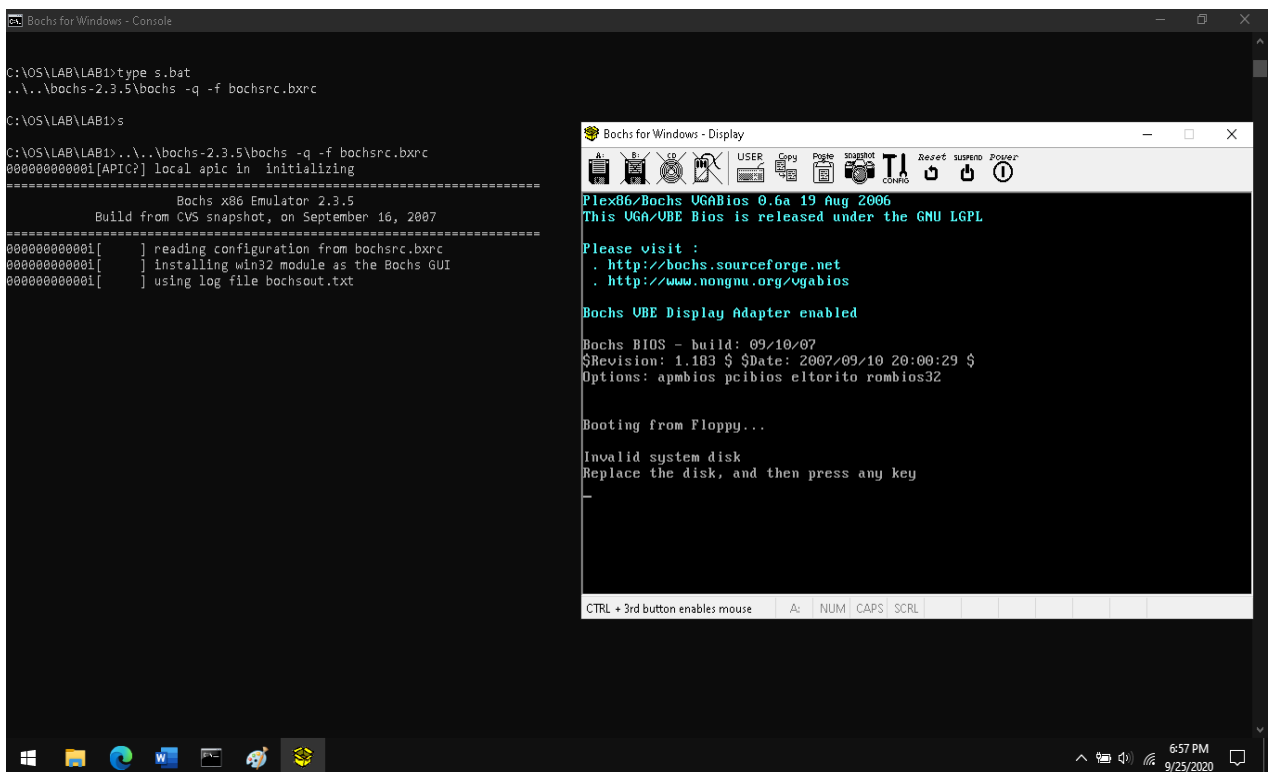
```
C:\OS\Bochs-2.3.5>cd "C:\os\lab\lab1"

C:\OS\LAB\LAB1>type boots.bin
Turbo Dump  Version 5.0.16.12 Copyright (c) 1988, 2000 Inprise Corporation
Display of File BOOTS.BIN

000000: EB 3C 00 4D 53 57 49 4E 34 2E 31 00 02 01 01 00 .<.MSWIN4.1....
000010: 02 E0 00 40 08 F0 09 00 12 00 02 00 00 00 00 00 ...@.....
000020: 00 00 00 00 00 00 29 E7 0F 15 2C 4E 4F 20 4E 41 .....).NO NA
000030: 4D 45 20 20 20 20 46 41 54 31 32 20 20 20 33 C9 ME FAT12 3.
000040: 8E D1 0C FC 7B 16 07 80 78 00 C5 76 00 1E 56 16 ....{...X.v.v.V.
000050: 55 BF 22 05 89 7E 00 89 4E 02 B1 08 FC F3 A4 06 U."...N.....
000060: 1F BD 00 7C C6 45 FE 0F 38 4E 24 70 20 8B C1 99 ...|.E..8N$} ...
000070: E8 7E 01 83 EB 3A 66 A1 1C 7C 66 3B 07 8A 57 FC ...|.f..|fj..W.
000080: 75 06 80 CA 02 88 56 02 80 C3 10 73 ED 33 C9 FE u.....V....s.3..
000090: 06 D8 7D 8A 46 10 98 F7 66 16 03 46 1C 13 56 1E ...}.F...f..F..V.
0000A0: 03 46 0E 13 D1 8B 76 11 60 89 46 FC 89 56 FE B8 .F...v...F..V..
0000B0: 20 00 F7 E6 8B 5E 0B 03 C3 48 F7 F3 01 46 FC 11 .....^...H...F..
0000C0: 4E FE 61 BF 00 07 E8 28 01 72 3E 38 2D 74 17 60 N.a....(r)8-t.`
0000D0: B1 0B BE D8 7D F3 A6 61 74 3D 4E 74 09 83 C7 20 ....).at=Nt...
0000E0: 3B FB 72 E7 EB DD FE BE D8 7D 7B A7 BE 7F 70 AC ;r.....}{....
0000F0: 98 03 F0 AC 98 40 74 0C 48 74 13 B4 0E BB 07 00 .....@t.Ht....
000100: CD 10 EF BF BE 82 7D EB E6 BE 80 7D EB E1 CD 16 .....}......
000110: 5E 1F 66 8F 04 CD 19 BE 81 7D 8B 7D 1A 8D 45 FE ^f.....}.E..
000120: 8A 4E 0D F7 E1 03 46 FC 13 56 FE B1 04 E8 C2 00 .N...F..V.....
000130: 72 D7 EA 00 02 78 00 52 50 06 53 6A 01 6A 10 91 r...p.RP.Sj.j..
000140: 8B 46 18 A2 26 05 96 02 33 D2 F7 F6 91 F7 F6 42 .F.&...3.....B
000150: 87 CA F7 76 1A 8A F2 8A E8 C0 CC 02 0A CC B8 01 ...v.....V$.
000160: 02 80 7E 02 0E 75 04 B4 42 8B F4 8A 56 24 CD 13 ...u..B...V$.
000170: 61 61 72 0A 40 75 01 42 03 5E 0B 49 75 77 C3 03 aar.@u.B.^Iuw..
000180: 18 01 27 0D 0A 49 6E 76 61 6C 69 64 20 73 79 73 ...'.Invalid sys
000190: 74 65 6D 20 64 69 73 68 FF 0D 0A 44 69 73 68 20 tem disk...Disk
0001A0: 49 2F 4F 28 65 72 72 6F 72 FF 0D 0A 52 65 70 6C I/O error...Repl
0001B0: 61 63 65 20 74 68 65 20 64 69 73 68 2C 20 61 6E ace the disk, an
0001C0: 64 20 74 68 65 6E 20 70 72 65 73 73 20 61 6E 79 d then press any
0001D0: 20 68 65 79 0D 0A 00 00 49 4F 20 20 20 20 20 20 key...IO
0001E0: 53 59 53 4D 53 44 4F 53 20 20 20 53 59 53 7F 01 SYMSDOS SYS..
0001F0: 00 41 BB 00 07 00 66 6A 00 E9 3B FF 00 00 55 AA .A...~fj...;...U.

C:\OS\LAB\LAB1>type s.bat
..\..\bochs-2.3.5\bochs -q -f bochssrc.bxrc
```

13. To load the file system, call 'dosfp'.

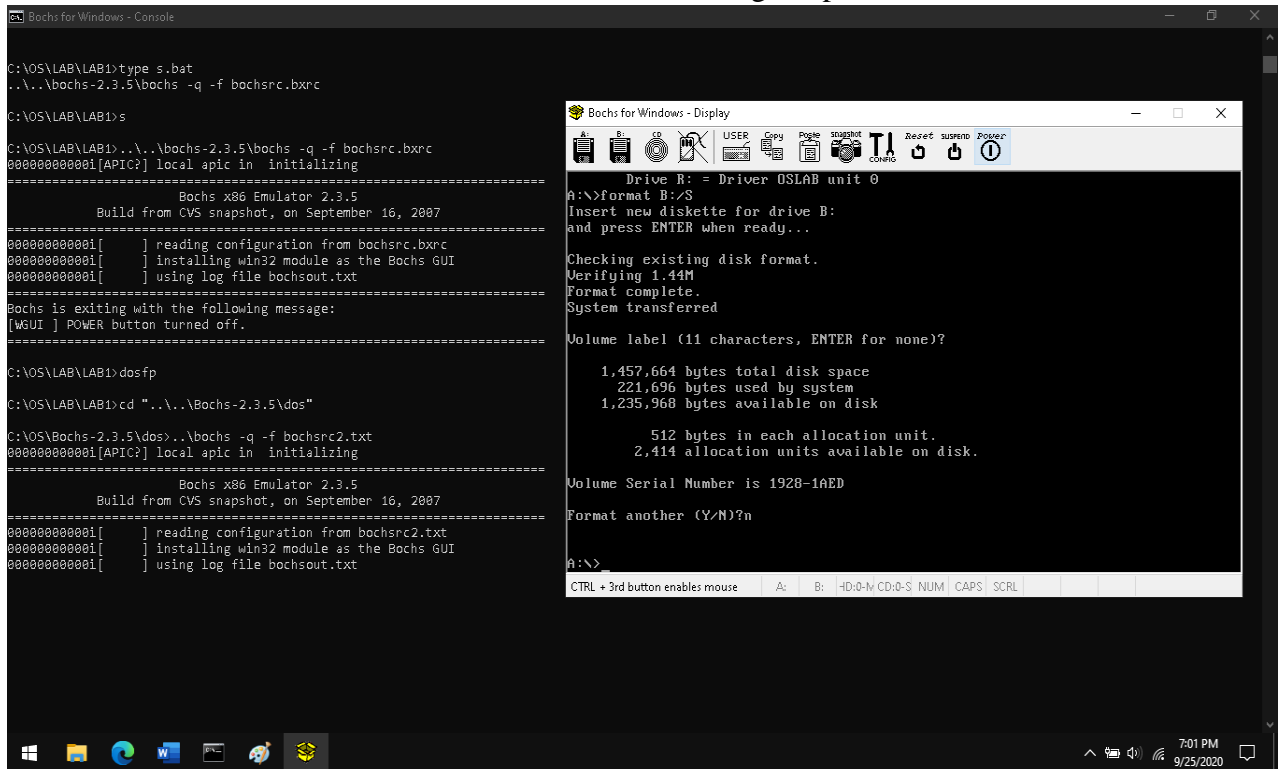


```
C:\OS\LAB\LAB1>type s.bat
..\..\bochs-2.3.5\bochs -q -f bochssrc.bxrc

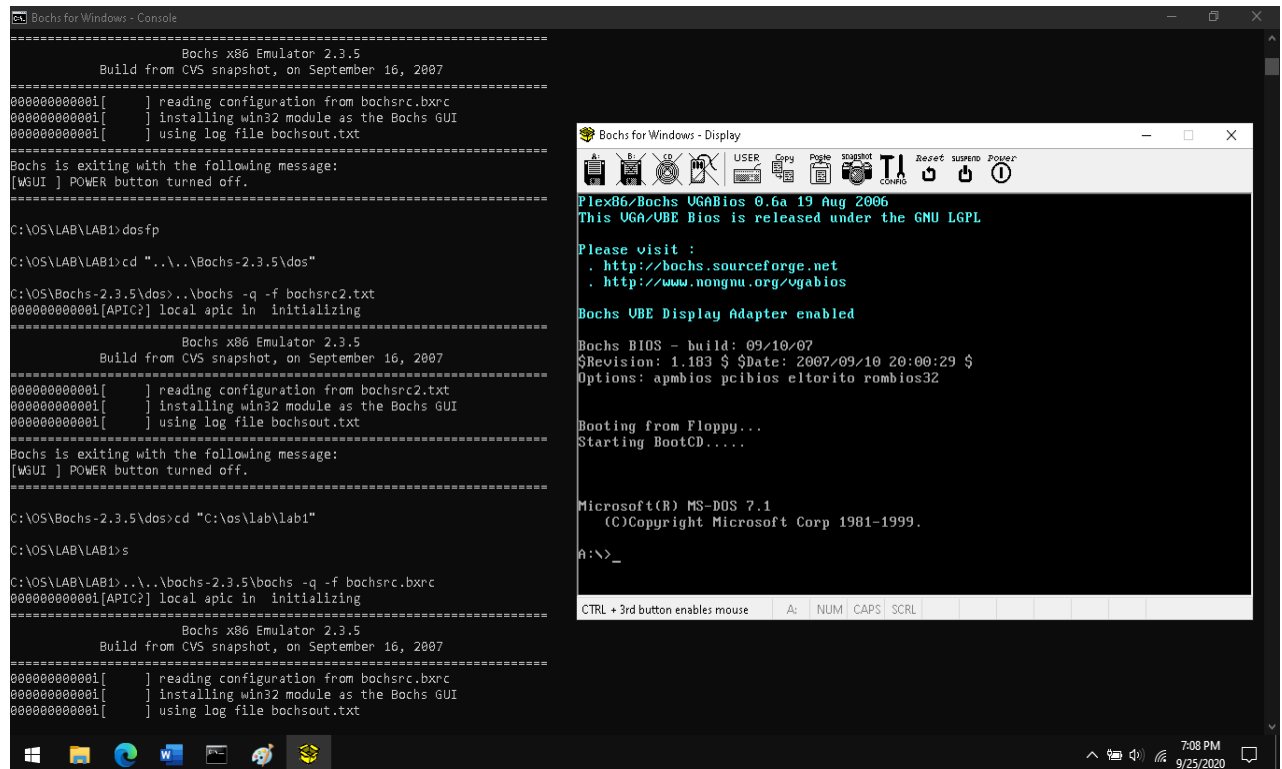
C:\OS\LAB\LAB1>s
C:\OS\LAB\LAB1>..\..\bochs-2.3.5\bochs -q -f bochssrc.bxrc
00000000001[APIC?] local apic in initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
00000000001[ ] reading configuration from bochssrc.bxrc
00000000001[ ] installing win32 module as the Bochs GUI
00000000001[ ] using log file bochsout.txt

C:\OS\LAB\LAB1>type s.bat
..\..\bochs-2.3.5\bochs -q -f bochssrc.bxrc
00000000001[APIC?] local apic in initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
00000000001[ ] reading configuration from bochssrc.bxrc
00000000001[ ] installing win32 module as the Bochs GUI
00000000001[ ] using log file bochsout.txt
```

14. Format with the command 'format B: / S' then close it using the power button.



15. Double check with the 's' command.



Assignment:

1. What is the meaning of 'ASCII' code, make standard ASCII table not need to be extended, write ASCII code in decimal, binary and hexadecimal number format and encoded characters and symbols.

Answer:

ASCII stands for (American Standard Code for Information Interchange), and the meaning of ASCII itself is an international standard in letter and symbol codes such as Hex and Unicode but ASCII is more universal, for example 124 is for the character "|".

Table of ASCII

| Binary | Dec | Hex | Character | Binary | Dec | Hex | Character | Binary | Dec | Hex | Character |
|----------|-----|-----|-----------|---------|-----|-----|-----------|---------|-----|-----|-----------|
| 0100000 | 32 | 20 | © | 0100000 | 64 | 40 | @ | 1100000 | 96 | 60 | ` |
| 0100001 | 33 | 21 | ! | 0100001 | 65 | 41 | A | 1100001 | 97 | 61 | a |
| 0100010 | 34 | 22 | “ | 1000010 | 66 | 42 | B | 1100010 | 98 | 62 | b |
| 0100011 | 35 | 23 | # | 1000011 | 67 | 43 | C | 1100011 | 99 | 63 | c |
| 0100100 | 36 | 24 | \$ | 1000100 | 68 | 44 | D | 1100100 | 100 | 64 | d |
| 0100101 | 37 | 25 | % | 1000101 | 69 | 45 | E | 1100101 | 101 | 65 | e |
| 0100110 | 38 | 26 | & | 1000110 | 70 | 46 | F | 1100110 | 102 | 66 | f |
| 0100111 | 39 | 27 | ‘ | 1000111 | 71 | 47 | G | 1100111 | 103 | 67 | g |
| 0101000 | 40 | 28 | (| 1001000 | 72 | 48 | H | 1101000 | 104 | 68 | h |
| 0101001 | 41 | 29 |) | 1001001 | 73 | 49 | I | 1101001 | 105 | 69 | i |
| i0101010 | 42 | 2A | * | 1001010 | 74 | 4A | J | 1101010 | 106 | 6A | j |
| 0101011 | 43 | 2B | + | 1001011 | 75 | 4B | K | 1101011 | 107 | 6B | k |
| 0101100 | 44 | 2C | , | 1001100 | 76 | 4C | L | 1101100 | 108 | 6C | l |
| 0101101 | 45 | 2D | - | 1001101 | 77 | 4D | M | 1101101 | 109 | 6D | m |
| 0101110 | 46 | 2E | . | 1001110 | 78 | 4E | N | 1101110 | 110 | 6E | n |
| 0101111 | 47 | 2F | / | 1001111 | 79 | 4F | O | 1101111 | 111 | 6F | o |
| 0110000 | 48 | 30 | 0 | 101000 | 80 | 50 | P | 1110000 | 112 | 70 | p |
| 0110001 | 49 | 31 | 1 | 1010001 | 81 | 51 | Q | 1110001 | 113 | 71 | q |
| 0110010 | 50 | 32 | 2 | 1010010 | 82 | 52 | R | 1110010 | 114 | 72 | r |
| 0110011 | 51 | 33 | 3 | 1010011 | 83 | 53 | S | 1110011 | 115 | 73 | s |
| 0110100 | 52 | 34 | 4 | 1010100 | 84 | 54 | T | 1110100 | 116 | 74 | t |
| 0110101 | 53 | 35 | 5 | 1010101 | 85 | 55 | U | 1110101 | 117 | 75 | u |
| 0110111 | 54 | 36 | 6 | 1010110 | 86 | 56 | V | 1110110 | 118 | 76 | v |
| 0110111 | 55 | 37 | 7 | 1010111 | 87 | 57 | W | 1110111 | 119 | 77 | w |
| 0111000 | 56 | 38 | 8 | 1011000 | 88 | 58 | X | 1111000 | 120 | 78 | x |
| 0111001 | 57 | 39 | 9 | 1011001 | 89 | 59 | Y | 1111001 | 121 | 79 | y |
| 0111010 | 58 | 3A | : | 1011010 | 90 | 5A | Z | 1111010 | 122 | 7A | z |
| 0111011 | 59 | 3B | : | 1011011 | 91 | 5B | [| 1111011 | 123 | 7B | { |
| 0111100 | 60 | 3C | < | 1011100 | 92 | 5C | \ | 1111100 | 124 | 7C | |
| 0111101 | 61 | 3D | = | 1011101 | 93 | 5D |] | 1111101 | 125 | 7D | } |
| 0111110 | 62 | 3E | > | 1011110 | 94 | 5E | ^ | 1111110 | 126 | 7E | N~ |
| 0111111 | 63 | 3F | ? | 1011111 | 95 | 5F | _ | | | | |

2. Look for the complete list of intel x86 family assembly commands (from books as well as the internet). This command list can be used as a guide to understanding the "boot.asm" and "karnel.asm" programs.

Answer:

1. ACALL (Absolute Call).

ACALL functions to call program sub routines.

2. ADD (Add Immediate Data).

ADD functions to add 8 bits of data directly into the contents of the accumulator and store the results in the accumulator.

3. ADDC (Add Carry Plus Immediate Data to Accumulator).

ADDC functions to add the contents of the carry flag (0 or 1) to the contents of the accumulator. 8 bit direct data is added to the accumulator.

4. AJMP (Absolute Jump)

AJMP is an absolute jump command. Jumps in 2 KB start at the address following the AJMP command. AJMP functions to transfer control of the program to a location where the address is calculated in the same way as the ACALL command. The program counter is added twice where the AJMP command is a 2-byte command. The program counter is loaded with a10 - a0 11 bits, to form a 16-bit destination address.

5. ANL (logical AND memory to accumulator)

ANL functions to contain the contents of the data address with the contents of the accumulator.

6. CJNE (Compare Indirect Address to Immediate Data)

CJNE functions to compare data directly with the memory location addressed by register R or Accumulator A. if not the same, the instruction will go to the code address.

Format: CJNE R, # data, address code.

7. CLR (Clear Accumulator)

CLR functions to reset the accumulator data to 00H.

Format: CLR A.

8. CPL (Complement Accumulator)

CPL serves to complement the contents of the accumulator.

9. DA (Decimal Adjust Accumulator)

DA functions to adjust the contents of the accumulator to BCD equivalents, after adding two BCD numbers.

10. DEC (Decrement Indirect Address)

DEC functions to reduce the contents of the memory location designated by register R by 1, and the results are stored at that location.

11. DIV (Divide Accumulator by B)

DIV functions to divide the contents of the accumulator with the contents of register B.

The accumulator contains the quotient, register B contains the remainder of the division.

12. DJNZ (Decrement Register And Jump Id Not Zero)

DJNZ functions to reduce the register value by 1 and if the result is 0 then the next instruction will be executed. If not 0, it will go to the address code.

13. INC (Increment Indirect Address)

INC functions to add memory contents by 1 and store it at that address.

14. JB (Jump if Bit is Set)

JB functions to read data by one bit, if the data is 1 then it will go to the code address and if it is 0 it will not go to the code address.

15. JBC (Jump if Bit Set and Clear Bit)

Bit JBC, functions as a test command rail specified by bit. If the bit is set, Jump is performed to the relative address and the specified bit in the command is cleared. The following program segment tests the least significant bit (LSB: Least Significant Byte), and if it is found that it is set, the program jumps to the READ location. JBC also functions to clean the LSB from the accumulator.

16. JC (Jump if Carry is Set)

The JC instruction is used to test the contents of the carry flag. If it contains 1, execution goes to the address code, if it contains 0, the next instruction will be executed.

17. JMP (Jump to sum of Accumulator and Data Pointer)

The JMP instruction is used to instruct the jump to a specific code address.

Format: JMP address code.

18. JNB (Jump if Bit is Not Set)

The JNB instruction is used to read data by one bit, if the data is 0 then it will go to the code address and if it is 1 it will not go to the code address.

Format: JNB address bits, address codes.

19. JNC (Jump if Carry Not Set)

JNC functions to test the Carry bit, and if it is not set, a jump will be made to the specified relative address.

20. JNZ (Jump if Accumulator Not Zero)

JNZ is the mnemonic for the jump if not zero instruction. In this case a jump will occur when the zero flag is "clear", and no jump will occur when the zero flag is set. Suppose that the JNZ 7800H is stored at location 2100H. If Z = 0, the next instruction will come from location 7800H: and when Z = 1, the program will descend to the next sequence instruction at location 2101H.

21. JZ (Jump if Accumulator is Zero)

JZ serves to test accumulator contents. If it is not zero, the jump is made to the relative address specified in the command.

22. LCALL (Long Call)

LCALL functions to allow calls to subroutines located anywhere in the 64K program memory. The LCALL operation goes like this:

- Adding to the program counter as many as 3, because the command is a 3-byte command.
- Add stack pointer by 1.
- Stores lower bytes of program counter on the stack.
- Adding a stack pointer.
- Stores a higher byte of the program on the stack.
- Load program counter with 16-bit destination address.

23. LJMP (Long Jump)

Long Jump functions to allow unconditional jump anywhere within the scope of program memory 64K. LCALL is a 3-byte command. The 16-bit destination address is specified directly in the command. This destination address is loaded into the program counter by the LJMP command.

24. MOV (Move From Memory)

MOV functions to move the contents of the accumulator / register or data from outside values or other addresses.

25. MOVC (Move From Codec Memory)

The MOVC instruction fills the accumulator with code bytes or constants from program memory. The byte address is the sum of the unsigned 8 bits in the accumulator and the 16-bit base register, which can be a data pointer or a program counter. This instruction does not affect any flags.