

LAPORAN PRAKTIKUM SISTEM OPERASI

KEGIATAN MODUL 1



DOSEN PENGAMPU

Heru Setya Nugraha, S.T., M.Kom.

DISUSUN OLEH

Nama : Asyam Daffa' Tsaqif

NIM : L200190227

Kelas : F

UNIVERSITAS MUHAMMADIYAH SURAKARTA

FAKULTAS KOMUNIKASI DAN INFORMATIKA

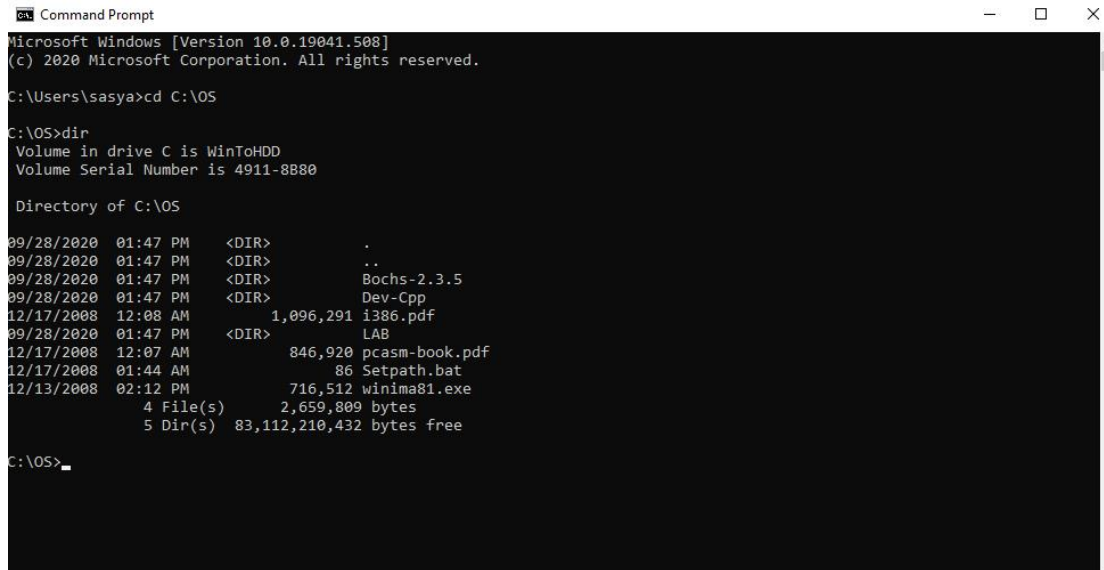
PRODI INFORMATIKA

2020 / 2021

Langkah Kerja

Menuju ke direktori kerja

1. Pertama buka Command Prompt
2. Masukkan perintah `cd C:\OS` untuk masuk ke direktori kerja `C:\OS`
3. Untuk melihat isi dari direktori masukan perintah `dir`. Seperti dibawah :



```
Command Prompt
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\sasya>cd C:\OS

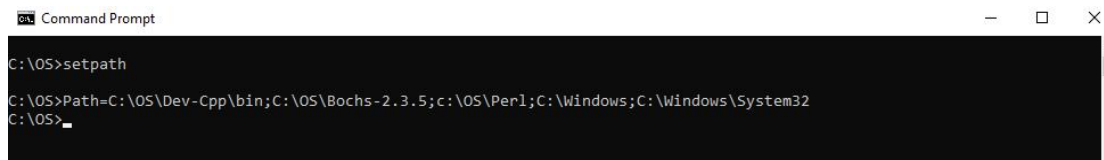
C:\OS>dir
Volume in drive C is WinToHDD
Volume Serial Number is 4911-8880

Directory of C:\OS

09/28/2020 01:47 PM <DIR>      .
09/28/2020 01:47 PM <DIR>      ..
09/28/2020 01:47 PM <DIR>      Bochs-2.3.5
09/28/2020 01:47 PM <DIR>      Dev-Cpp
12/17/2008 12:08 AM          1,096,291 i386.pdf
09/28/2020 01:47 PM <DIR>      LAB
12/17/2008 12:07 AM          846,920 pcasm-book.pdf
12/17/2008 01:44 AM             86 Setpath.bat
12/13/2008 02:12 PM          716,512 winima81.exe
               4 File(s)      2,659,809 bytes
               5 Dir(s)      83,112,210,432 bytes free

C:\OS>
```

3. Jalankan file setpath, untuk menjalankan ketik 'setpath' tekan <ENTER>

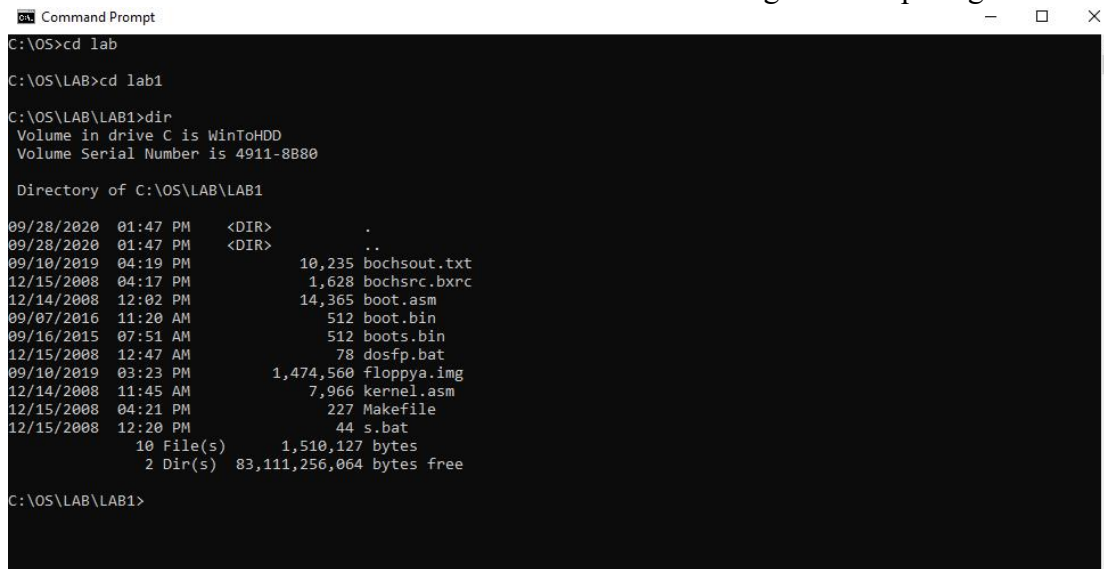


```
Command Prompt

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>
```

4. Masuk ke direktori kerja pada '`C:\OS\LAB\LAB1`' dengan cara `cd lab` lalu <ENTER> `cd lab1` <ENTER> dan buka isi direktori dengan `dir`. Seperti gambar dibawah :



```
Command Prompt

C:\OS>cd lab

C:\OS\LAB>cd lab1

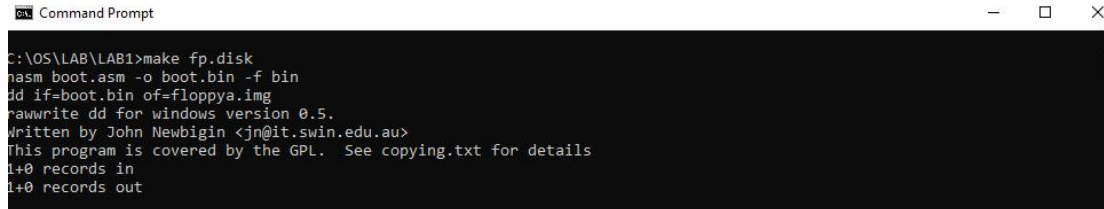
C:\OS\LAB\LAB1>dir
Volume in drive C is WinToHDD
Volume Serial Number is 4911-8880

Directory of C:\OS\LAB\LAB1

09/28/2020 01:47 PM <DIR>      .
09/28/2020 01:47 PM <DIR>      ..
09/10/2019 04:19 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
09/07/2016 11:20 AM           512 boot.bin
09/16/2015 07:51 AM           512 boots.bin
12/15/2008 12:47 AM             78 dosfp.bat
09/10/2019 03:23 PM       1,474,560 floppy.a.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)      83,111,256,064 bytes free

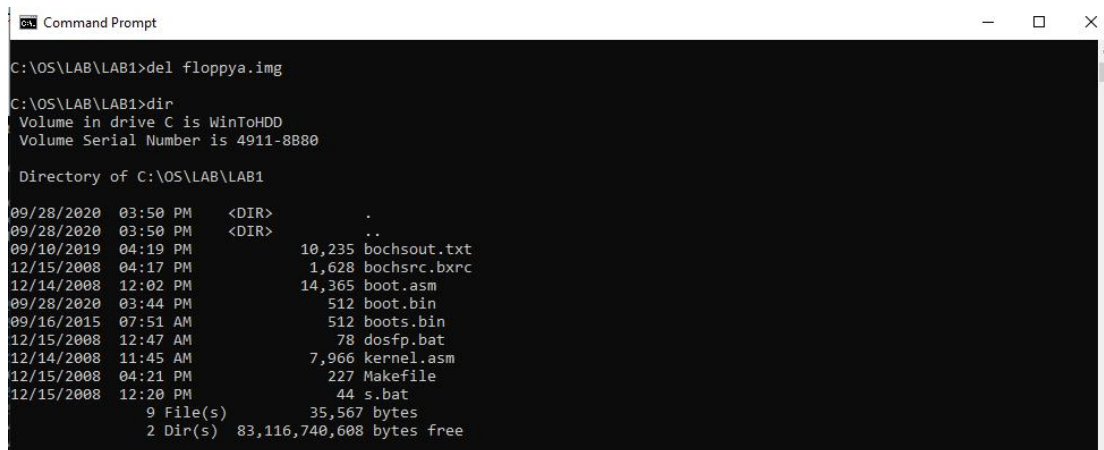
C:\OS\LAB\LAB1>
```

5. Ketik `make fp.disk` untuk melakukan kompilasi hadap source code program 'boot.asm' sebagai outputnya file 'boot.bin' dan isinya disalin ke dalam bootsector file image floppy.img. Seperti gambar dibawah ini :



```
Command Prompt
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 Newbiggin <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
```

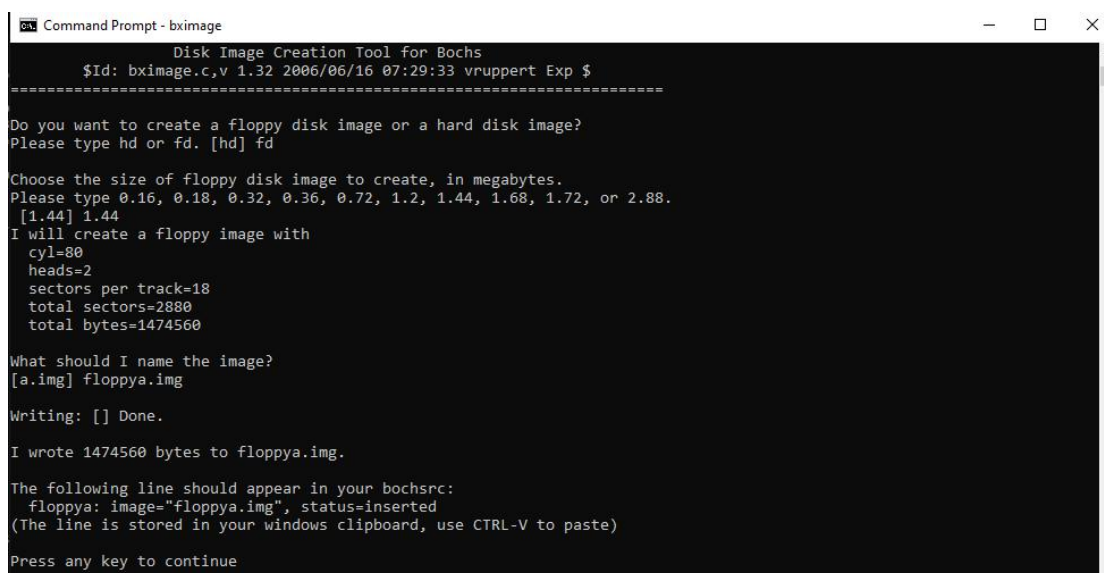
6. Hapuslah file 'floppya.img' jika sudah ada pada direktori kerja anda, dari 'Command Prompt' (lakukan dari direktori kerja) ketik `del floppya.img /P` lanjutkan dengan tekan Y dan <ENTER>. Pastikan bahwa file sudah benar benar terhapus dengan perintah 'dir'. Selanjutnya panggil 'bimage' sehingga ditampilkan window seperti pada gambar berikut :



```
Command Prompt
C:\OS\LAB\LAB1>del floppya.img
C:\OS\LAB\LAB1>dir
Volume in drive C is WinToHDD
Volume Serial Number is 4911-8B80

Directory of C:\OS\LAB\LAB1

09/28/2020  03:50 PM  <DIR>          .
09/28/2020  03:50 PM  <DIR>          ..
09/10/2019  04:19 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
09/28/2020  03:44 PM           512  boot.bin
09/16/2015  07:51 AM           512  boots.bin
12/15/2008  12:47 AM            78  dosfp.bat
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
               9 File(s)        35,567 bytes
               2 Dir(s)  83,116,740,608 bytes free
```



```
Command Prompt - bimage
Disk Image Creation Tool for Bochs
$Id: bimage.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] 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

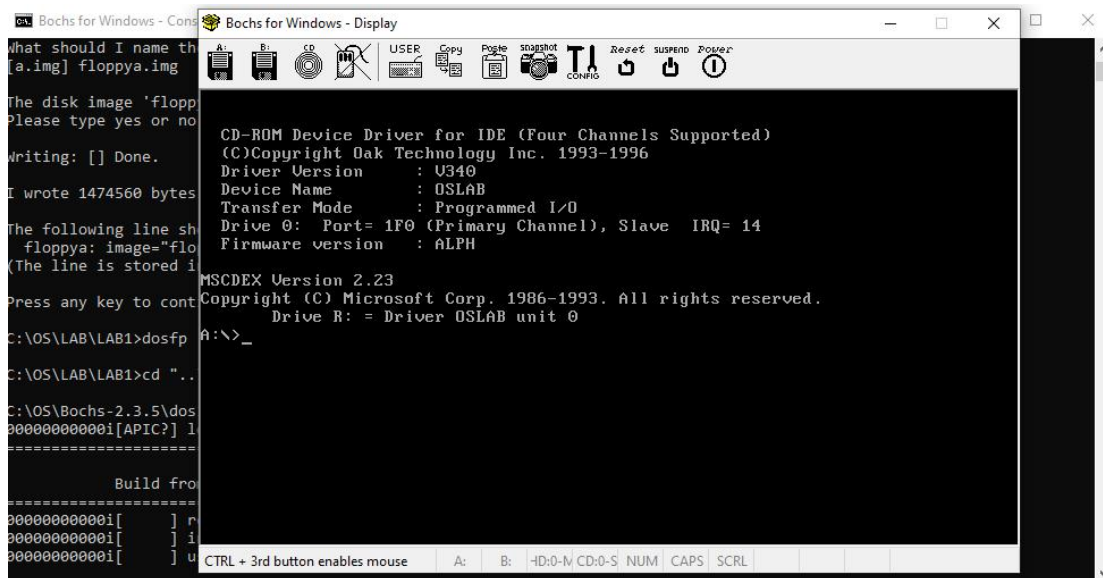
Writing: [ ] Done.

I wrote 1474560 bytes to floppya.img.

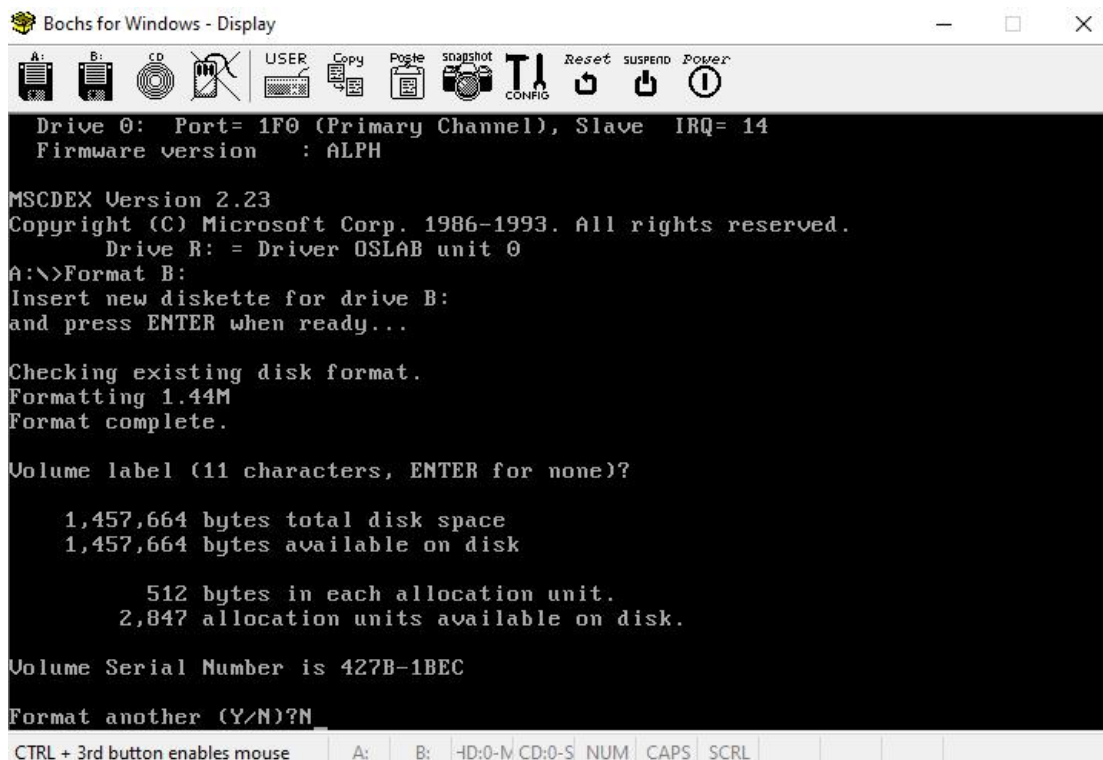
The following line should appear in your bochsrc:
  floppya: image="floppya.img", status=inserted
(The line is stored in your windows clipboard, use CTRL-V to paste)

Press any key to continue
```

7. Lakukan pemformatan pada file floppy.img dengan mengetik perintah 'dosfp' untuk membuka konfigurasi pc-simulator.



8. Ketikkan 'Format B:' setelah format selesai, klik enter. Lalu ketik 'n' dan tutup simulator dengan mengklik tombol power.



9. Ketik 'tdump boots.bin' lalu tekan enter maksud dari perintah ini adalah memindahkan data pada file 'boots.bin' ke dalam memory kerja 'tdump'. Seperti gambar dibawah :

```

Command Prompt

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

000000: EB 3C 90 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 BC FC 7B 16 07 BD 78 00 C5 76 00 1E 56 16 ...{...x..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 7D 20 8B C1 99 ...|.E..8N$} ...
000070: E8 7E 01 83 EB 3A 66 A1 1C 7C 66 3B 07 8A 57 FC ~...:f..|f;..W.
000080: 75 06 80 CA 02 88 56 02 80 C3 10 73 ED 33 C9 FE u....V....s.3..
000090: 06 D8 7D 8A 46 10 98 F7 66 16 03 46 1C 13 56 1E ..).F...f..F..V.
0000A0: 03 46 0E 13 D1 8B 76 11 60 89 46 FC 89 56 FE B8 .F...v...F..V..
0000B0: 20 00 F7 E6 8B 5E 08 03 C3 48 F7 F3 01 46 FC 11 ....^...H...F..
0000C0: 4E FE 61 BF 00 07 E8 28 01 72 3E 38 2D 74 17 60 N.a....(r>8-t.`
0000D0: B1 08 BE D8 7D F3 A6 61 74 3D 4E 74 09 83 C7 20 ....}.at=Nt...
0000E0: 3B FB 72 E7 EB DD FE 0E D8 7D 7B A7 BE 7F 7D AC ;.r.....){...}.
0000F0: 98 03 F0 AC 98 40 74 0C 48 74 13 B4 0E BB 07 00 .....@t.Ht.....
000100: CD 10 EB EF BE 82 7D EB E6 BE 80 7D EB E1 CD 16 .....}.....}....
000110: 5E 1F 66 8F 04 CD 19 BE 81 7D 8B 7D 1A 8D 45 FE ^.f.....}.E..
000120: 8A 4E 0D F7 E1 03 46 FC 13 56 FE B1 04 E8 C2 00 .N....F..V.....
000130: 72 D7 EA 00 02 70 00 52 50 06 53 6A 01 6A 10 91 r....p.RP.Sj.j..
000140: 8B 46 18 A2 26 05 96 92 33 D2 F7 F6 01 F7 F6 42 .F.&...3.....B
000150: 87 CA F7 76 1A 8A F2 8A E8 C0 CC 02 0A CC B8 01 ...v.....}....
000160: 02 80 7E 02 0E 75 04 B4 42 8B F4 8A 56 24 CD 13 ..~..u..B...V$..
000170: 61 61 72 0A 40 75 01 42 03 5E 08 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 6B FF 0D 0A 44 69 73 6B 20 tem disk...Disk
0001A0: 49 2F 4F 20 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 6B 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 6B 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 SYSMSDOS SYS..
0001F0: 00 41 B8 00 07 60 66 6A 00 E9 3B FF 00 00 55 AA .A...`fj.;...U.

```

10. Masukkan perintah dengan mengetik ‘s’ untuk membuka simulator.

```

Bochs for Windows - Display

Plex86/Bochs UGABios 0.6a 19 Aug 2006
This UGA/UBE Bios is released under the GNU LGPL

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

Bochs UBE Display Adapter enabled

Bochs BIOS - build: 09/10/07
$Revision: 1.183 $ $Date: 2007/09/10 20:00:29 $
Options: apmbios pcibios eltorito rombios32

Booting from Floppy...

Invalid system disk
Replace the disk, and then press any key

CTRL + 3rd button enables mouse  A: NUM CAPS SCRL

```

11. Boot invalid karena floppy.img belum diisi system file, langkah berikutnya yaitu mengetik perintah ‘del floppya.img’, membuat img baru dengna perintah ‘bximage’. Lalu pilih type fd dan ukuran 1.44 MB dan dinamai dengan floppya.img.


```
Command Prompt
$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] 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] floppy.img

Writing: [] Done.

I wrote 1474560 bytes to floppy.img.

The following line should appear in your bochsrc:
  floppy: image="floppy.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>
```

12. Buka simulator dengan mengetik 'dosfp', lalu masukkan perintah 'Format B:/S', kemudian tekan enter untuk lanjut.

```
Bochs for Windows - Console
total sectors=2880
total bytes=1474560

What should I name the i
[a.img] floppy.img

Writing: [] Done.

I wrote 1474560 bytes to

The following line shoul
  floppy: image="floppy
(The line is stored in y
Press any key to continu

C:\OS\LAB\LAB1>dosfp

C:\OS\LAB\LAB1>cd "..\..

C:\OS\Bochs-2.3.5\dos>..
00000000000i[APIC?] loca
=====
Build from C
=====
00000000000i[ ] read
00000000000i[ ] inst
00000000000i[ ] usin

Bochs for Windows - Display
MSCDEx Version 2.23
Copyright (C) Microsoft Corp. 1986-1993. All rights reserved.
Drive B: = Driver OSLAB unit 0
A:\>Format B:/S
Insert new diskette for drive B:
and press ENTER when ready...
Checking existing disk format.
Formatting 1.44M
Format complete.
System transferred
Volume label (11 characters, ENTER for none)?

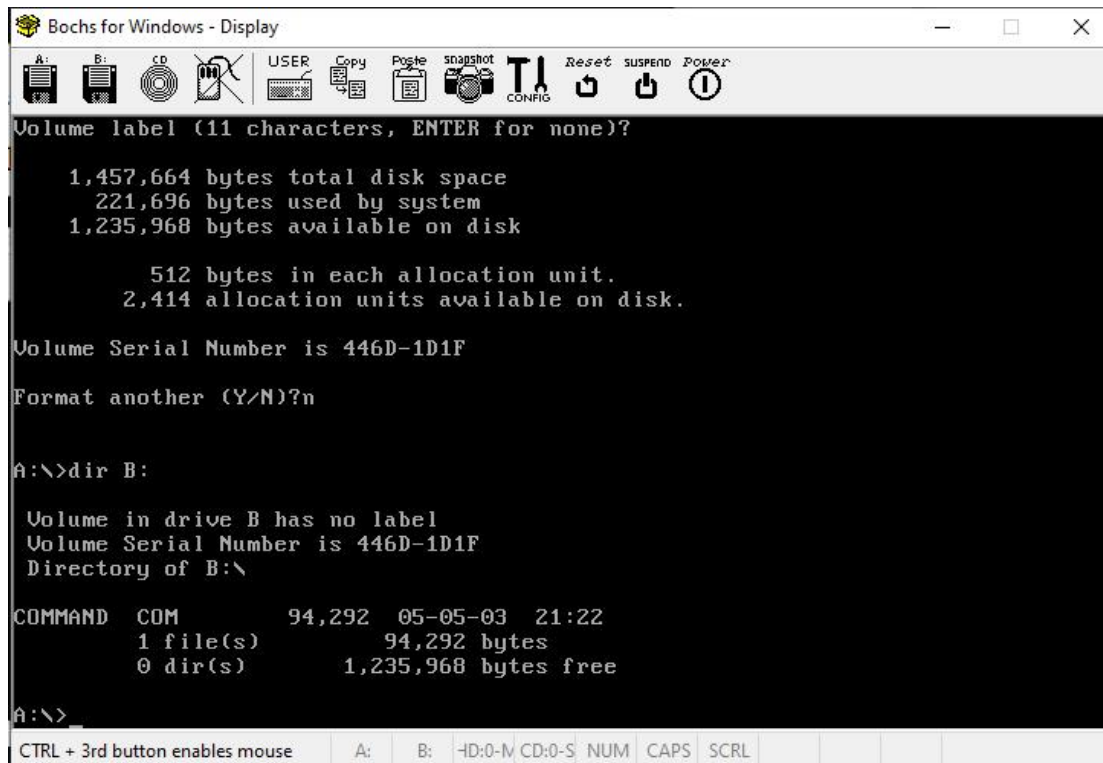
1,457,664 bytes total disk space
221,696 bytes used by system
1,235,968 bytes available on disk

512 bytes in each allocation unit.
2,414 allocation units available on disk.

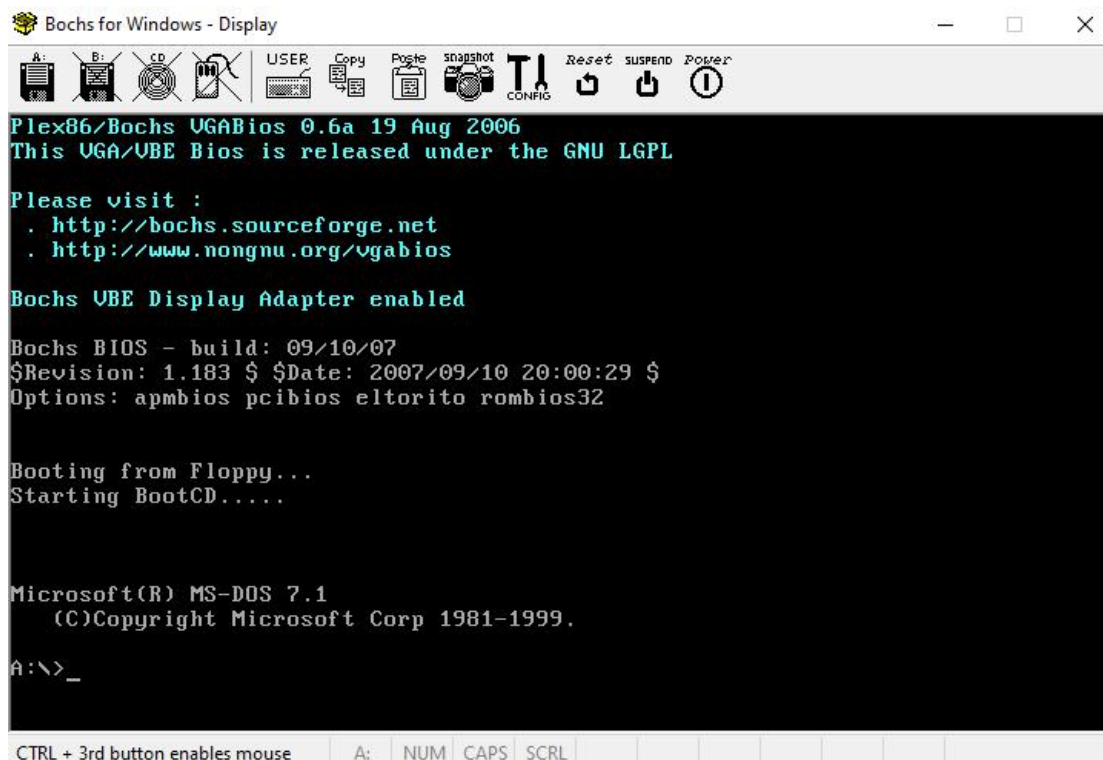
Volume Serial Number is 2742-1B16
Format another (Y/N)?

CTRL + 3rd button enables mouse  A:  B:  -HD:0-IV CD:0-S NUM CAPS SCRL
```

13. Format lagi dengan mengetik 'y', lalu tekan enter untuk lanjut. Jika format sudah selesai ketik 'n', kemudian ketik 'dir B:' untuk mengecek bahwa floppy.img telah terisi system file, lalu tutup simulator dengan klik tombol power.



14. Mencoba menggunakan floppy.img sebagai boot.disk dengan perintah 's'.



'starting BootCD' artinya boot.disk yang pertama dibuat menggunakan sistem dari CD, tapi proses boot yang sebenarnya dari floppy.img.

TUGAS

1. Apa yang dimaksud dengan kode ASCII, buatlah tabel kode ASCII lengkap cukup kode ASCII yang standar tidak perlu extend, tuliskan kode ASCII dalam format angka decimal, binary dan hexadecimal serta karakter dan symbol yang dikodekan.

ASCII merupakan kepanjangan dari (*American Standard Code for Information Interchange*), dan pengertian dari ASCII sendiri adalah suatu standar internasional dalam kode huruf dan simbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|". Ia selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks.

Kode ini merepresentasikan angka, huruf serta tombol standar, Enter, Escape, Backspace dan Space.

| Nilai ANSI ASCII (Desimal) | Nilai Unicode (Heksa Desimal) | Karakter | Keterangan | Binary |
|----------------------------------|----------------------------------|----------|-----------------------|---------|
| 13 | 00D | CR | Enter/CR | 0001101 |
| 16 | 0010 | DLE | Escape | 0010000 |
| 8 | 0008 | BS | Backspace | 0001000 |
| 32 | 0020 | spasi | Spasi | 0100000 |
| 48 | 0030 | 0 | Angka nol | 0110000 |
| 49 | 0031 | 1 | Angka satu | 0110001 |
| 50 | 0032 | 2 | Angka dua | 0110010 |
| 51 | 0033 | 3 | Angka tiga | 0110011 |
| 52 | 0034 | 4 | Angka empat | 0110100 |
| 53 | 0035 | 5 | Angka lima | 0110101 |
| 54 | 0036 | 6 | Angka enam | 0110110 |
| 55 | 0037 | 7 | Angka tujuh | 0110111 |
| 56 | 0038 | 8 | Angka delapan | 0111000 |
| 57 | 0039 | 9 | Angka sembilan | 0111001 |
| 127 | 007F | DEL | Delete | 1111111 |
| 66 | 0042 | B | Huruf latin B kapital | 1000010 |
| 67 | 0043 | C | Huruf latin C kapital | 1000011 |
| 68 | 0044 | D | Huruf latin D kapital | 1000100 |
| 69 | 0045 | E | Huruf latin E kapital | 1000101 |
| 70 | 0046 | F | Huruf latin F kapital | 1000110 |
| 71 | 0047 | G | Huruf latin G kapital | 1000111 |
| 72 | 0048 | H | Huruf latin H kapital | 1001000 |
| 73 | 0049 | I | Huruf latin I kapital | 1001001 |
| 74 | 004A | J | Huruf latin J kapital | 1001010 |

| | | | | |
|-----|------|---|-----------------------|---------|
| 75 | 004B | K | Huruf latin K kapital | 1001011 |
| 76 | 004C | L | Huruf latin L kapital | 1001100 |
| 77 | 004D | M | Huruf latin M kapital | 1001101 |
| 78 | 004E | N | Huruf latin N kapital | 1001110 |
| 79 | 004F | O | Huruf latin O kapital | 1001111 |
| 80 | 0050 | P | Huruf latin P kapital | 1010000 |
| 81 | 0051 | Q | Huruf latin Q kapital | 1010001 |
| 82 | 0052 | R | Huruf latin R kapital | 1010010 |
| 83 | 0053 | S | Huruf latin S kapital | 1010011 |
| 84 | 0054 | T | Huruf latin T kapital | 1010100 |
| 85 | 0055 | U | Huruf latin U kapital | 1010101 |
| 86 | 0056 | V | Huruf latin V kapital | 1010110 |
| 87 | 0057 | W | Huruf latin W kapital | 1010111 |
| 88 | 0058 | X | Huruf latin X kapital | 1011000 |
| 89 | 0059 | Y | Huruf latin Y kapital | 1011001 |
| 90 | 005A | Z | Huruf latin Z kapital | 1011010 |
| 97 | 0061 | a | Huruf latin a kecil | 1100001 |
| 98 | 0062 | b | Huruf latin b kecil | 1100010 |
| 99 | 0063 | c | Huruf latin c kecil | 1100011 |
| 100 | 0064 | d | Huruf latin d kecil | 1100100 |
| 101 | 0065 | e | Huruf latin e kecil | 1100101 |
| 102 | 0066 | f | Huruf latin f kecil | 1100110 |
| 103 | 0067 | g | Huruf latin g kecil | 1100111 |
| 104 | 0068 | h | Huruf latin h kecil | 1101000 |
| 105 | 0069 | i | Huruf latin i kecil | 1101001 |
| 106 | 006A | j | Huruf latin j kecil | 1101010 |
| 107 | 006B | k | Huruf latin k kecil | 1101011 |
| 108 | 006C | l | Huruf latin l kecil | 1101100 |
| 109 | 006D | m | Huruf latin m kecil | 1101101 |
| 110 | 006E | n | Huruf latin n kecil | 1101110 |
| 111 | 006F | o | Huruf latin o kecil | 1101111 |
| 112 | 0070 | p | Huruf latin p kecil | 1110000 |
| 113 | 0071 | q | Huruf latin q kecil | 1110001 |
| 114 | 0072 | r | Huruf latin r kecil | 1110010 |
| 115 | 0073 | s | Huruf latin s kecil | 1110011 |
| 116 | 0074 | t | Huruf latin t kecil | 1110100 |
| 117 | 0075 | u | Huruf latin u kecil | 1110101 |
| 118 | 0076 | v | Huruf latin v kecil | 1110110 |
| 119 | 0077 | w | Huruf latin w kecil | 1110111 |
| 120 | 0078 | x | Huruf latin x kecil | 1111000 |
| 121 | 0079 | y | Huruf latin y kecil | 1111001 |
| 122 | 007A | z | Huruf latin z kecil | 1111010 |

2. Carilah daftar perintah bahasa assembly untuk mesin intelkeluarga x86 lengkap (dari buku refrensi atau internet). Daftar perintah ini dapat digunakan sebagai pedoman untuk memahami program ‘boot.asm’ dan ‘kernel.asm’.

Tabel Mnemonik Perintah Assembly AT89S51

| No | PERINTA | MNEMONIK |
|-----------|--|-----------------|
| 1. | ADD | ADD |
| 2. | ADD WITH CARRY | ADC |
| 3. | SUB WITH BORROW | SBB |
| 4. | INCREMENT | INC |
| 5. | DECREMENT | DEC |
| 6. | MULTIPLY | MUL |
| 7. | DEVIDE | DIV |
| 8. | AND LOGIC | ANL |
| 9. | OR LOGIC | ORL |
| 10. | EXLUSIVE OR LOGIK | XRL |
| 11. | DECIMAL ADJUST ACCUMULATOR | DAA |
| 12. | CLEAR ACCUMULATOR | CLR A |
| 13. | COMPLEMENT ACCUMULATOR | CPL A |
| 14. | ROTATE ACCUMULATOR LEFT | RLA |
| 15. | ROTATE ACCUMULATOR LEFT THROUGH | RLCA |
| 16. | ROTATE ACCUMULATOR RIGHT | RRA |
| 17. | ROTATE ACCUMULATOR RIGHT THROUGH CARRY | RRCA |
| 18. | SWAPP NIBBLE WITHIN ACCUMULATOR | SWAP |
| 19. | PUSH DIRECT BYTE KE STACK | PUSH |
| 20. | POP DIRECT BYTE DARI STACK | POP |
| 21. | JUMP IF CARRY SET C=1 | JC |
| 22. | JUMP IF CARRY NOT SET C = 0 | JNC |
| 23. | JUMP IF DIRECT BIT SET | JB |
| 24. | JUMP IF DIRECT BIT NOT SET | JNB |
| 25. | JUMP IF DIRECT BIT SET & CLEAR BIT | JBC |
| 26. | ABSOLUTE CALL | ACAL |
| 27. | LONG CALL | LCALL |
| 28. | RETURN | RET |
| 29. | RETURN FROM INTERRUPT | RETI |
| 30. | ABSOLUTE JUMP | AJMP |
| 31. | LONG JUMP | LJMP |
| 32. | SHORT JUMP | SJMP |
| 33. | JUMP INDIRECT | JMP |
| 34. | JUMP IF ACCUMULATOR ZERRO | JZ |
| 35. | JUMP IF ACCUMULATOT NOT ZERRO | JNZ |
| 36. | COMPARE AND JUMP IF NOT EQUAL | CJNE |
| 37. | DECREAMENT AND JUMP IF NOT ZERO | DJNZ |
| 38. | NO OPERATION | NOP |

1. ANL (logical AND memori ke akumulator)

ANL berfungsi untuk mengAND-kan isi alamat data dengan isi akumulator.

2. ADD (Add Immediate Data)

ADD berfungsi untuk menambah 8 bit data langsung ke dalam isi akumulator dan menyimpan hasilnya pada akumulator.

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

ADDC berfungsi untuk menambahkan isi carry flag (0 atau 1) ke dalam isi akumulator. Data langsung 8 bit ditambahkan ke akumulator.

4. AJMP (Absolute Jump)

AJMP adalah perintah jump mutlak. Jump dalam 2 KB dimulai dari alamat yang mengikuti perintah AJMP. AJMP berfungsi untuk mentransfer kendali program ke lokasi dimana alamat dikalkulasi dengan cara yang sama dengan perintah ACALL. Konter program ditambahkan dua kali dimana perintah AJMP adalah perintah 2-byte. Konter program di-load dengan a10 – a0 11 bits, untuk membentuk alamat tujuan 16-bit.

5. ACALL (Absolute Call)

ACALL berfungsi untuk memanggil sub rutin program

6. CJNE (Compare Indirect Address to Immediate Data)

CJNE berfungsi untuk membandingkan data langsung dengan lokasi memori yang dialamati oleh register R atau Akumulator A. apabila tidak sama maka instruksi akan menuju ke alamat kode.

Format : CJNE R,#data,Alamat kode.

7. CLR (Clear Accumulator)

CLR berfungsi untuk mereset data akumulator menjadi 00H.

Format : CLR A

8. CPL (Complement Accumulator)

CPL berfungsi untuk mengkomplemen isi akumulator.

9. DA (Decimal Adjust Accumulator)

DA berfungsi untuk mengatur isi akumulator ke padanan BCD, steleah penambahan dua angka BCD.

10. DEC (Decrement Indirect Address)

DEC berfungsi untuk mengurangi isi lokasi memori yang ditujukan oleh register R dengan 1, dan hasilnya disimpan pada lokasi tersebut.

11. DIV (Divide Accumulator by B)

DIV berfungsi untuk membagi isi akumulator dengan isi register B. Akumulator berisi hasil bagi, register B berisi sisa pembagian.

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

DJNZ berfungsi untuk mengurangi nilai register dengan 1 dan jika hasilnya sudah 0 maka instruksi selanjutnya akan dieksekusi. Jika belum 0 akan menuju ke alamat kode.

13. INC (Increment Indirect Address)

INC berfungsi untuk menambahkan isi memori dengan 1 dan menyimpannya pada alamat tersebut.

14. JB (Jump if Bit is Set)

JB berfungsi untuk membaca data per satu bit, jika data tersebut adalah 1 maka akan menuju ke alamat kode dan jika 0 tidak akan menuju ke alamat kode.

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

Bit JBC, berfungsi sebagai perintah rel menguji yang terspesifikasikan secara bit. Jika bit di-set, maka Jump dilakukan ke alamat relatif dan yang terspesifikasi secara bit di dalam perintah dibersihkan. Segmen program berikut menguji bit yang kurang signifikan (LSB: Least Significant Byte), dan jika ditemukan bahwa ia telah di-set, program melompat ke READ lokasi. JBC juga berfungsi membersihkan LSB dari akumulator.

16. JC (Jump if Carry is Set)

Instruksi JC berfungsi untuk menguji isi carry flag. Jika berisi 1, eksekusi menuju ke alamat kode, jika berisi 0, instruksi selanjutnya yang akan dieksekusi.

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

Instruksi JMP berfungsi untuk memerintahkan loncat kesuatu alamat kode tertentu.

Format : JMP alamat kode.

18. JNB (Jump if Bit is Not Set)

Instruksi JNB berfungsi untuk membaca data per satu bit, jika data tersebut adalah 0 maka akan menuju ke alamat kode dan jika 1 tidak akan menuju ke alamat kode.

Format : JNB alamat bit, alamat kode.

19. JNC (Jump if Carry Not Set)

JNC berfungsi untuk menguji bit Carry, dan jika tidak di-set, maka sebuah lompatan akan dilakukan ke alamat relatif yang telah ditentukan.

20. JNZ (Jump if Accumulator Not Zero)

JNZ adalah mnemonik untuk instruksi jump if not zero (lompat jika tidak nol). Dalam hal ini suatu lompatan akan terjadi bilamana bendera nol dalam keadaan “clear”, dan tidak akan terjadi lompatan bilamana bendera nol tersebut dalam keadaan set. Andaikan bahwa JNZ 7800H disimpan pada lokasi 2100H. Jika Z=0, instruksi berikutnya akan berasal dari lokasi 7800H: dan bilamana Z=1, program akan turun ke instruksi urutan berikutnya pada lokasi 2101H.

21. JZ (Jump if Accumulator is Zero)

JZ berfungsi untuk menguji konten-konten akumulator. Jika bukan nol, maka lompatan dilakukan ke alamat relatif yang ditentukan dalam perintah.

22. LCALL (Long Call)

LCALL berfungsi untuk memungkinkan panggilan ke subrutin yang berlokasi dimanapun dalam memori program 64K. Operasi LCALL berjalan seperti berikut:

- Menambahkan ke dalam konter program sebanyak 3, karena perintahnya adalah perintah 3-byte.
- Menambahkan penunjuk stack sebanyak 1.
- Menyimpan byte yang lebih rendah dari konter program ke dalam stack.
- Menambahkan penunjuk stack.
- Menyimpan byte yang lebih tinggi dari program ke dalam stack.
- Me-load konter program dengan alamat tujuan 16-bit.

23. . LJMP (Long Jump)

Long Jump berfungsi untuk memungkinkan lompatan tak bersyarat kemana saja dalam lingkup ruang memori program 64K. LCALL adalah perintah 3-byte. Alamat tujuan 16-bit ditentukan secara langsung dalam perintah tersebut. Alamat tujuan ini di-load ke dalam konter program oleh perintah LJMP.

24. MOV (Move From Memory)

MOV berfungsi untuk memindahkan isi akumulator/register atau data dari nilai luar atau alamat lain.

25. MOVC (Move From Codec Memory)

Instruksi MOVC berfungsi untuk mengisi accumulator dengan byte kode atau konstanta dari program memory. Alamat byte tersebut adalah hasil penjumlahan unsigned 8 bit pada accumulator dan 16 bit register basis yang dapat berupa data pointer atau program counter. Instruksi ini tidak mempengaruhi flag apapun juga.

26. MOVX (Move Accumulator to External Memory Addressed by Data Pointer)

MOVX berfungsi untuk memindahkan isi akumulator ke memori data eksternal yang alamatnya ditunjukkan oleh isi data pointer.

27. MUL (Multiply)

MUL AB berfungsi untuk mengalikan unsigned 8 bit integer pada accumulator dan register B. Byte rendah (low order) dari hasil perkalian akan disimpan dalam accumulator sedangkan byte tinggi (high order) akan disimpan dalam register B. Jika hasil perkalian lebih besar dari 255 (0FFh), overflow flag akan bernilai '1'. Jika hasil perkalian lebih kecil atau sama dengan 255, overflow flag akan bernilai '0'. Carry flag akan selalu dikosongkan.

28. NOP (No Operation)

Fungsi NOP adalah eksekusi program akan dilanjutkan ke instruksi berikutnya. Selain PC, instruksi ini tidak mempengaruhi register atau flag apapun juga.

29. ORL (Logical OR Immediate Data to Accumulator)

Instruksi ORL berfungsi sebagai instruksi Gerbang logika OR yang akan menjumlahkan Accumulator terhadap nilai yang ditentukan.

Format : ORL A,#data.

30. POP (Pop Stack to Memory)

Instruksi POP berfungsi untuk menempatkan byte yang ditunjukkan oleh stack pointer ke suatu alamat data.

31. PUSH (Push Memory onto Stack)

Instruksi PUSH berfungsi untuk menaikkan stack pointer kemudian menyimpan isinya ke suatu alamat data pada lokasi yang ditunjuk oleh stack pointer.

32. RET (Return from subroutine)

Intruksi RET berfungsi untuk kembali dari suatu subrutin program ke alamat terakhir subrutin tersebut di panggil.

33. RETI (Return From Interrupt)

RETI berfungsi untuk mengambil nilai byte tinggi dan rendah dari PC dari stack dan mengembalikan kondisi logika interrupt agar dapat menerima interrupt lain dengan prioritas yang sama dengan prioritas interrupt yang baru saja diproses. Stack pointer akan dikurangi dengan 2. Instruksi ini tidak mempengaruhi flag apapun juga. Nilai PSW tidak akan dikembalikan secara otomatis ke kondisi sebelum interrupt. Eksekusi program akan dilanjutkan pada alamat yang diambil tersebut. Umumnya alamat tersebut adalah alamat setelah lokasi dimana terjadi interrupt. Jika interrupt dengan prioritas sama atau lebih rendah tertunda saat RETI dieksekusi, maka satu instruksi lagi akan dieksekusi sebelum interrupt yang tertunda tersebut diproses.

34. RL (Rotate Accumulator Left)

Instruksi RL berfungsi untuk memutar setiap bit dalam akumulatur satu posisi ke kiri.

35. . RLC (Rotate Left through Carry)

Fungsi : Memutar (Rotate) Accumulator ke Kiri (Left) Melalui Carry Flag. Kedelapan bit accumulator dan carry flag akan diputar satu bit ke kiri secara bersama-sama. Bit 7 akan dirotasi ke carry flag, nilai carry flag akan berpindah ke posisi bit 0. Instruksi ini tidak mempengaruhi flag lain

36. RR (Rotate Right)

Fungsi : Memutar (Rotate) Accumulator ke Kanan (Right). Kedelapan bit accumulator akan diputar satu bit ke kanan. Bit 0 akan dirotasi ke posisi bit 7. Instruksi ini tidak mempengaruhi flag apapun juga.

37. RRC (Rotate Right through Carry)

Fungsi : Memutar (Rotate) Accumulator ke Kanan (Right) Melalui Carry Flag. Kedelapan bit accumulator dan carry flag akan diputar satu bit ke kanan secara bersama-sama. Bit 0 akan dirotasi ke carry flag, nilai carry flag akan berpindah ke posisi bit 7. Instruksi ini tidak mempengaruhi flag lain.

38. SETB (set Carry flag)

Instruksi SETB berfungsi untuk menset carry flag.

39. SJMP (Short Jump)

Sebuah Short Jump berfungsi untuk mentransfer kendali ke alamat tujuan dalam 127 bytes yang mengikuti dan 128 yang mengawali perintah SJMP. Alamat tujuannya ditentukan sebagai sebuah alamat relative 8-bit. Ini adalah Jump tidak bersyarat. Perintah SJMP menambahkan konter program sebanyak 2 dan menambahkan alamat relatif ke dalamnya untuk mendapatkan alamat tujuan. Alamat relatif tersebut ditentukan dalam perintah sebagai 'SJMP rel'.

40. SUBB (Subtract With Borrow)

Fungsi : Pengurangan (Subtract) dengan Peminjaman (Borrow). SUBB mengurangi variabel yang tertera pada operand kedua dan carry flag sekaligus dari accumulator dan menyimpan hasilnya pada accumulator. SUBB akan memberi nilai '1' pada carry flag jika peminjaman ke bit 7 dibutuhkan dan mengosongkan C jika tidak dibutuhkan peminjaman. Jika C bernilai '1' sebelum mengeksekusi SUBB, hal ini menandakan bahwa terjadi peminjaman pada proses pengurangan sebelumnya, sehingga carry flag dan source byte akan dikurangkan dari accumulator secara bersama-sama. AC akan bernilai '1' jika peminjaman ke bit 3 dibutuhkan dan mengosongkan AC jika tidak dibutuhkan peminjaman. OV akan bernilai '1' jika ada peminjaman ke bit 6 namun tidak ke bit 7 atau ada peminjaman ke bit 7 namun tidak ke bit 6. Saat mengurangi signed integer, OV menandakan adanya angka negative sebagai hasil dari pengurangan angka negatif dari angka positif atau adanya angka positif sebagai hasil dari pengurangan angka positif dari angka negative. Addressing mode yang dapat digunakan adalah: register, direct, register indirect, atau immediate data.

41. SWAP (Swap Nibbles)

Fungsi : Menukar (Swap) Upper Nibble dan Lower Nibble dalam Accumulator. SWAP A akan menukar nibble (4 bit) tinggi dan nibble rendah dalam accumulator. Operasi ini dapat dianggap sebagai rotasi 4 bit dengan RR atau RL. Instruksi ini tidak mempengaruhi flag apapun juga.

42. XCH (Exchange Bytes)

Fungsi : Menukar (Exchange) Accumulator dengan Variabel Byte. XCH akan mengisi accumulator dengan variabel yang tertera pada operand kedua dan pada saat yang sama juga akan mengisikan nilai accumulator ke dalam variabel tersebut. Addressing mode yang dapat digunakan adalah: register, direct, atau register indirect.

43. XCHD (Exchange Digits)

Fungsi : Menukar (Exchange) Digit. XCHD menukar nibble rendah dari accumulator, yang umumnya mewakili angka heksadesimal atau BCD, dengan nibble rendah dari internal data memory yang diakses secara indirect. Nibble tinggi kedua register tidak akan terpengaruh. Instruksi ini tidak mempengaruhi flag apapun juga.

44. XRL (Exclusive OR Logic)

Fungsi : Logika Exclusive OR untuk Variabel Byte XRL akan melakukan operasi bitwise logika exclusive OR antara kedua variabel yang dinyatakan. Hasilnya akan disimpan pada destination byte. Instruksi ini tidak mempengaruhi flag apapun juga. Kedua operand mampu menggunakan enam kombinasi addressing mode. Saat destination byte adalah accumulator, source byte dapat berupa register, direct, register indirect, atau immediate data. Saat destination byte berupa direct address, source byte dapat berupa accumulator atau immediate data.