

LAPORAN KEGIATAN PRAKTIKUM SISTEM OPERASI



Modul 1

Pengenalan Sistem Pengembangan OS dengan PC Simulator 'Bochs'

Nama : Lencia Putri Septa Riani

NIM : L200190267

Kelas : G

- Untuk menuju ke direktori kerja C:\OS, masukkan perintah 'cd os' lalu tekan <ENTER>
- Lalu masukkan perintah dir, maka akan muncul tampilan seperti dibawah

```

Command Prompt
Microsoft Windows [Version 10.0.19041.508]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\moogystore>cd c:\os

c:\OS>dir
Volume in drive C has no label.
Volume Serial Number is 5A22-F524

Directory of c:\OS

22/09/2020 12:51 <DIR>      .
22/09/2020 12:51 <DIR>      ..
22/09/2020 12:51 <DIR>      Bochs-2.3.5
22/09/2020 12:51 <DIR>      Dev-Cpp
17/12/2008 00:08           1.096.291 i386.pdf
22/09/2020 12:51 <DIR>      LAB
17/12/2008 00:07           846.920 pcasm-book.pdf
17/12/2008 01:44            86 Setpath.bat
13/12/2008 14:12           716.512 winima81.exe
               4 File(s)      2.659.809 bytes
               5 Dir(s)    139.716.841.472 bytes free

c:\OS>

```

- Jalankan file setpath dengan mengetik 'setpath' lalu tekan <ENTER>

```

Command Prompt
Microsoft Windows [Version 10.0.19041.508]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\moogystore>cd c:\os

c:\OS>dir
Volume in drive C has no label.
Volume Serial Number is 5A22-F524

Directory of c:\OS

22/09/2020 12:51 <DIR>      .
22/09/2020 12:51 <DIR>      ..
22/09/2020 12:51 <DIR>      Bochs-2.3.5
22/09/2020 12:51 <DIR>      Dev-Cpp
17/12/2008 00:08           1.096.291 i386.pdf
22/09/2020 12:51 <DIR>      LAB
17/12/2008 00:07           846.920 pcasm-book.pdf
17/12/2008 01:44            86 Setpath.bat
13/12/2008 14:12           716.512 winima81.exe
               4 File(s)      2.659.809 bytes
               5 Dir(s)    139.716.841.472 bytes free

c:\OS>setpath

c:\OS>Path=C:\OS\Dev-Cpp\bin;C:\OS\Bochs-2.3.5;c:\OS\Perl;C:\Windows;C:\Windows\Syste
m32
c:\OS>

```

Melihat isi direktori kerja

- Cobalah untuk membuka file tersebut dengan perintah berikut, dari 'COMMAND PROMPT', ketikkan 'Notepad boot.asm' dan tekan <ENTER>. Lalu tutup kembali program notepadnya.

The screenshot shows a Windows Command Prompt window with the following commands and output:

```

C:\Users\moogystore>cd c:\os
c:\OS>dir
Volume in drive C has no label
Volume Serial Number is 5A22

Directory of c:\OS

22/09/2020 12:51 <DIR>
22/09/2020 12:51 <DIR>
22/09/2020 12:51 <DIR>
22/09/2020 12:51 <DIR>
17/12/2008 00:08 1.0
22/09/2020 12:51 <DIR>
17/12/2008 00:07 8
17/12/2008 01:44 7
13/12/2008 14:12 5 Dir(s) 139.

c:\OS>setpath
c:\OS>Path=C:\OS\Dev-Cpp\bin;
m32
c:\OS>cd lab
c:\OS\LAB>cd lab1
c:\OS\LAB\LAB1>Notepad boot.asm
c:\OS\LAB\LAB1>

```

The Notepad window displays the content of boot.asm:

```

; *****
; LAB-1 : boot-strap loader - real mode
; untuk memindahkan file OS dari floppy disk format DOS
; *****
; atur mode kerja 16 bit (real-mode)
[BITS 16]
; Menentukan lokasi awal dari program
[ORG 0x0000]
; loncat ke label START
jmp START
; Keterangan format floppy disk format FAT12
OEM_ID db "QUASI-OS"
BytesPerSector dw 0x0200
SectorsPerCluster db 0x01

```

Sekarang bukalah file 'Makefile', dari 'Command Prompt' untuk mengetahui script makefile dengan cara:

- bukalah direktori kerja anda 'C:\OS\LAB\LAB1' selanjutnya ketik 'Notepad M' tekan tombol 'TAB' sehingga muncul 'Notepad Makefile' dan tekan <ENTER>.

The screenshot shows a Windows Command Prompt window with the following commands and output:

```

C:\Users\moogystore>cd c:\os
c:\OS>dir
Volume in drive C has no label
Volume Serial Number is 5A22

Directory of c:\OS

22/09/2020 12:51 <DIR>
22/09/2020 12:51 <DIR>
22/09/2020 12:51 <DIR>
22/09/2020 12:51 <DIR>
17/12/2008 00:08 1.0
22/09/2020 12:51 <DIR>
17/12/2008 00:07 8
17/12/2008 01:44 7
13/12/2008 14:12 5 Dir(s) 139.

c:\OS>setpath
c:\OS>Path=C:\OS\Dev-Cpp\bin;
m32
c:\OS>cd lab
c:\OS\LAB>cd lab1
c:\OS\LAB\LAB1>Notepad boot.asm
c:\OS\LAB\LAB1>Notepad Makefile
c:\OS\LAB\LAB1>

```

The Notepad window displays the content of Makefile:

```

#
# LAB01 - Makefile
#

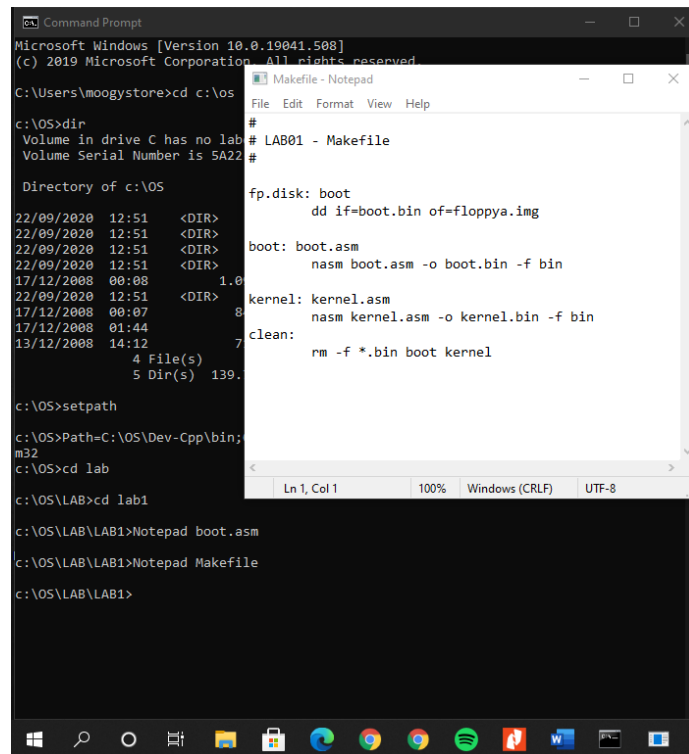
fp.disk: boot
dd if=boot.bin of=floppya.img

boot: boot.asm
nasm boot.asm -o boot.bin -f bin

kernel: kernel.asm
nasm kernel.asm -o kernel.bin -f bin

clean:
rm -f *.bin boot kernel

```



```
Microsoft Windows [Version 10.0.19041.508]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\moogystore>cd c:\os

c:\OS>dir
Volume in drive C has no label
Volume Serial Number is 5A22

Directory of c:\OS

22/09/2020  12:51    <DIR>
22/09/2020  12:51    <DIR>
22/09/2020  12:51    <DIR>
22/09/2020  12:51    <DIR>
17/12/2008  00:08             1.0
22/09/2020  12:51    <DIR>
17/12/2008  00:07             8
17/12/2008  01:44             7
13/12/2008  14:12             4 File(s)
                    5 Dir(s)  139.

c:\OS>setpath

c:\OS>Path=C:\OS\Dev-Cpp\bin;
m32

c:\OS>cd lab

c:\OS\LAB>cd lab1

c:\OS\LAB\LAB1>Notepad boot.asm

c:\OS\LAB\LAB1>Notepad Makefile

c:\OS\LAB\LAB1>
```

```
# LAB01 - Makefile

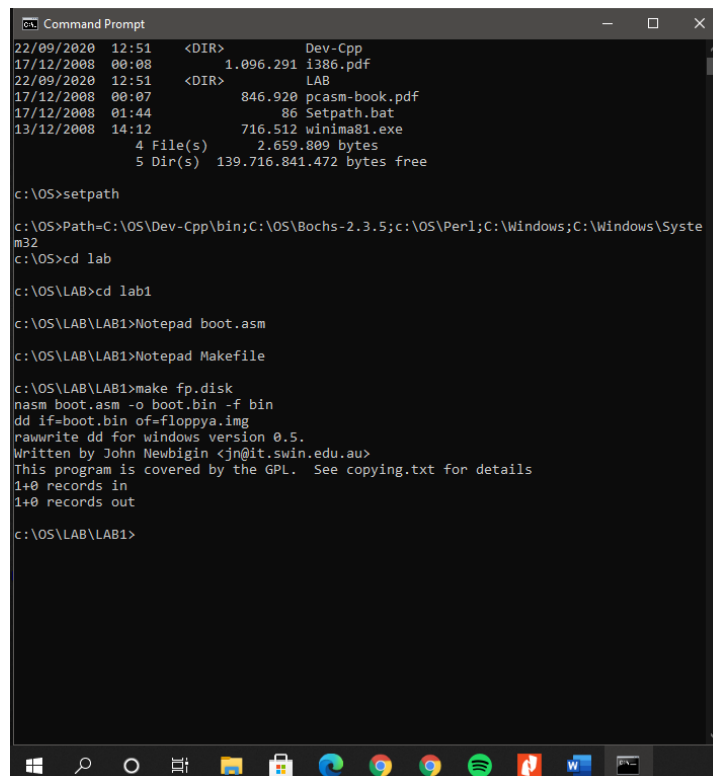
fp.disk: boot
    dd if=boot.bin of=floppya.img

boot: boot.asm
    nasm boot.asm -o boot.bin -f bin

kernel: kernel.asm
    nasm kernel.asm -o kernel.bin -f bin

clean:
    rm -f *.bin boot kernel
```

- bukanlah ‘Command Prompt’ dan buka direktori kerja ‘LAB1’ selanjutnya ketik ‘make fp.disk’
- Periksa hasil kompilasi dengan memasukan perintah ‘dir’, sekarang pada direktori kerja terdapat tambahan file baru, yaitu ‘boot.bin’ dan isinya sudah disalin kedalam bootsector ‘floppya.img’.



```
22/09/2020  12:51    <DIR>      Dev-Cpp
17/12/2008  00:08      1.096.291 i386.pdf
22/09/2020  12:51    <DIR>      LAB
17/12/2008  00:07      846.920 pcasm-book.pdf
17/12/2008  01:44           86 Setpath.bat
13/12/2008  14:12      716.512 winima81.exe
                    4 File(s)    2.659.809 bytes
                    5 Dir(s)  139.716.841.472 bytes free

c:\OS>setpath

c:\OS>Path=C:\OS\Dev-Cpp\bin;C:\OS\Bochs-2.3.5;c:\OS\Perl;c:\Windows;c:\Windows\Syste
m32
c:\OS>cd lab

c:\OS\LAB>cd lab1

c:\OS\LAB\LAB1>Notepad boot.asm

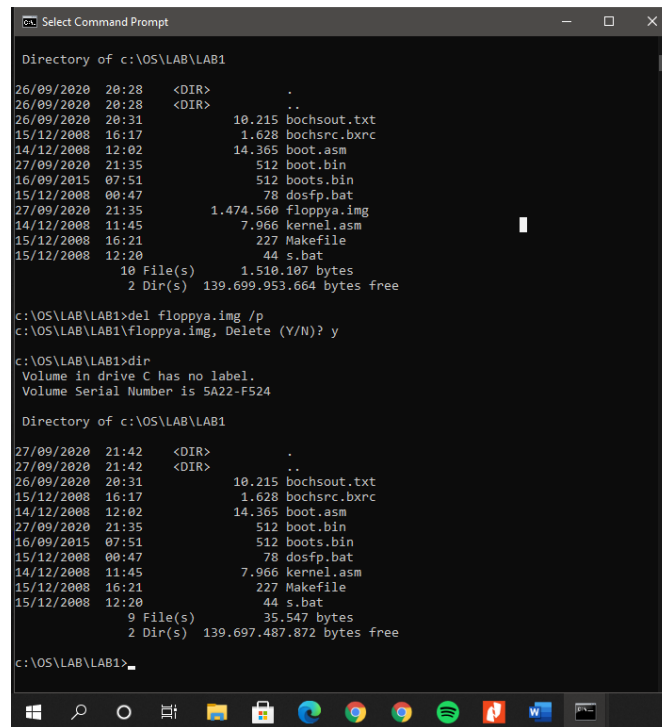
c:\OS\LAB\LAB1>Notepad Makefile

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>
```

Mengenal 'BOOT DISK'

- 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'.



```
Select Command Prompt

Directory of c:\OS\LAB\LAB1
26/09/2020 20:28 <DIR> .
26/09/2020 20:28 <DIR> ..
26/09/2020 20:31      10.215 bochsout.txt
15/12/2008 16:17      1.628 bochsrc.bxrc
14/12/2008 12:02     14.365 boot.asm
27/09/2020 21:35      512 boot.bin
16/09/2015 07:51     512 boots.bin
15/12/2008 00:47      78 dosfp.bat
27/09/2020 21:35    1.474.560 floppya.img
14/12/2008 11:45     7.966 kernel.asm
15/12/2008 16:21     227 Makefile
15/12/2008 12:20      44 s.bat
               10 File(s)      1.510.107 bytes
               2 Dir(s)  139.699.953.664 bytes free

c:\OS\LAB\LAB1>del floppya.img /p
c:\OS\LAB\LAB1>floppya.img, Delete (Y/N)? y

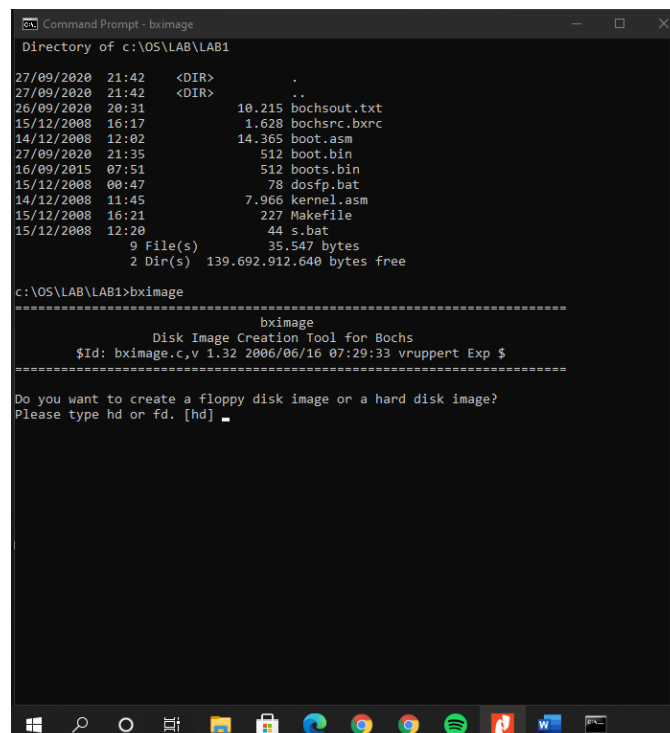
c:\OS\LAB\LAB1>dir
Volume in drive C has no label.
Volume Serial Number is 5A22-F524

Directory of c:\OS\LAB\LAB1

27/09/2020 21:42 <DIR> .
27/09/2020 21:42 <DIR> ..
26/09/2020 20:31      10.215 bochsout.txt
15/12/2008 16:17      1.628 bochsrc.bxrc
14/12/2008 12:02     14.365 boot.asm
27/09/2020 21:35      512 boot.bin
16/09/2015 07:51     512 boots.bin
15/12/2008 00:47      78 dosfp.bat
14/12/2008 11:45     7.966 kernel.asm
15/12/2008 16:21     227 Makefile
15/12/2008 12:20      44 s.bat
               9 File(s)       35.547 bytes
               2 Dir(s)  139.697.487.872 bytes free

c:\OS\LAB\LAB1>
```

- Selanjutnya panggil 'bximage' lalu tekan <ENTER>



```
Command Prompt - bximage

Directory of c:\OS\LAB\LAB1
27/09/2020 21:42 <DIR> .
27/09/2020 21:42 <DIR> ..
26/09/2020 20:31      10.215 bochsout.txt
15/12/2008 16:17      1.628 bochsrc.bxrc
14/12/2008 12:02     14.365 boot.asm
27/09/2020 21:35      512 boot.bin
16/09/2015 07:51     512 boots.bin
15/12/2008 00:47      78 dosfp.bat
14/12/2008 11:45     7.966 kernel.asm
15/12/2008 16:21     227 Makefile
15/12/2008 12:20      44 s.bat
               9 File(s)       35.547 bytes
               2 Dir(s)  139.692.912.640 bytes free

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] _
```

- Ada dua pilihan file image yaitu [hd] untuk membuat harddisk image atau [fd] untuk membuat floppy image. Kita akan membuat floppy image (karena ukuran filenya lebih kecil), selanjutnya ketikan 'fd' dan tekan <ENTER>

```
Command Prompt - bxiimage
Directory of c:\OS\LAB\LAB1
27/09/2020 21:42 <DIR>      .
27/09/2020 21:42 <DIR>      ..
26/09/2020 20:31             10.215 bochsout.txt
15/12/2008 16:17             1.628 bochsrc.bxrc
14/12/2008 12:02             14.365 boot.asm
27/09/2020 21:35             512 boot.bin
16/09/2015 07:51             512 boots.bin
15/12/2008 00:47             78 dosfp.bat
14/12/2008 11:45             7.966 kernel.asm
15/12/2008 16:21            227 Makefile
15/12/2008 12:20             44 s.bat
                9 File(s)      35.547 bytes
                2 Dir(s)    139.692.912.640 bytes free

c:\OS\LAB\LAB1>bxiimage
=====
                bxiimage
                Disk Image Creation Tool for Bochs
                $Id: bxiimage.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]
```

- Ada beberapa tipe (kapasitas) yang ditawarkan, pilih tipe yang paling banyak digunakan saat ini yaitu tipe floppy dengan kapasitas '1.44MB', ditunjukan oleh angka [1.44] selanjutnya tekan <ENTER>

```
Command Prompt - bxiimage
15/12/2008 00:47             78 dosfp.bat
14/12/2008 11:45             7.966 kernel.asm
15/12/2008 16:21            227 Makefile
15/12/2008 12:20             44 s.bat
                9 File(s)      35.547 bytes
                2 Dir(s)    139.692.912.640 bytes free

c:\OS\LAB\LAB1>bxiimage
=====
                bxiimage
                Disk Image Creation Tool for Bochs
                $Id: bxiimage.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] _
```

- untuk memberikan nama file, ketikan 'floppya.img' dan <ENTER>

```
Command Prompt - bximage
15/12/2008 12:20 44 s.bat
9 File(s) 35.547 bytes
2 Dir(s) 139.692.912.640 bytes free

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] 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
```

- Pastikan keberadaan file image tersebut dengan perintah 'dir'

```
Command Prompt
[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 5A22-F524

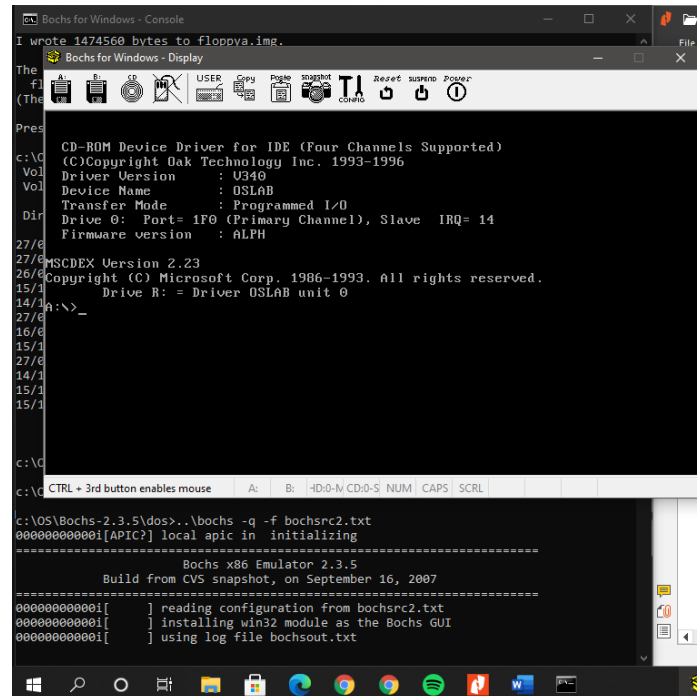
Directory of c:\OS\LAB\LAB1

27/09/2020 21:49 <DIR> .
27/09/2020 21:49 <DIR> ..
26/09/2020 20:31 10.215 bochsout.txt
15/12/2008 16:17 1.628 bochsrc.bxrc
14/12/2008 12:02 14.365 boot.asm
27/09/2020 21:35 512 boot.bin
16/09/2015 07:51 512 boots.bin
15/12/2008 00:47 78 dosfp.bat
27/09/2020 21:49 1.474.560 floppya.img
14/12/2008 11:45 7.966 kernel.asm
15/12/2008 16:21 227 Makefile
15/12/2008 12:20 44 s.bat
10 File(s) 1.510.107 bytes
2 Dir(s) 139.684.945.920 bytes free

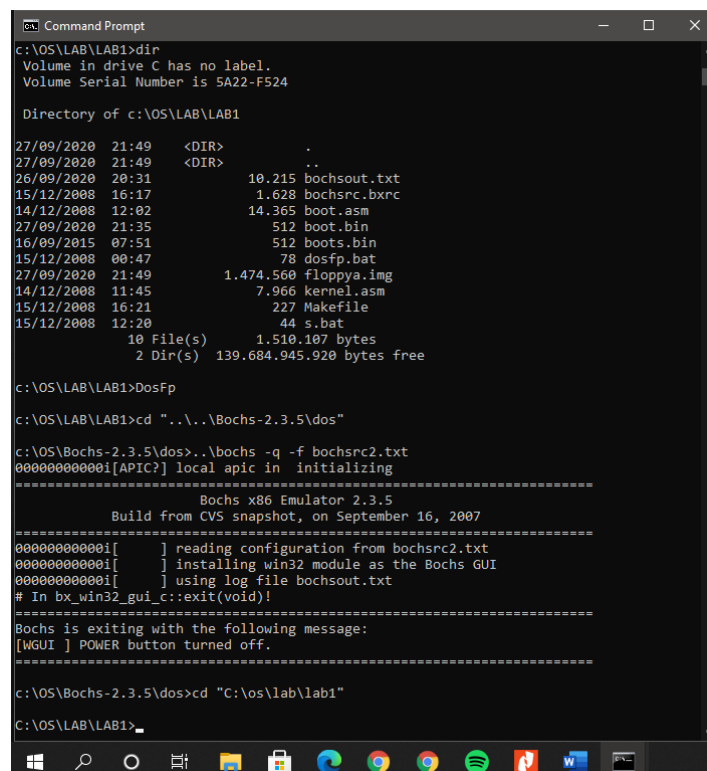
c:\OS\LAB\LAB1>
```

Langkah-langkah untuk memformat 'Floppya.img' adalah:

- Jalankan PC-Simulator dari 'Command Prompt' dengan perintah 'DosFp'.
- pada konfigurasi PC-Simulator file 'floppya.img' terpasang pada 'drive B:'. Selanjutnya dari prompt 'A:>' ketikkan 'Format B:' <ENTER> [2x]



- Tutup kembali PC-Simulator dengan klik pada tombol power, di bagian menu atas-kanan.



- Untuk mengcopy 512 byte data bootsektor ke dalam sebuah file terpisah caranya ketik ‘dd if=floppya.img of=boots.bin count=1’

```

C:\OS\LAB\LAB1>cd "..\..\Bochs-2.3.5\dos"

C:\OS\Bochs-2.3.5\dos>..\bochs -q -f bochsrc2.txt
0000000000i[APIC?] local apic in  initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
0000000000i[      ] reading configuration from bochsrc2.txt
0000000000i[      ] installing win32 module as the Bochs GUI
0000000000i[      ] using log file bochsout.txt
# In bx_win32_gui c::exit(void)!
=====
Bochs is exiting with the following message:
[VGUI ] POWER button turned off.
=====

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

C:\OS\LAB\LAB1>dd if=floppya.img of=boots.bin count=1
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>
  
```

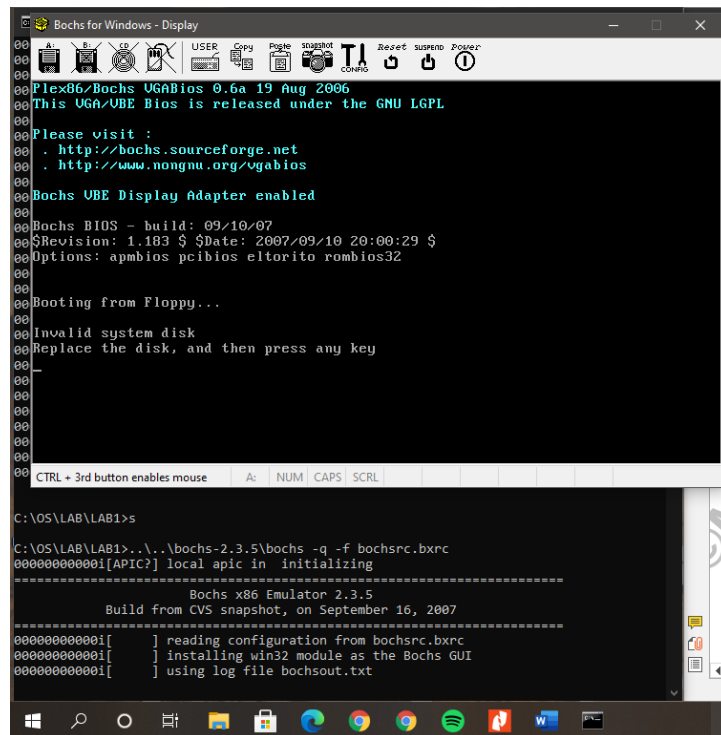
- Untuk melihat isinya ketik ‘tdump boots.bin’ lalu tekan <ENTER>

```

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 0B F0 09 00 12 00 02 00 00 00 00 00 ...@.....
000020: 00 00 00 00 00 00 29 EC 1D 6B 0D 4E 4F 20 4E 41 .....).k.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 0B FC F3 A4 06 U."...~.N$} ...
000060: 1F BD 00 7C C6 45 FE 0F 38 4E 24 7D 20 88 C1 99 ...|.E..8N$} ...
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 88 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 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 06 92 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.....
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 4A 69 73 68 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 68 7C 20 61 6E ace the disk, an
0001C0: 64 20 74 68 65 6E 20 70 72 65 73 73 20 61 6E 70 d then press any
0001D0: 20 68 65 79 0D 0A 00 00 40 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 BB 00 07 60 66 6A 00 E9 3B FF 00 00 55 AA .A...`fj...U..
  
```

- Selanjutnya masukan perintah 's' <ENTER>, akan ditampilkan windows 'Bochs for windows – display' yang sedang melakukan proses 'booting' namun tidak berhasil karena tidak menemukan diskboot, seperti ditampilkan pada gambar di bawah lalu tekan power.



```
Bochs for Windows - Display
Plex86/Bochs UGABios 0.6a 19 Aug 2006
This UGA/UEFI Bios is released under the GNU LGPL

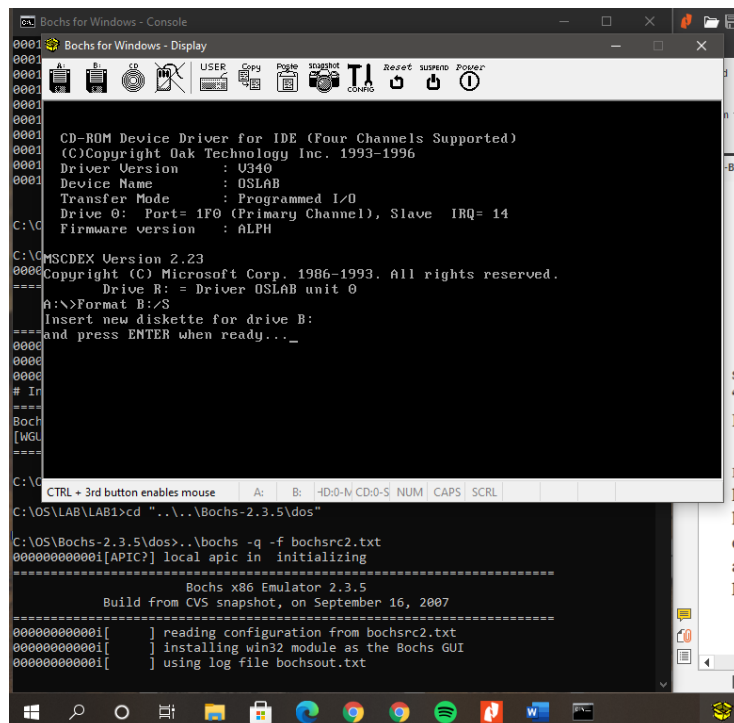
Please visit :
. http://bochs.sourceforge.net
. http://www.nongnu.org/ugabios

Bochs UEFI 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
```

- Pada windows 'Bochs' masukan perintah 'A:>format B:/S ' <ENTER>



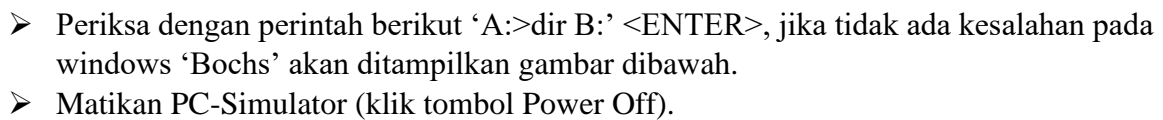
```
Bochs for Windows - Console
CD-ROM Device Driver for IDE (Four Channels Supported)
(C)Copyright Oak Technology Inc. 1993-1996
Driver Version : U340
Device Name : OSLAB
Transfer Mode : Programmed I/O
Drive 0: Port= 1F0 (Primary Channel), Slave IRQ= 14
Firmware version : ALPH

C:\CD\MSCDEx Version 2.23
Copyright (C) Microsoft Corp. 1986-1993. All rights reserved.
====
Drive R: = Driver OSLAB unit 0
A:>format B:/S
Insert new diskette for drive B:
and press ENTER when ready..._

# In
Boch
[WGL

C:\CD
CTRL + 3rd button enables mouse  A: B: ID-0-IV CD-0-S NUM CAPS SCRL

C:\OS\LAB\LAB1>cd "..\..\Bochs-2.3.5\dos"
C:\OS\Bochs-2.3.5\dos>..\bochs -q -f bochsrc2.txt
0000000000i[APIC?] local apic in initializing
-----
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
-----
0000000000i[ ] reading configuration from bochsrc2.txt
0000000000i[ ] installing win32 module as the Bochs GUI
0000000000i[ ] using log file bochsout.txt
```



- Sekarang kita coba untuk menggunakan 'floppya.img' sebagai 'boot disk' lagi, ketik 'S' <ENTER>. Jika tidak ada kesalahan akan menampilkan seperti gambar dibawah.
- Lalu klik tombol power off

```

Bochs for Windows - Console
=====
Bochs for Windows - Display
=====
000
000
000
# I Plex86/Bochs VGABios 0.6a 19 Aug 2006
=====
This VGA/VBE Bios is released under the GNU LGPL
Boc
[WG Please visit :
=====
. http://bochs.sourceforge.net
. http://www.nongnu.org/vgabios
C:\
Bochs VBE Display Adapter enabled
C:\
Bochs BIOS - build: 09/10/07
C:\$Revision: 1.183 $ $Date: 2007/09/10 20:00:29 $
000Options: apmbios pcibios eltorito rombios32
=====
Booting from Floppy...
Starting BootCD.....
000
000
000
# I Microsoft(R) MS-DOS 7.1
=====
(C)Copyright Microsoft Corp 1981-1999.
Boc
[WGA:\>_
=====
C:\
CTRL + 3rd button enables mouse A: NUM CAPS SCRL
C:\OS\LAB\LAB1>s
C:\OS\LAB\LAB1>..\..\bochs-2.3.5\bochs -q -f bochsrc.bxrc
00000000000i[APIC?] local apic in initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
00000000000i[ ] reading configuration from bochsrc.bxrc
00000000000i[ ] installing win32 module as the Bochs GUI
00000000000i[ ] using log file bochsout.txt

```

TUGAS 1

1. Apa yang dimaksud dengan kode 'ASCII', buatlah tabel kode ASCII lengkap cukup kode ASCII yang standar tidak perlu extended, tuliskan kode ASCII dalam format angka desimal, binary dan hexadesimal serta karakter dan simbol yang dikodekan.

Kode ASCII adalah **ASCII** (American Standard Code for Information Interchange) merupakan Kode Standar Amerika untuk Pertukaran Informasi atau sebuah standar internasional dalam pengkodean huruf dan simbol.

Decimal	Hex	Binary	Karakter
32	20	100000	Spasi
33	21	100001	!
34	22	100010	"
35	23	100011	#
36	24	100100	\$
37	25	100101	%
38	26	100110	&
39	27	100111	'
40	28	101000	(
41	29	101001)
42	2A	101010	*
43	2B	101011	+
44	2C	101100	,
45	2D	101101	-
46	2E	101110	.
47	2F	101111	/
48	30	110000	0
49	31	110001	1
50	32	110010	2
51	33	110011	3
52	34	110100	4
53	35	110101	5
54	36	110110	6
55	37	110111	7
56	38	111000	8
57	39	111001	9
58	3A	111010	:
59	3B	111011	;
60	3C	111100	<
61	3D	111101	=
62	3E	111110	>
63	3F	111111	?
64	40	1000000	@
65	41	1000001	A

66	42	1000010	B
67	43	1000011	C
68	44	1000100	D
69	45	1000101	E
70	46	1000110	F
71	47	1000111	G
72	48	1001000	H
73	49	1001001	I
74	4A	1001010	J
75	4B	1001011	K
76	4C	1001100	L
77	4D	1001101	M
78	4E	1001110	N
79	4F	1001111	O
80	50	1010000	P
81	51	1010001	Q
82	52	1010010	R
83	53	1010011	S
84	54	1010100	T
85	55	1010101	U
86	56	1010110	V
87	57	1010111	W
88	58	1011000	X
89	59	1011001	Y
90	5A	1011010	Z
91	5B	1011011	[
92	5C	1011100	\
93	5D	1011101]
94	5E	1011110	^
95	5F	1011111	_
96	60	1100000	`
97	61	1100001	a
98	62	1100010	b
99	63	1100011	c
100	64	1100100	d
101	65	1100101	e
102	66	1100110	f
103	67	1100111	g
104	68	1101000	h
105	69	1101001	i
106	6A	1101010	j
107	6B	1101011	k
108	6C	1101100	l
109	6D	1101101	m
110	6E	1101110	n
111	6F	1101111	o
112	70	1110000	p
113	71	1110001	q

114	72	1110010	r
115	73	1110011	s
116	74	1110100	t
117	75	1110101	u
118	76	1110110	v
119	77	1110111	w
120	78	1111000	x
121	79	1111001	y
122	7A	1111010	z
123	7B	1111011	{
124	7C	1111100	
125	7D	1111101	}
126	7E	1111110	~
127	7F	1111111	DEL

- Carilah daftar perintah bahasa assembly untuk mesin intel keluarga x86 lengkap (dari buku referensi atau internet). Daftar perintah ini dapat digunakan sebagai pedoman untuk memahami program 'boot.asm' dan 'kernel.asm'.

Terbagi menjadi 3 bagian utama yaitu :

1. Komentar

Komentar diawali dengan tanda titik koma (;).

; ini adalah komentar

2. Label

Label diakhiri dengan tanda titik dua (:).

Contoh: main: ,loop: ,proses: ,keluar:

3. Assembler directives

Directives adalah perintah yang ditujukan kepada assembler ketika sedang menerjemahkan program kita ke bahasa mesin.

Directive dimulai dengan tanda titik. **.model** : memberitahu assembler berapa memori yang akan dipakai oleh program kita.

Ada model tiny, model small, model compact, model medium, model large, dan model huge.

.data : memberitahu assembler bahwa bagian di bawah ini adalah data program.

.code : memberitahu assembler bahwa bagian di bawah ini adalah instruksi program.

.stack : memberitahu assembler bahwa program kita memiliki stack.

Program EXE harus punya stack. Kira-kira yang penting itu dulu.

Semua directive yang dikenal assembler adalah: .186 .286 .286c .286p .287 .386 .386c .386p .387 .486 .486p .8086 .8087 .alpha .break .code .const .continue .cref .data .data? .dosseg .else .elseif .endif .endw .err .err1 .err2 .errb .errdef .errdif .errdifi .erre .erridn .erridni .errnb .errndef .errnz .exit .fardata .fardata? .if .lall .lfcond .list .listall .listif .listmacro .listmacroall .model .no87 .nocref .nolist .nolistif .nolistmacro .radix .repeat .sall .seq .sfcond .stack .startup .tfcond .type .until .untilcxz .while .xall .xcref .xlist.

Definisi data

DB : define bytes. Membentuk data byte demi byte. Data bisa data numerik maupun teks.

catatan: untuk membentuk data string, pada akhir string harus diakhiri tanda dolar (\$).

sintaks: {label} DB {data} contoh: teks1 db "Hello world \$" **DW** : define words.

Membentuk data word demi word (1 word = 2 byte).

sintaks: {label} DW {data} contoh: kucing dw ?, ?, ? ;mendefinisikan tiga slot 16-bit yang isinya don't care

(disimbolkan dengan tanda tanya)

DD : define double words. Membentuk data doubleword demi doubleword (4 byte).

sintaks: {label} DD {data} **EQU** : equals. Membentuk konstanta. sintaks: {label}

EQU {data}

contoh: sepuluh EQU 10

Ada assembly yang melibatkan bilangan pecahan (floating point), bilangan bulat (integer), DF (define far words),

DQ (define quad words), dan DT (define ten bytes).

Perpindahan data

MOV : move. Memindahkan suatu nilai dari register ke memori, memori ke register, atau register ke register.

sintaks: MOV {tujuan}, {sumber}

contoh:

mov AX, 4C00h ;mengisi register AX dengan 4C00(hex).

mov BX, AX ;menyalin isi AX ke BX. mov CL, [BX] ;mengisi register CL dengan data di memori yang alamatnya ditunjuk BX.

mov CL, [BX] + 2 ;mengisi CL dengan data di memori yang alamatnya ditunjuk BX lalu geser maju 2 byte.

mov [BX], AX ;menyimpan nilai AX pada tempat di memori yang ditunjuk BX. mov [BX] - 1, 00101110b ;menyimpan 00101110(bin) pada alamat yang ditunjuk BX lalu geser mundur 1 byte.

LEA : load effective address. Mengisi suatu register dengan alamat offset sebuah data.

sintaks: LEA {register}, {sumber} contoh: lea DX, teks1 **XCHG** : exchange.

Menukar dua buah register langsung.

sintaks: XCHG {register 1}, {register 2} Kedua register harus punya ukuran yang sama.

Bila sama-sama 8 bit (misalnya AH dengan BL) atau sama-sama 16 bit (misalnya CX dan DX),

maka pertukaran bisa dilakukan. Sebenarnya masih banyak perintah perpindahan data, misalnya IN, OUT, LODS, LODSB, LODSW, MOVS, MOVSB, MOVSW, LDS, LES, LAHF, SAHF, dan XLAT.

Operasi logika

AND : melakukan bitwise and. sintaks: AND {register}, {angka} AND {register 1}, {register 2} hasil disimpan di register 1.

contoh: mov AL, 00001011b mov AH, 11001000b and AL, AH ;sekarang AL berisi 00001000(bin),

sedangkan AH tidak berubah.

OR : melakukan bitwise or. sintaks: OR {register}, {angka} OR {register 1}, {register 2} hasil disimpan di register 1.

NOT : melakukan bitwise not (*one's complement*) sintaks: NOT {register} hasil disimpan di register itu sendiri.

XOR : melakukan bitwise eksklusif or. sintaks: XOR {register}, {angka} XOR {register 1}, {register 2} hasil disimpan di register 1. Tips: sebuah register yang di-XOR-kan dengan dirinya sendiri akan menjadi berisi nol.

SHL : shift left. Menggeser bit ke kiri. Bit paling kanan diisi nol. sintaks: SHL {register}, {banyaknya}

SHR : shift right. Menggeser bit ke kanan. Bit paling kiri diisi nol. sintaks: SHR {register}, {banyaknya}

ROL : rotate left. Memutar bit ke kiri. Bit paling kiri jadi paling kanan kali ini. sintaks: ROL {register}, {banyaknya} Bila banyaknya rotasi tidak disebutkan, maka nilai yang ada di CL akan digunakan sebagai banyaknya rotasi.

ROR : rotate right. Memutar bit ke kanan. Bit paling kanan jadi paling kiri. sintaks: ROR {register}, {banyaknya} Bila banyaknya rotasi tidak disebutkan, maka nilai yang ada di CL akan digunakan sebagai banyaknya rotasi.

Ada lagi : RCL dan RCR.

Operasi matematika

ADD : add. Menjumlahkan dua buah register.

sintaks: ADD {tujuan}, {sumber} operasi yang terjadi: $\text{tujuan} = \text{tujuan} + \text{sumber}$.
carry (bila ada) disimpan di CF.

ADC : add with carry. Menjumlahkan dua register dan carry flag (CF).

sintaks: ADC {tujuan}, {sumber} operasi yang terjadi: $\text{tujuan} = \text{tujuan} + \text{sumber} + \text{CF}$.

carry (bila ada lagi) disimpan lagi di CF.

INC : increment. Menjumlah isi sebuah register dengan 1.

Bedanya dengan ADD, perintah INC hanya memakan 1 byte memori sedangkan ADD pakai 3 byte.

sintaks: INC {register}

SUB : subtract. Mengurangkan dua buah register.

sintaks: SUB {tujuan}, {sumber} operasi yang terjadi: $\text{tujuan} = \text{tujuan} - \text{sumber}$.
borrow (bila terjadi) menyebabkan CF bernilai 1.

SBB : subtract with borrow. Mengurangkan dua register dan carry flag (CF).

sintaks: SBB {tujuan}, {sumber} operasi yang terjadi: $\text{tujuan} = \text{tujuan} - \text{sumber} - \text{CF}$.
borrow (bila terjadi lagi) menyebabkan CF dan SF (sign flag) bernilai 1.

DEC : decrement. Mengurang isi sebuah register dengan 1.

Jika SUB memakai 3 byte memori, DEC hanya memakai 1 byte. sintaks: DEC {register}

MUL : multiply. Mengalikan register dengan AX atau AH.

sintaks: MUL {sumber} Bila register sumber adalah 8 bit,

maka isi register itu dikali dengan isi AL, kemudian disimpan di AX.

Bila register sumber adalah 16 bit, maka isi register itu dikali dengan isi AX,

kemudian hasilnya disimpan di DX:AX. Maksudnya, DX berisi high order byte-nya, AX berisi low order byte-nya.

IMUL : signed multiply. Sama dengan MUL,

hanya saja IMUL menganggap bit-bit yang ada di register sumber sudah dalam bentuk *two's complement*.

sintaks: IMUL {sumber}

DIV : divide. Membagi AX atau DX:AX dengan sebuah register.

sintaks: DIV {sumber} Bila register sumber adalah 8 bit (misalnya: BL), maka operasi yang terjadi: -AX dibagi BL,

-hasil bagi disimpan di AL, -sisa bagi disimpan di AH.

Bila register sumber adalah 16 bit (misalnya: CX), maka operasi yang terjadi: -

DX:AX dibagi CX, -hasil bagi disimpan di AX, -sisa bagi disimpan di DX.

IDIV : signed divide. Sama dengan DIV, hanya saja IDIV menganggap bit-bit yang ada di register sumber sudah dalam bentuk *two's complement*.

sintaks: IDIV {sumber}

NEG : negate. Membuat isi register menjadi negatif (*two's complement*).

Bila mau *one's complement*, gunakan perintah NOT. sintaks: NEG {register} hasil disimpan di register itu sendiri.

Pengulangan

LOOP : loop. Mengulang sebuah proses. Pertama register CX dikurangi satu.

Bila CX sama dengan nol, maka looping berhenti. Bila tidak nol, maka lompat ke label tujuan.

sintaks: LOOP {label tujuan} Tips: isi CX dengan nol untuk mendapat jumlah pengulangan terbanyak.

Karena nol dikurang satu sama dengan -1, atau dalam notasi *two's complement* menjadi FFFF(hex) yang sama dengan 65535(dec).

LOOPE : loop while equal. Melakukan pengulangan selama $CX \neq 0$ dan $ZF = 1$. CX tetap dikurangi 1 sebelum diperiksa.

sintaks: LOOP {label tujuan}

LOOPZ : loop while zero. Identik dengan LOOPE.

LOOPNE : loop while not equal.

Melakukan pengulangan selama $CX \neq 0$ dan $ZF = 0$. CX tetap dikurangi 1 sebelum diperiksa.

sintaks: LOOPNE {label tujuan}

LOOPNZ : loop while not zero. Identik dengan LOOPNE.

REP : repeat. Mengulang perintah sebanyak CX kali. sintaks: REP {perintah assembly} contoh:

mov CX, 05 rep inc BX ;register BX ditambah 1 sebanyak 5x.

REPE : repeat while equal. Mengulang perintah sebanyak CX kali, tetapi pengulangan segera dihentikan bila didapati $ZF = 1$.

sintaks: REPE {perintah assembly}

REPZ : repeat while zero. Identik dengan REPE.

REPNE : repeat while not equal. Mengulang perintah sebanyak CX kali, tetapi pengulangan segera dihentikan bila didapati $ZF = 0$.

sintaks: REPNE {perintah assembly}

REPNZ : repeat while not zero. Identik dengan REPNE.

Perbandingan

CMP : compare. Membandingkan dua buah operand. Hasilnya mempengaruhi sejumlah flag register.

sintaks: CMP {operand 1}, {operand 2}. Operand ini bisa register dengan register, register dengan isi memori, atau register dengan angka.

CMP tidak bisa membandingkan isi memori dengan isi memori. Hasilnya adalah:

Kasus	Bila operand 1 < operand 2	Bila operand 1 = operand 2	Bila operand 1 > operand 2
Signed binary	OF = 1, SF = 1, ZF = 0	OF = 0, SF = 0, ZF = 1	OF = 0, SF = 0, ZF = 0
Unsigned binary	CF = 1, ZF = 0	CF = 0, ZF = 1	CF = 0, ZF = 0

Lompat-lompat

JMP: jump. Lompat tanpa syarat. Lompat begitu saja. sintaks: JMP {label tujuan}

Lompat bersyarat sintaksnya sama dengan JMP, yaitu perintah jump diikuti label tujuan.

PERINTAH	ARTI	SYARAT	KASUS	KETERANGAN ("OP" = OPERAND)	MENGIKUTI CMP?
JA	jump if above	$CF = 0 \wedge ZF = 0$	unsigned	lompat bila $op\ 1 > op\ 2$	ya
JNBE	jump if not below or equal				
JB	jump if below	$CF = 1 \wedge ZF = 0$	unsigned	lompat bila $op\ 1 < op\ 2$	ya
JNAE	jump if not above or equal				
JAE	jump if above or equal	$CF = 0 \vee ZF = 1$	unsigned	lompat bila $op\ 1 \geq op\ 2$	ya
JNB	jump if not below				
JBE	jump if below or equal	$CF = 1 \vee ZF = 1$	unsigned	lompat bila $op\ 1 \leq op\ 2$	ya
JNA	jump if not above				
JG	jump if greater	$OF = 0 \wedge ZF = 0$	signed	lompat bila $op\ 1 > op\ 2$	ya
JNLE	jump if not less or equal				
JGE	jump if greater or equal	$OF = 0 \vee ZF = 1$	signed	lompat bila $op\ 1 \geq op\ 2$	ya
JNL	jump if not less than				
JL	jump if less than	$OF = 1 \wedge ZF = 0$	signed	lompat bila $op\ 1 < op\ 2$	ya
JNGE	jump if not greater or equal				
JLE	jump if less or equal	$OF = 1 \vee ZF = 1$	signed	lompat bila $op\ 1 \leq op\ 2$	ya
JNG	jump if not greater				
JE	jump if equal	$ZF = 1$	keduanya	lompat bila $op\ 1 = op\ 2$	ya
JZ	jump if zero	$ZF = 1$	keduanya	lompat bila $op\ 1 = op\ 2$	ya
JNE	jump if not equal	$ZF = 0$	keduanya	lompat bila $op\ 1 \neq op\ 2$	ya
JNZ	jump if not zero	$ZF = 0$	keduanya	lompat bila $op\ 1 \neq op\ 2$	ya
JC	jump if carry	$CF = 1$	N/A	lompat bila carry flag = 1	tidak
JNC	jump if not carry	$CF = 0$	N/A	lompat bila carry flag = 0	tidak

JP	jump on parity	PF = 1	N/A	lompat bila parity flag = 1	tidak selalu
JPE	jump on parity even			lompat bila bilangan genap	
JNP	jump on not parity	PF = 0	N/A	lompat bila parity flag = 0	tidak selalu
JPO	jump on parity odd			lompat bila bilangan ganjil	
JO	jump if overflow	OF = 1	N/A	lompat bila overflow flag = 1	tidak
JNO	jump if not overflow	OF = 0	N/A	lompat bila overflow flag = 0	tidak
JS	jump if sign	SF = 1	N/A	lompat bila bilangan negatif	tidak
JCXZ	jump if CX is zero	CX = 0000	N/A	lompat bila CX berisi nol	tidak

Operasi stack

PUSH : push. Menambahkan sesuatu ke stack.

Sesuatu ini harus register berukuran 16 bit (pada 386+ harus 32 bit), tidak boleh angka, tidak boleh alamat memori.

Maka Anda tidak bisa mem-push register 8-bit seperti AH, AL, BH, BL, dan kawan-kawannya.

sintaks: push {register 16-bit sumber}

contoh: push DX push AX Setelah operasi push, register SP (stack pointer) otomatis dikurangi 2 (karena datanya 2 byte).

Makanya, “top” dari stack seakan-akan “tumbuh turun”.

POP : pop. Mengambil sesuatu dari stack.

Sesuatu ini akan disimpan di register tujuan dan harus 16-bit. Maka Anda tidak bisa mem-pop menuju AH, AL, dkk.

sintaks: POP {register 16-bit tujuan}

contoh: POP BX Setelah operasi pop, register SP otomatis ditambah 2 (karena 2 byte), sehingga “top” dari stack “naik” lagi.

Tip: karena register segmen tidak bisa diisi langsung nilainya, Anda bisa menggunakan stack sebagai perantara.

Contoh kodenya: mov AX, seg teks1 push AX pop DS

PUSHF : push flags. Mem-push **semua** isi register flag ke dalam stack.

Biasa dipakai untuk mem*backup* data di register flag sebelum operasi matematika.

Sintaks: PUSHF ;(saja).

POPF : pop flags. Lawan dari pushf. Sintaks: POPF ;(saja).

POPA : pop all general-purpose registers.

Adalah ringkasan dari sejumlah perintah dengan urutan:

pop DI pop SI pop BP pop SP pop BX pop DX pop CX pop AX

Urutan sudah ditetapkan seperti itu.

sintaks: POPA ;(saja). Jauh lebih cepat mengetikkan POPA daripada mengetik POP-POP-POP yang banyak itu.

PUSHA : push all general-purpose registers. Lawan dari POPA,

dimana PUSHA adalah singkatan dari sejumlah perintah dengan urutan yang sudah ditetapkan:

push AX push CX push DX push BX push SP push BP push SI push DI

Operasi pada register flag

CLC : clear carry flag. Menjadikan CF = 0. Sintaks: CLC ;(saja).

STC : set carry flag. Menjadikan CF = 1. Sintaks: STC ;(saja).

CMC : complement carry flag. Melakukan operasi NOT pada CF. Yang tadinya 0 menjadi 1, dan sebaliknya.

CLD : clear direction flag. Menjadikan DF = 0. Sintaks: CLD ;(saja).

STD : set direction flag. Menjadikan DF = 1.

CLI : clear interrupt flag. Menjadikan IF = 0, sehingga interrupt ke CPU akan di-disable.

Biasanya perintah CLI diberikan sebelum menjalankan sebuah proses penting yang riskan gagal bila diganggu.

STI : set interrupt flag. Menjadikan IF = 1.

Perintah lainnya

ORG : origin. Mengatur awal dari program (bagian static data).

Analoginya seperti mengatur dimana letak titik (0, 0) pada koordinat Cartesius.

sintaks: ORG {alamat awal}

Pada program COM (program yang berekstensi .com), harus ditulis “ORG 100h” untuk mengatur alamat mulai dari progam pada 0100(hex),

karena dari alamat 0000(hex) sampai 00FF(hex) sudah dipesan oleh sistem operasi (DOS).

INT : interrupt. Menginterupsi prosesor.

Prosesor akan:

1. Membackup data registernya saat itu,
2. Menghentikan apa yang sedang dikerjakannya,
3. Melompat ke bagian interrupt-handler (entah dimana kita tidak tahu, sudah ditentukan BIOS dan DOS),
4. Melakukan interupsi,
5. Mengembalikan data registernya,
6. Meneruskan pekerjaan yang tadi ditunda.

sintaks: INT {nomor interupsi}

IRET : interrupt-handler return.

Kita bisa membuat interrupt-handler sendiri dengan berbagai cara.

Perintah IRET adalah perintah yang menandakan bahwa interrupt-handler kita selesai, dan prosesor boleh melanjutkan pekerjaan yang tadi tertunda.

CALL : call procedure. Memanggil sebuah prosedur.

sintaks: CALL {label nama prosedur}

RET : return. Tanda selesai prosedur.

Setiap prosedur harus memiliki RET di ujungnya.

sintaks: RET ;(saja)

HLT : halt. Membuat prosesor menjadi tidak aktif.

Prosesor harus mendapat interupsi dari luar atau di-reset supaya aktif kembali.

Jadi, jangan gunakan perintah HLT untuk mengakhiri program!!

Sintaks: HLT ;(saja). **NOP** : no operation.

Perintah ini memakan 1 byte di memori tetapi tidak menyuruh prosesor melakukan apa-apa selama 3 clock prosesor.

Berikut contoh potongan program untuk melakukan *delay* selama 0,1 detik pada prosesor Intel 80386 yang berkecepatan 16 MHz.

mov ECX, 533333334d ;ini adalah bilangan desimal idle: nop loop idle.