

# **LAPORAN KEGIATAN PRAKTIKUM**



## **MODUL 1**

### **SISTEM OPERASI**

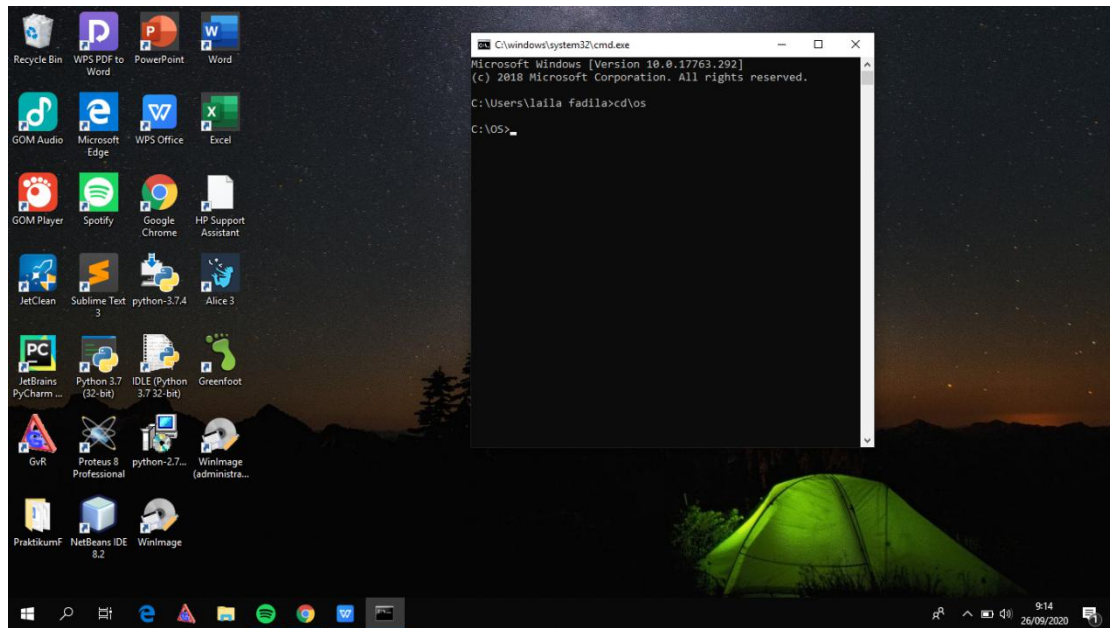
#### **PENGENALAN SISTEM PENGEMBANG OS**

**Nama : Laila Fadila Burhani**

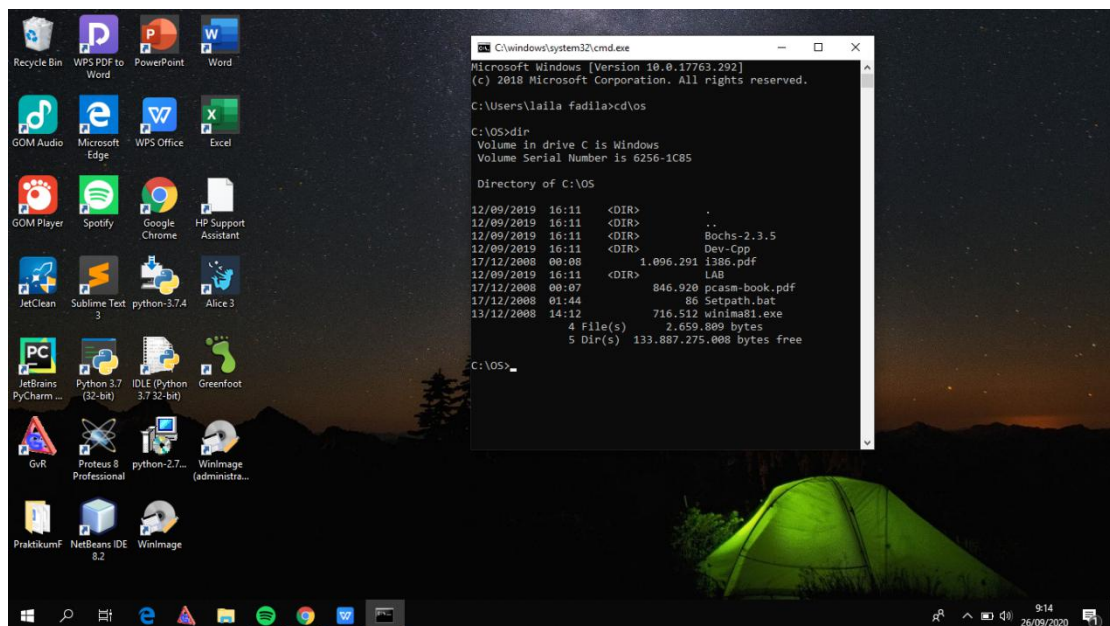
**Nim : L200190261**

**Kelas : G**

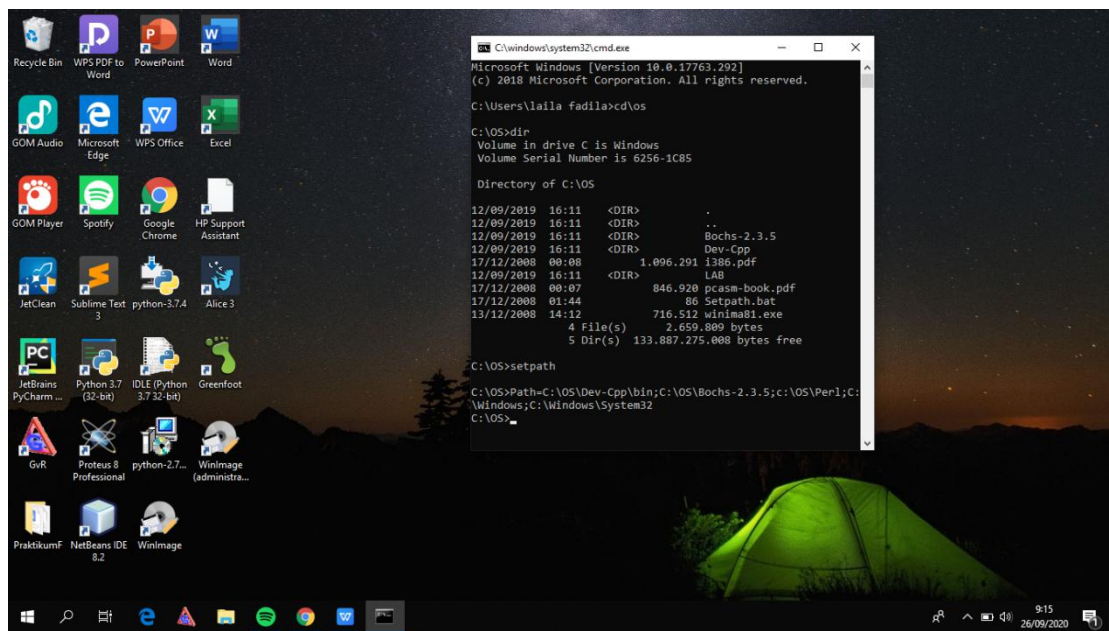
1. Masuk ke direktori kerja '**C:\OS**', dengan perintah '**cd\os**' kemudian <ENTER>



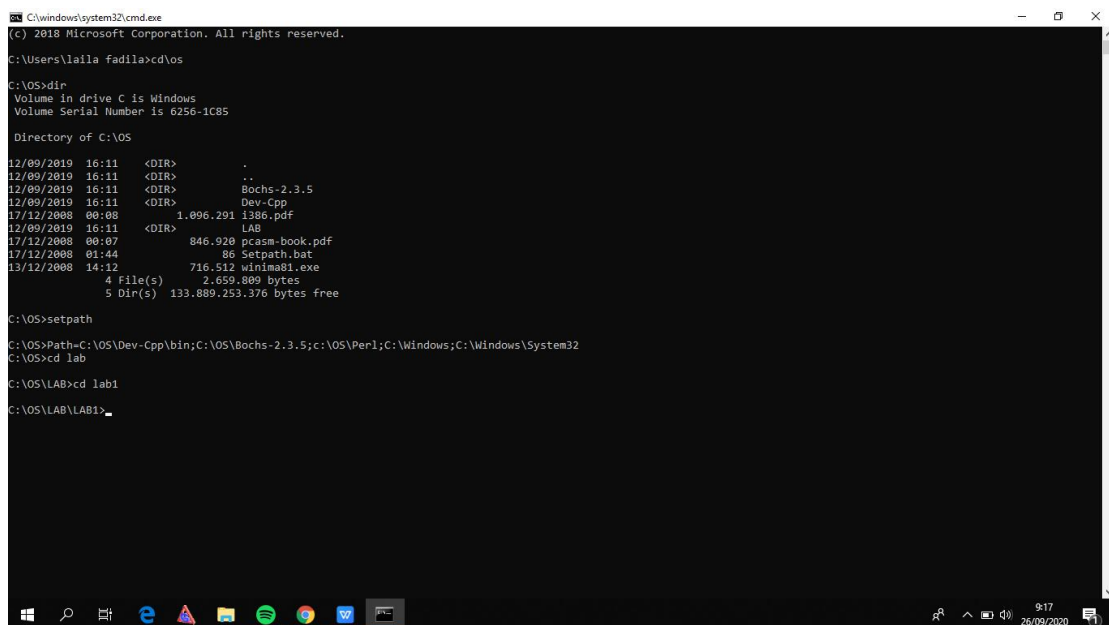
2. Masukan perintah **dir** kemudian <ENTER>. Dir digunakan untuk melihat isi direktori di dalam folder tersebut



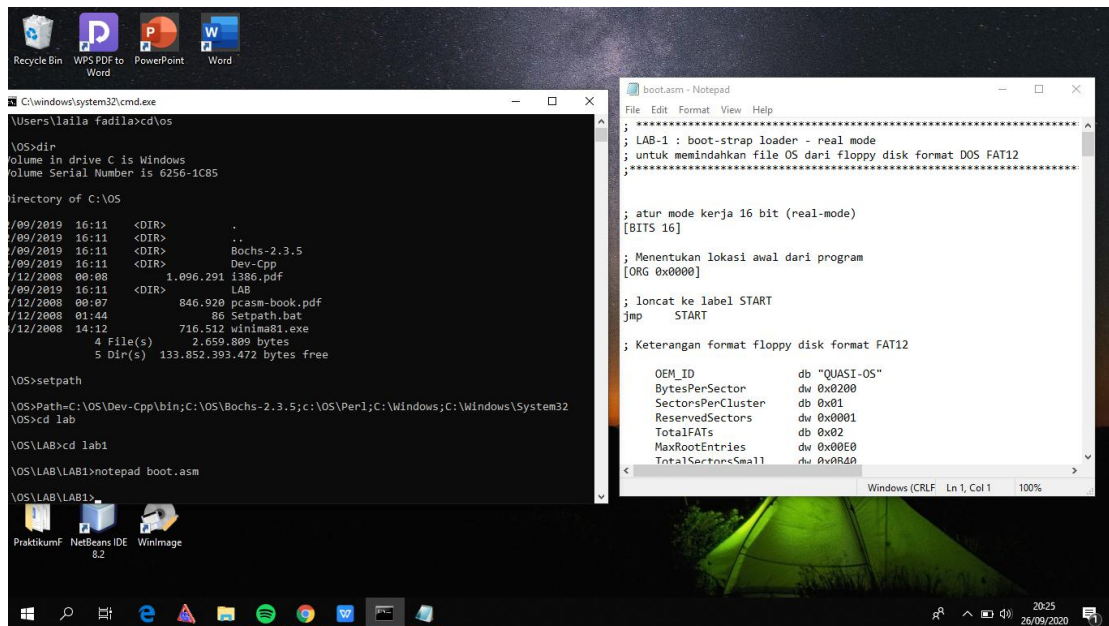
3. Ketik **'setpath'** untuk menjalankan file setpath. lalu tekan <ENTER>



4. Masuk ke direktori kerja pada **'C:\OS\LAB\LAB1'**



5. Ketik **'Notepad boot.asm'** dan tekan <ENTER>. Notepad boot.asm digunakan untuk melihat isi dari boot.asm di notepad



```
C:\Windows\system32\cmd.exe
\Users\laila fadila>cd \os
\OS>dir
Volume in drive C is Windows
Volume Serial Number is 6256-1C85

Directory of C:\OS

09/2019 16:11 <DIR>          .
09/2019 16:11 <DIR>          ..
09/2019 16:11 <DIR>          Bochs-2.3.5
09/2019 16:11 <DIR>          Dev-Cpp
12/2008 00:00 <DIR>          1.096.291 1386.pdf
09/2019 16:11 <DIR>          LAB
12/2008 00:07             846.920 pcasm-book.pdf
12/2008 01:44             86 Setpath.bat
12/2008 14:12             716.512 winima81.exe
               4 File(s)      2.659.809 bytes
               5 Dir(s)      133.852.393.472 bytes free

\OS>setpath
\OS>Path=C:\OS\Dev-Cpp\bin;C:\OS\Bochs-2.3.5;c:\OS\Perl;c:\Windows;C:\Windows\System32
\OS>cd lab
\OS\LAB>cd lab1
\OS\LAB\LAB1>notepad boot.asm
\OS\LAB\LAB1>
```

```
boot.asm - Notepad
File Edit Format View Help
; *****
; LAB-1 : boot-strap loader - real mode
; untuk memindahkan file OS dari floppy disk format DOS FAT12
; *****

; atur mode kerja 16 bit (real-mode)
[BITS 16]

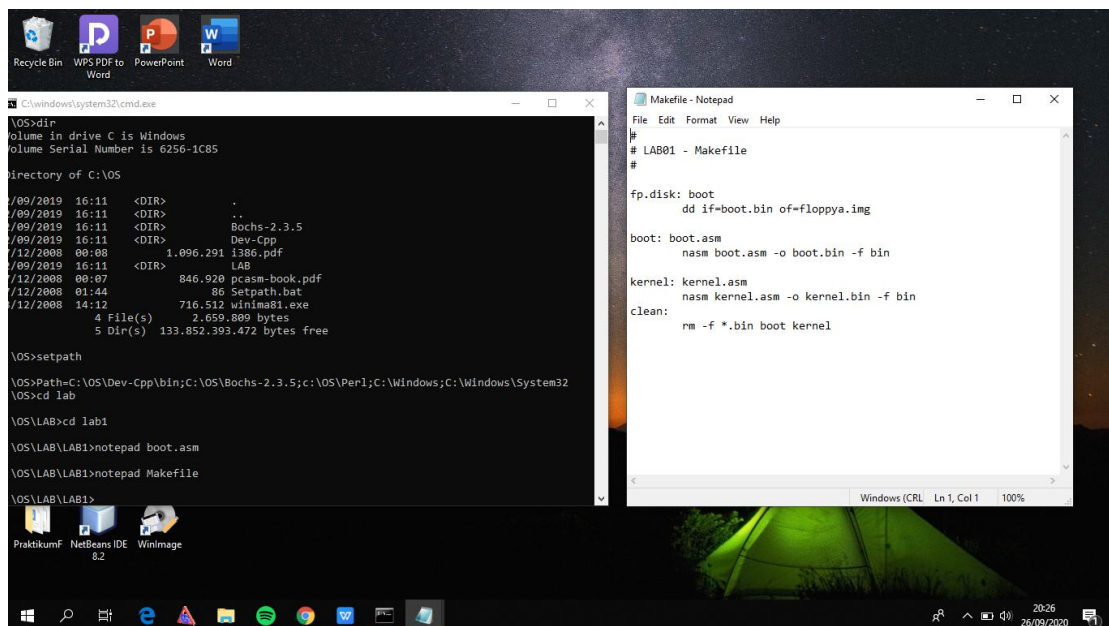
; Menentukan lokasi awal dari program
[ORG 0x0000]

; loncat ke label START
jmp START

; Keterangan format floppy disk format FAT12

OEM_ID          db "QUASI-OS"
BytesPerSector  dw 0x0200
SectorsPerCluster dw 0x01
ReservedSectors dw 0x0001
TotalFATs       dw 0x02
MaxRootEntries  dw 0x00E0
TotalSectorsSmall dw 0xFFFF
```

6. Ketik **'Notepad Makefile'** dan tekan <ENTER>



```
C:\Windows\system32\cmd.exe
\OS>dir
Volume in drive C is Windows
Volume Serial Number is 6256-1C85

Directory of C:\OS

09/2019 16:11 <DIR>          .
09/2019 16:11 <DIR>          ..
09/2019 16:11 <DIR>          Bochs-2.3.5
09/2019 16:11 <DIR>          Dev-Cpp
12/2008 00:00 <DIR>          1.096.291 1386.pdf
09/2019 16:11 <DIR>          LAB
12/2008 00:07             846.920 pcasm-book.pdf
12/2008 01:44             86 Setpath.bat
12/2008 14:12             716.512 winima81.exe
               4 File(s)      2.659.809 bytes
               5 Dir(s)      133.852.393.472 bytes free

\OS>setpath
\OS>Path=C:\OS\Dev-Cpp\bin;C:\OS\Bochs-2.3.5;c:\OS\Perl;c:\Windows;C:\Windows\System32
\OS>cd lab
\OS\LAB>cd lab1
\OS\LAB\LAB1>notepad boot.asm
\OS\LAB\LAB1>notepad Makefile
\OS\LAB\LAB1>
```

```
Makefile - Notepad
File Edit Format View Help
#
# LAB01 - Makefile
#

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

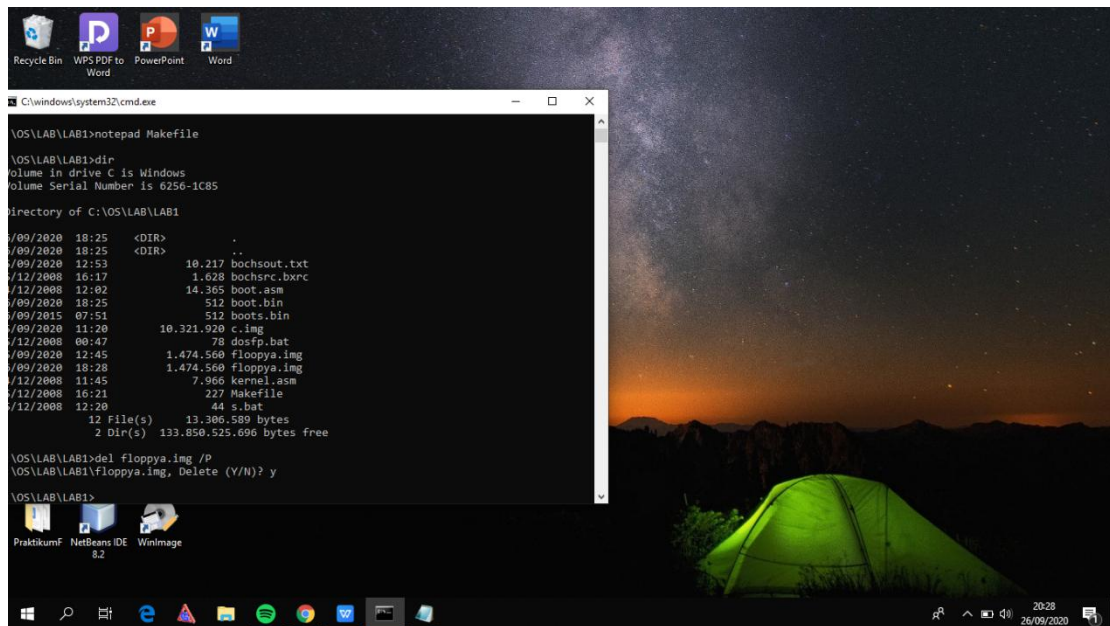
boot: boot.asm
nasm boot.asm -o boot.bin -f bin

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

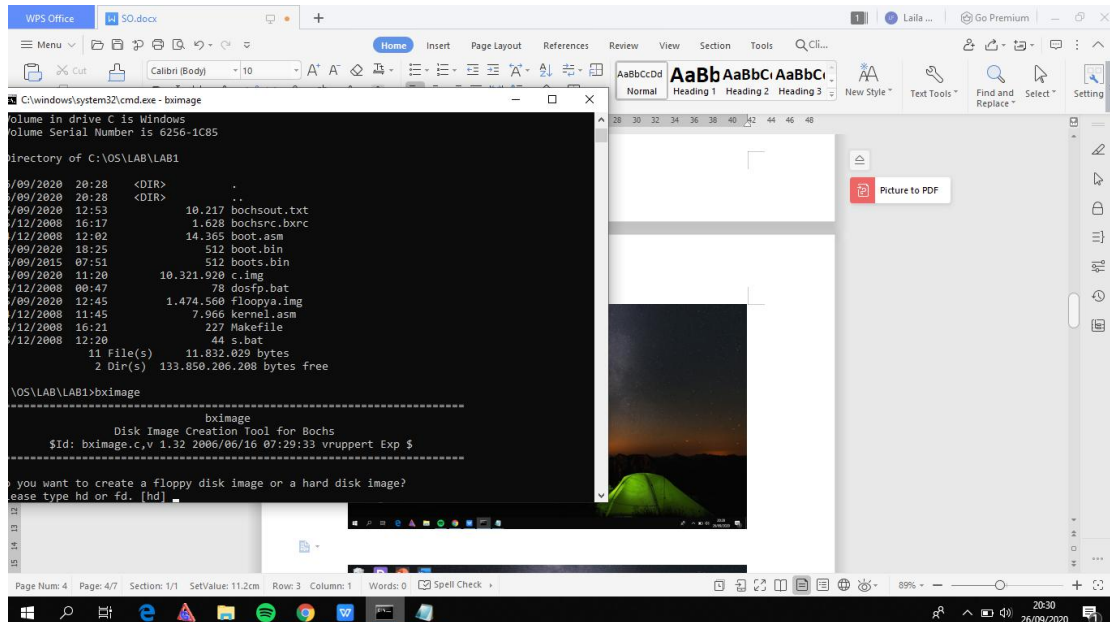
clean:
rm -f *.bin boot kernel
```



7. Ketik **'del floppy.img /P'** lanjutkan dengan tekan **'Y'** dan <ENTER>. Ini bertujuan untuk menghapus file **'floppy.img'** jika sudah ada di direktori kerja.



8. Ketik **'dir'** untuk memastikan bahwa file sudah benar-benar terhapus. Selanjutnya panggil **'bximage'**



9. Selanjutnya ketikkan **'fd'** dan tekan <ENTER>. Kenapa kita memilih fd karena kita akan membuat floppy image yang ukurannya lebih kecil

```
C:\windows\system32\cmd.exe - bximage
Volume in drive C is Windows
Volume Serial Number is 6256-1C85

Directory of C:\OS\LAB\LAB1
26/09/2020 20:28 <DIR>      .
26/09/2020 20:28 <DIR>      ..
25/09/2020 12:53          10.217 bochsout.txt
15/12/2008 16:17          1.628 bochsrc.bxrc
14/12/2008 12:02          14.365 boot.asm
26/09/2020 19:25          512 boot.bin
16/09/2015 07:51          512 boots.bin
25/09/2020 11:20       10.321.920 c.img
15/12/2008 00:47           78 dosfp.bat
25/09/2020 12:45       1.474.560 floppy.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
11 File(s)      11.832.029 bytes
2 Dir(s)        133.850.206.208 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] _
```

10. Pilih tipe yang paling banyak digunakan saat ini yaitu tipe floppy dengan kapasitas **'1.44MB'**

```
C:\windows\system32\cmd.exe - bximage
Volume in drive C is Windows
Volume Serial Number is 6256-1C85

Directory of C:\OS\LAB\LAB1
26/09/2020 20:28 <DIR>      .
26/09/2020 20:28 <DIR>      ..
25/09/2020 12:53          10.217 bochsout.txt
15/12/2008 16:17          1.628 bochsrc.bxrc
14/12/2008 12:02          14.365 boot.asm
26/09/2020 19:25          512 boot.bin
16/09/2015 07:51          512 boots.bin
25/09/2020 11:20       10.321.920 c.img
15/12/2008 00:47           78 dosfp.bat
25/09/2020 12:45       1.474.560 floppy.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
11 File(s)      11.832.029 bytes
2 Dir(s)        133.850.206.208 bytes free

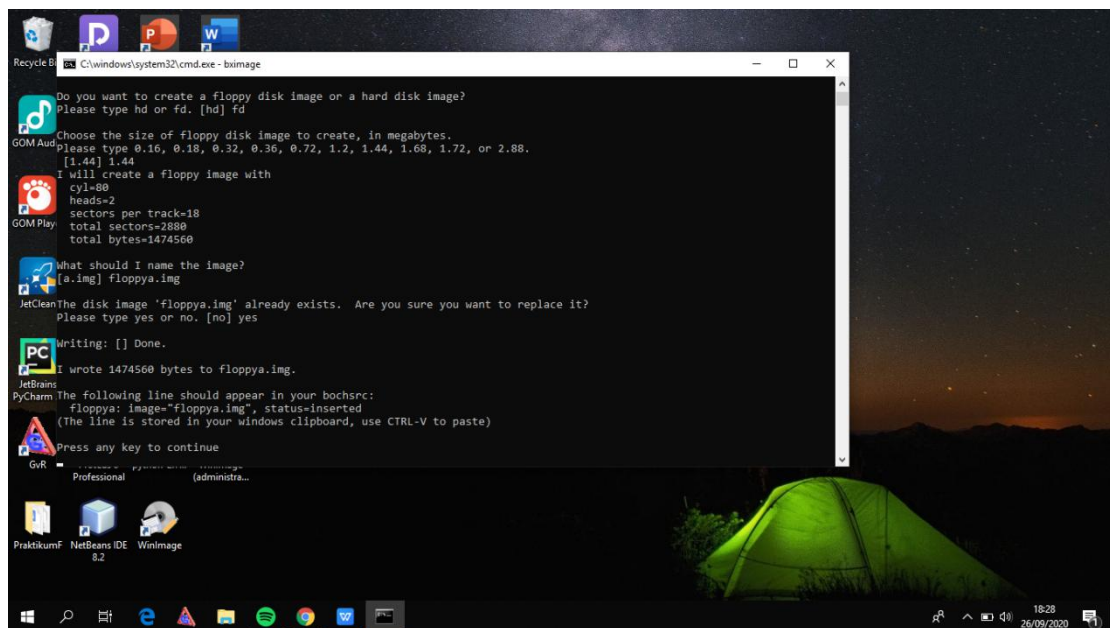
C:\OS\LAB\LAB1>bximage
-----
bximage
Disk Image Creation Tool for Bochs
$Id: bximage.c,v 1.32 2006/06/16 07:29:33 vruppert Exp $
-----

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

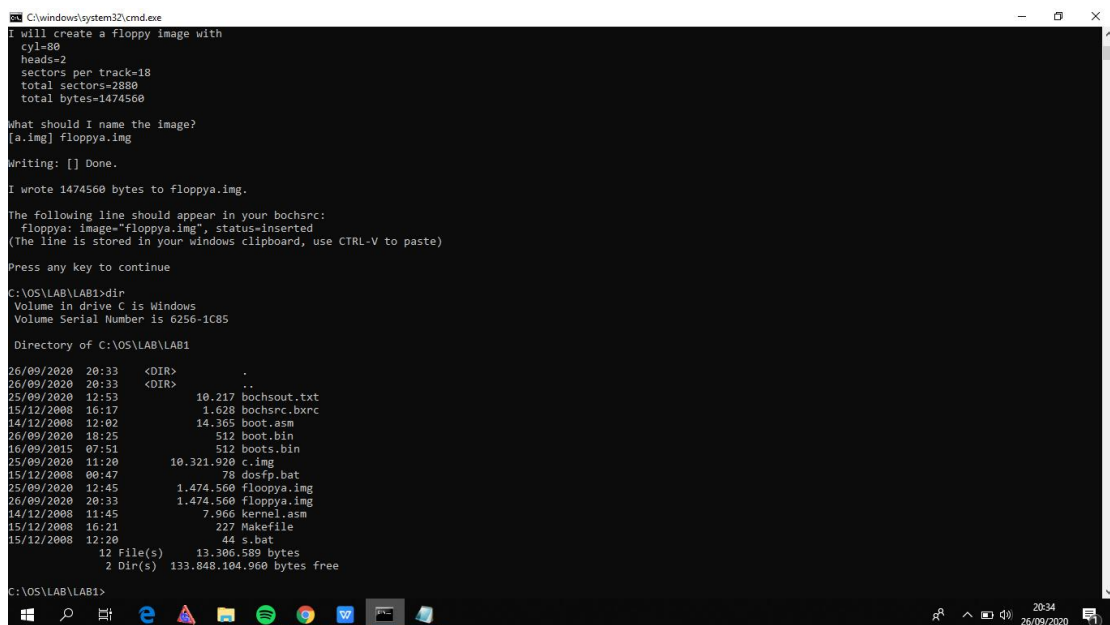
Choose the size of floppy disk image to create, in megabytes.
Please type 0.16, 0.18, 0.32, 0.36, 0.72, 1.2, 1.44, 1.68, 1.72, or 2.88.
[1.44] 1.44
I will create a floppy image with
cyl=80
heads=2
sectors per track=18
total sectors=2880
total bytes=1474560

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

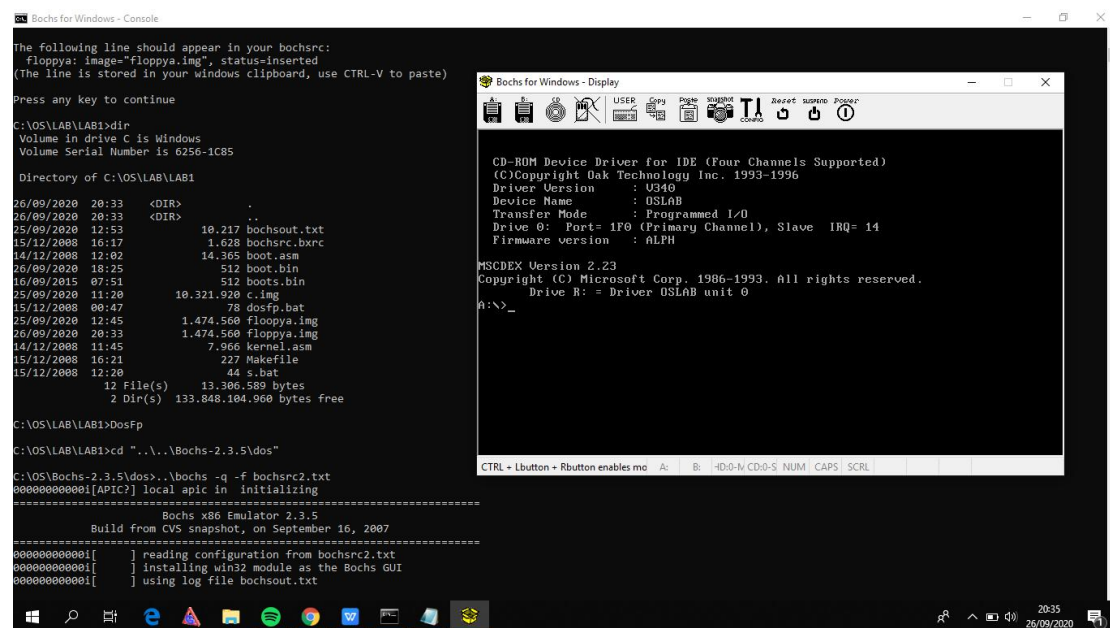
## 11. Terakhir memberi nama file, ketikan **'floppya.img'** dan <ENTER>



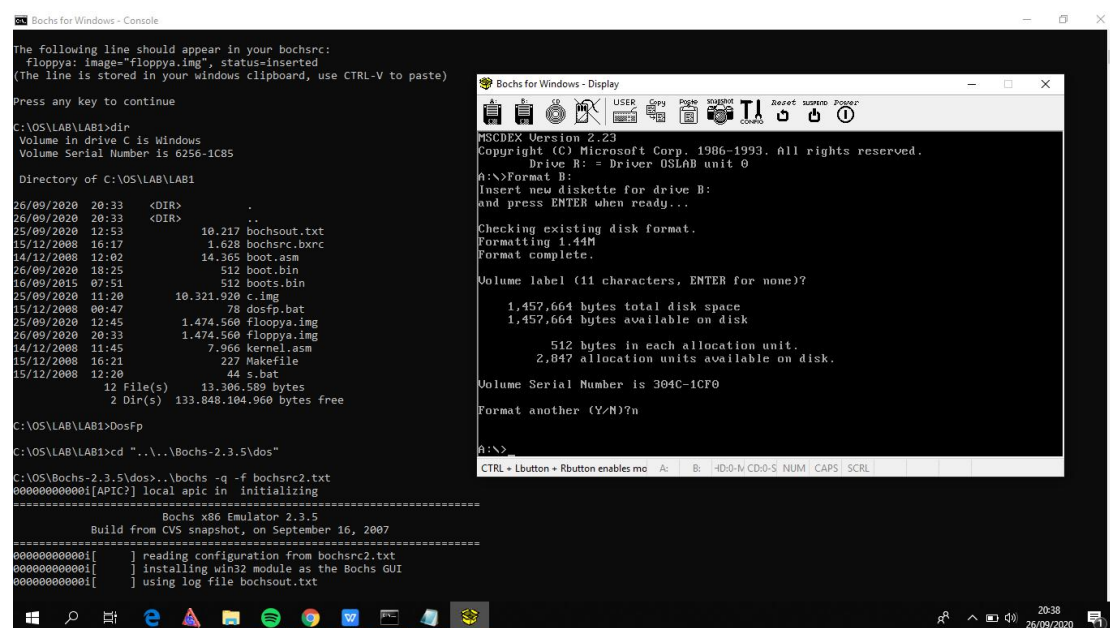
## 12. Ketik perintah **'dir'** untuk memastikan keberadaan file image tersebut



### 13. Ketik perintah 'DosFp' untuk menjalankan PC-Simulator dari 'Command Prompt'

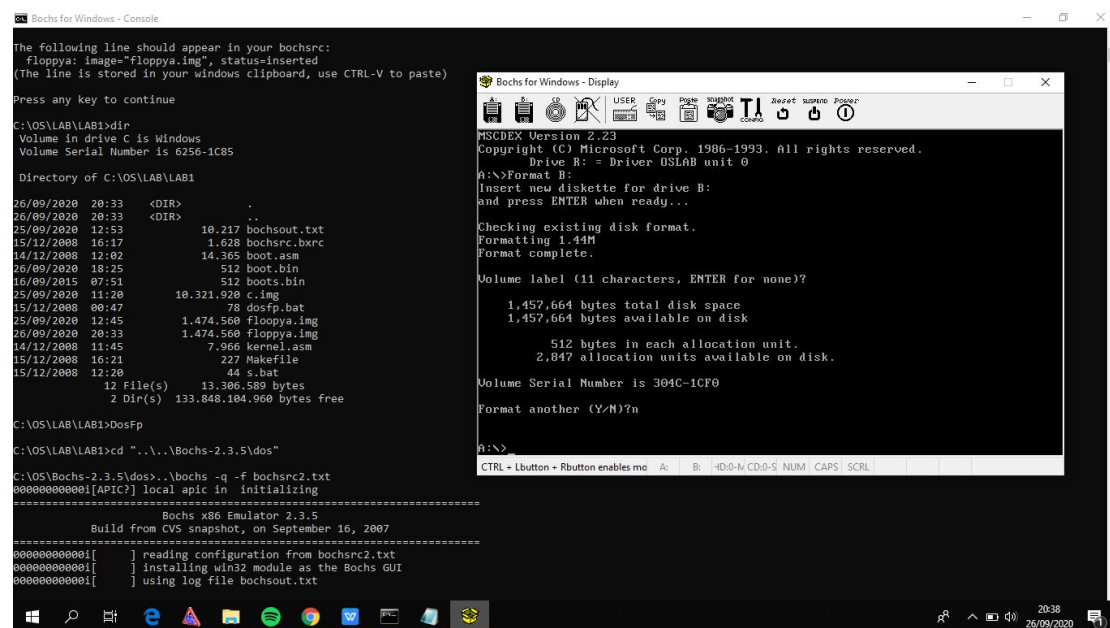


### 14. Selanjutnya dari prompt 'A:>' ketikan 'Format B:' kemudian tekan <ENTER> (2x)

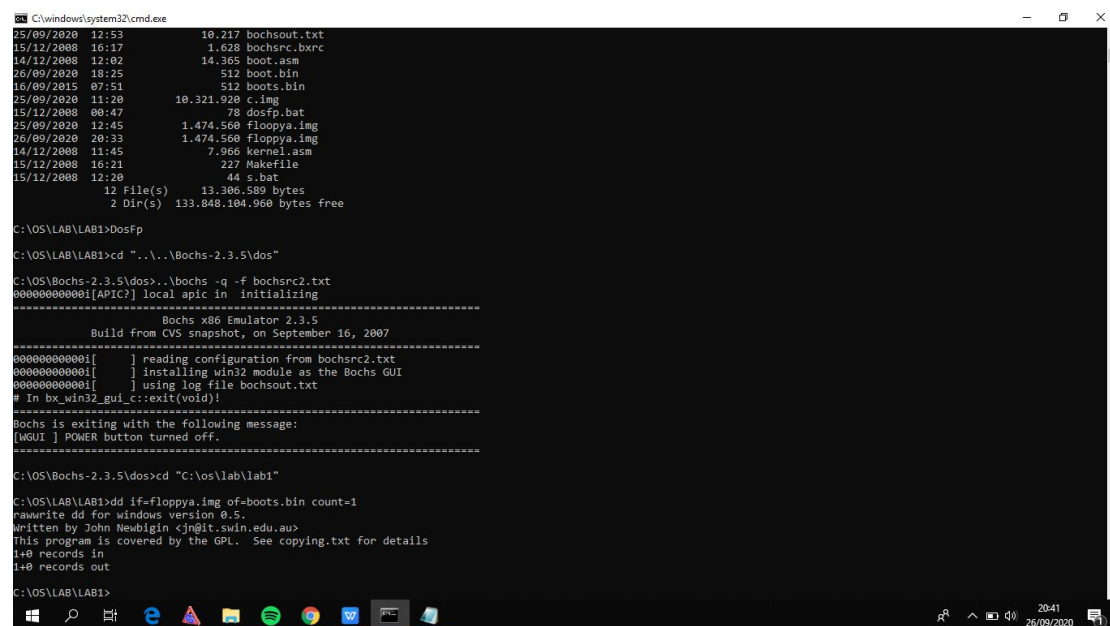




## 15. Tutup kembali PC-Simulator dengan klik pada ‘tombol power’



## 16. Ketik ‘dd if=floppya.img of=boots.bin count=1’ dari Command Prompt untuk mencopy 512 byte data bootsektor ke dalam sebuah file terpisah



## 17. ketikan 'tdump boots.bin' kemudian <ENTER>

```
C:\windows\system32\cmd.exe
Written by John Newbigin <jn@it.swin.edu.au>
This program is covered by the GPL. See copying.txt for details
140 records in
140 records out

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 29 F0 1C 4C 30 4E 4F 20 4E 41 .....).KMO 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 00 FC F3 A4 06 U..{..N.....
000060: 1F BD 00 7C C6 45 FE 0F 38 4E 24 7D 20 88 C1 99 ...|.E..BNS} ...
000070: 58 7E E1 83 EB 3A 66 A1 1C 7C 66 3B 07 8A 57 FC .....f..|f..M.
000080: 75 06 00 CA 02 80 56 02 80 C3 10 73 ED 33 C9 FE u.....V.....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 8F 00 07 E8 28 01 72 3E 38 2D 74 17 08 N.a....(r>B-t.
0000D0: B1 00 BE D8 7D F3 A6 61 74 3D 4E 74 00 83 C7 20 .....).at=Net...
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 B8 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 88 7D 1A 8D 45 FE ^f.....}.E.
000120: 8A 4E 00 F7 E1 03 46 FC 13 56 FE B1 04 E8 C2 00 .M....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: 88 46 18 A2 26 05 96 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 88 F4 8A 56 24 CD 13 ...u..B...V$.
000170: 61 61 72 0A 40 75 01 42 03 5E 00 49 75 77 C3 03 aar.@u.B.^Iuw..
000180: 18 01 27 00 0A 40 6E 76 61 6C 69 64 20 73 79 73 ...Invalid sys
000190: 74 65 6D 20 64 69 73 68 FF 00 0A 44 69 73 68 20 tem disk...Disk
0001A0: 49 2F 4F 20 65 72 72 6F 72 FF 00 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 00 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 50 53 7F 01 SYMSDOS SYS..
0001F0: 00 41 B8 00 07 60 66 6A 00 E9 3B FF 00 00 55 AA .A...^fj...;..U.

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

## 18. Ketik perintah 'type s.bat' untuk melihat isi file 's.bat' kemudian <ENTER>

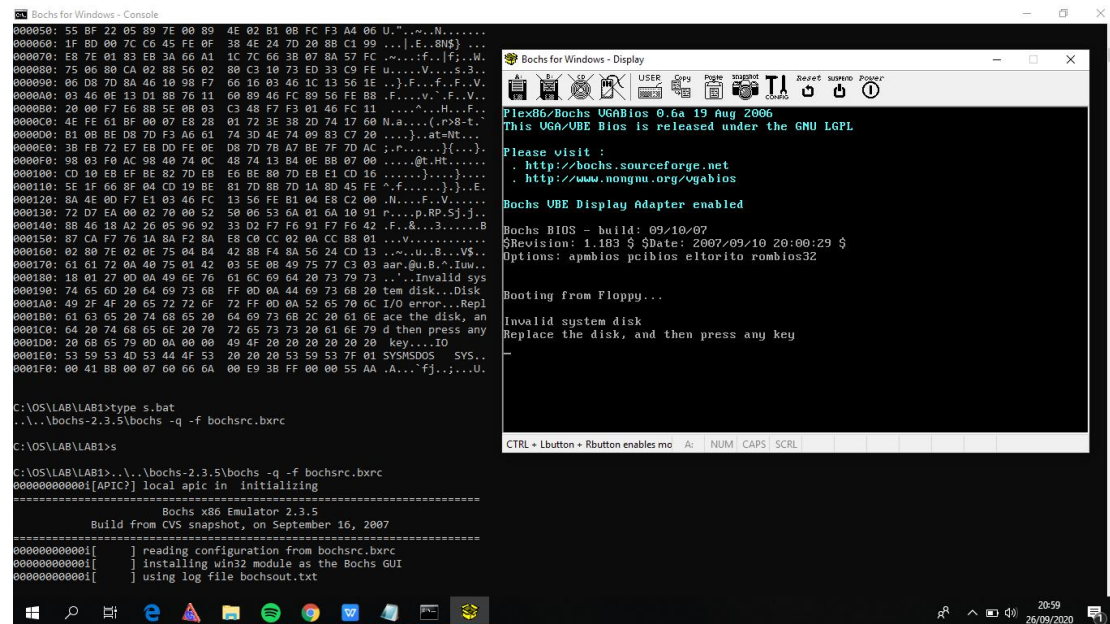
```
C:\windows\system32\cmd.exe
Turbo Dump Version 5.0.16.12 Copyright (c) 1988, 2000 Inprise Corporation
Display of File BOOTS.BIN

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 29 D1 1C 6B 1E 4E 4F 20 4E 41 .....).KMO 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 00 FC F3 A4 06 U..{..N.....
000060: 1F BD 00 7C C6 45 FE 0F 38 4E 24 7D 20 88 C1 99 ...|.E..BNS} ...
000070: 58 7E E1 83 EB 3A 66 A1 1C 7C 66 3B 07 8A 57 FC .....f..|f..M.
000080: 75 06 00 CA 02 80 56 02 80 C3 10 73 ED 33 C9 FE u.....V.....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 8F 00 07 E8 28 01 72 3E 38 2D 74 17 08 N.a....(r>B-t.
0000D0: B1 00 BE D8 7D F3 A6 61 74 3D 4E 74 00 83 C7 20 .....).at=Net...
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 B8 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 88 7D 1A 8D 45 FE ^f.....}.E.
000120: 8A 4E 00 F7 E1 03 46 FC 13 56 FE B1 04 E8 C2 00 .M....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: 88 46 18 A2 26 05 96 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 88 F4 8A 56 24 CD 13 ...u..B...V$.
000170: 61 61 72 0A 40 75 01 42 03 5E 00 49 75 77 C3 03 aar.@u.B.^Iuw..
000180: 18 01 27 00 0A 40 6E 76 61 6C 69 64 20 73 79 73 ...Invalid sys
000190: 74 65 6D 20 64 69 73 68 FF 00 0A 44 69 73 68 20 tem disk...Disk
0001A0: 49 2F 4F 20 65 72 72 6F 72 FF 00 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 00 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 50 53 7F 01 SYMSDOS SYS..
0001F0: 00 41 B8 00 07 60 66 6A 00 E9 3B FF 00 00 55 AA .A...^fj...;..U.

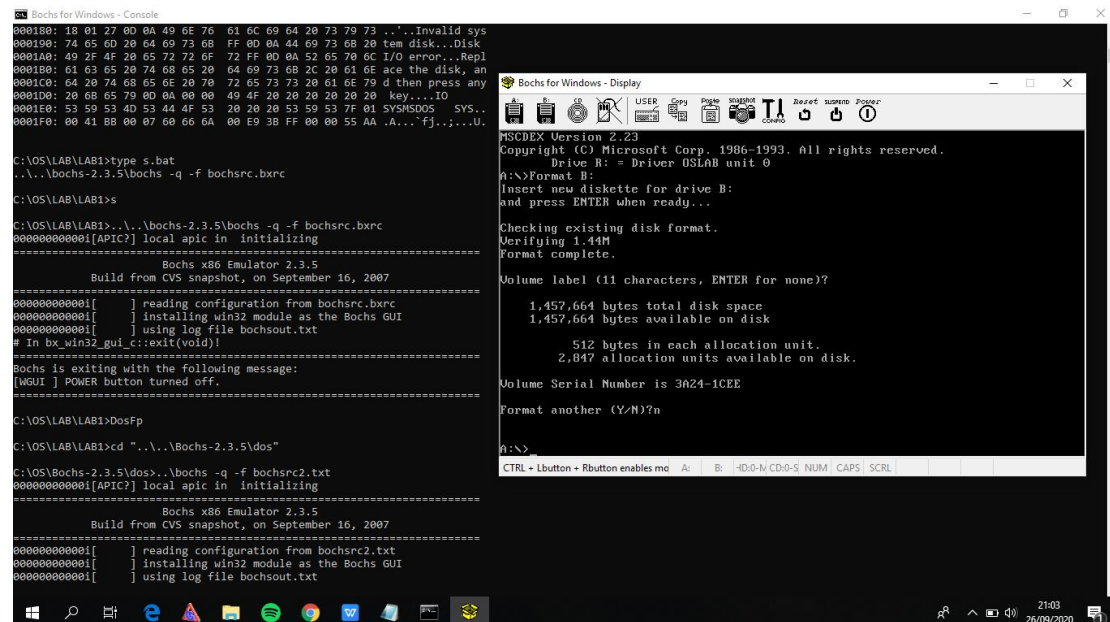
C:\OS\LAB\LAB1>type s.bat
...\\bochs-2.3.5\bochs -q -f bochsrtc.bxrc

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

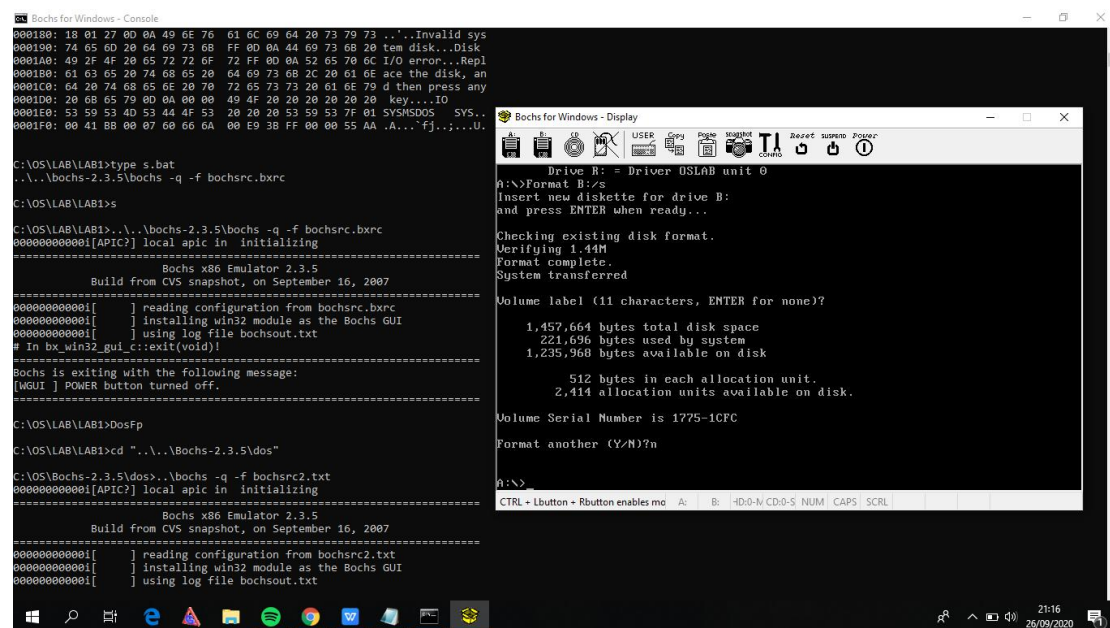
19. Selanjutnya masukan perintah **'s'** , kemudian akan ditampilkan windows **'Bochs for windows – display'** yang sedang melakukan proses **'booting'**. akan tetapi tidak berhasil karena tidak menemukan diskboot



20. Panggil **'DosFp'** . Pada windows **'Bochs'** kemudian masukan perintah **'A:>format B:/S'** kemudian **<ENTER>**



## 21. Matikan PC-Simulator ‘tombol Power’, kemudian ketik ‘S’ dan <ENTER>



```
Bochs for Windows - Console
000100: 18 01 27 00 0A 09 0E 70 01 6C 69 64 20 73 79 73 ...Invalid sys
000100: 74 65 60 20 64 60 73 68 FF 00 0A 44 60 73 68 20 tem disk...Disk
0001A0: 49 2F 4F 20 65 72 72 6F 72 FF 00 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 00 0A 00 00 49 4F 20 20 20 20 20 key....IO
0001E0: 53 59 53 40 53 44 4F 53 20 20 20 53 59 53 7F 01 SYSHSD05 SYS...
0001F0: 00 41 88 00 07 60 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 bochsrc.bxrc
C:\OS\LAB\LAB1>s
C:\OS\LAB\LAB1>...\\bochs-2.3.5\bochs -q -f bochsrc.bxrc
0000000000[APIC?] local apic in initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
0000000000[ ] reading configuration from bochsrc.bxrc
0000000000[ ] installing win32 module as the Bochs GUI
0000000000[ ] using log file bochsout.txt
# In bx_win32_gui.c:exit(void)!
=====
Bochs is exiting with the following message:
[MGUI ] POWER button turned off.
=====
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 bochsrc2.txt
0000000000[APIC?] local apic in initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
0000000000[ ] reading configuration from bochsrc2.txt
0000000000[ ] installing win32 module as the Bochs GUI
0000000000[ ] using log file bochsout.txt

Bochs for Windows - Display
Drive R: = Driver OSLAB unit 0
A:\>Format B: /s
Insert new diskette for drive B:
and press ENTER when ready...

Checking existing disk format.
Verifying 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 1775-1CFC

Format another (Y/N)?n
A:\>
CTRL + Lbutton + Rbutton enables me A: B: HD-0-N CD-0-S NUM CAPS SCRL
```



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.

- Code-code ASCII

Decimal	Hexadecimal	Binary	Character	Description
32	20	00100000	Spasi	Space
33	21	00100001	!	Exclamation mark
34	22	00100010	"	Double quote
35	23	00100011	#	Number
36	24	00100100	\$	Dollar
37	25	00100101	%	Percent
38	26	00100110	&	Ampersand
39	27	00100111	'	Single quote
40	28	00101000	(	Left parenthesis
41	29	00101001	)	Right parenthesis
42	2A	00101010	*	Asterisk
43	2B	00101011	+	Plus
44	2C	00101100	,	comma
45	2D	00101101	-	Minus
46	2E	00101110	.	Period
47	2F	00101111	/	Slash
48	30	00110000	0	Zero
49	31	00110001	1	One
50	32	00110010	2	Two
51	33	00110011	3	Three
52	34	00110100	4	Four
53	35	00110101	5	Five
54	36	00110110	6	Six
55	37	00110111	7	Seven
56	38	00111000	8	Eight
57	39	00111001	9	Nine

58	3A	00111010	:	Colon
59	3B	00111011	;	Semicolon
60	3C	00111100	<	Less than
61	3D	00111101	=	Equality sign
62	3E	00111110	>	Greather then
63	3F	00111111	?	Question mark
64	40	01000000	@	At sign
65	41	01000001	A	
66	42	01000010	B	
67	43	01000011	C	
68	44	01000100	D	
69	45	01000101	E	
70	46	01000110	F	
71	47	01000111	G	
72	48	01001000	H	
73	49	01001001	I	
74	4A	01001010	J	
75	4B	01001011	K	
76	4C	01001100	L	
77	4D	01001101	M	
78	4E	01001110	N	
79	4F	01001111	O	
80	50	01010000	P	
81	51	01010001	Q	
82	52	01010010	R	
83	53	01010011	S	
84	54	01010100	T	
85	55	01010101	U	
86	56	01010110	V	
87	57	01010111	W	
88	58	01011000	X	
89	59	01011001	Y	

90	5A	01011010	Z	
91	5B	01011011	[	Left square bracket
92	5C	01011100	\	Backslash
93	5D	01011101	]	Right square bracket
94	5E	01011110	^	Caret/circumflex
95	5F	01011111	_	Underscore
96	60	01100000	`	Grave/accent
97	61	01100001	a	
98	62	01100010	b	
99	63	01100011	c	
100	64	01100100	d	
101	65	01100101	e	
102	66	01100110	f	
103	67	01100111	g	
104	68	01101000	h	
105	69	01101001	i	
106	6A	01101010	j	
107	6B	01101011	k	
108	6C	01101100	l	
109	6D	01101101	m	
110	6E	01101110	n	
111	6F	01101111	o	
112	70	01110000	p	
113	71	01110010	q	
114	72	01110011	r	
115	73	01110100	s	
116	74	01110101	t	
117	75	01110111	u	
118	76	01111000	v	
119	77	01111001	w	
120	78	01111010	x	
121	79	01111001	y	

122	7A	01111010	z	
123	7B	01111011	{	Left curly bracket
124	7C	01111100		Vertical bar
125	7D	01111101	}	Right curly bracket
126	7E	01111110	~	Tilde
127	7F	01111111	DEL	Delete

2. 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? .dos seg .else .elseif .endif .endw .err .err1 .err2 .errb.errdef .errdif .errdifi .erre .erridn .erri dni .errnb .errndef .errnz .exit .fardata .fardata? .if .lall .lfcond .list .listall .listif .listma cro.listmacroall .model .no87 .nocref .nolist .nolistif .nolistmacro .radix .repeat .sall .s eq .sfcond .stack.startup .tfcond .type .until .untilcxz .while .xall .xcref .xlist.



## 1. 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).

## 2. 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.

### 3. 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.

### 4. Operasi matematika

**ADD** : add. Menjumlahkan dua buah register. sintaks: ADD {tujuan}, {sumber} operasi yang terjadi: tujuan = tujuan + 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: tujuan = tujuan + sumber + 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.

## 5. 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.

## 6. 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

## 7. 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?
<b>JA</b>	jump if above	$CF = 0 \wedge ZF = 0$	unsigned	lompat bila op 1 > op 2	ya
<b>JNBE</b>	jump if not below or equal				
<b>JB</b>	jump if below	$CF = 1 \wedge ZF = 0$	unsigned	lompat bila op 1 < op 2	ya
<b>JNAE</b>	jump if not above or equal				
<b>JAE</b>	jump if above or equal	$CF = 0 \vee ZF = 1$	unsigned	lompat bila op 1 $\geq$ op 2	ya
<b>JNB</b>	jump if not below				
<b>JBE</b>	jump if below or equal	$CF = 1 \vee ZF = 1$	unsigned	lompat bila op 1 $\leq$ op 2	ya
<b>JNA</b>	jump if not above				
<b>JG</b>	jump if greater	$OF = 0 \wedge ZF = 0$	signed	lompat bila op 1 > op 2	ya
<b>JNLE</b>	jump if not less or equal				
<b>JGE</b>	jump if greater or equal	$OF = 0 \vee ZF = 1$	signed	lompat bila op 1 $\geq$ op 2	ya
<b>JNL</b>	jump if not less than				
<b>JL</b>	jump if less than	$OF = 1 \wedge ZF = 0$	signed	lompat bila op 1 < op 2	ya
<b>JNGE</b>	jump if not greater or equal				
<b>JLE</b>	jump if less or equal	$OF = 1 \vee ZF = 1$	signed	lompat bila op 1 $\leq$ op 2	ya
<b>JNG</b>	jump if not greater				
<b>JE</b>	jump if equal	$ZF = 1$	keduanya	lompat bila op 1 = op 2	ya
<b>JZ</b>	jump if zero	$ZF = 1$	keduanya	lompat bila op 1 = op 2	ya
<b>JNE</b>	jump if not equal	$ZF = 0$	keduanya	lompat bila op 1 $\neq$ op 2	ya
<b>JNZ</b>	jump if not zero	$ZF = 0$	keduanya	lompat bila op 1 $\neq$ op 2	ya
<b>JC</b>	jump if carry	$CF = 1$	N/A	lompat bila carry flag = 1	tidak
<b>JNC</b>	jump if not carry	$CF = 0$	N/A	lompat bila carry flag = 0	tidak
<b>JP</b>	jump on parity	$PF = 1$	N/A	lompat bila parity flag = 1	tidak selalu
<b>JPE</b>	jump on parity even			lompat bila bilangan genap	

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

## 8. 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 perantaranya. 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 *membackup* 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*

## 9. 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.

## 10. 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 program 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. **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.*