

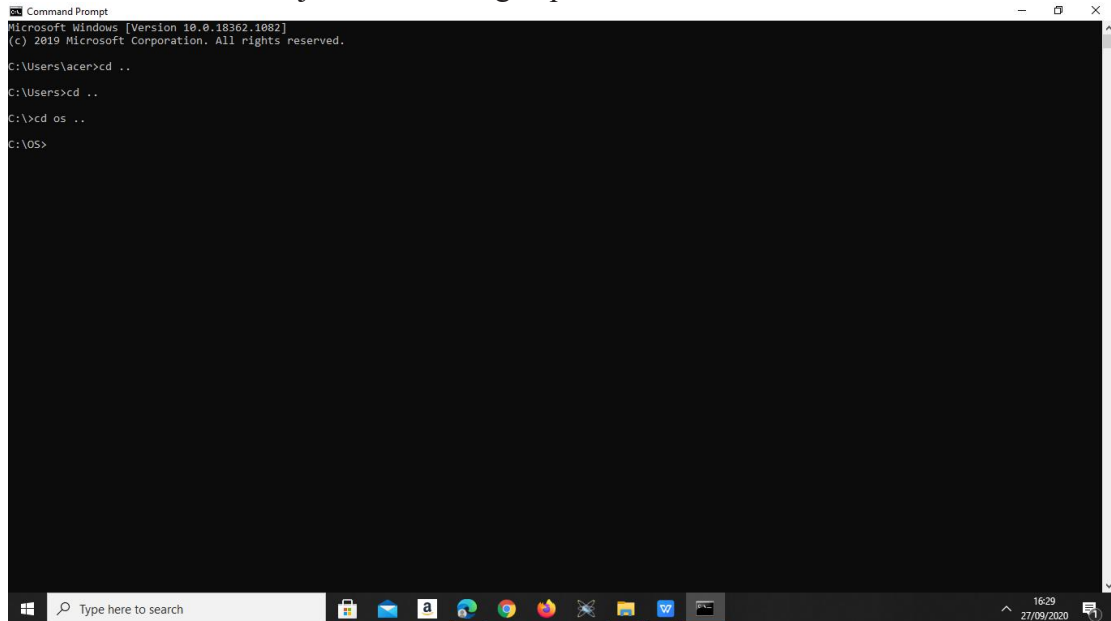
LAPORAN PRAKTIKUM
SISTEM OPERASI
LANGKAH KERJA



NAMA : NILA DWI RAHMAWATI
NIM : L 200190254
KELAS : G

FAKULTAS KOMUNIKASI DAN INFORMATIKA
UNIVERSITAS MUHAMMADIYAH SURAKARTA

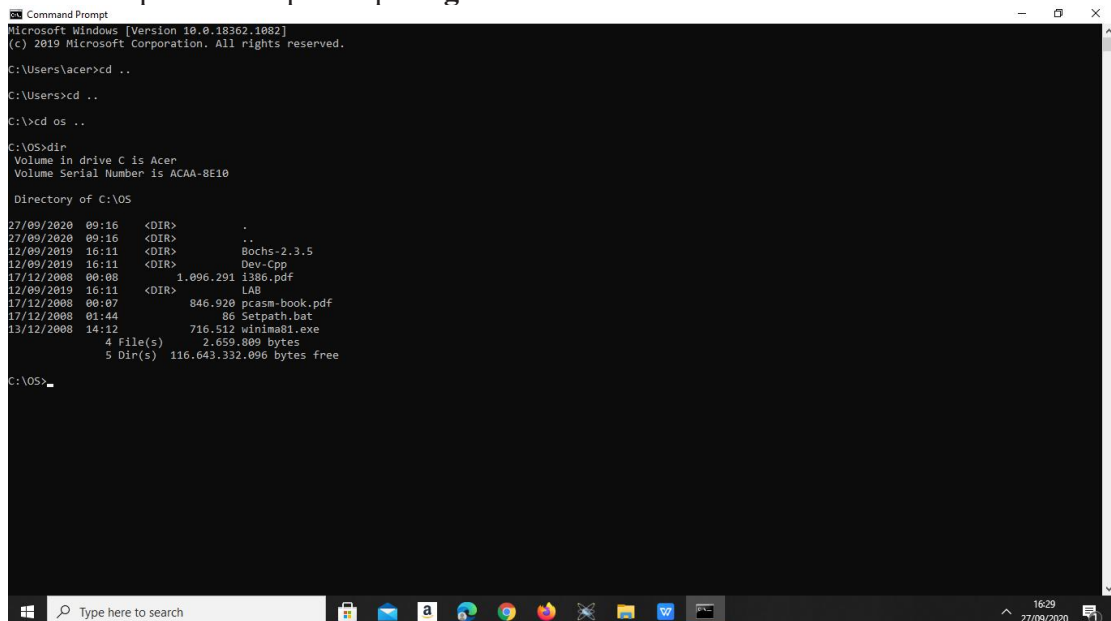
1. Jalankan program command prompt atau cmd.
2. Masuk ke direktori kerja 'C:\OS', dengan perintah 'cd os' <ENTER>.



```
Command Prompt
Microsoft Windows [Version 10.0.18362.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\acer>cd ..
C:\Users>cd ..
C:\>cd os ..
C:\OS>
```

3. Masukkan perintah dir, untuk melihat isi direktori di dalam folder tersebut. Akan muncul seperti di tampilan pada **gambar**



```
Command Prompt
Microsoft Windows [Version 10.0.18362.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

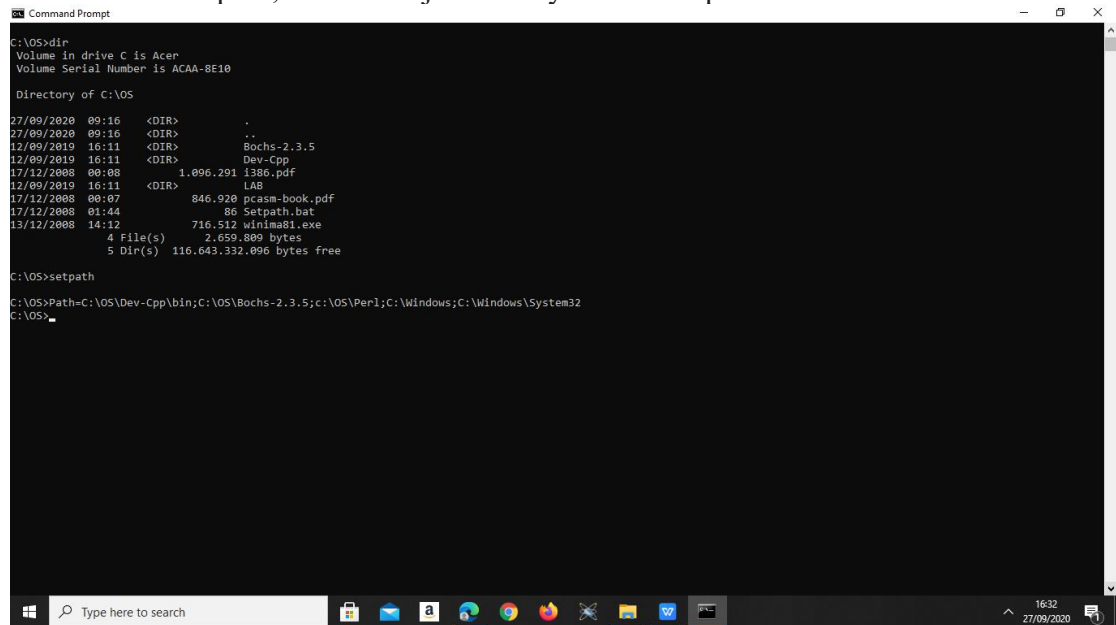
C:\Users\acer>cd ..
C:\Users>cd ..
C:\>cd os ..
C:\OS>dir
Volume in drive C is Acer
Volume Serial Number is ACAA-8E10

Directory of C:\OS

27/09/2020  09:16    <DIR>        .
27/09/2020  09:16    <DIR>        ..
12/09/2019  16:11    <DIR>        Bochs-2.3.5
12/09/2019  16:11    <DIR>        Dev-Cpp
17/12/2008  00:08    <DIR>        1.096.291 i386.pdf
12/09/2019  16:11    <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 winima01.exe
               4 File(s)          2.659.809 bytes
               5 Dir(s)  116.643.332.096 bytes free

C:\OS>
```

4. Jalankan file setpath, untuk menjalankannya ketik 'setpath' tekan <ENTER>



The screenshot shows a Windows Command Prompt window titled 'Command Prompt'. The user has entered the command 'dir' at the prompt 'C:\OS>'. The output displays the directory contents of 'C:\OS', including files like 'i386.pdf', 'pcasm-book.pdf', 'Setpath.bat', and 'winima81.exe'. The user then enters the command 'setpath' at the prompt 'C:\OS>'. The output shows the path 'C:\OS\Path=C:\OS\Dev-Cpp\bin;C:\OS\Bochs-2.3.5;c:\OS\Perl;C:\Windows;C:\Windows\System32' and the prompt returns to 'C:\OS>'.

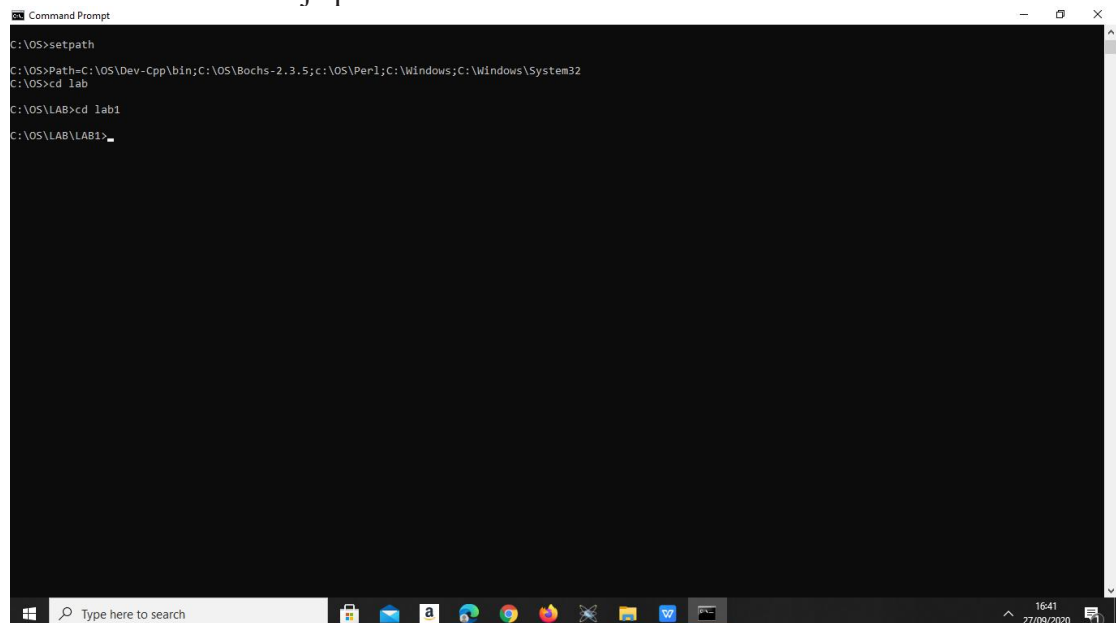
```
C:\OS>dir
Volume in drive C is Acer
Volume Serial Number is ACAA-8E10

Directory of C:\OS

27/09/2020  09:16    <DIR>        .
27/09/2020  09:16    <DIR>        ..
12/09/2019  16:11    <DIR>        Bochs-2.3.5
12/09/2019  16:11    <DIR>        Dev-Cpp
17/12/2008  00:08           1,096,291 i386.pdf
12/09/2019  16:11    <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)        116,643,332 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\System32
C:\OS>
```

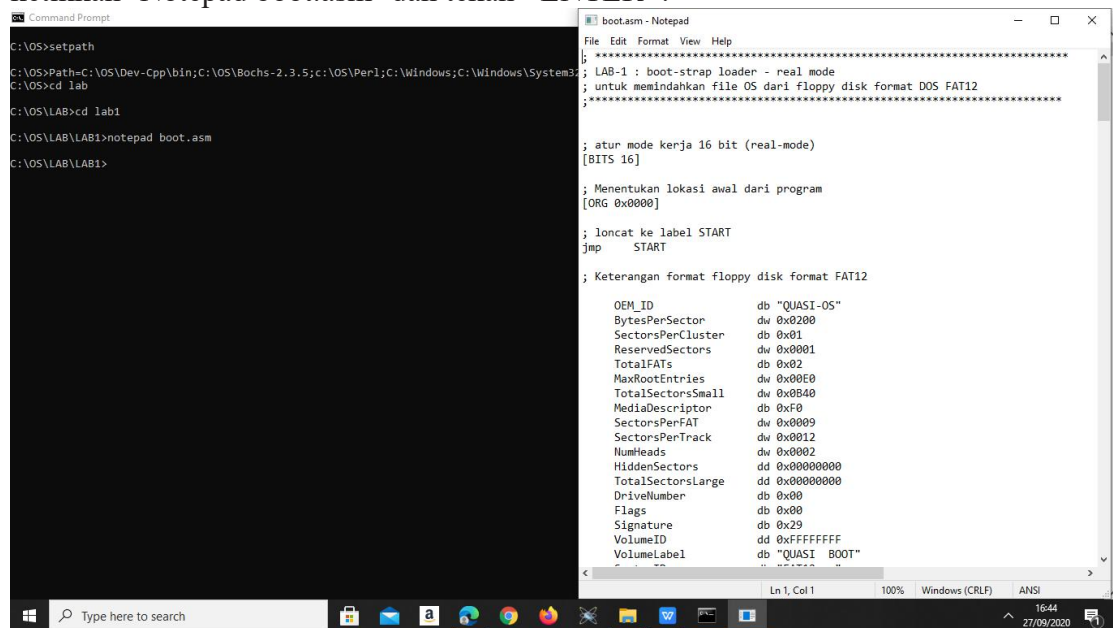
5. Masuk ke direktori kerja pada 'C:\OS\LAB\LAB1'.



The screenshot shows a Windows Command Prompt window titled 'Command Prompt'. The user has entered the command 'cd lab' at the prompt 'C:\OS>'. The output shows the current directory as 'C:\OS\LAB'. The user then enters the command 'cd lab1' at the prompt 'C:\OS\LAB>'. The output shows the current directory as 'C:\OS\LAB\LAB1' and the prompt returns to 'C:\OS\LAB\LAB1>'.

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

6. Membuka file tersebut dengan perintah berikut, dari 'COMMAND PROMPT', ketikkan 'Notepad boot.asm' dan tekan <ENTER>.



The screenshot shows two windows. On the left is the Command Prompt with the following commands: `C:\OS>setpath`, `C:\OS>path=C:\OS\Dev-Cpp\bin;C:\OS\Bochs-2.3.5;c:\OS\Perl;c:\Windows\System32`, `C:\OS>cd lab`, `C:\OS\LAB>cd lab1`, `C:\OS\LAB\LAB1>notepad boot.asm`, and `C:\OS\LAB\LAB1>`. On the right is the Notepad window titled 'boot.asm - Notepad' showing the content of the file. The content includes comments in Indonesian, assembly instructions for setting real mode, jumping to the START label, and a table of floppy disk format parameters for FAT12.

```
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]

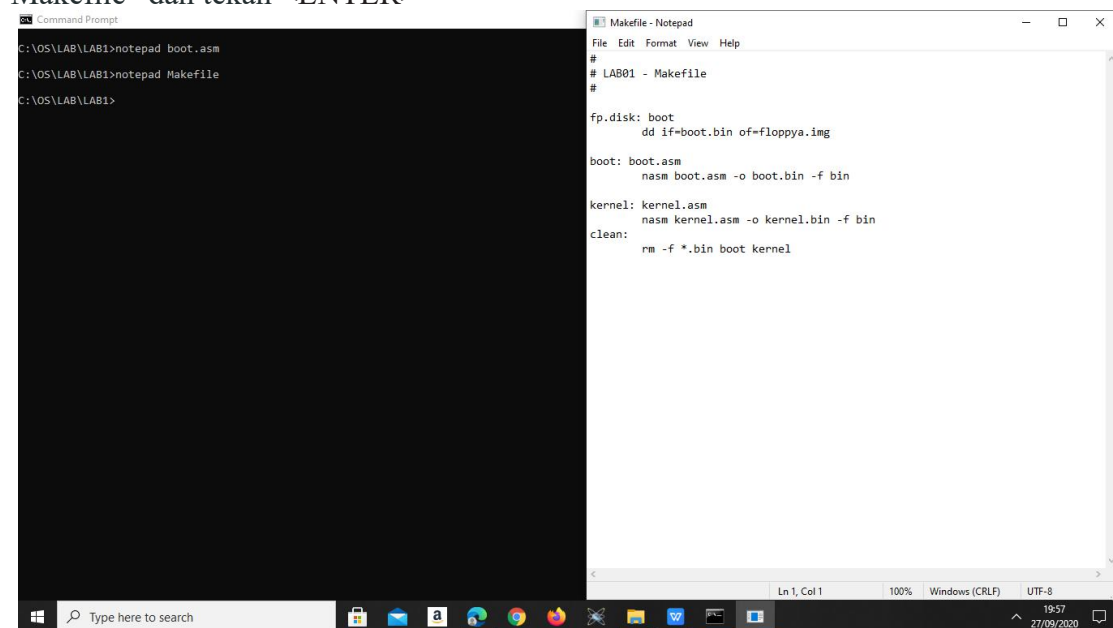
; Menentukan lokasi awal dari program
[ORG 0x0000]

; 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 0x0B40
MediaDescriptor db 0xF0
SectorsPerFAT dw 0x0009
SectorsPerTrack dw 0x0012
NumHeads       dw 0x0002
HiddenSectors  dd 0x00000000
TotalSectorsLarge dd 0x00000000
DriveNumber    db 0x00
Flags          db 0x00
Signature      db 0x29
VolumeID       dd 0xFFFFFFFF
VolumeLabel    db "QUASI BOOT"
```

7. Ketik "Notepad M" lalu tekan tombol "Tab" maka akan muncul "Notepad Makefile" dan tekan <ENTER>



The screenshot shows two windows. On the left is the Command Prompt with the following commands: `C:\OS\LAB\LAB1>notepad boot.asm`, `C:\OS\LAB\LAB1>notepad Makefile`, and `C:\OS\LAB\LAB1>`. On the right is the Notepad window titled 'Makefile - Notepad' showing the content of the Makefile. The content includes a comment, a target for the boot disk, and rules for building the boot and kernel binaries.

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

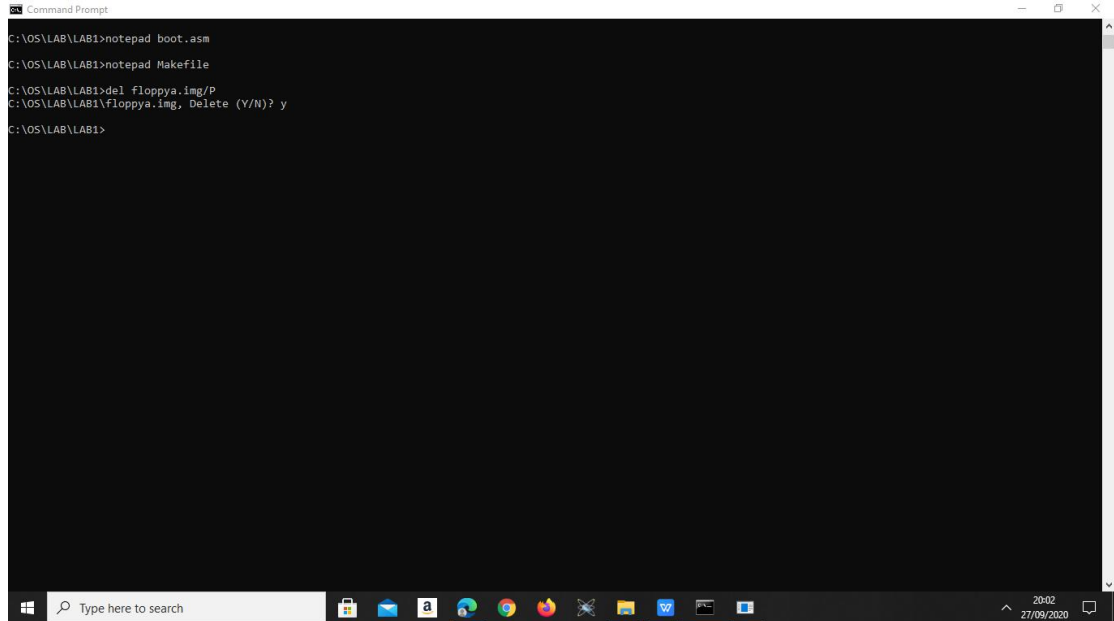
fp.disk: boot
    dd if=boot.bin of=floppya.img

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

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

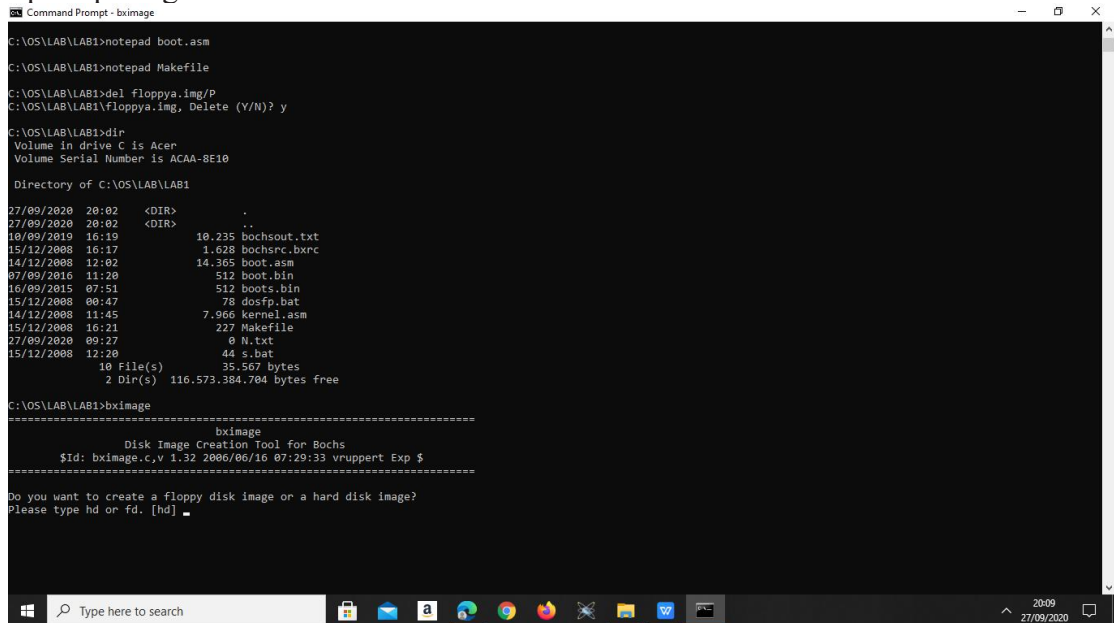
clean:
    rm -f *.bin boot kernel
```

8. Ketik 'del floppy.img /P' <ENTER> dan tambahkan 'y' lalu <ENTER> lagi.



```
Command Prompt
C:\OS\LAB\LAB1>notepad boot.asm
C:\OS\LAB\LAB1>notepad Makefile
C:\OS\LAB\LAB1>del floppy.img/P
C:\OS\LAB\LAB1>floppy.img, Delete (Y/N)? y
C:\OS\LAB\LAB1>
```

9. Masukkan perintah 'dir'. Selanjutnya panggil 'bxiimage' sehingga ditampilkan seperti pada gambar



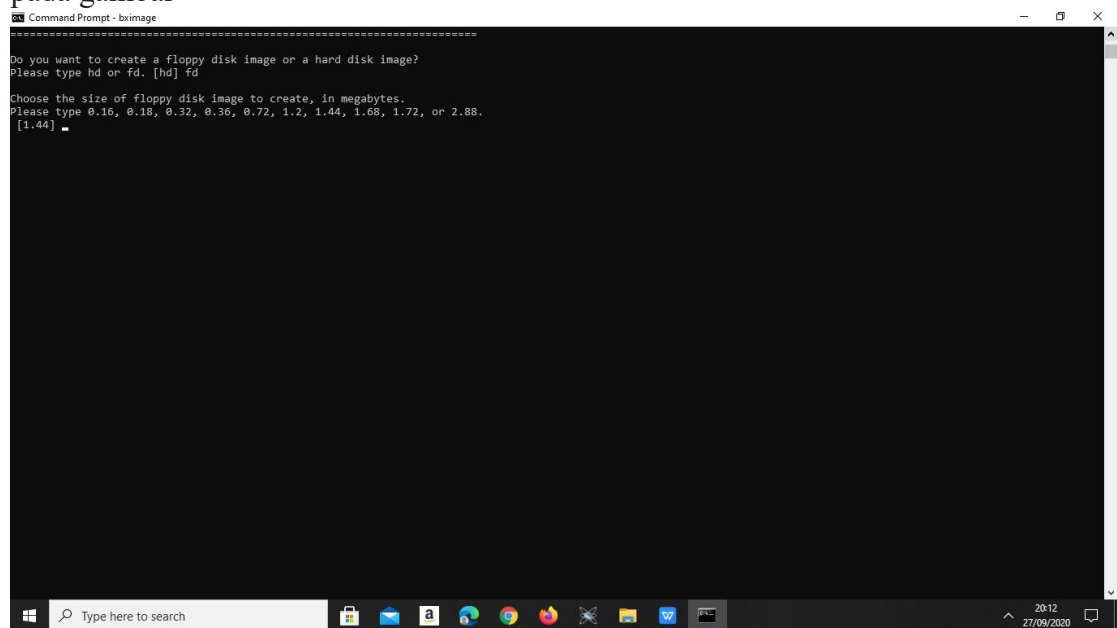
```
Command Prompt - bxiimage
C:\OS\LAB\LAB1>notepad boot.asm
C:\OS\LAB\LAB1>notepad Makefile
C:\OS\LAB\LAB1>del floppy.img/P
C:\OS\LAB\LAB1>floppy.img, Delete (Y/N)? y
C:\OS\LAB\LAB1>dir
Volume in drive C is Acer
Volume Serial Number is ACAA-8E10

Directory of C:\OS\LAB\LAB1

27/09/2020  20:02    <DIR>          .
27/09/2020  20:02    <DIR>          ..
10/09/2019  16:19             10,225 bochsout.txt
15/12/2000  16:17              1,628 bochssrc.bxrc
14/12/2008  12:02             14,365 boot.asm
07/09/2016  11:20              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
27/09/2020  09:27                0 N.txt
15/12/2008  12:20               44 s.bat
               10 File(s)      35,567 bytes
               2 Dir(s)      116,573,384,704 bytes free

C:\OS\LAB\LAB1>bxiimage
=====
                bxiimage
      Disk Image Creation Tool for Bochs
      $Id: bxiimage.c,v 1.32 2000/06/16 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] _
```

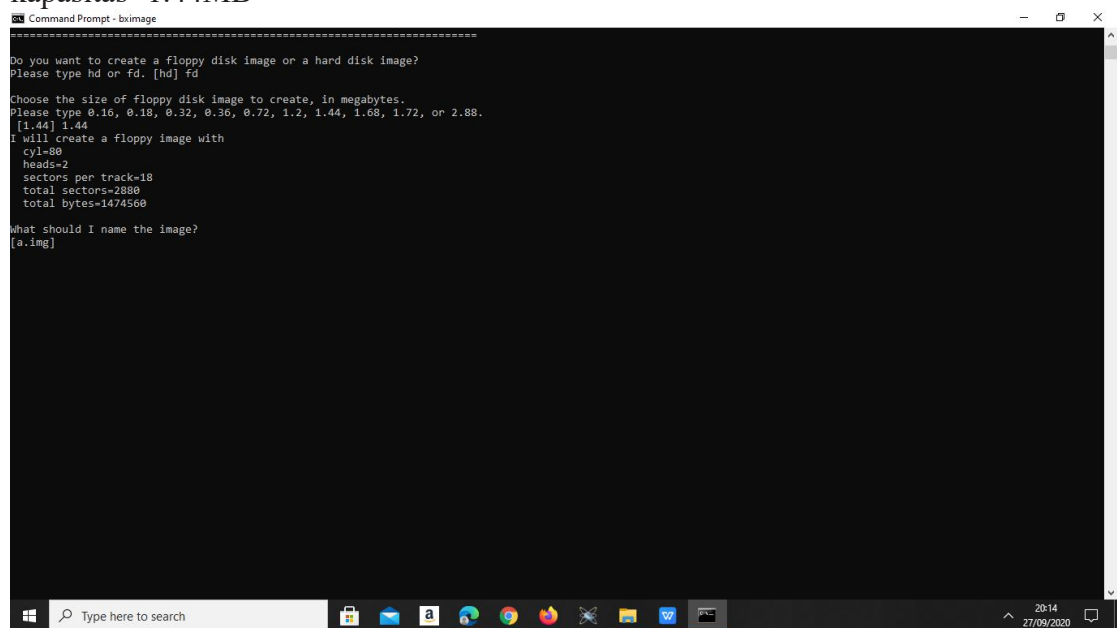
10. Ketikkan 'fd' dan tekan <ENTER> sehingga muncul tahapan berikutnya seperti pada gambar



```
Command Prompt - bimage
=====
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]
```

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



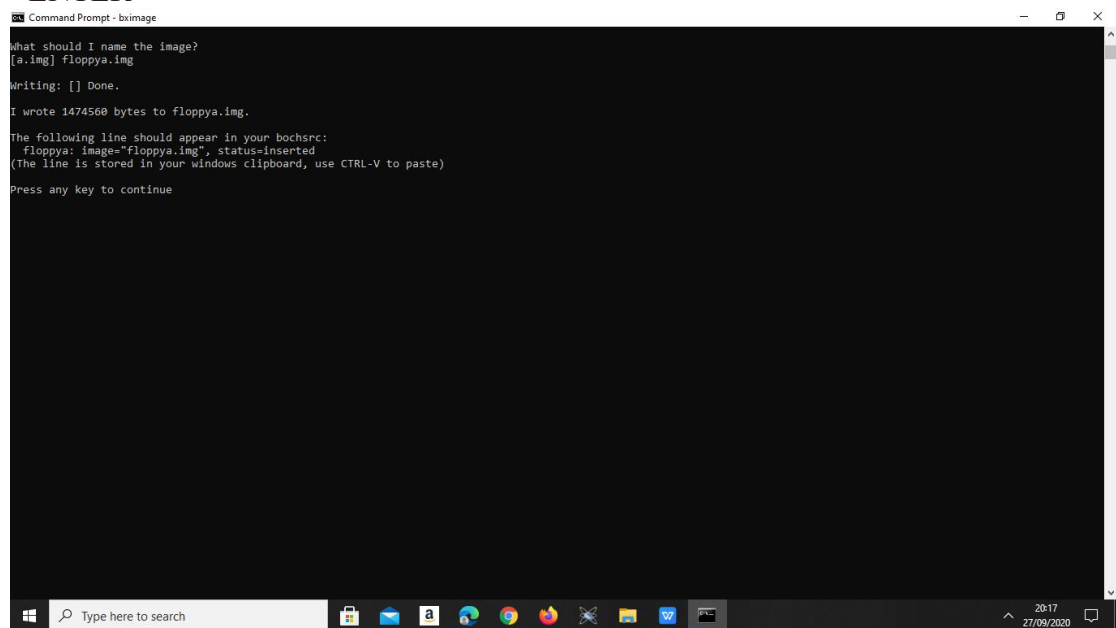
```
Command Prompt - bimage
=====
Do you want to create a floppy disk image or a hard disk image?
Please type hd or fd. [hd] fd

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]
```

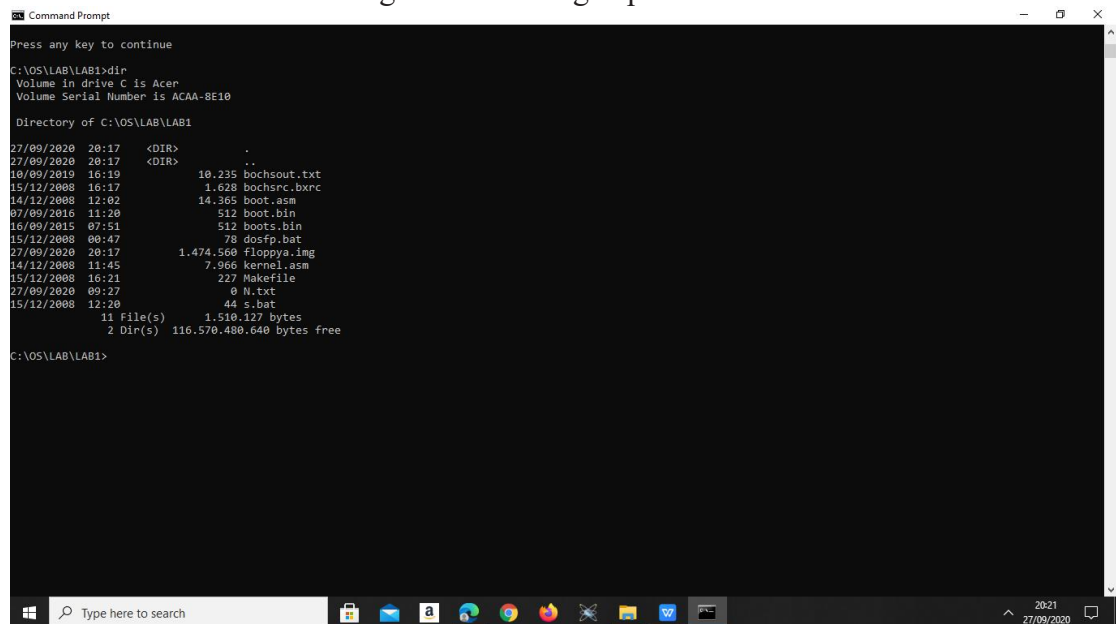
12. Ketika diminta untuk memberikan nama file, ketikan 'floppya.img' dan <ENTER>



```
Command Prompt - bimage

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 bochssrc:
  floppya: image="floppya.img", status=inserted
(The line is stored in your windows clipboard, use CTRL-V to paste)
Press any key to continue
```

13. Pastikan keberadaan file image tersebut dengan perintah 'dir'



```
Command Prompt

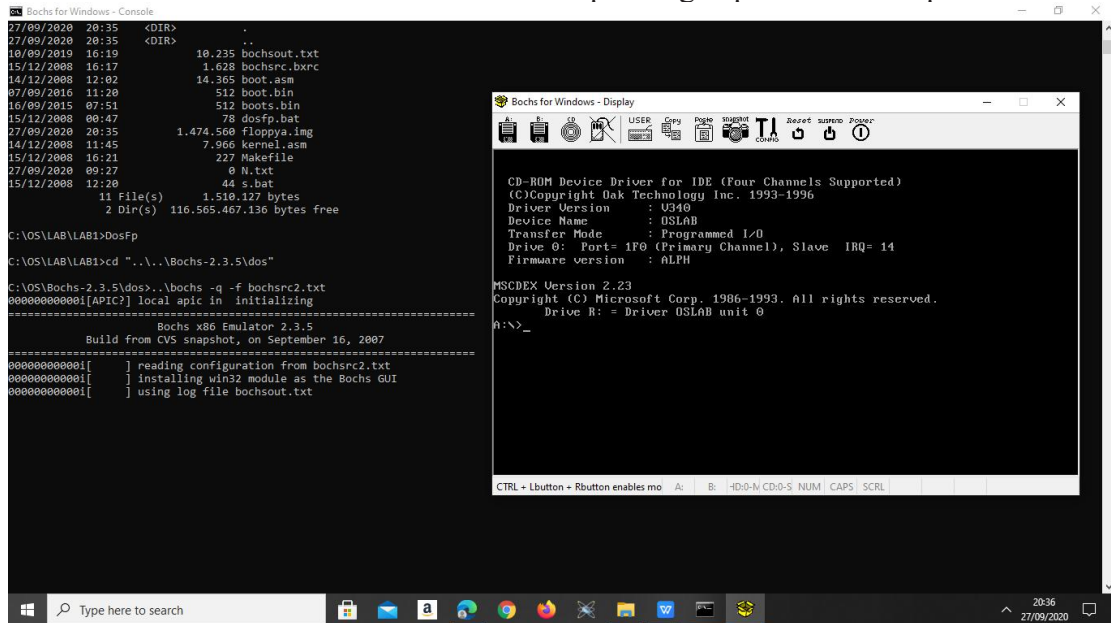
Press any key to continue
C:\OS\LAB\LAB1>dir
Volume in drive C is Acer
Volume Serial Number is ACAA-BE10

Directory of C:\OS\LAB\LAB1

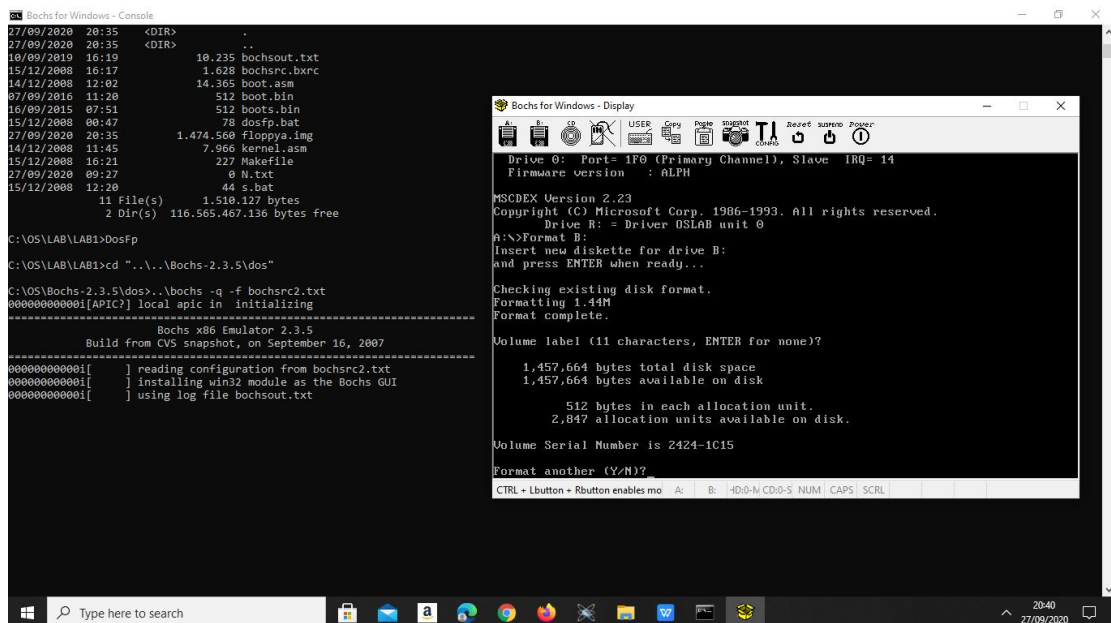
27/09/2020  20:17    <DIR>        .
27/09/2020  20:17    <DIR>        ..
10/09/2019  16:19             10.235 bochsout.txt
15/12/2008  16:17             1.628 bochssrc.bxrc
14/12/2008  12:02             14.365 boot.asm
07/09/2016  11:20              512 boot.bin
16/09/2015  07:51              512 boots.bin
15/12/2008  00:47              78 dosfp.bat
27/09/2020  20:17           1.474.560 floppya.img
14/12/2008  11:45             7.966 kernel.asm
15/12/2008  10:21             227 Makefile
27/09/2020  09:27               0 N.txt
15/12/2008  12:20              44 s.bat
               11 File(s)          1.510.127 bytes
               2 Dir(s)          116.570.480 bytes free

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

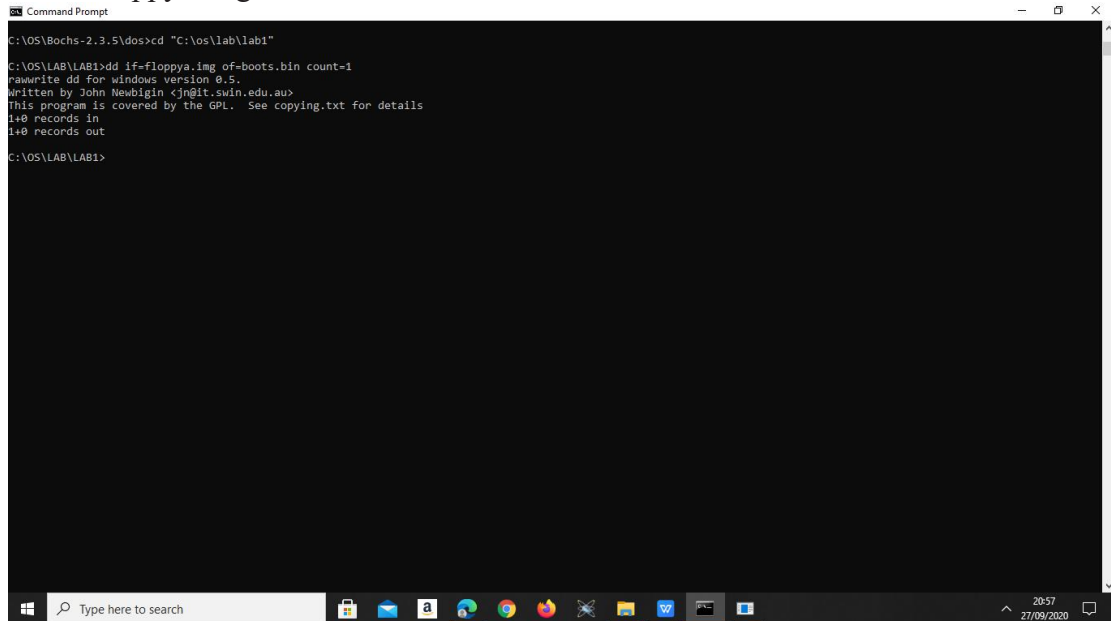
14. Jalankan PC-Simulator dari 'Command Prompt' dengan perintah 'DosFp'



15. Selanjutnya dari prompt 'A:>' ketikkan 'Format B:' <ENTER> [2x]

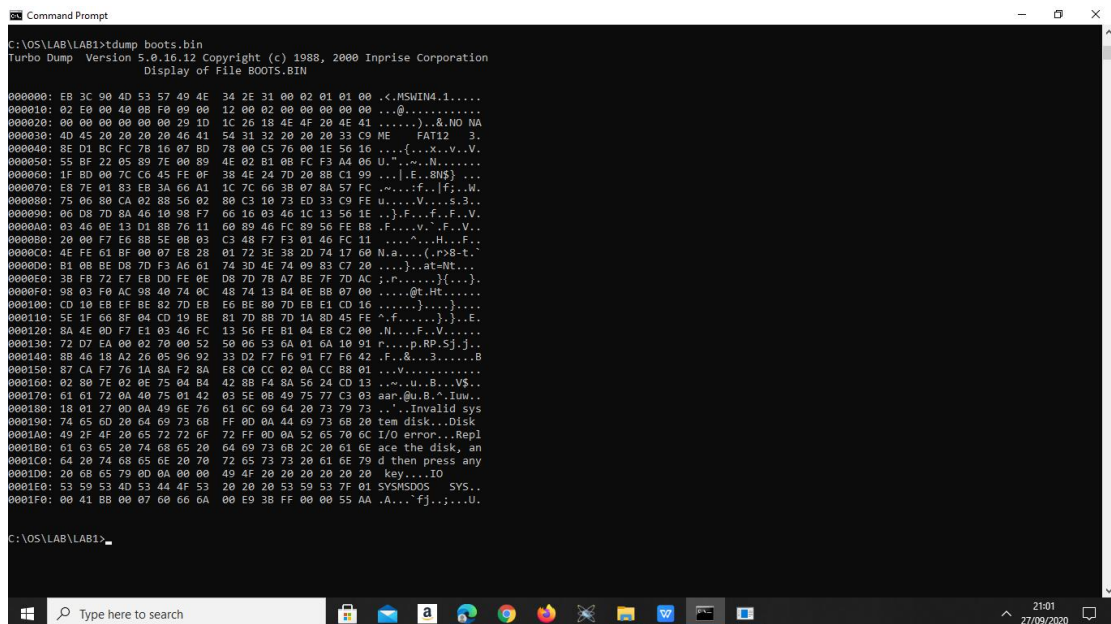


16. Tutup kembali PC-Simulator dengan klik pada tombol power. Copy 512 byte data bootsektor ke dalam sebuah file terpisah, caranya dari ‘Command Prompt’ ketik ‘dd if=floppya.img of=boots.bin count=1’



```
Command Prompt
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>
```

17. Ketikkan “tdump boots.bin” lalu <ENTER>



```
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 .<..MSMIN4.1....
000010: 02 E0 00 40 08 F0 09 00 12 00 02 00 00 00 00 00 ...@.....
000020: 00 00 00 00 00 29 1D 1C 26 18 4E 4F 2B 4E 41 .....&.ND NA
000030: 4D 45 20 20 20 20 46 41 54 31 32 20 20 20 33 C9 ME FAT12 3.
000040: BE 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..8M$} ...
000070: E8 7E 01 83 EB 3A 06 A1 1C 7C 60 38 07 8A 57 FC .~...f...|fj..W.
000080: 75 06 00 CA 02 08 56 02 00 C2 10 73 ED 33 C0 FE u.....V.....$3..
000090: 00 D8 7D 8A 46 10 98 F7 66 16 03 46 1C 13 56 1E .}.F...f..F..V..
0000A0: 03 46 0E 13 D1 8B 76 11 60 89 46 FC 89 56 FE B8 .F...v..F..V..
0000B0: 20 00 F7 E6 88 5E 08 03 C3 48 F7 F3 01 46 FC 11 .F...^...H...F..
0000C0: 4E FE 01 BF 00 07 E8 28 01 72 3E 38 2D 74 17 60 N.a....(>x8-t.'
0000D0: B1 0B BE D8 7D F3 A6 61 74 3D 4E 74 09 83 C7 20 ...}..@t..ht...
0000E0: 3B F8 72 E7 EB DD FE 0E D8 7D 7B A7 BE 7F 7D AC ;.F.....}{...}.
0000F0: 98 03 F0 AC 98 40 74 0C 48 74 13 B4 0E B8 07 00 ....@t..ht.....
000100: CD 10 EB EF BE 82 7D E8 E6 BE 80 7D E8 E1 CD 16 .....}.....}....
000110: 5E 1F 06 8F 04 CD 19 BE 81 7D 8B 7D 1A 8D 45 FE ^..f.....}.E..
000120: 8A 4E 0D F7 E1 03 46 FC 13 56 FE 01 04 E8 C2 00 ..n...F..V.....
000130: 72 D7 EA 00 02 70 00 52 50 00 53 6A 01 6A 10 91 P...p.RP.Sj..
000140: 88 46 18 A2 26 05 96 92 33 D2 F7 F6 91 F7 F6 42 .F..&...3.....B
000150: 87 CA F7 76 1A 8A F2 8A E8 C0 CC 02 0A CC B8 01 ...v.....
000160: 02 80 7E 02 0E 75 04 B4 42 8B F4 8A 56 24 CD 13 ...u..B...V$..
000170: 61 61 72 0A 40 75 01 42 03 5E 0B 49 75 77 C3 03 aar.@.B..Iuw..
000180: 18 01 27 00 0A 49 6E 76 61 6C 69 64 20 73 70 73 ...Invalid sys
000190: 74 65 6D 20 64 69 73 68 FF 0D 0A 44 69 73 68 20 tem disk...Disk
0001A0: 49 2F 4F 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 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 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 59 53 7F 01 SYSMSDOS SYS..
0001F0: 00 41 B8 00 07 00 66 6A 00 E9 3B FF 00 00 55 AA .A...`fj.;...U.
C:\OS\LAB\LAB1>
```

18. Untuk melihat isi file 's.bat' masukkan perintah 'type s.bat' lalu <ENTER>

```

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

C:\OS\LAB\LAB1>

```

19. Selanjutnya masukan perintah 's' <ENTER>, akan ditampilkan windows 'Bochs for windows – display' yang sedang melakukan proses 'booting' namun tidak berhasil karena tidak menemukan diskboot.

The screenshot displays a Windows 10 desktop environment. On the left, a terminal window titled "Bochs for Windows - Console" shows the execution of a batch file named s.bat. The commands executed are:

```
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
00000000001[APIC?] local apic in initializing
=====
          Bochs x86 Emulator 2.3.5
      Build from CVS snapshot, on September 16, 2007
=====
0000000000i[          ] reading configuration from bochsrc.bxrc
0000000000i[          ] installing win32 module as the Bochs GUI
0000000000i[          ] using log file bochsout.txt
```

On the right, a window titled "Bochs for Windows - Display" shows the Bochs BIOS boot screen. The text displayed is:

```

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/ugabios

Bochs UBE Display Adapter enabled

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

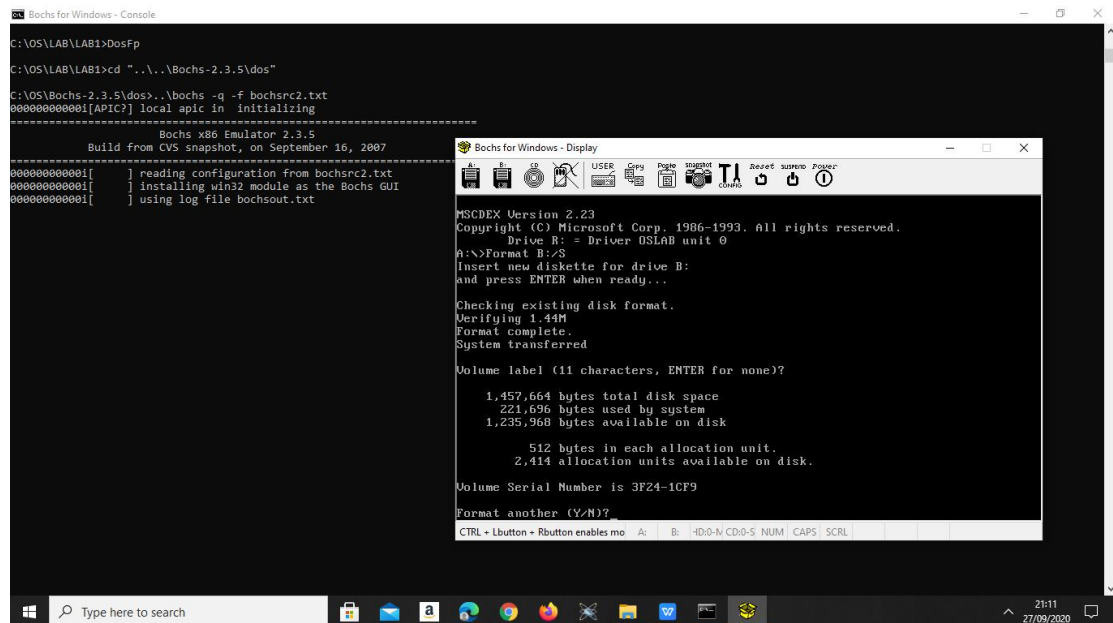
Booting from Floppy...

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

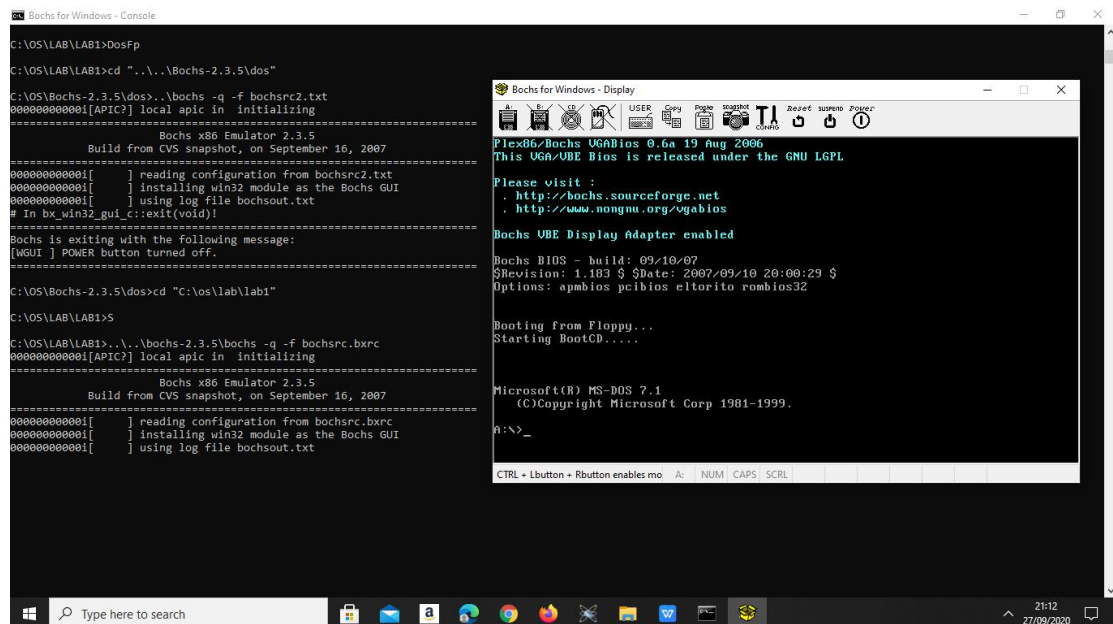
CTRL + Lbutton + Rbutton enables mo  AL  NUM  CAPS  SCRL
```

The taskbar at the bottom shows the Start button, a search bar, and several pinned application icons including File Explorer, Edge, and various utility programs. The system clock in the bottom right corner indicates the time is 21:07 on 27/09/2020.

20. Tekan tombol Power, ketikkan DosFp pada 'COMMAND PROMPT' <ENTER> lalu masukkan "A:>format B:/S" lalu <ENTER>



21. Matikan PC-Simulator (klik tombol Power Off), ketik 'S' <ENTER>.



LAPORAN PRAKTIKUM
SISTEM OPERASI
MODUL I
Pengenalan Sistem Pengembang OS



NAMA : NILA DWI RAHMAWATI
NIM : L 200190254
KELAS : G

FAKULTAS KOMUNIKASI DAN INFORMATIKA
UNIVERSITAS MUHAMMADIYAH SURAKARTA

TUGAS

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.

Jawab :

a. Pengertian ASCII

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

b. Tabel ASCII :

DEC	HEX	BIN	Symbol	Description
0	00	00000000	NUL	Null char
1	01	00000001	SOH	Start of Heading
2	02	00000010	STX	Start of Text
3	03	00000011	ETX	End of Text
4	04	00000100	EOT	End of Transmission
5	05	00000101	ENQ	Enquiry
6	06	00000110	ACK	Acknowledgment
7	07	00000111	BEL	Bell
8	08	00001000	BS	Back Space
9	09	00001001	HT	Horizontal Tab
10	0A	00001010	LF	Line Feed
11	0B	00001011	VT	Vertical Tab
12	0C	00001100	FF	Form Feed
13	0D	00001101	CR	Carriage Return
14	0E	00001110	SO	Shift Out / X-On
15	0F	00001111	SI	Shift In / X-Off
16	10	00010000	DLE	Data Line Escape
17	11	00010001	DC1	Device Control 1 (oft. XON)
18	12	00010010	DC2	Device Control 2
19	13	00010011	DC3	Device Control 3 (oft. XOFF)
20	14	00010100	DC4	Device Control 4
21	15	00010101	NAK	Negative

				Acknowledgement
22	16	00010110	SYN	Synchronous Idle
23	17	00010111	ETB	End of Transmit Block
24	18	00011000	CAN	Cancel
25	19	00011001	EM	End of Medium
26	1A	00011010	SUB	Substitute
27	1B	00011011	ESC	Escape
28	1C	00011100	FS	File Separator
29	1D	00011101	GS	Group Separator
30	1E	00011110	RS	Record Separator
31	1F	00011111	US	Unit Separator
32	20	00100000		Space
33	21	00100001	!	Exclamation mark
34	22	00100010	"	Double quotes (or speech marks)
35	23	00100011	#	Number
36	24	00100100	\$	Dollar
37	25	00100101	%	Per cent sign
38	26	00100110	&	Ampersand
39	27	00100111	'	Single quote
40	28	00101000	(Open parenthesis (or open bracket)
41	29	00101001)	Close parenthesis (or close bracket)
42	2A	00101010	*	Asterisk
43	2B	00101011	+	Plus
44	2C	00101100	,	Comma
45	2D	00101101	-	Hyphen
46	2E	00101110	.	Period, dot or full stop
47	2F	00101111	/	Slash or divide
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 (or open angled

				bracket)
61	3D	00111101	=	Equals
62	3E	00111110	>	Greater than (or close angled bracket)
63	3F	00111111	?	Question mark
64	40	01000000	@	At symbol
65	41	01000001	A	Uppercase A
66	42	01000010	B	Uppercase B
67	43	01000011	C	Uppercase C
68	44	01000100	D	Uppercase D
69	45	01000101	E	Uppercase E
70	46	01000110	F	Uppercase F
71	47	01000111	G	Uppercase G
72	48	01001000	H	Uppercase H
73	49	01001001	I	Uppercase I
74	4A	01001010	J	Uppercase J
75	4B	01001011	K	Uppercase K
76	4C	01001100	L	Uppercase L
77	4D	01001101	M	Uppercase M
78	4E	01001110	N	Uppercase N
79	4F	01001111	O	Uppercase O
80	50	01010000	P	Uppercase P
81	51	01010001	Q	Uppercase Q
82	52	01010010	R	Uppercase R
83	53	01010011	S	Uppercase S
84	54	01010100	T	Uppercase T
85	55	01010101	U	Uppercase U
86	56	01010110	V	Uppercase V
87	57	01010111	W	Uppercase W
88	58	01011000	X	Uppercase X
89	59	01011001	Y	Uppercase Y
90	5A	01011010	Z	Uppercase Z
91	5B	01011011	[Opening bracket
92	5C	01011100	\	Backslash
93	5D	01011101]	Closing bracket
94	5E	01011110	^	Caret - circumflex
95	5F	01011111	_	Underscore
96	60	01100000	`	Grave accent
97	61	01100001	a	Lowercase a
98	62	01100010	b	Lowercase b
99	63	01100011	c	Lowercase c
100	64	01100100	d	Lowercase d
101	65	01100101	e	Lowercase e

102	66	01100110	f	Lowercase f
103	67	01100111	g	Lowercase g
104	68	01101000	h	Lowercase h
105	69	01101001	i	Lowercase i
106	6A	01101010	j	Lowercase j
107	6B	01101011	k	Lowercase k
108	6C	01101100	l	Lowercase l
109	6D	01101101	m	Lowercase m
110	6E	01101110	n	Lowercase n
111	6F	01101111	o	Lowercase o
112	70	01110000	p	Lowercase p
113	71	01110001	q	Lowercase q
114	72	01110010	r	Lowercase r
115	73	01110011	s	Lowercase s
116	74	01110100	t	Lowercase t
117	75	01110101	u	Lowercase u
118	76	01110110	v	Lowercase v
119	77	01110111	w	Lowercase w
120	78	01111000	x	Lowercase x
121	79	01111001	y	Lowercase y
122	7A	01111010	z	Lowercase z
123	7B	01111011	{	Opening brace
124	7C	01111100		Vertical bar
125	7D	01111101	}	Closing brace
126	7E	01111110	~	Equivalency sign - tilde
127	7F	01111111		Delete

Sumber : <https://www.ascii-code.com/>

- 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'.

Jawab :

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: 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: tujuan = tujuan – 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: tujuan = tujuan – sumber – 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.

Tips: 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*.

❖ *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 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. Membakup 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