

PRAKTIKUM SISTEM OPERASI

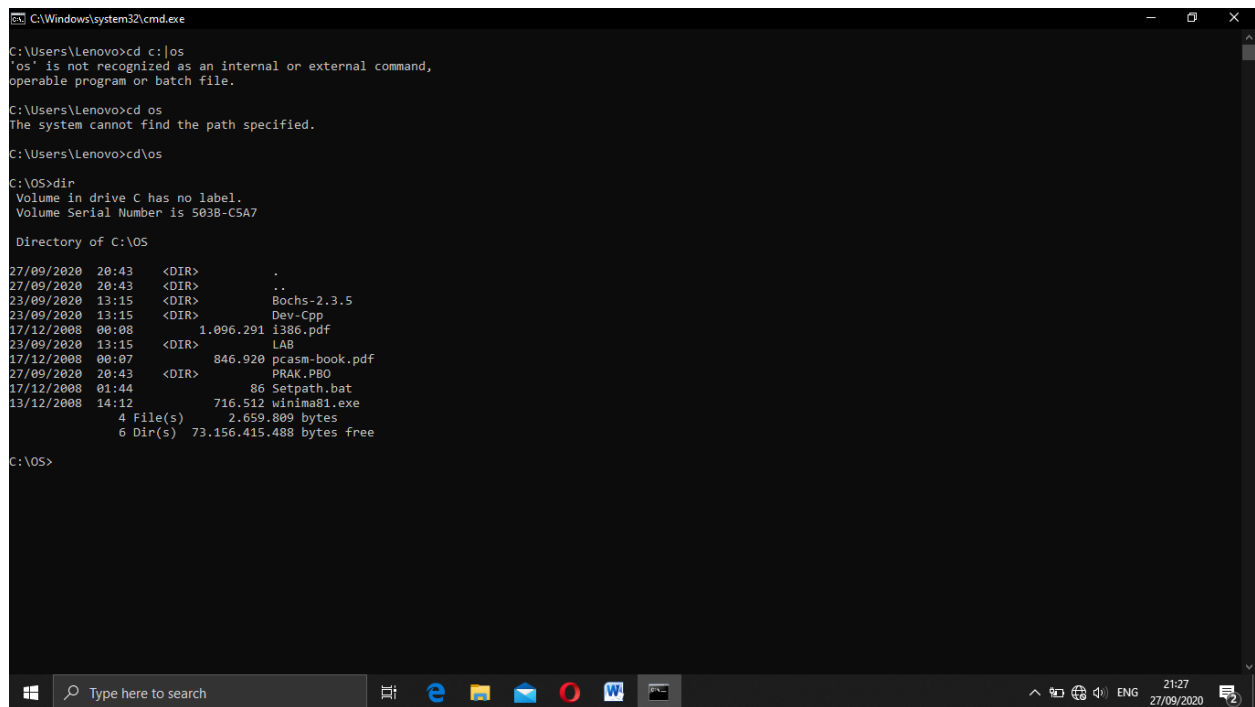
Modul 1: Pengenalan Sistem Pengembangan OS Dengan PC Simulator 'Bochs'

Nama : Zaimatul Ummah

NIM : L200190237

Kelas : Prak SO G

Berikut percobaan saya menggunakan cmd



```
C:\Windows\system32\cmd.exe

C:\Users\Lenovo>cd c:\os
'os' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Lenovo>cd os
The system cannot find the path specified.

C:\Users\Lenovo>cd \os
C:\OS>dir
Volume in drive C has no label.
Volume Serial Number is 5038-C5A7

Directory of C:\OS

27/09/2020  20:43    <DIR>          .
27/09/2020  20:43    <DIR>          ..
23/09/2020  13:15    <DIR>          Bochs-2.3.5
23/09/2020  13:15    <DIR>          Dev-Cpp
17/12/2008  00:08             1.096.291  i386.pdf
23/09/2020  13:15    <DIR>          LAB
17/12/2008  00:07             846.920  pcasm-book.pdf
27/09/2020  20:43    <DIR>          PRAK.PBO
17/12/2008  01:44             86 Setpath.bat
13/12/2008  14:12             716.512  winima81.exe
               4 File(s)      2.659.809 bytes
               6 Dir(s)  73.156.415.488 bytes free

C:\OS>
```

```
C:\Windows\system32\cmd.exe

C:\Users\Lenovo>cd c:\os
'os' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Lenovo>cd os
The system cannot find the path specified.

C:\Users\Lenovo>cd \os

C:\OS>dir
Volume in drive C has no label.
Volume Serial Number is 5038-C5A7

Directory of C:\OS

27/09/2020  20:43    <DIR>          .
27/09/2020  20:43    <DIR>          ..
23/09/2020  13:15    <DIR>          Bochs-2.3.5
23/09/2020  13:15    <DIR>          Dev-Cpp
17/12/2008  00:08    <DIR>          1.096.291  i386.pdf
23/09/2020  13:15    <DIR>          LAB
17/12/2008  00:07    <DIR>          846.920  pcasm-book.pdf
27/09/2020  20:43    <DIR>          PRAK.P80
17/12/2008  01:44    <DIR>          86  Setpath.bat
13/12/2008  14:12    <DIR>          716.512  winima81.exe
               4 File(s)      2.659.809 bytes
               6 Dir(s)   73.156.415.488 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>cd LAB

C:\OS\LAB>
```

```
C:\Windows\system32\cmd.exe

Directory of C:\OS

27/09/2020  20:43    <DIR>          .
27/09/2020  20:43    <DIR>          ..
23/09/2020  13:15    <DIR>          Bochs-2.3.5
23/09/2020  13:15    <DIR>          Dev-Cpp
17/12/2008  00:08    <DIR>          1.096.291  i386.pdf
23/09/2020  13:15    <DIR>          LAB
17/12/2008  00:07    <DIR>          846.920  pcasm-book.pdf
27/09/2020  20:43    <DIR>          PRAK.P80
17/12/2008  01:44    <DIR>          86  Setpath.bat
13/12/2008  14:12    <DIR>          716.512  winima81.exe
               4 File(s)      2.659.809 bytes
               6 Dir(s)   73.156.415.488 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>cd LAB

C:\OS\LAB>cd Lab1

C:\OS\LAB\LAB1>dir
Volume in drive C has no label.
Volume Serial Number is 5038-C5A7

Directory of C:\OS\LAB\LAB1

26/09/2020  21:28    <DIR>          .
26/09/2020  21:28    <DIR>          ..
10/09/2019  16:19    <DIR>          10.235  bochsout.txt
15/12/2008  16:17    <DIR>          1.628  bochsrc.bxrc
14/12/2008  12:02    <DIR>          14.365  boot.asm
26/09/2020  21:28    <DIR>          512  boot.bin
16/09/2015  07:51    <DIR>          512  boots.bin
15/12/2008  00:47    <DIR>          78  dosfp.bat
26/09/2020  21:33    <DIR>          1.474.560  floppy.img
14/12/2008  11:45    <DIR>          7.966  kernel.asm
15/12/2008  16:21    <DIR>          227  Makefile
15/12/2008  12:20    <DIR>          44  s.bat
               10 File(s)      1.510.127 bytes
               2 Dir(s)   73.155.747.840 bytes free

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

```
C:\Windows\system32\cmd.exe
13/12/2008 14:12 716,512 winima81.exe
4 File(s) 2,659,809 bytes
6 Dir(s) 73,156,415,488 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>cd LAB

C:\OS\LAB>cd Lab1

C:\OS\LAB\LAB1>dir
Volume in drive C has no label.
Volume Serial Number is 5038-C5A7

Directory of C:\OS\LAB\LAB1

26/09/2020 21:28 <DIR> .
26/09/2020 21:28 <DIR> ..
10/09/2019 16:19 10,235 bochsout.txt
15/12/2008 16:17 1,628 bochsrc.bxrc
14/12/2008 12:02 14,365 boot.asm
26/09/2020 21:28 512 boot.bin
16/09/2015 07:51 512 boots.bin
15/12/2008 00:47 78 dosfp.bat
26/09/2020 21:33 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
10 File(s) 1,510,127 bytes
2 Dir(s) 73,155,747,840 bytes free

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

C:\OS\LAB\LAB1>make fp.disk
nasm boot.asm -o boot.bin -f bin
dd if=boot.bin of=floppy.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>
```

```
C:\Windows\system32\cmd.exe
4 File(s) 2,659,809 bytes
6 Dir(s) 73,157,292,032 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>cd Lab

C:\OS\LAB>cd Lab1

C:\OS\LAB\LAB1>dir
Volume in drive C has no label.
Volume Serial Number is 5038-C5A7

Directory of C:\OS\LAB\LAB1

27/09/2020 21:32 <DIR> .
27/09/2020 21:32 <DIR> ..
10/09/2019 16:19 10,235 bochsout.txt
15/12/2008 16:17 1,628 bochsrc.bxrc
14/12/2008 12:02 14,365 boot.asm
27/09/2020 21:31 512 boot.bin
16/09/2015 07:51 512 boots.bin
27/09/2020 21:32 1,474,560 floppy.img
15/12/2008 00:47 78 dosfp.bat
27/09/2020 21:31 512 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) 1,510,639 bytes
2 Dir(s) 73,157,226,496 bytes free

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

C:\OS\LAB\LAB1>make fp.disk
nasm boot.asm -o boot.bin -f bin
dd if=boot.bin of=floppy.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>
```

```
C:\Windows\system32\cmd.exe - bximage
27/09/2020 21:31      512 boot.bin
16/09/2015 07:51      512 boots.bin
27/09/2020 21:32    1,474,560 floppy.img
15/12/2008 00:47       78 dosfp.bat
27/09/2020 21:31      512 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)    1,510,639 bytes
          2 Dir(s)    73,157,226,496 bytes free

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

C:\OS\LAB\LAB1>make fp.disk
nasm boot.asm -o boot.bin -f bin
dd if=boot.bin of=floppy.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>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]
```

```
C:\Windows\system32\cmd.exe
dd if=boot.bin of=floppy.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>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] floppy.img

The disk image 'floppy.img' already exists. Are you sure you want to replace it?
Please type yes or no. [no] yes

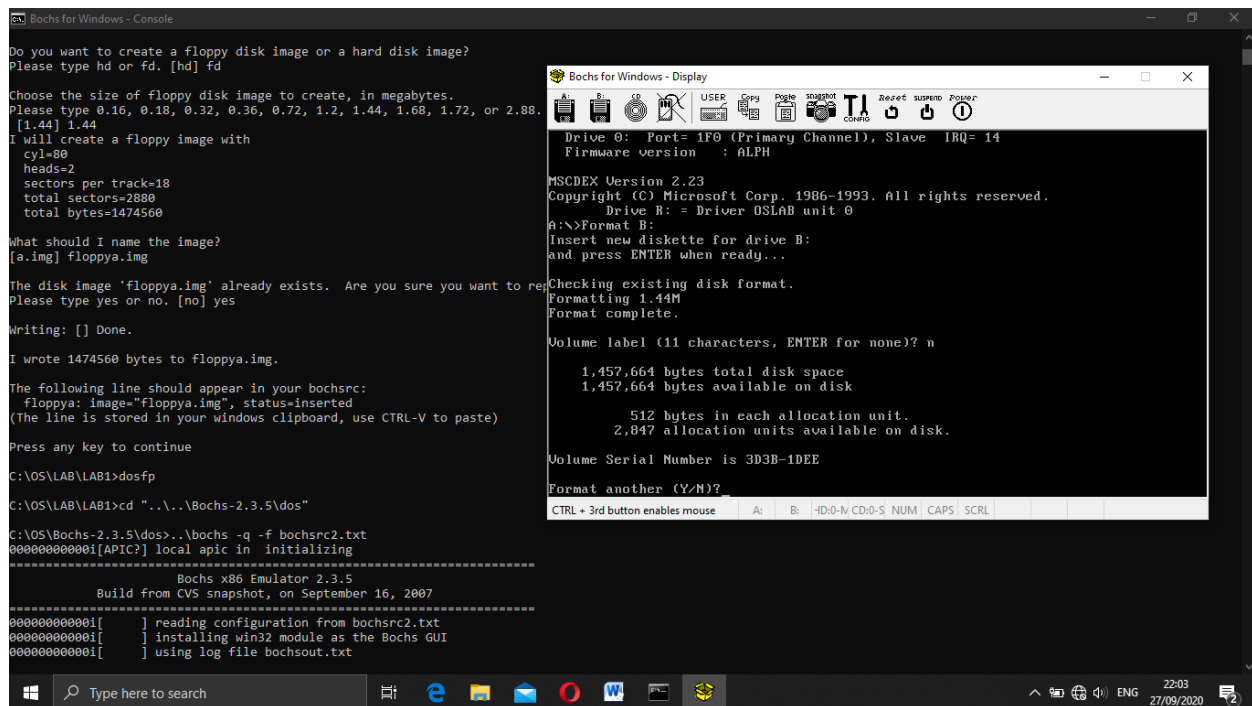
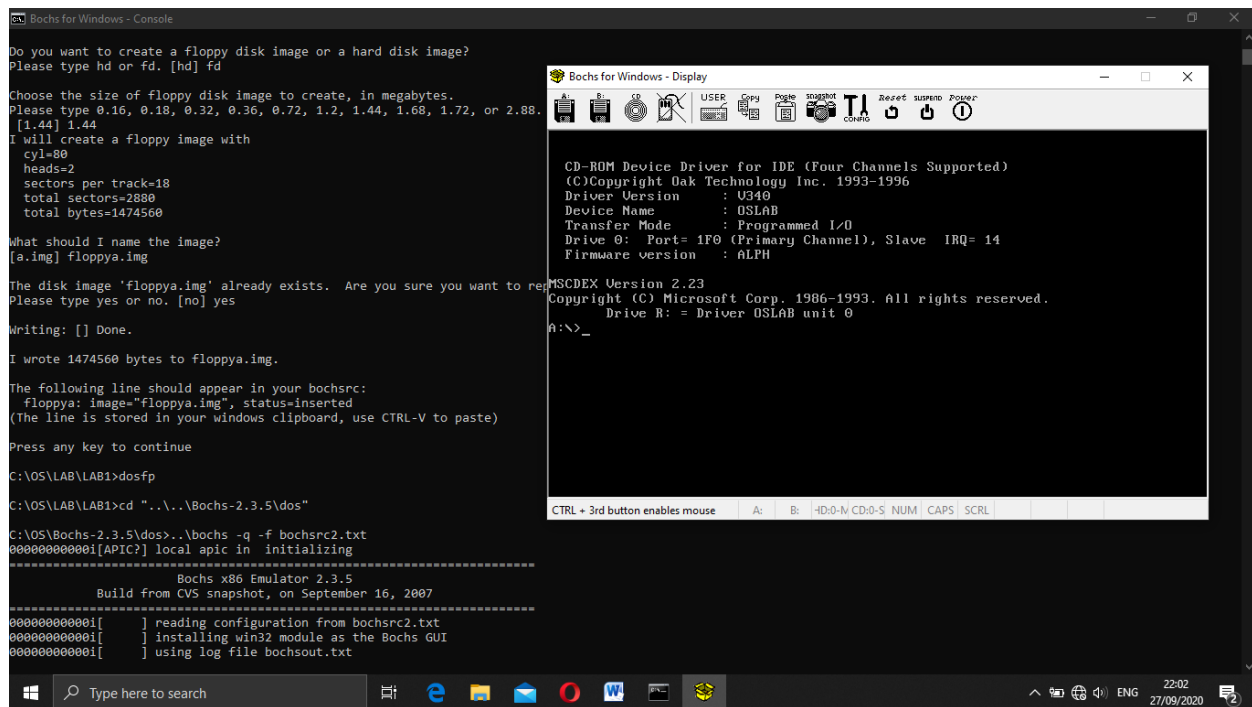
Writing: [] Done.

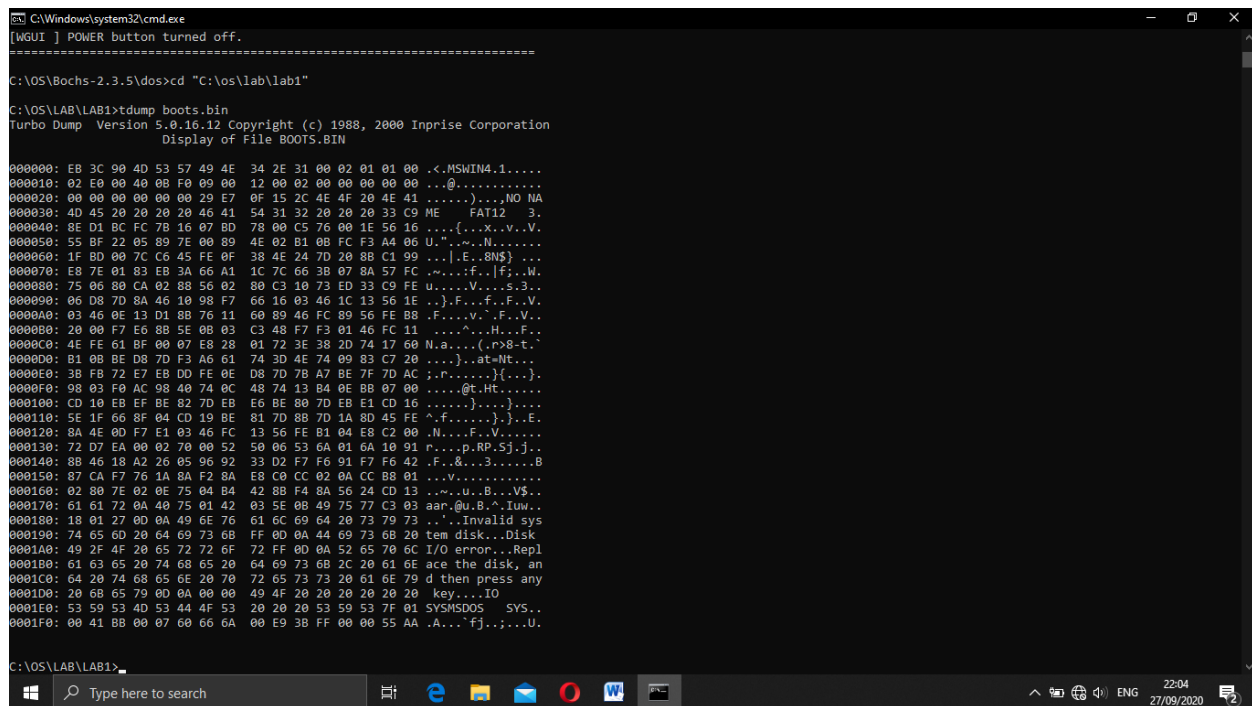
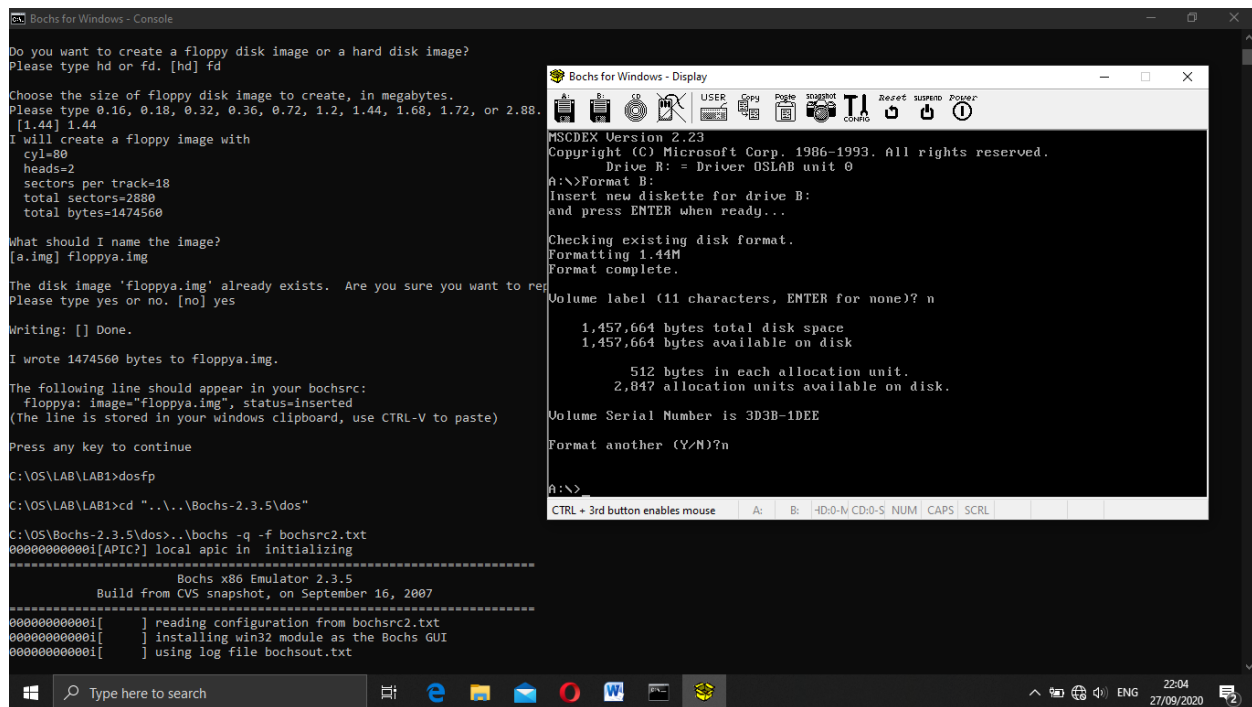
I wrote 1474560 bytes to floppy.img.

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

Press any key to continue

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





```
C:\Windows\system32\cmd.exe
000000: 75 06 80 CA 02 88 56 02 80 C3 10 73 ED 33 C9 FE u....V....3..
000000: 06 D8 7D 8A 46 10 98 F7 66 16 03 46 1C 13 56 1E ..).F...F..F..V.
0000A0: 03 46 0E 13 D1 8B 76 11 60 89 46 FC 89 56 FE B8 .F...V..F..V..
0000B0: 20 00 F7 E6 8B 5E 0B 03 C3 48 F7 F3 01 46 FC 11 .....H...F..
0000C0: 4E FE 61 BF 00 07 E8 28 01 72 3E 38 2D 74 17 60 N.a....(r>8-t..
0000D0: B1 0B BE D8 7D F3 A6 61 74 3D 4E 74 09 83 C7 20 ....).at-Nt...
0000E0: 3B FB 72 E7 EB DD FE 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 E0 EF 0E 82 7D EB E6 BE 80 7D EB E1 CD 16 .....}.....
000110: 5C 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 .N...f..V.....
000130: 72 D7 EA 00 02 70 00 52 50 06 53 6A 01 6A 10 91 r....p.RP.Sj.j..
000140: 8B 46 18 A2 26 05 96 92 33 D2 F7 F6 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 08 49 75 77 C3 03 aar.@u.B.^Iuw..
000180: 18 01 27 0D 0A 49 6E 76 61 6C 69 64 20 73 79 73 ...Invalid sys
000190: 74 65 6D 20 64 69 73 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 68 65 79 0D 0A 00 00 49 4F 20 20 20 20 20 20 key....IO
0001E0: 53 59 53 4D 53 44 4F 53 20 20 20 53 59 53 7F 01 SYSMSDOS SYS..
0001F0: 00 41 8B 00 07 60 66 6A 00 E9 3B FF 00 55 AA .A...`fj.;...U.

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
# In bx_win32_gui_c::exit(void)!
=====
Bochs is exiting with the following message:
[WGUi ] POWER button turned off.
=====

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

```
C:\Windows\system32\cmd.exe - bximage
0001F0: 00 41 8B 00 07 60 66 6A 00 E9 3B FF 00 55 AA .A...`fj.;...U.

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
# In bx_win32_gui_c::exit(void)!
=====
Bochs is exiting with the following message:
[WGUi ] POWER button turned off.
=====

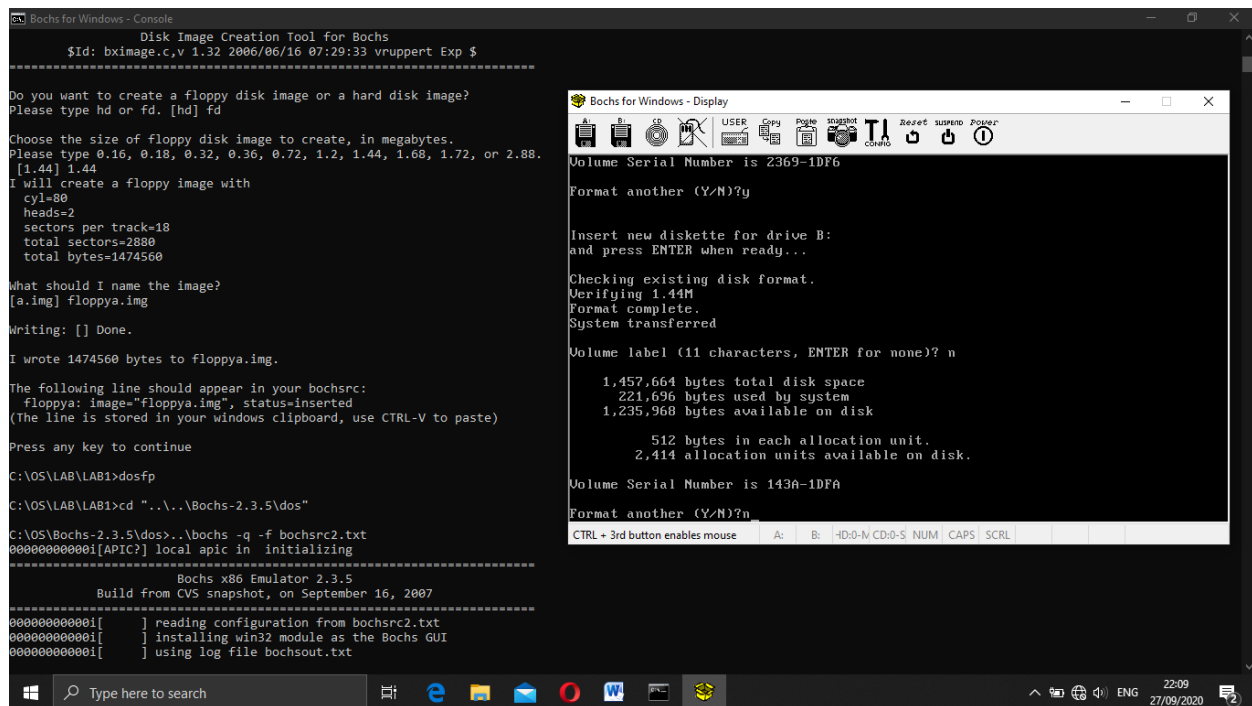
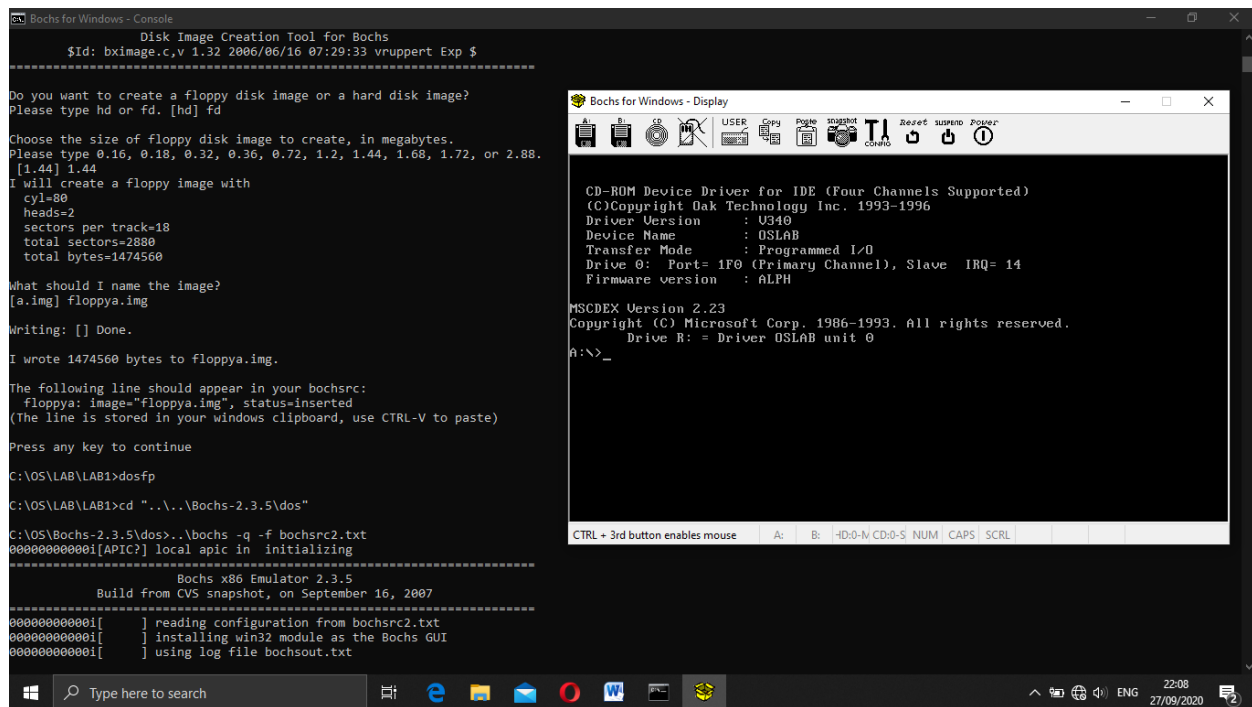
C:\OS\LAB\LAB1>del floppy.a.img

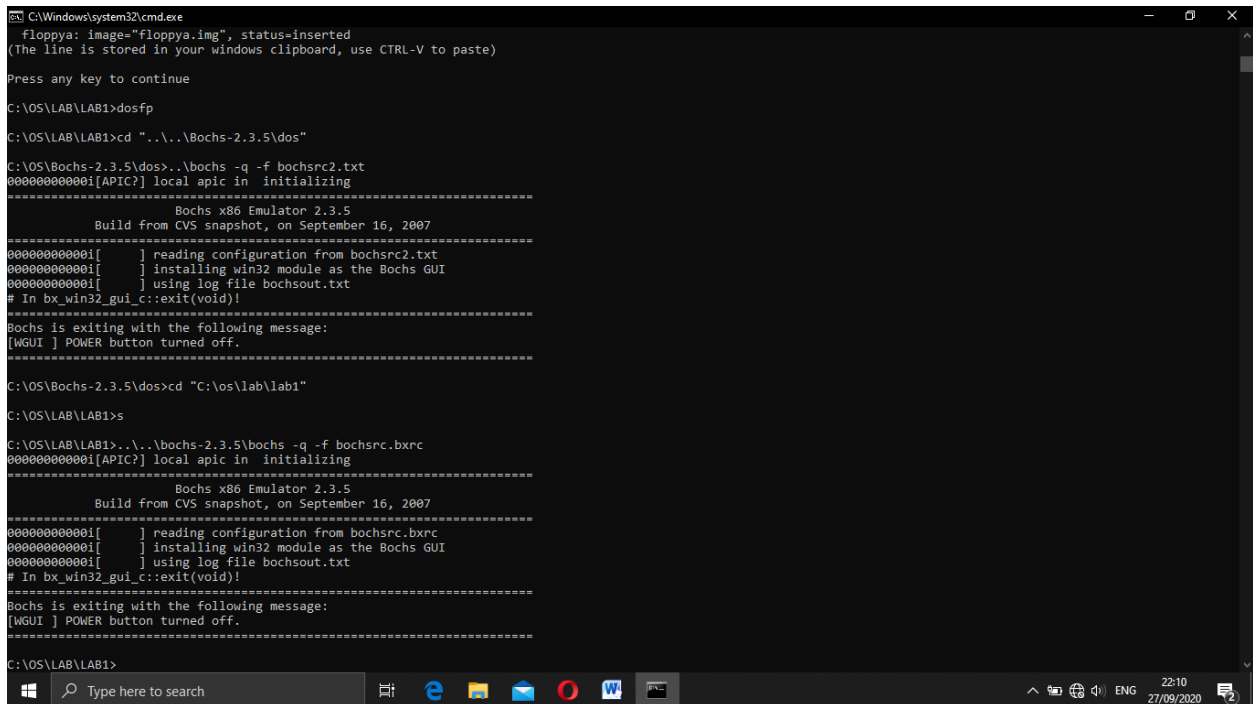
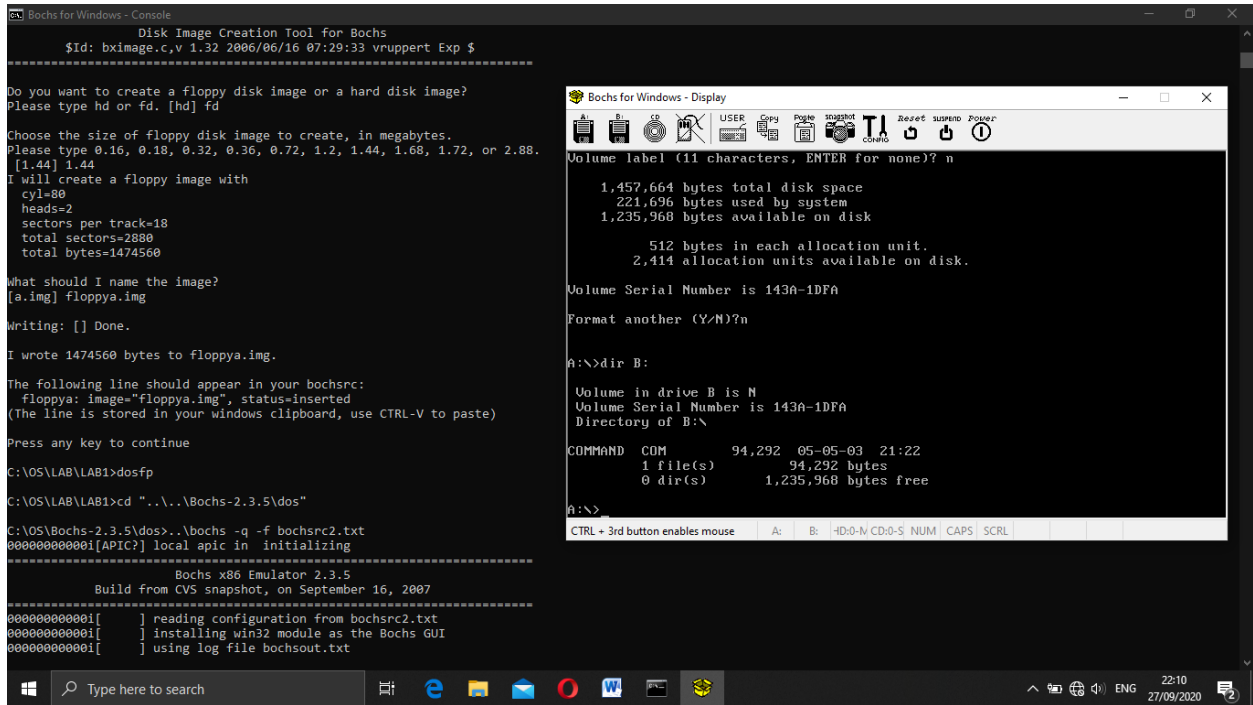
C:\OS\LAB\LAB1>bximage
=====
bximage
Disk Image Creation Tool for Bochs
$Id: bximage.c,v 1.32 2006/06/16 07:29:33 vruppert Exp $
=====

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

Choose the size of floppy disk image to create, in megabytes.
Please type 0.16, 0.18, 0.32, 0.36, 0.72, 1.2, 1.44, 1.68, 1.72, or 2.88.
[1.44] 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]
```





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

Kode ASCII merupakan singkatan dari American Standard Code for information Interchange atau Kode Standar Amerika Untuk Pertukaran Informasi. Kode ASCII merupakan suatu standar internasional dalam kode huruf dan symbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, kode ASCII juga digunakan untuk mewakili karakter-karakter angka maupun huruf didalam computer.

Kode ASCII yang standar tidak perlu extended:

Dec	Hex	Char	40	28	(48	30	0	56	38	8
33	21	!	41	29)	49	31	1	57	39	9
34	22	"	42	2A	*	50	32	2	58	3A	:
35	23	#	43	2B	+	51	33	3	59	3B	;
36	24	\$	44	2C	,	52	34	4	60	3C	<
37	25	%	45	2D	-	53	35	5	61	3D	=
38	26	&	46	2F	.	54	36	6	62	3E	>
39	27	'	47	30	/	55	37	7	63	3F	?

64	40	@	72	48	H	80	50	P	88	58	X
65	41	A	73	49	I	81	51	Q	89	59	Y
66	42	B	74	4A	J	82	52	R	90	5A	Z
67	43	C	75	4B	K	83	53	S	91	5B	[
68	44	D	76	4C	L	84	54	T	92	5C	\
69	45	E	77	4D	M	85	55	U	93	5D]
70	46	F	78	4E	N	86	56	V	94	5E	^
71	47	G	79	4F	O	87	57	W	95	5F	_

96	60	.	104	68	h	112	70	p	120	78	x
97	61	a	105	68	i	113	71	q	121	79	y
98	62	b	106	6A	j	114	72	r	122	7A	z
99	63	c	107	6B	k	115	73	s	123	7B	{
100	64	d	108	6C	l	116	74	t	124	7C	
101	65	e	109	6D	m	117	75	u	125	7D	}
102	66	f	110	6E	n	118	76	v	126	7E	~
103	67	g	111	6F	o	119	77	w	127	7F	DEL

TABEL ASCII

Binary	Oct	Dec	Hex	Glyph
010 0000	040	32	20	
010 0001	041	33	21	!
010 0010	042	34	22	"
010 0011	043	35	23	#
010 0100	044	36	24	\$
010 0101	045	37	25	%
010 0110	046	38	26	&

010 0111	047	39	27	`
010 1000	050	40	28	(
010 1001	051	41	29)
010 1010	052	42	2A	*
010 1011	053	43	2B	+
010 1100	054	44	2C	,
010 1101	055	45	2D	-
010 1110	056	46	2E	.
010 1111	057	47	2F	/
011 0000	060	48	30	0
011 0001	061	49	31	1

011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	071	57	39	9
011 1010	072	58	3A	:
011 1011	073	59	3B	;
011 1100	074	60	3C	<
011 1101	075	61	3D	=
011 1110	076	62	3E	>

011 1111	077	63	3F	?
100 0000	100	64	40	@
100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D
100 0101	105	69	45	E
100 0110	106	70	46	F
100 0111	107	72	47	G
100 1000	110	72	48	H
100 1001	111	73	49	I
100 1010	112	74	4A	J

100 1011	113	75	4B	K
100 1100	114	76	4C	L
100 1101	115	77	4D	M
100 1110	116	78	4E	N
100 1111	117	79	4F	O
101 0000	120	80	50	P
101 0001	121	81	51	Q
101 0010	122	82	52	R
101 0011	123	83	53	S
101 0100	124	84	54	T
101 0101	125	85	55	U
101 0110	126	86	56	V
101 0111	127	87	57	W

101 1000	130	88	58	X
101 1001	131	89	59	Y
101 1010	132	90	5A	Z
101 1011	133	91	5B	[
101 1100	134	92	5C	\
101 1101	135	93	5D]
101 1110	136	94	5E	
101 1111	137	95	5F	—
110 0000	140	96	60	`
110 0001	141	97	61	a
110 0010	142	98	62	b
110 0011	143	99	63	C

110 0100	144	100	64	d
110 0101	145	101	65	e
110 0110	146	102	66	f
110 0111	147	103	67	g
110 1000	150	104	68	h
110 1001	151	105	69	i
110 1010	152	106	6A	j
110 1011	153	107	6B	k
110 1100	154	108	6C	l
110 1101	155	109	6D	m
110 1110	156	110	6E	n
110 1111	157	111	6F	o

111 0000	160	112	70	p
111 0001	161	113	71	q
111 0010	162	114	72	r
111 0011	163	115	73	s
111 0100	164	116	74	t
111 0101	165	117	75	u
111 0110	166	118	76	v
111 0111	167	119	77	w
111 1000	170	120	78	x
111 1001	171	121	79	y
111 1010	172	122	7A	z
111 1011	173	123	7B	{

111 1010	172	122	7A	z
111 1011	173	123	7B	{
111 1100	174	124	7C	
111 1101	175	125	7D	}
111 1110	176	126	7E	~

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? .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	$CF = 0$			
JB	jump if below	$CF = 1 \wedge ZF = 0$	unsigned	lompat bila $op\ 1 < op\ 2$	ya
JNAE	jump if not above or equal	$CF = 0$			
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 ∨ ZF = 1	signed	lompat bila op 1 ≤ op 2	ya
JNG	jump if not greater				
JE	jump if equal	ZF = 1	keduan ya	lompat bila op 1 = op 2	ya
JZ	jump if zero	ZF = 1	keduan ya	lompat bila op 1 = op 2	ya
JNE	jump if not equal	ZF = 0	keduan ya	lompat bila op 1 ≠ op 2	ya
JNZ	jump if not zero	ZF = 0	keduan ya	lompat bila op 1 ≠ 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			lompat bila	

	on parity even			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 overfl ow	OF = 1	N/A	lompat bila overflow flag = 1	tidak
JNO	jump if not overfl ow	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 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 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