

**SISTEM OPERASI**  
**MODUL 1**  
**PENGENALAN SISTEM PENGEMBANGAN OS**  
**DENGAN PC SIMULATOR 'BOCHS'**



**PENYUSUN:**

**NAMA : ROYAN SAIFUR ROBBI**

**NIM : L200190196**

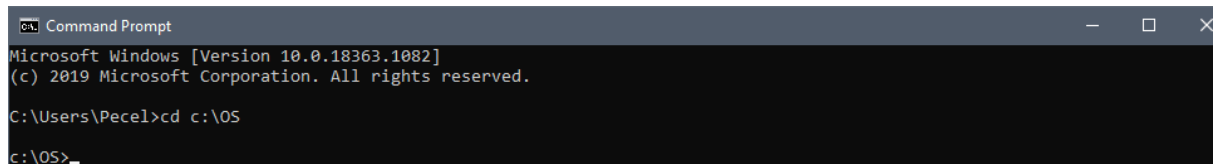
**KELAS : F**

**PROGRAM STUDI INFORMATIKA**  
**FAKULTAS KOMUNIKASI DAN INFORMATIKA**  
**UNIVERSITAS MUHAMMADIYAH SURAKARTA**  
**TAHUN 2020**

## Langkah Kerja

### Menuju ke direktori kerja.

- Jalankan program 'Command Prompt' atau 'cmd'.
- Masuk ke direktori kerja 'C:\OS', dengan perintah 'cd c:\OS' <ENTER>.

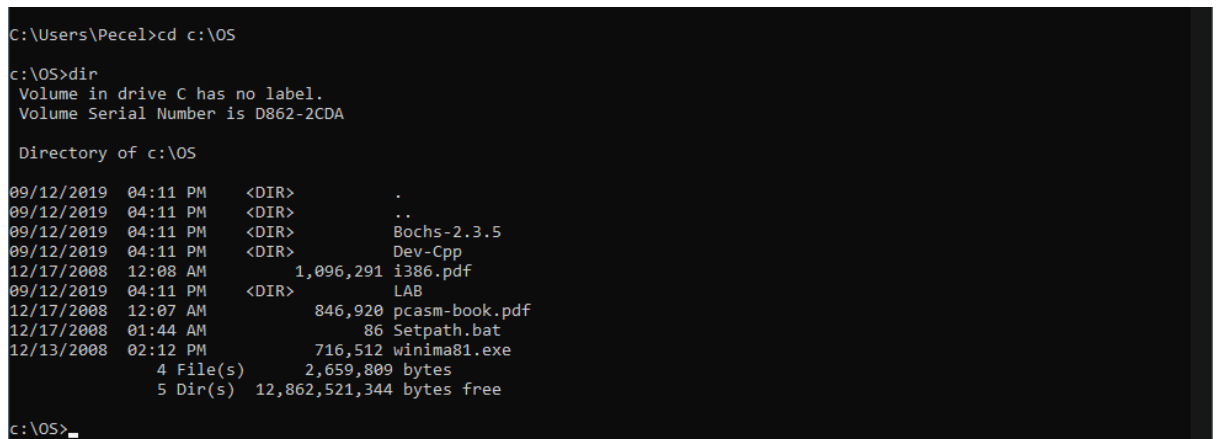


```
Command Prompt
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Pecel>cd c:\OS

c:\OS>
```

- Masukan perintah dir, untuk melihat isi direktori di dalam folder tersebut.



```
C:\Users\Pecel>cd c:\OS

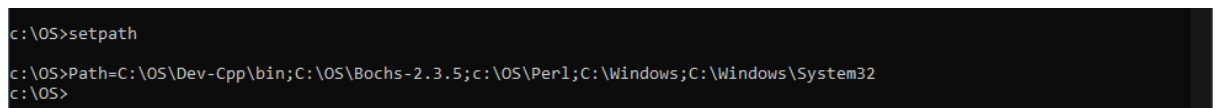
c:\OS>dir
Volume in drive C has no label.
Volume Serial Number is D862-2CDA

Directory of c:\OS

09/12/2019  04:11 PM  <DIR>          .
09/12/2019  04:11 PM  <DIR>          ..
09/12/2019  04:11 PM  <DIR>          Bochs-2.3.5
09/12/2019  04:11 PM  <DIR>          Dev-Cpp
12/17/2008  12:08 AM             1,096,291  i386.pdf
09/12/2019  04:11 PM  <DIR>          LAB
12/17/2008  12:07 AM             846,920  pcasm-book.pdf
12/17/2008  01:44 AM              86  Setpath.bat
12/13/2008  02:12 PM             716,512  winima81.exe
               4 File(s)          2,659,809 bytes
               5 Dir(s)    12,862,521,344 bytes free

c:\OS>
```

- Jalankan file setpath dengan ketik 'setpath' <ENTER>.

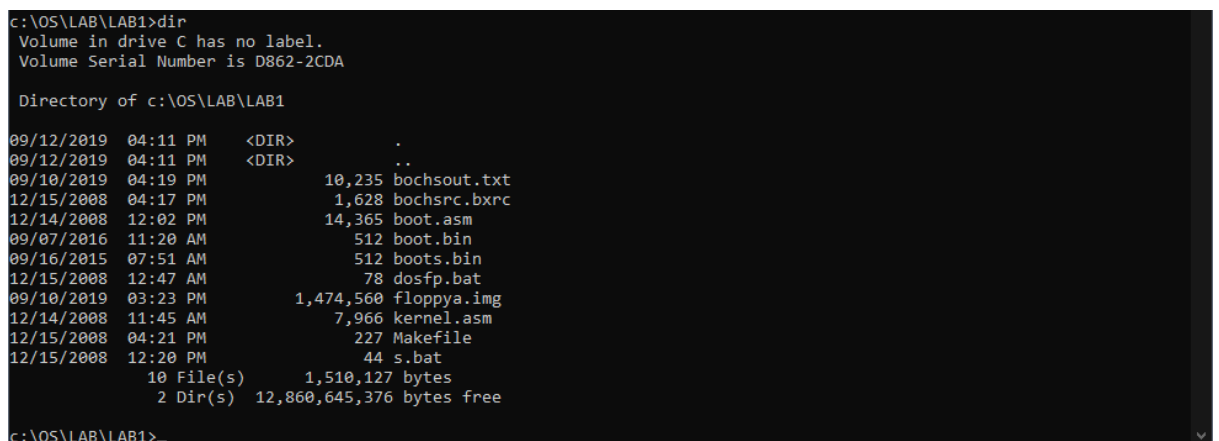


```
c:\OS>setpath

c:\OS>Path=C:\OS\Dev-Cpp\bin;C:\OS\Bochs-2.3.5;c:\OS\Perl;C:\Windows;C:\Windows\System32
c:\OS>
```

### Melihat isi direktori kerja

- Jalankan 'Command Prompt' atau 'cmd'.
- Masuk ke direktori kerja pada 'C:\OS\LAB\LAB1' dengan cara ketik 'cd lab\lab1' <ENTER>, kemudian ketik 'dir' untuk membuka isi direktori.



```
c:\OS\LAB\LAB1>dir
Volume in drive C has no label.
Volume Serial Number is D862-2CDA

Directory of c:\OS\LAB\LAB1

09/12/2019  04:11 PM  <DIR>          .
09/12/2019  04:11 PM  <DIR>          ..
09/10/2019  04:19 PM             10,235  bochsout.txt
12/15/2008  04:17 PM             1,628  bochsrc.bxrc
12/14/2008  12:02 PM             14,365  boot.asm
09/07/2016  11:20 AM              512  boot.bin
09/16/2015  07:51 AM              512  boots.bin
12/15/2008  12:47 AM              78  dosfp.bat
09/10/2019  03:23 PM          1,474,560  floppy.img
12/14/2008  11:45 AM             7,966  kernel.asm
12/15/2008  04:21 PM              227  Makefile
12/15/2008  12:20 PM              44  s.bat
               10 File(s)          1,510,127 bytes
               2 Dir(s)    12,860,645,376 bytes free

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

- c. Ketikkan 'notepad boot.asm' <ENTER> untuk membuka file boot.asm.

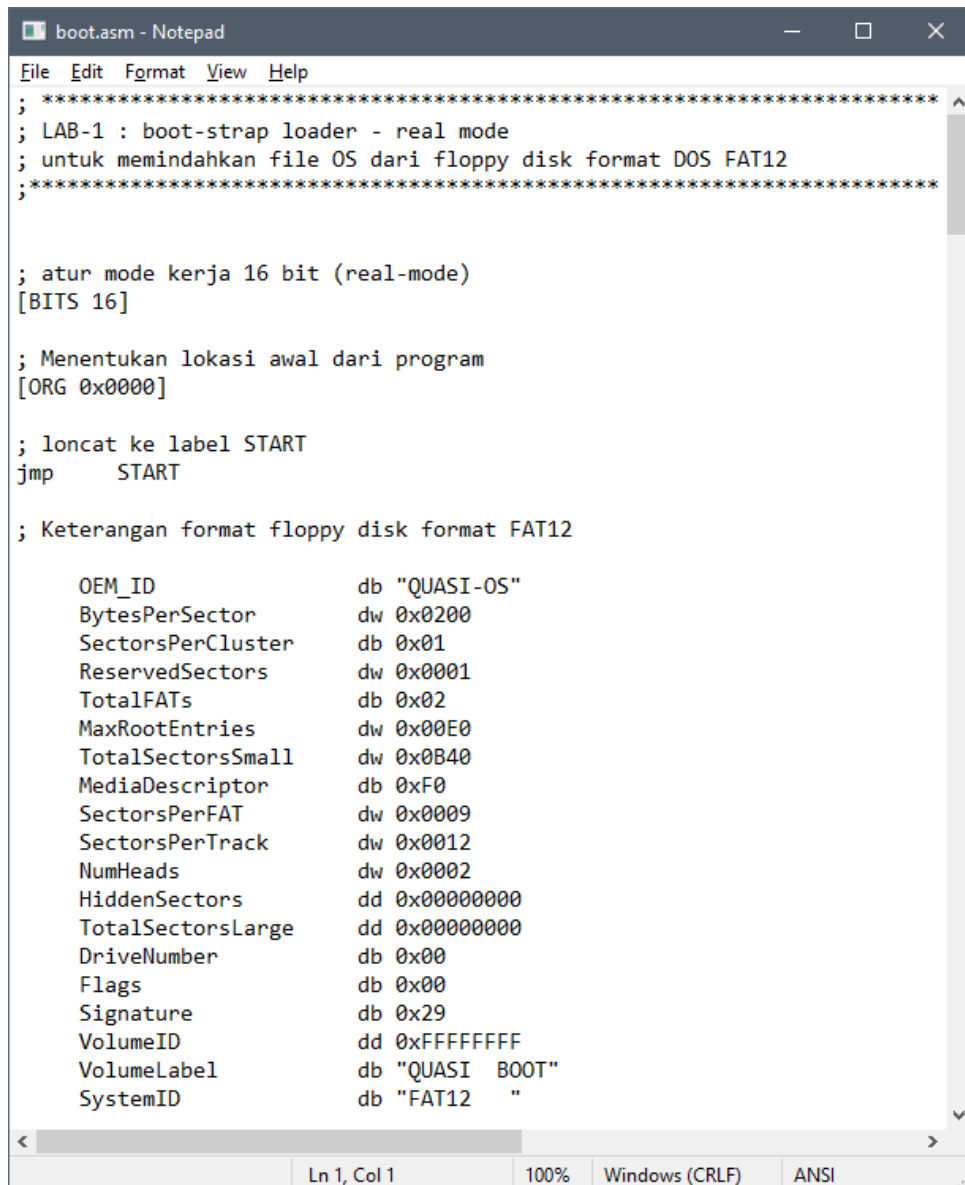
```
c:\OS\LAB\LAB1>dir
Volume in drive C has no label.
Volume Serial Number is D862-2CDA

Directory of c:\OS\LAB\LAB1

09/12/2019  04:11 PM  <DIR>          .
09/12/2019  04:11 PM  <DIR>          ..
09/10/2019  04:19 PM               10,235 bochsout.txt
12/15/2008  04:17 PM               1,628 bochsrc.bxrc
12/14/2008  12:02 PM              14,365 boot.asm
09/07/2016  11:20 AM               512 boot.bin
09/16/2015  07:51 AM               512 boots.bin
12/15/2008  12:47 AM                78 dosfp.bat
09/10/2019  03:23 PM            1,474,560 floppy.img
12/14/2008  11:45 AM               7,966 kernel.asm
12/15/2008  04:21 PM                227 Makefile
12/15/2008  12:20 PM                44 s.bat
               10 File(s)            1,510,127 bytes
               2 Dir(s)  12,860,645,376 bytes free

c:\OS\LAB\LAB1>notepad boot.asm
c:\OS\LAB\LAB1>
```

Jika benar, akan muncul seperti gambar di bawah.



```
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   db 0x01
    ReservedSectors     dw 0x0001
    TotalFATs           db 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"
    SystemID            db "FAT12  "
```

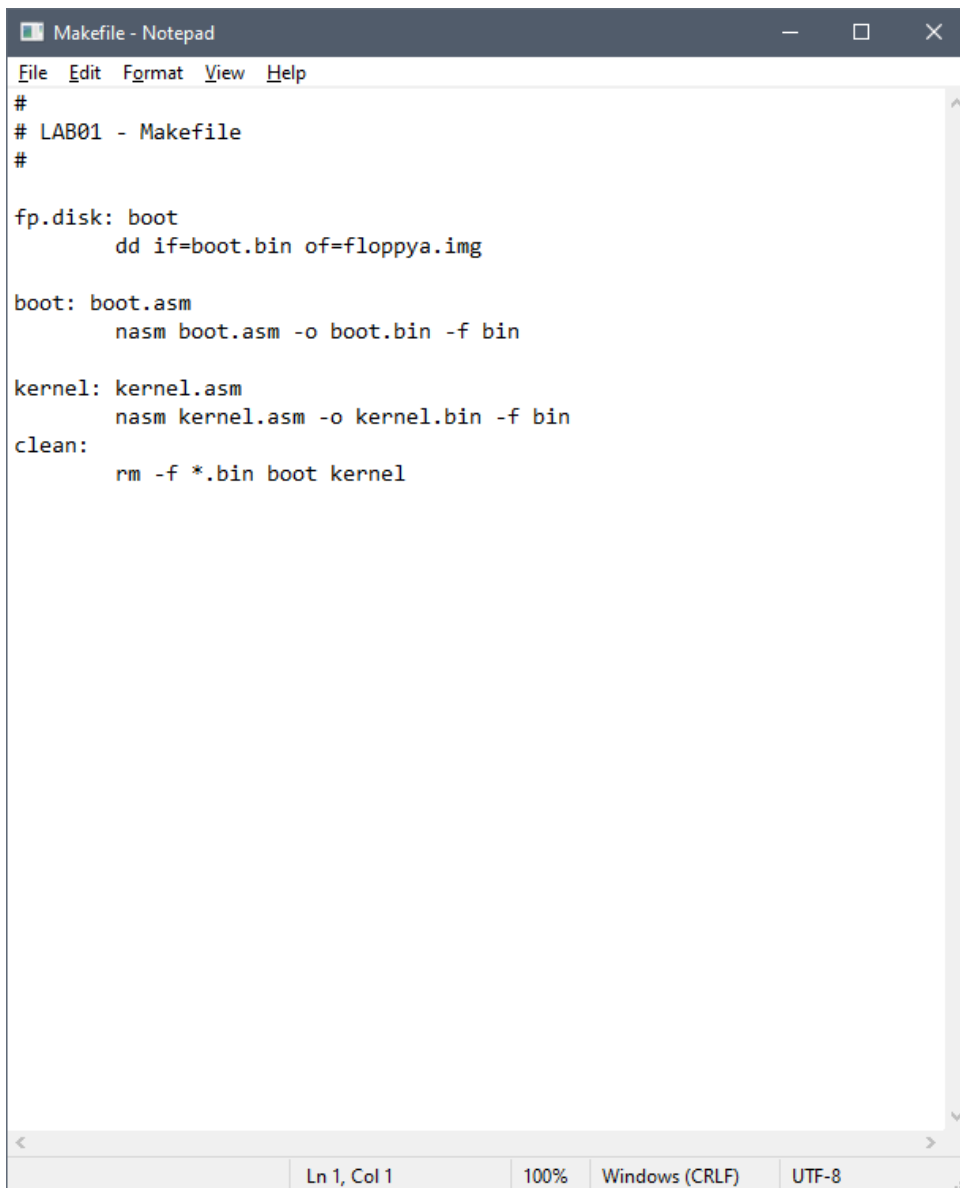
Jangan lakukan modifikasi terhadap program diatas dan tutup kembali program 'notepad'nya.

### Sekilas tentang Makefile

- a. Membuka file 'Makefile' dari 'Command Prompt' atau 'cmd' dari direktori 'C:\OS\LAB\LAB1' dengan cara ketikkan 'notepad m' dan tekan tombol 'Tab' hingga muncul 'notepad Makefile' lalu <ENTER>.

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

Kemudian akan muncul file notepad seperti dibawah.



```
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
```

Jangan memodifikasi isi file dan tutup program 'notepad'nya.

- b. Kembali ke 'cmd', pada direktori 'C:\OS\LAB\LAB1' ketikkan 'make fp.disk' lalu <ENTER> untuk melakukan kompilasi terhadap source code program 'boot.asm',

sebagai outputnya file 'boot.bin' dan isinya disalin ke dalam bootsector file image floppy 'floppya.img'.

```
c:\OS\LAB\LAB1>make fp.disk
nasm boot.asm -o boot.bin -f bin
dd if=boot.bin of=floppya.img
rawwrite dd for windows version 0.5.
Written by John Newbiggin <jn@it.swin.edu.au>
This program is covered by the GPL. See copying.txt for details
1+0 records in
1+0 records out
```

## Mengenal 'BOOT DISK'

- Hapus file 'floppya.img' jika sudah ada pada direktori kerja dengan 'cmd'. Ketik 'del floppya.img /P' <ENTER> lalu tekan 'Y' <ENTER>.

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

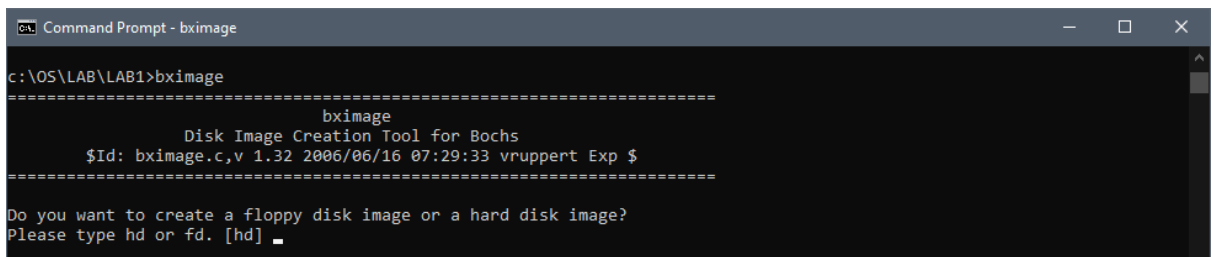
- Ketik 'dir' untuk memastikan bahwa file 'floppya.img' sudah terhapus.

```
c:\OS\LAB\LAB1>dir
Volume in drive C has no label.
Volume Serial Number is D862-2CDA

Directory of c:\OS\LAB\LAB1

09/24/2020  10:36 PM  <DIR>          .
09/24/2020  10:36 PM  <DIR>          ..
09/10/2019  04:19 PM             10,235 bochsout.txt
12/15/2008  04:17 PM             1,628 bochsrc.bxrc
12/14/2008  12:02 PM            14,365 boot.asm
09/24/2020  10:23 PM              512 boot.bin
09/16/2015  07:51 AM              512 boots.bin
12/15/2008  12:47 AM              78 dosfp.bat
12/14/2008  11:45 AM           7,966 kernel.asm
12/15/2008  04:21 PM              227 Makefile
12/15/2008  12:20 PM              44 s.bat
               9 File(s)          35,567 bytes
               2 Dir(s) 12,861,071,360 bytes free
```

- Jika sudah ketikkan 'bximage' untuk menampilkan window seperti dibawah.

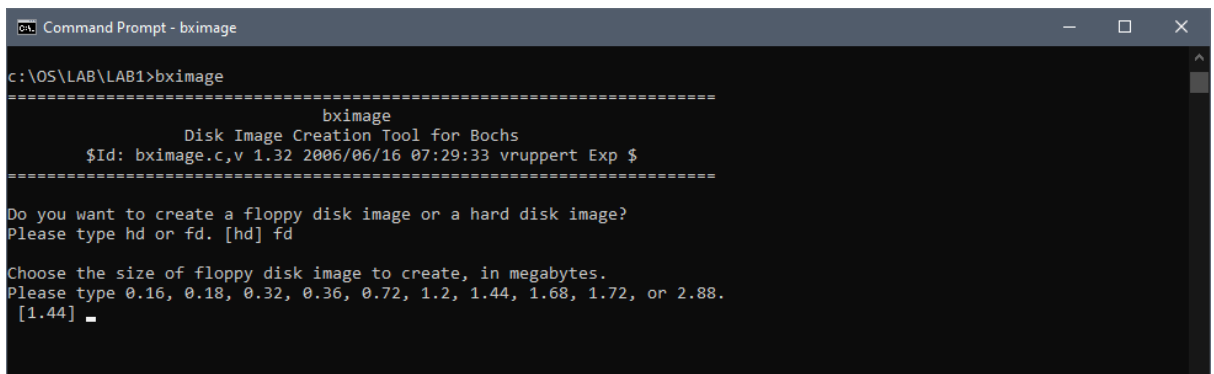


```
Command Prompt - bximage

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
```

- Kita akan membuat floppy image, ketikkan 'fd' <ENTER>.



```
Command Prompt - bximage

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
```

Tekan <ENTER> hingga seperti di bawah.

```
Command Prompt - bximage

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]
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]
```

Lalu ketikkan 'floppy.img' dan <ENTER>.

```
Command Prompt - 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]
I will create a floppy image with
  cyl=80
  heads=2
  sectors per track=18
  total sectors=2880
  total bytes=1474560

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

Writing: [] Done.

I wrote 1474560 bytes to floppy.img.

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

Press any key to continue
```

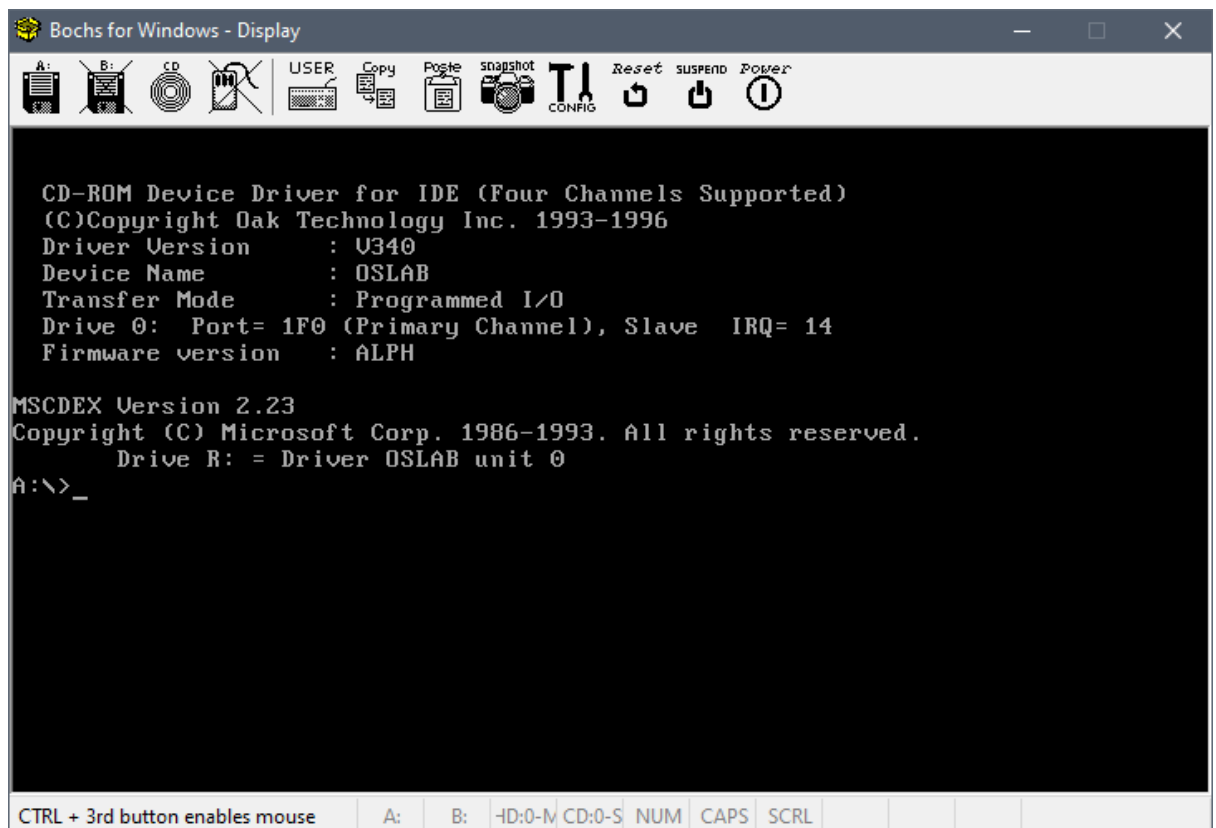
- e. Tekan keyboard bebas dan jalankan PC-Simulator dari 'Command Prompt' dengan perintah 'DosFp'<ENTER>.

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

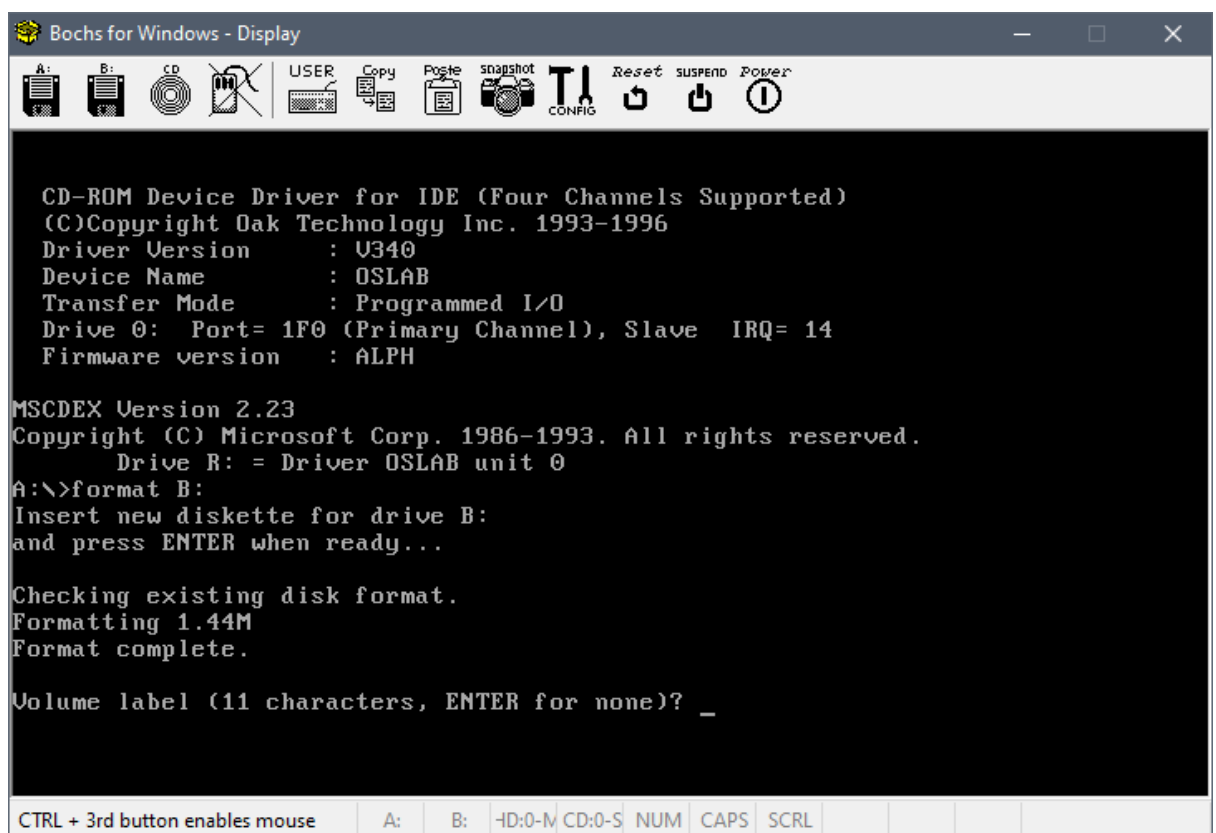
c:\OS\LAB\LAB1>cd "..\..\Bochs-2.3.5\dos"

c:\OS\Bochs-2.3.5\dos>..\bochs -q -f bochsrc2.txt
00000000000i[APIC?] local apic in  initializing
=====
                Bochs x86 Emulator 2.3.5
                Build from CVS snapshot, on September 16, 2007
=====
00000000000i[      ] reading configuration from bochsrc2.txt
00000000000i[      ] installing win32 module as the Bochs GUI
00000000000i[      ] using log file bochsout.txt
```

Lalu akan muncul seperti di bawah.



- f. Selanjutnya dari prompt 'A:\>' ketikkan 'format B:' <ENTER>[2x].



Lalu klik tombol power yang berada di kanan atas.

## Melihat data dalam boot sector

- Pada 'cmd' ketikkan 'dd if=floppya.img of=boots.bin count=1' <ENTER> untuk menyalin byte data dari 'floppya.img' ke dalam file 'boots.bin' sebanyak satu sektor mulai dari sektor 0.

```
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
```

- Ketik 'tdump boots.bin' <ENTER>, akan muncul seperti ini.

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

000000: EB 3C 90 4D 53 57 49 4E 34 2E 31 00 02 01 01 00 .<.MSWIN4.1....
000010: 02 E0 00 40 0B F0 09 00 12 00 02 00 00 00 00 00 ...@.....
000020: 00 00 00 00 00 00 29 F1 1E 23 13 4E 4F 20 4E 41 .....).#NO NA
000030: 4D 45 20 20 20 20 46 41 54 31 32 20 20 20 33 C9 ME FAT12 3.
000040: 8E D1 8C FC 7B 16 07 BD 78 00 C5 76 00 1E 56 16 ....{...x...v..V.
000050: 55 BF 22 05 89 7E 00 89 4E 02 B1 0B FC F3 A4 06 U."...~..N.....
000060: 1F BD 00 7C C6 45 FE 0F 38 4E 24 7D 20 8B C1 99 ...|.E..8N$} ...
000070: E8 7E 01 83 EB 3A 66 A1 1C 7C 66 38 07 8A 57 FC ~...:f...|f;..W.
000080: 75 06 80 CA 02 88 56 02 80 C3 10 73 ED 33 C9 FE u....V....s.3..
000090: 06 D8 7D 8A 46 10 98 F7 66 16 03 46 1C 13 56 1E ..}.F...f..F..V.
0000A0: 03 46 0E 13 D1 8B 76 11 60 89 46 FC 89 56 FE B8 .F....v..`F..V..
0000B0: 20 00 F7 E6 8B 5E 0B 03 C3 48 F7 F3 01 46 FC 11 ....^...H...F..
0000C0: 4E FE 61 BF 00 07 E8 28 01 72 3E 38 2D 74 17 60 N.a....(.r>8-t.`
0000D0: B1 0B BE D8 7D F3 A6 61 74 3D 4E 74 09 83 C7 20 ....}.at=Nt...
0000E0: 3B F8 72 E7 EB DD FE 0E D8 7D 7B A7 BE 7F 7D AC ;r.....}{...}.
0000F0: 98 03 F0 AC 98 40 74 0C 48 74 13 B4 0E BB 07 00 ....@t.Ht.....
000100: CD 10 EB EF BE 82 7D EB E6 BE 80 7D EB E1 CD 16 .....}.....}....
000110: 5E 1F 66 8F 04 CD 19 BE 81 7D 8B 7D 1A 8D 45 FE ^.f.....}...E.
000120: 8A 4E 0D F7 E1 03 46 FC 13 56 FE B1 04 E8 C2 00 .N....F..V.....
000130: 72 D7 EA 00 02 70 00 52 50 06 53 6A 01 6A 10 91 r....p.RP.Sj.j..
000140: 8B 46 18 A2 26 05 96 92 33 D2 F7 F6 91 F7 F6 42 .F...&...3.....B
000150: 87 CA F7 76 1A 8A F2 8A E8 C0 CC 02 0A CC B8 01 ...v.....
000160: 02 80 7E 02 0E 75 04 B4 42 8B F4 8A 56 24 CD 13 ..~..u..B...V$.
000170: 61 61 72 0A 40 75 01 42 03 5E 0B 49 75 77 C3 03 aar.@u.B.^Iuw..
000180: 18 01 27 0D 0A 49 6E 76 61 6C 69 64 20 73 79 73 ...Invalid sys
000190: 74 65 6D 20 64 69 73 6B FF 0D 0A 44 69 73 6B 20 tem disk...Disk
0001A0: 49 2F 4F 20 65 72 72 6F 72 FF 0D 0A 52 65 70 6C I/O error...Repl
0001B0: 61 63 65 20 74 68 65 20 64 69 73 6B 2C 20 61 6E ace the disk, an
0001C0: 64 20 74 68 65 6E 20 70 72 65 73 73 20 61 6E 79 d then press any
```

## 'Boot' PC-Simulator dengan file image 'floppya.img'

- Ketikkan 'type s.bat' <ENTER> pada 'cmd'.

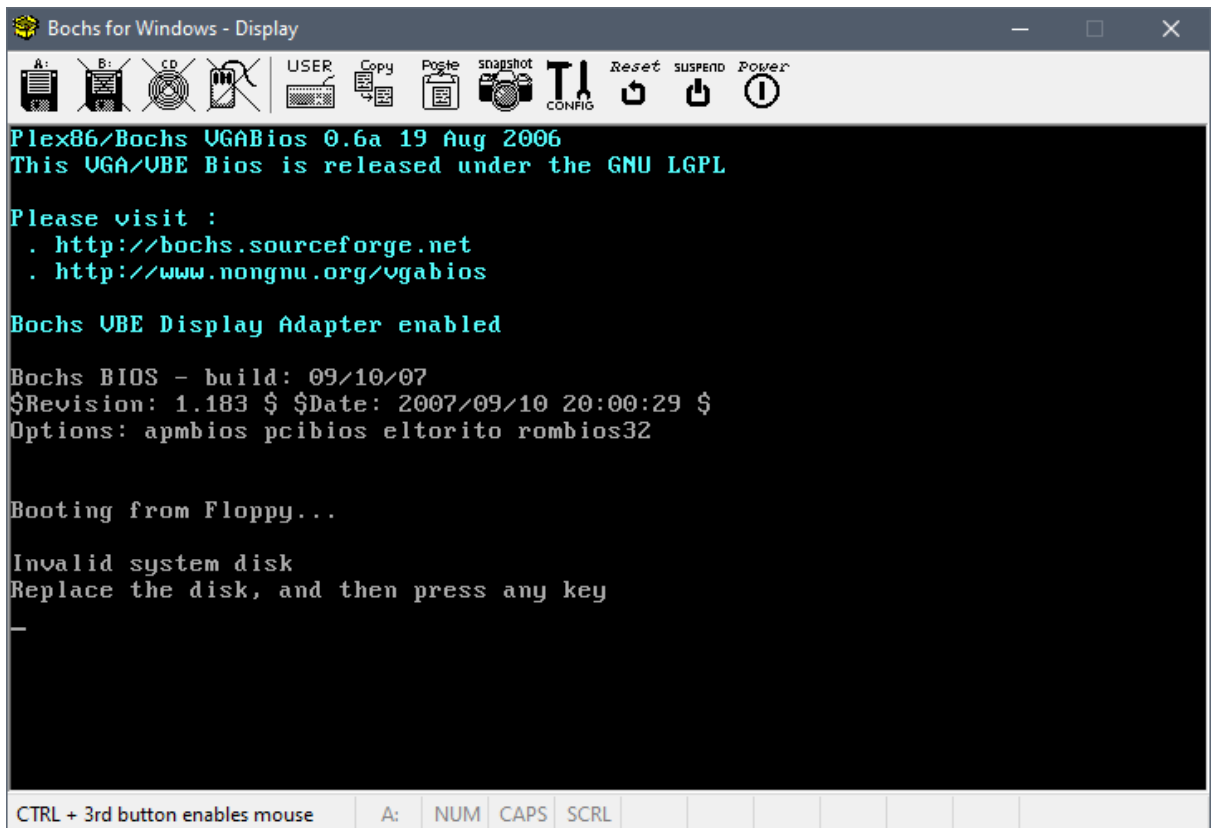
```
C:\OS\LAB\LAB1>type s.bat
..\..\bochs-2.3.5\bochs -q -f bochsrc.bxrc
C:\OS\LAB\LAB1>
```

Selanjutnya ketik 's' <ENTER> yang akan nampak seperti ini.

```
C:\OS\LAB\LAB1>s
C:\OS\LAB\LAB1>..\..\bochs-2.3.5\bochs -q -f bochsrc.bxrc
0000000000i[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
```

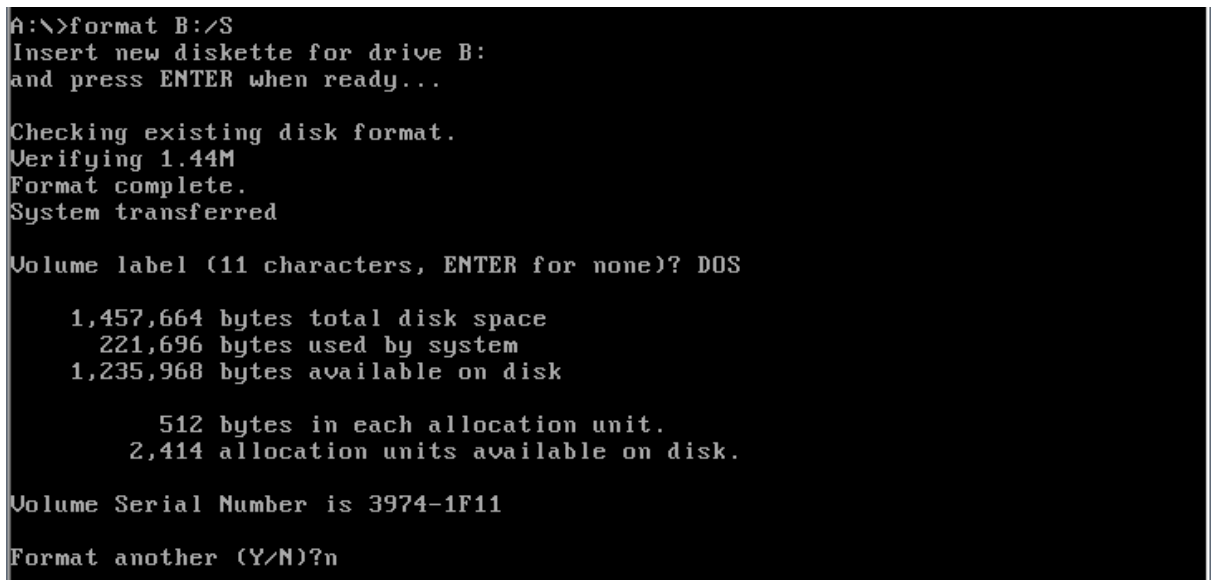
Lalu akan muncul tampilan seperti di bawah.





Hal diatas merupakan tampilan gagal booting karena 'floppya.img' belum terisi 'system file' atau 'kernel'. Klik tombol power lalu panggil 'DosFp' dari 'cmd'.

- b. Format 'floppya.img' dengan ketik 'format B:/S' <ENTER> pada 'Boschs for Windows – Display' direktori 'A:\>' dan beri nama jika perlu, contoh 'DOS'.



- c. Pastikan bahwa floppy pada drive 'B:' terisi dengan 'system file' dengan ketikkan 'dir B:'.

```
A:\>dir B:

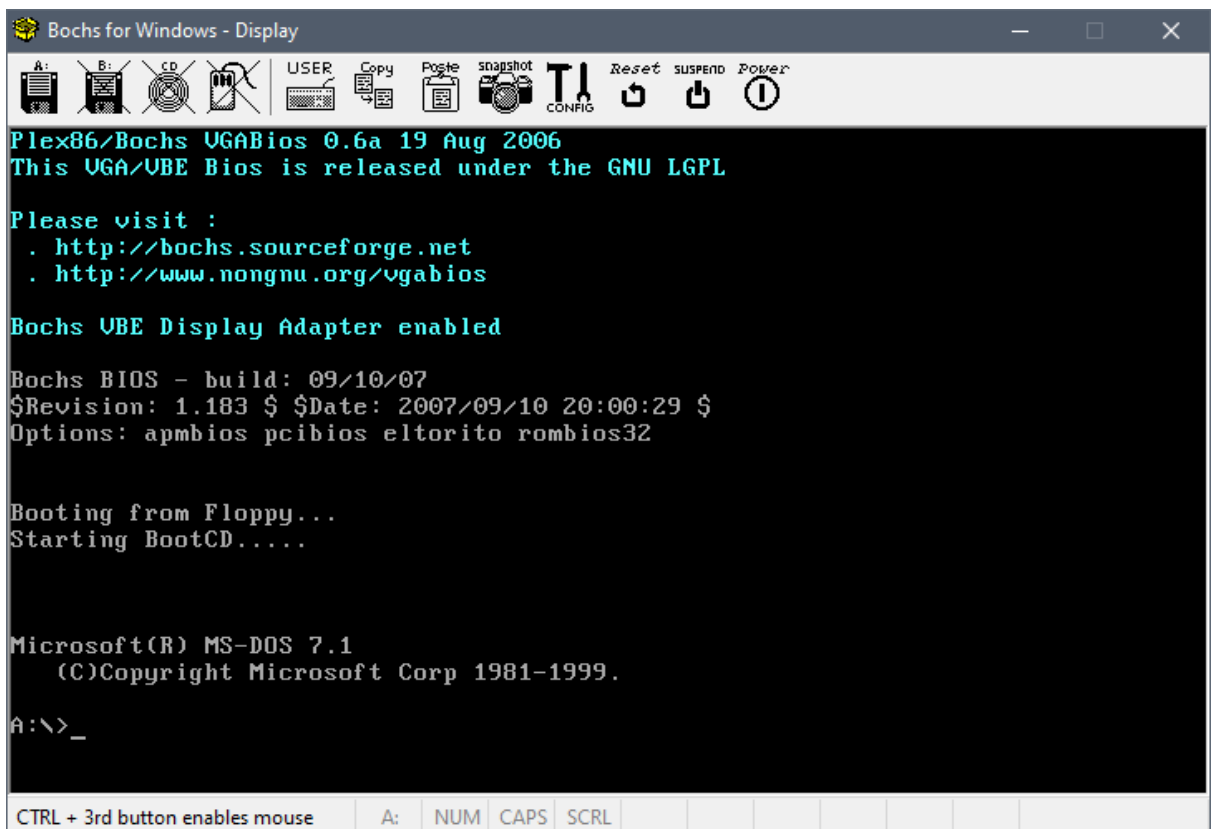
Volume in drive B is DOS
Volume Serial Number is 3974-1F11
Directory of B:\

COMMAND  COM           94,292   05-05-03   21:22
          1 file(s)           94,292 bytes
          0 dir(s)           1,235,968 bytes free

A:\>
```

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

- d. Matikan PC-Simulator dengan klik tombol power dan coba ketik 's' lalu <ENTER> pada 'cmd'. Jika berhasil akan seperti gambar di bawah.



```
Bochs for Windows - Display
A: B: CD USER Copy Paste Snapshot CONFIG Reset Suspend Power
Plex86/Bochs UGABios 0.6a 19 Aug 2006
This UGA/VBE Bios is released under the GNU LGPL

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

Bochs VBE Display Adapter enabled

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

Booting from Floppy...
Starting BootCD.....

Microsoft(R) MS-DOS 7.1
(C)Copyright Microsoft Corp 1981-1999.

A:\>
```

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

## 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:

Singkatan dari *American Standard Code for Information Interchange* yang merupakan standar pengkodean untuk alat komunikasi. Kode ASCII standar:

Bin.	Hex.	Dec.	ASCII Symbol	Explanation
0000000	0	0	NUL	The null character prompts the device to do nothing
0000001	1	1	SOH	Initiates a header (Start of Heading)
0000010	2	2	STX	Ends the header and marks the beginning of a message. (start of text)
0000011	3	3	ETX	Indicates the end of the message (end of text)
0000100	4	4	EOT	Marks the end of a completes transmission (End of Transmission)
0000101	5	5	ENQ	A request that requires a response (Enquiry)
0000110	6	6	ACK	Gives a positive answer to the request (Acknowledge)
0000111	7	7	BEL	Triggers a beep (Bell)
0001000	8	8	BS	Let's the cursor move back one step (Backspace)
0001001	9	9	TAB (HT)	A horizontal tab that moves the cursor within a row to the next predefined position (Horizontal Tab)
0001010	A	10	LF	Causes the cursor to jump to the next line (Line Feed)
0001011	B	11	VT	The vertical tab lets the cursor jump to a predefined line (Vertical Tab)
0001100	C	12	FF	Requests a page break (Form Feed)
0001101	D	13	CR	Moves the cursor back to the first position of the line (Carriage Return)
0001110	E	14	SO	Switches to a special presentation (Shift Out)
0001111	F	15	SI	Switches the display back to the normal state (Shift In)
0010000	10	16	DLE	Changes the meaning of the following characters (Data Link Escape)
0010001	11	17	DC1	Control characters assigned depending on the device used (Device Control)
0010010	12	18	DC2	
0010011	13	19	DC3	
0010100	14	20	DC4	
0010101	15	21	NAK	Negative response to a request (Negative Acknowledge)
0010110	16	22	SYN	Synchronizes a data transfer, even if no signals are transmitted (Synchronous Idle)
0010111	17	23	ETB	Marks the end of a transmission block (End of Transmission Block)
0011000	18	24	CAN	Makes it clear that a transmission was faulty and the data must be discarded (Cancel)
0011001	19	25	EM	Indicates the end of the storage medium (End of Medium)
0011010	1A	26	SUB	Replacement for a faulty sign (Substitute)
0011011	1B	27	ESC	Initiates an escape sequence and thus gives the following characters a special meaning (Escape)
0011100	1C	28	FS	Marks the separation of logical data blocks and is hierarchically ordered: file as the largest unit, file as the smallest unit (File Separator, Group Separator, Record Separator, Unit Separator)
0011101	1D	29	GS	
0011110	1E	30	RS	

0011111	1F	31	US	
0100000	20	32	SP	Blank space (Space)
0100001	21	33	!	Exclamation mark
0100010	22	34	"	Only quotes above
0100011	23	35	#	Pound sign
0100100	24	36	\$	Dollar sign
0100101	25	37	%	Percentage sign
0100110	26	38	&	Commerical and
0100111	27	39	'	Apostrophe
0101000	28	40	(	Left bracket
0101001	29	41	)	Right bracket
0101010	2A	42	*	Asterisk
0101011	2B	43	+	Plus symbol
0101100	2C	44	,	Comma
0101101	2D	45	-	Dash
0101110	2E	46	.	Full stop
0101111	2F	47	/	Forward slash
0110000	30	48	0	
0110001	31	49	1	
0110010	32	50	2	
0110011	33	51	3	
0110100	34	52	4	
0110101	35	53	5	
0110110	36	54	6	
0110111	37	55	7	
0111000	38	56	8	
0111001	39	57	9	
0111010	3A	58	:	Colon
0111011	3B	59	;	Semicolon
0111100	3C	60	<	Small than bracket
0111101	3D	61	=	Equals sign
0111110	3E	62	>	Bigger than symbol
0111111	3F	63	?	Question mark
1000000	40	64	@	At symbol
1000001	41	65	A	
1000010	42	66	B	
1000011	43	67	C	
1000100	44	68	D	
1000101	45	69	E	
1000110	46	70	F	
1000111	47	71	G	
1001000	48	72	H	
1001001	49	73	I	
1001010	4A	74	J	
1001011	4B	75	K	
1001100	4C	76	L	
1001101	4D	77	M	
1001110	4E	78	N	

1001111	4F	79	O	
1010000	50	80	P	
1010001	51	81	Q	
1010010	52	82	R	
1010011	53	83	S	
1010100	54	84	T	
1010101	55	85	U	
1010110	56	86	V	
1010111	57	87	W	
1011000	58	88	X	
1011001	59	89	Y	
1011010	5A	90	Z	
1011011	5B	91	[	Left square bracket
1011100	5C	92	\	Inverse/backward slash
1011101	5D	93	]	Right square bracket
1011110	5E	94	^	Circumflex
1011111	5F	95	_	Underscore
1100000	60	96	`	Gravis (backtick)
1100001	61	97	a	
1100010	62	98	b	
1100011	63	99	c	
1100100	64	100	d	
1100101	65	101	e	
1100110	66	102	f	
1100111	67	103	g	
1101000	68	104	h	
1101001	69	105	i	
1101010	6A	106	j	
1101011	6B	107	k	
1101100	6C	108	l	
1101101	6D	109	m	
1101110	6E	110	n	
1101111	6F	111	o	
1110000	70	112	p	
1110001	71	113	q	
1110010	72	114	r	
1110011	73	115	s	
1110100	74	116	t	
1110101	75	117	u	
1110110	76	118	v	
1110111	77	119	w	
1111000	78	120	x	
1111001	79	121	y	
1111010	7A	122	z	
1111011	7B	123	{	Left curly bracket
1111100	7C	124		Vertical line
1111101	7D	125	}	Right curly brackets
1111110	7E	126	~	Tilde

1111111	7F	127	DEL	Deletes a character. Since this control character consists of the same number on all positions, during the typewriter era it was possible to invalidate another character by punching out all the positions (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'.

Jawab:

- **ACALL (Absolute Call)**  
ACALL berfungsi untuk memanggil sub rutin program.
- **ADD (Add Immediate Data)**  
ADD berfungsi untuk menambah 8 bit data langsung ke dalam isi akumulator dan menyimpan hasilnya pada akumulator.
- **ADDC (Add Carry Plus Immediate Data to Accumulator).**  
ADDC berfungsi untuk menambahkan isi carry flag (0 atau 1) ke dalam isi akumulator. Data langsung 8 bit ditambahkan ke akumulator.
- **AJMP (Absolute Jump)**  
AJMP adalah perintah jump mutlak. Jump dalam 2 KB dimulai dari alamat yang mengikuti perintah AJMP. AJMP berfungsi untuk mentransfer kendali program ke lokasi dimana alamat dikalkulasi dengan cara yang sama dengan perintah ACALL. Konter program ditambahkan dua kali dimana perintah AJMP adalah perintah 2-byte. Konter program di-load dengan a10 – a0 11 bits, untuk membentuk alamat tujuan 16-bit.
- **ANL (logical AND memori ke akumulator)**  
ANL berfungsi untuk mengAND-kan isi alamat data dengan isi akumulator.
- **CJNE (Compare Indirect Address to Immediate Data)**  
CJNE berfungsi untuk membandingkan data langsung dengan lokasi memori yang dialamati oleh register R atau Akumulator A. Apabila tidak sama maka instruksi akan menuju ke alamat kode.  
Format: CJNE R, #data. Alamat kode.
- **CLR (Clear Accumulator)**  
CLR berfungsi untuk mereset data akumulator menjadi 00H.  
Format: CLR A

- **CPL (Complement Accumulator)**  
CPL berfungsi untuk mengkomplemen isi akumulator.
- **DA (Decimal Adjust Accumulator)**  
DA berfungsi untuk mengatur isi akumulator ke padanan BCD, setelah penambahan dua angka BCD.
- **DEC (Decrement Indirect Address)**  
DEC berfungsi untuk mengurangi isi lokasi memori yang ditujukan oleh register R dengan 1, dan hasilnya disimpan pada lokasi tersebut.
- **DIV (Divide Accumulator by B)**  
DIV berfungsi untuk membagi isi akumulator dengan isi register B. Akumulator berisi hasil bagi, register B berisi sisa pembagian.
- **DJNZ (Decrement Register and Jump Id Not Zero)**  
DJNZ berfungsi untuk mengurangi nilai register dengan 1 dan jika hasilnya sudah 0 maka instruksi selanjutnya akan dieksekusi. Jika belum 0 akan menuju ke alamat kode.
- **INC (Increment Indirect Address)**  
INC berfungsi untuk menambahkan isi memori dengan 1 dan menyimpannya pada alamat tersebut.
- **JB (Jump if Bit is Set)**  
JB berfungsi untuk membaca data per satu bit, jika data tersebut adalah 1 maka akan menuju ke alamat kode dan jika 0 tidak akan menuju ke alamat kode.
- **JBC (Jump if Bit Set and Clear Bit)**  
Bit JBC, berfungsi sebagai perintah rel menguji yang terspesifikasikan secara bit. Jika bit di-set, maka Jump dilakukan ke alamat relatif dan yang terspesifikasi secara bit di dalam perintah dibersihkan. Segmen program berikut menguji bit yang kurang signifikan (LSB: Least Significant Byte), dan jika diketemukan bahwa ia telah di-set, program melompat ke READ lokasi. JBC juga berfungsi membersihkan LSB dari akumulator.
- **JC (Jump if Carry is Set)**  
Instruksi JC berfungsi untuk menguji isi carry flag. Jika berisi 1, eksekusi menuju ke alamat kode, jika berisi 0, instruksi selanjutnya yang akan dieksekusi.
- **JMP (Jump to sum of Accumulator and Data Pointer)**  
Instruksi JMP berfungsi untuk memerintahkan loncat kesuatu alamat kode tertentu.

Format: JMP alamat kode.

- **JNB (Jump if Bit is Not Set)**

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

Format: JNB alamat bit, alamat kode.

- **JNC (Jump if Carry Not Set)**

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

- **JNZ (Jump if Accumulator Not Zero)**

JNZ adalah mnemonik untuk instruksi jump if not zero (lompat jika tidak nol).

Dalam hal ini suatu lompatan akan terjadi bilamana bendera nol dalam keadaan “clear”, dan tidak akan terjadi lompatan bilamana bendera nol tersebut dalam keadaan set. Andaikan bahwa JNZ 7800H disimpan pada lokasi 2100H. Jika Z=0, instruksi berikutnya akan berasal dari lokasi 7800H: dan bilamana Z=1, program akan turun ke instruksi urutan berikutnya pada lokasi 2101H.

- **JZ (Jump if Accumulator is Zero)**

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

- **LCALL (Long Call)**

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

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

- **LJMP (Long Jump)**

Long Jump berfungsi untuk memungkinkan lompatan tak bersyarat kemana saja dalam lingkup ruang memori program 64K. LCALL adalah perintah 3-byte.

Alamat tujuan 16-bit ditentukan secara langsung dalam perintah tersebut. Alamat tujuan ini di-load ke dalam konter program oleh perintah LJMP.

- **MOV (Move from Memory)**



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

- **MOVC (Move from Codec Memory)**

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

- **MOVX (Move Accumulator to External Memory Addressed by Data Pointer)**

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

- **MUL (Multiply)**

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

- **NOP (No Operation)**

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

- **ORL (Logical OR Immediate Data to Accumulator)**

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

Format: ORL A,#data.

- **POP (Pop Stack to Memory)**

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

- **PUSH (Push Memory onto Stack)**

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

- **RET (Return from subroutine)**

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

- **RETI (Return from Interrupt)**

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

- **RL (Rotate Accumulator Left)**

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

- **RLC (Rotate Left through Carry)**

Fungsi: Memutar (Rotate) Accumulator ke Kiri (Left) Melalui Carry Flag.

Kedelapan bit accumulator dan carry flag akan diputar satu bit ke kiri secara bersama-sama. Bit 7 akan dirotasi ke carry flag, nilai carry flag akan berpindah ke posisi bit 0. Instruksi ini tidak mempengaruhi flag lain.

- **RR (Rotate Right)**

Fungsi: Memutar (Rotate) Accumulator ke Kanan (Right). Kedelapan bit accumulator akan diputar satu bit ke kanan. Bit 0 akan dirotasi ke posisi bit 7.

Instruksi ini tidak mempengaruhi flag apapun juga.

- **RRC (Rotate Right through Carry)**

Fungsi: Memutar (Rotate) Accumulator ke Kanan (Right) Melalui Carry Flag.

Kedelapan bit accumulator dan carry flag akan diputar satu bit ke kanan secara bersama-sama. Bit 0 akan dirotasi ke carry flag, nilai carry flag akan berpindah ke posisi bit 7. Instruksi ini tidak mempengaruhi flag lain.

- **SETB (set Carry flag)**

Instruksi SETB berfungsi untuk menset carry flag.

- **SJMP (Short Jump)**

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

menambahkan alamat relatif ke dalamnya untuk mendapatkan alamat tujuan.

Alamat relatif tersebut ditentukan dalam perintah sebagai 'SJMP rel'.

- **SUBB (Subtract with Borrow)**

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

- **SWAP (Swap Nibbles)**

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

- **XCH (Exchange Bytes)**

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

- **XCHD (Exchange Digits)**

Fungsi: Menukar (Exchange) Digit. XCHD menukar nibble rendah dari accumulator, yang umumnya mewakili angka heksadesimal atau BCD, dengan nibble rendah dari internal data memory yang diakses secara indirect. Nibble

tinggi kedua register tidak akan terpengaruh. Instruksi ini tidak mempengaruhi flag apapun juga.

- **XRL (Exclusive OR Logic)**

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