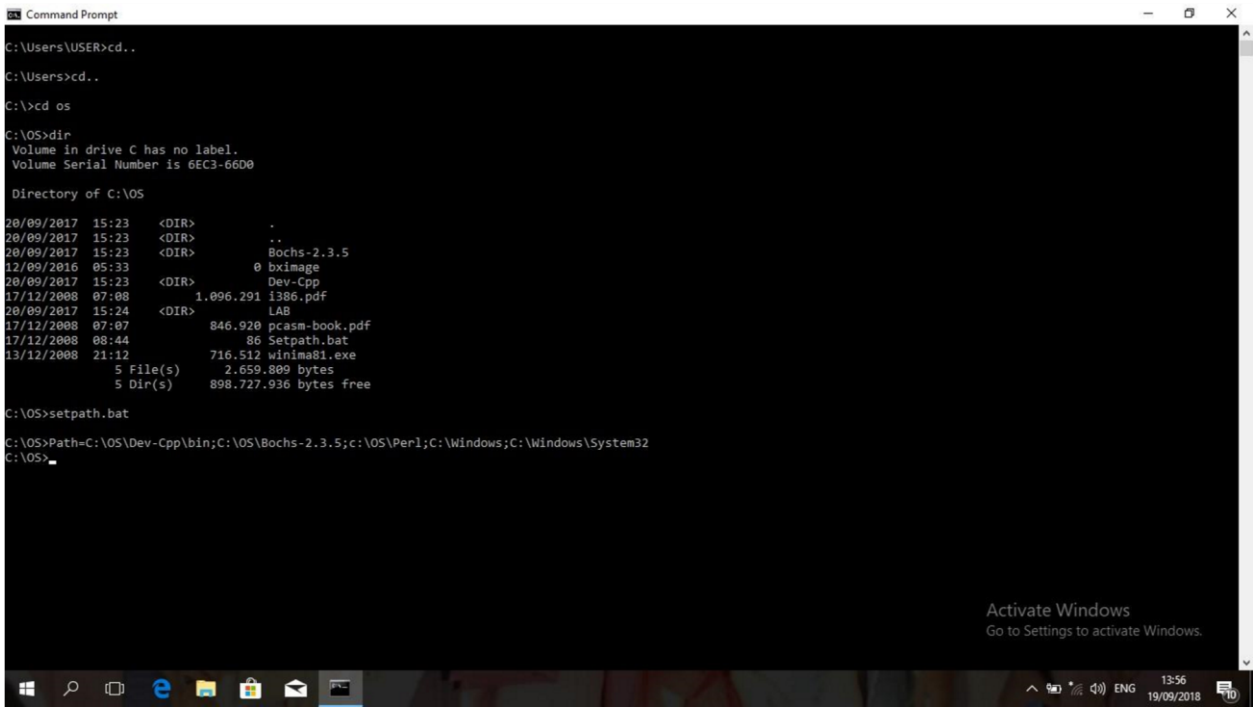


Nama : Helmi Lutfian  
NIM : L200170055  
Kelas : C

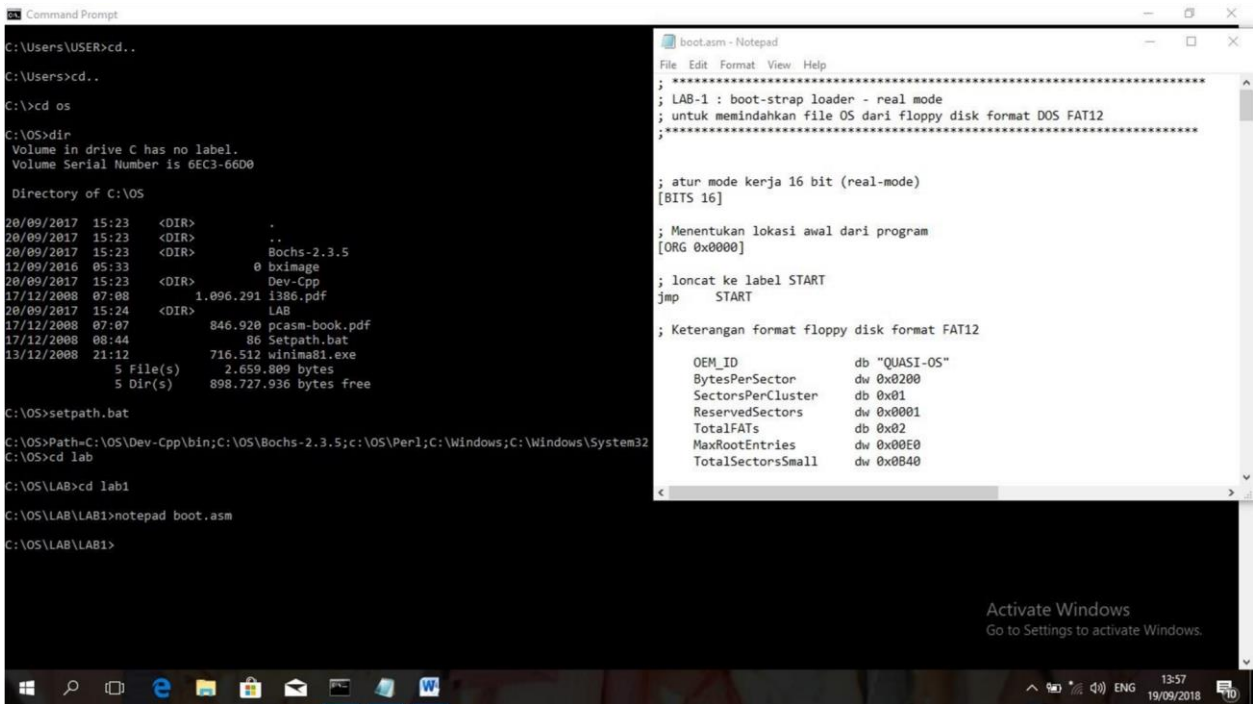
Menuju ke direktori kerja.

- Masuk ke direktori kerja „C:\OS“
- Melihat isi direktori di dalam folder OS
- Menjalankan file setpath untuk mengatur lingkungan kerja („path“)

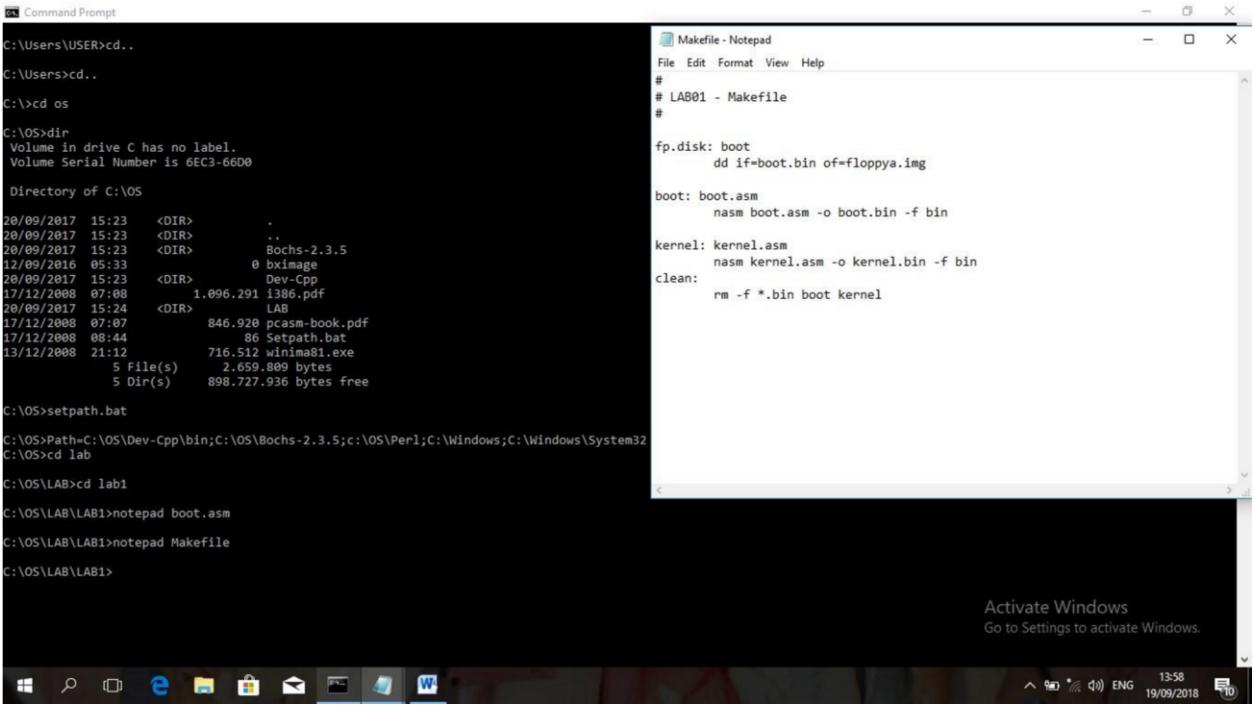


Melihat isi direktori kerja

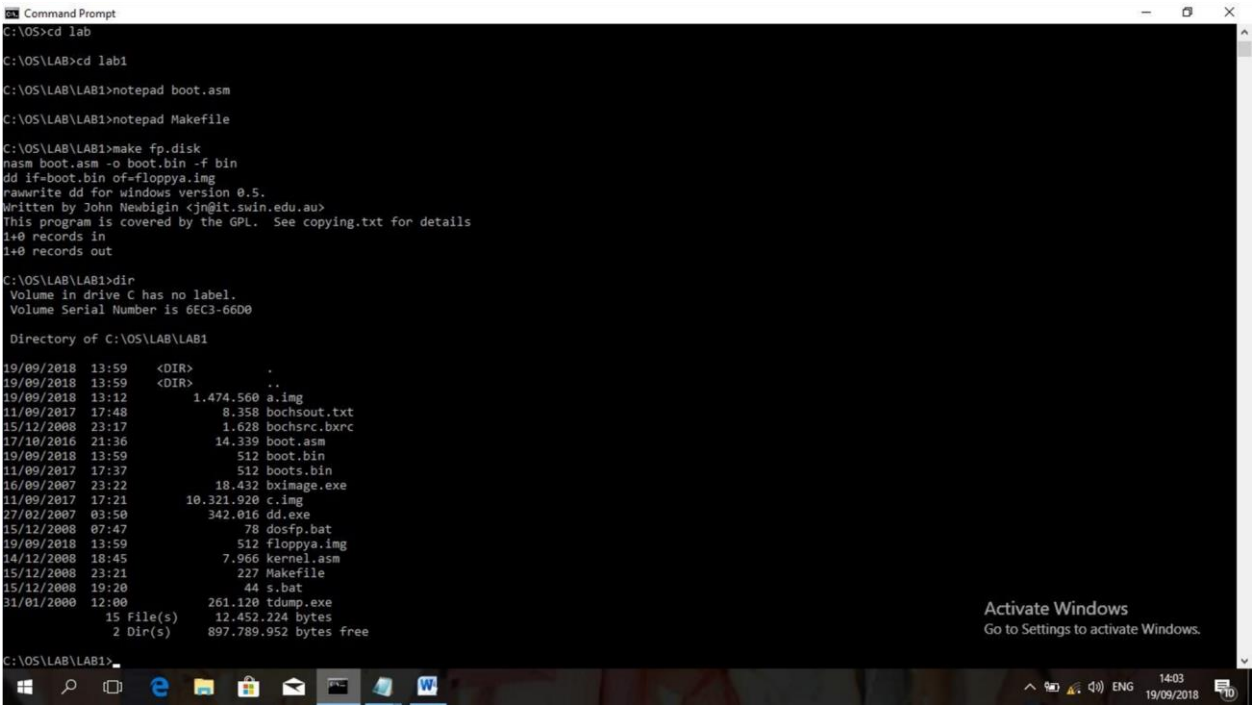
- Masuk ke direktori kerja pada „C:\OS\LAB\LAB1“.
- Membuka file dengan mengetikkan „Notepad boot.asm“



- Membuka file „Makefile“ dengan mengetikkan „Notepad Makefile“ untuk mengetahui script makefile

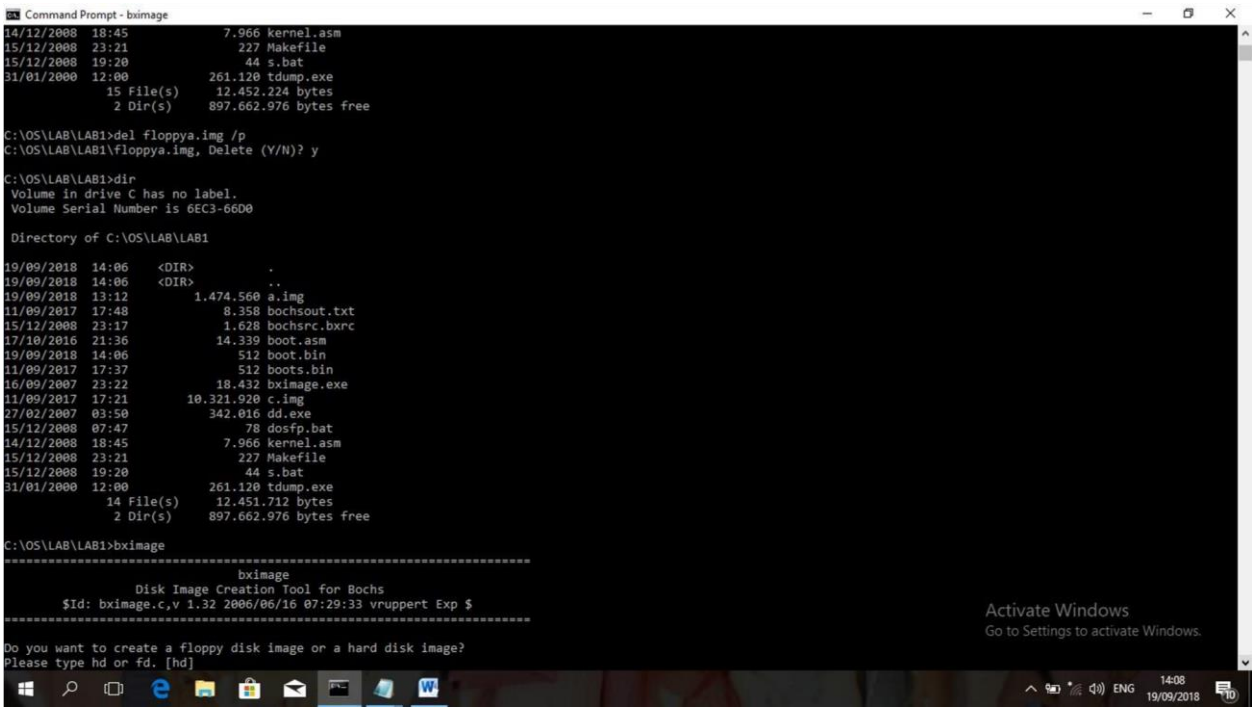


- Mengatur proses file makefile dengan mengetikkan „make fp.disk“
- Memeriksa hasil kompilasi dengan perintah „dir“



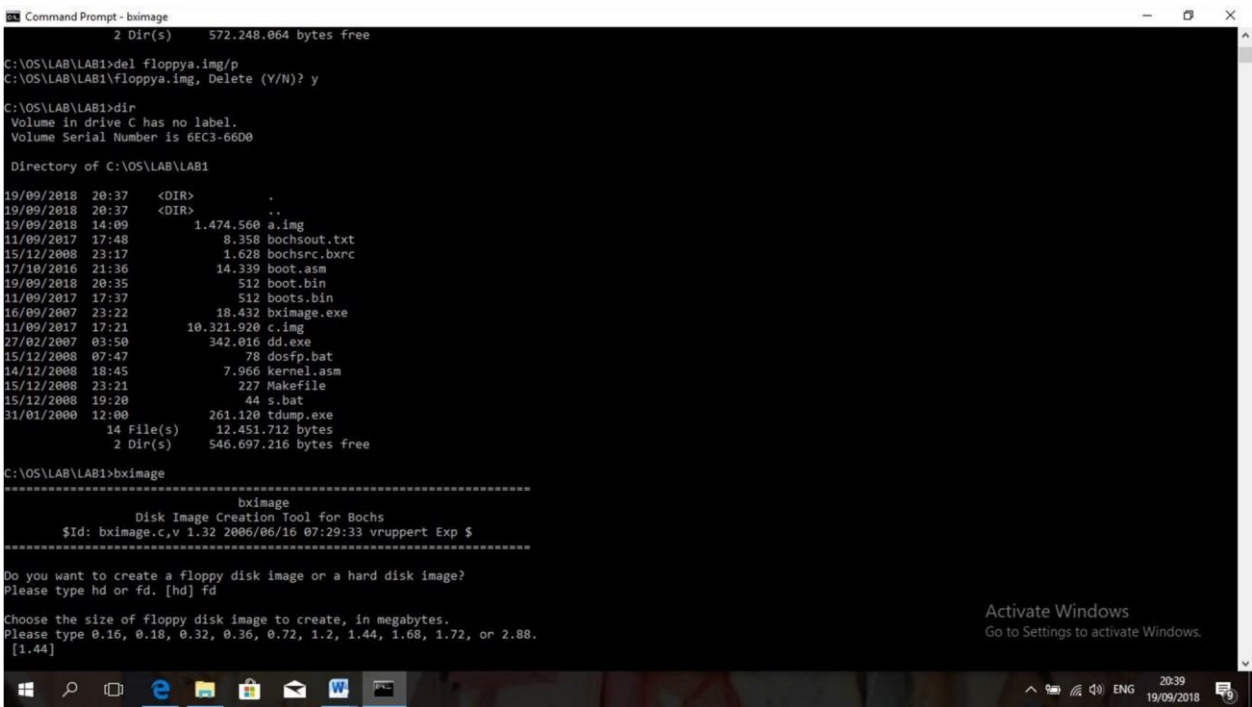
## ‘BOOT DISK’ Membuat file image floppy tahap-1

- Menghapus file „floppy.img“ dengan mengetikkan „del.floppya.img/p“ dan melanjutkan dengan tekan „Y“
  - Memastikan file sudah benar – benar terhapus dengan perintah „dir“
- Memanggil „bimage“



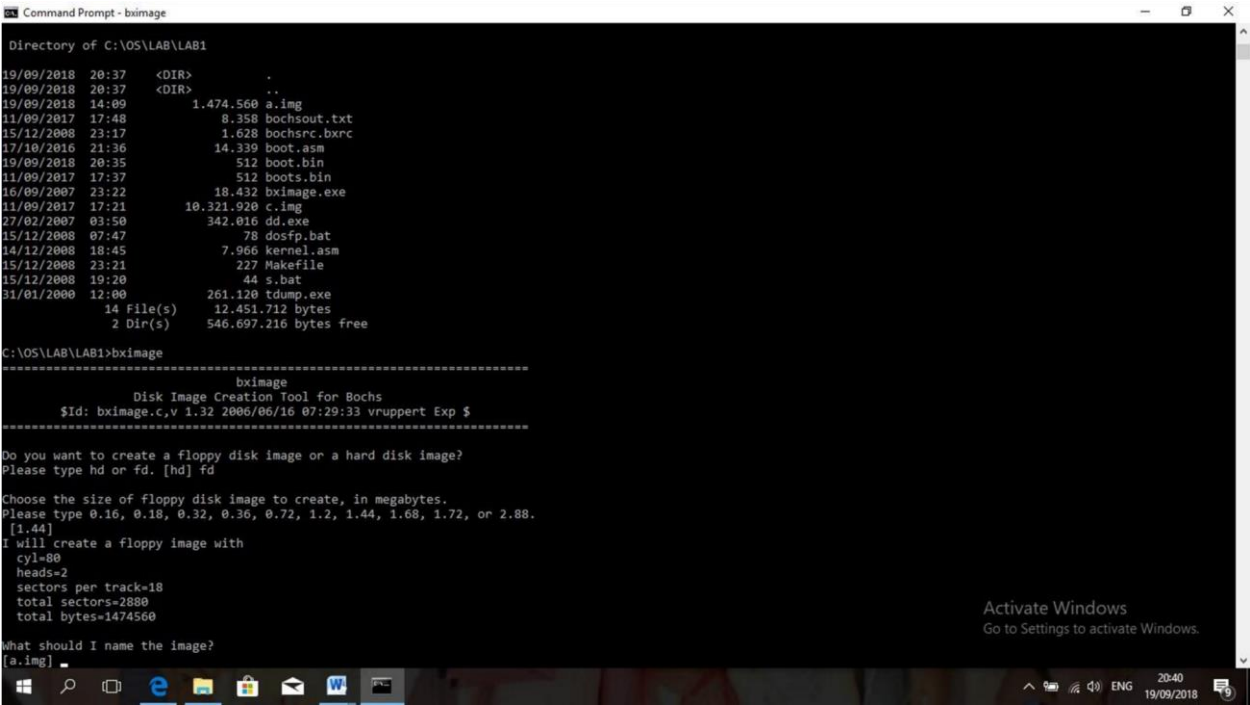
Membuat file image floppy tahap-2

- Membuat floppy image dengan mengetikkan „fd“



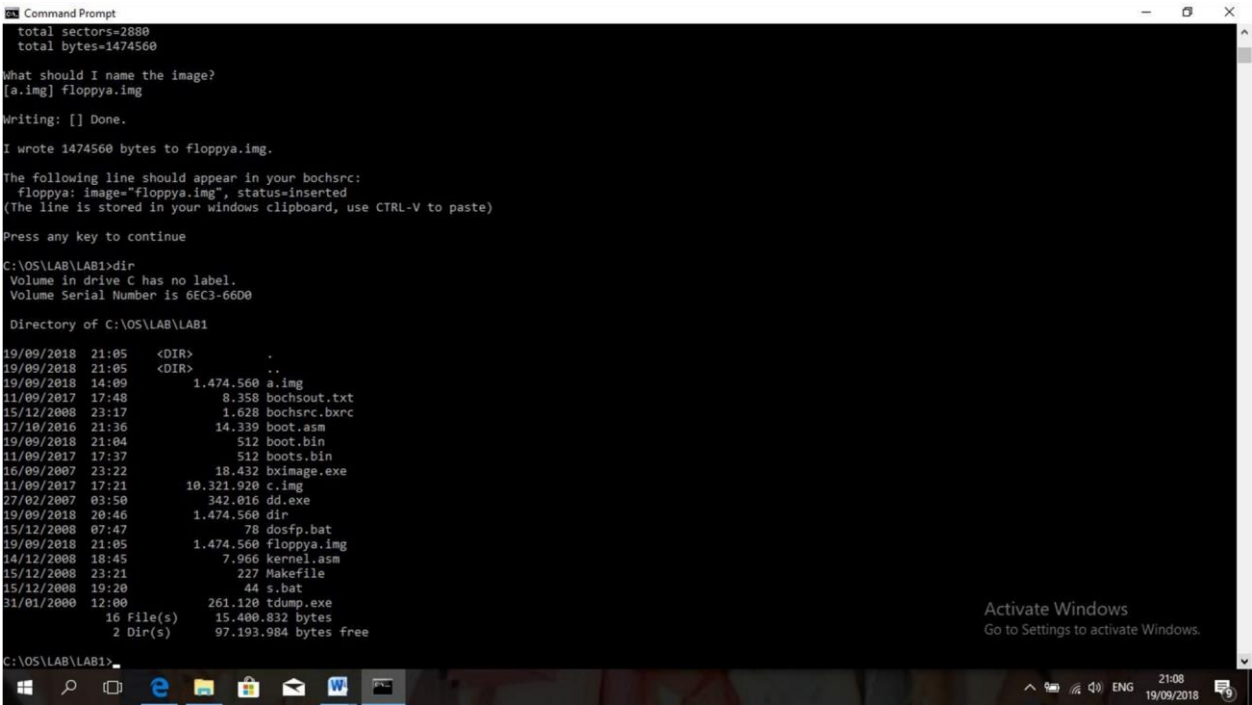
Membuat file image floppy tahap-3

- Memilih tipe floppy dengan kapasitas „1.44MB“, ditujukan oleh angka [1.44]



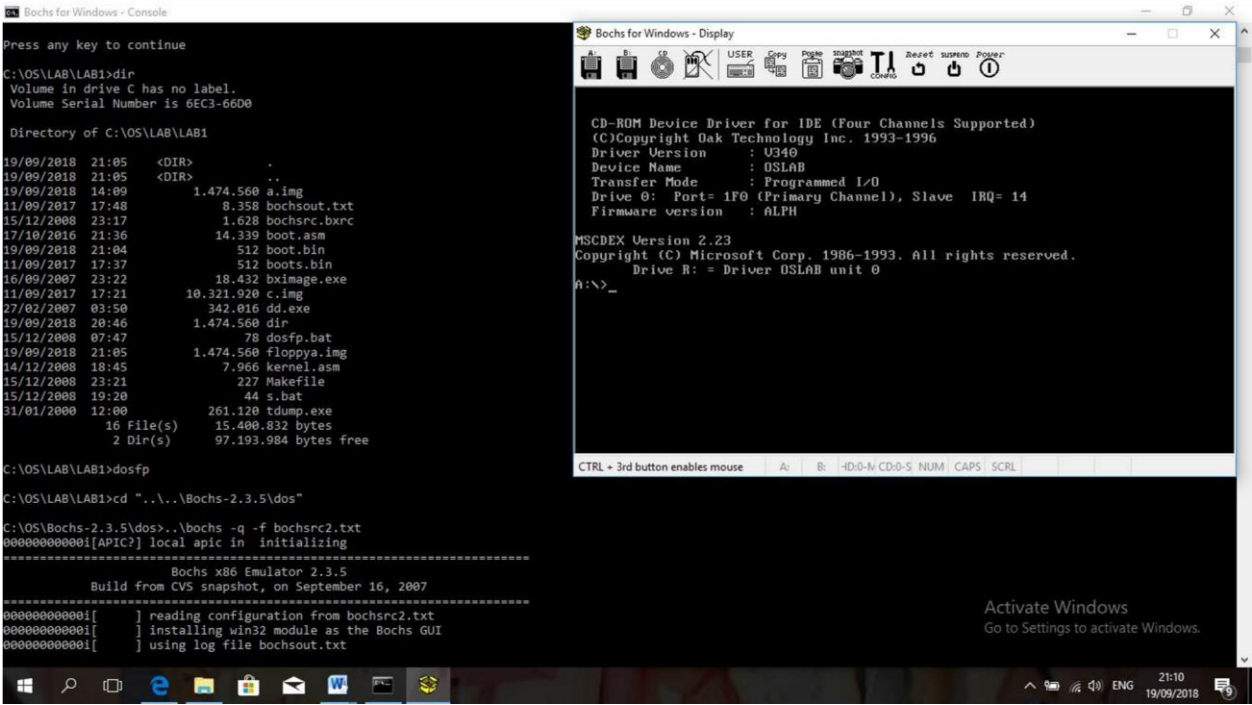
### Membuat file image floppy tahap akhir

- Memberikan nama file dengan mengetikkan „floppya.img“
- Memastikan keberadaan file image dengan perintah „dir“

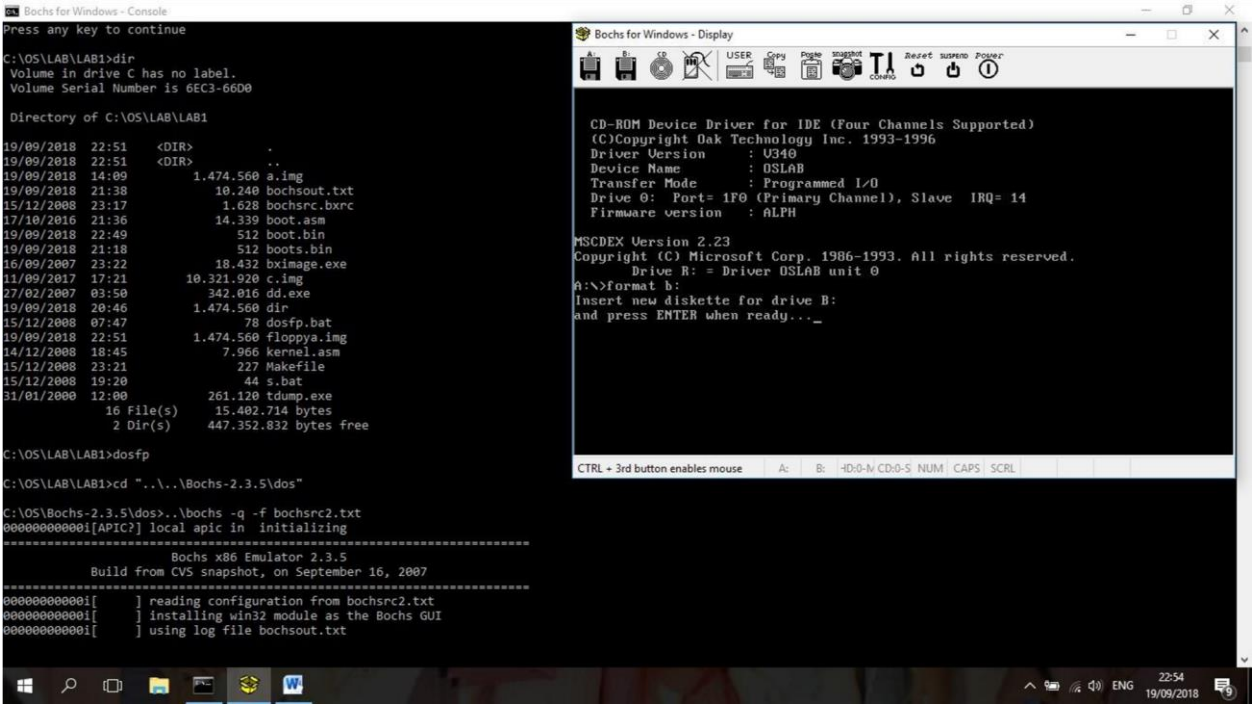


### Memformat ‘floppya.img’

- Menjalankan PC-Simulator dengan perintah „DosFp“



- Mengetikkan „Format B:“ pada prompt „A:>“



- Menutup kembali PC-Simulator dengan klik tombol power



```
Command Prompt
19/09/2018 22:51 <DIR> .
19/09/2018 22:51 <DIR> ..
19/09/2018 14:09 1.474.560 a.img
19/09/2018 21:38 10.240 bochsout.txt
15/12/2008 23:17 1.628 bochsrc.bxrc
17/10/2016 21:36 14.339 boot.asm
19/09/2018 22:49 512 boot.bin
19/09/2018 21:18 512 boots.bin
16/09/2007 23:22 18.432 bximage.exe
11/09/2017 17:21 10.321.920 c.img
27/02/2007 03:50 342.016 dd.exe
19/09/2018 20:46 1.474.560 dir
15/12/2008 07:47 78 dosfp.bat
19/09/2018 22:51 1.474.560 floppy.img
14/12/2008 18:45 7.966 kernel.asm
15/12/2008 23:21 227 Makefile
15/12/2008 19:20 44 s.bat
31/01/2000 12:00 261.120 tdump.exe
16 File(s) 15.402.714 bytes
2 Dir(s) 447.352.832 bytes free

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
000000000000[APIC?] local apic in initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
000000000000[ ] reading configuration from bochsrc2.txt
000000000000[ ] installing win32 module as the Bochs GUI
000000000000[ ] using log file bochsout.txt
# In bx_win32_gui.c::exit(void)
=====
Bochs is exiting with the following message:
[MGUI ] POWER button turned off.
=====
C:\OS\Bochs-2.3.5\dos>cd "C:\os\lab\lab1"
C:\OS\LAB\LAB1>
```

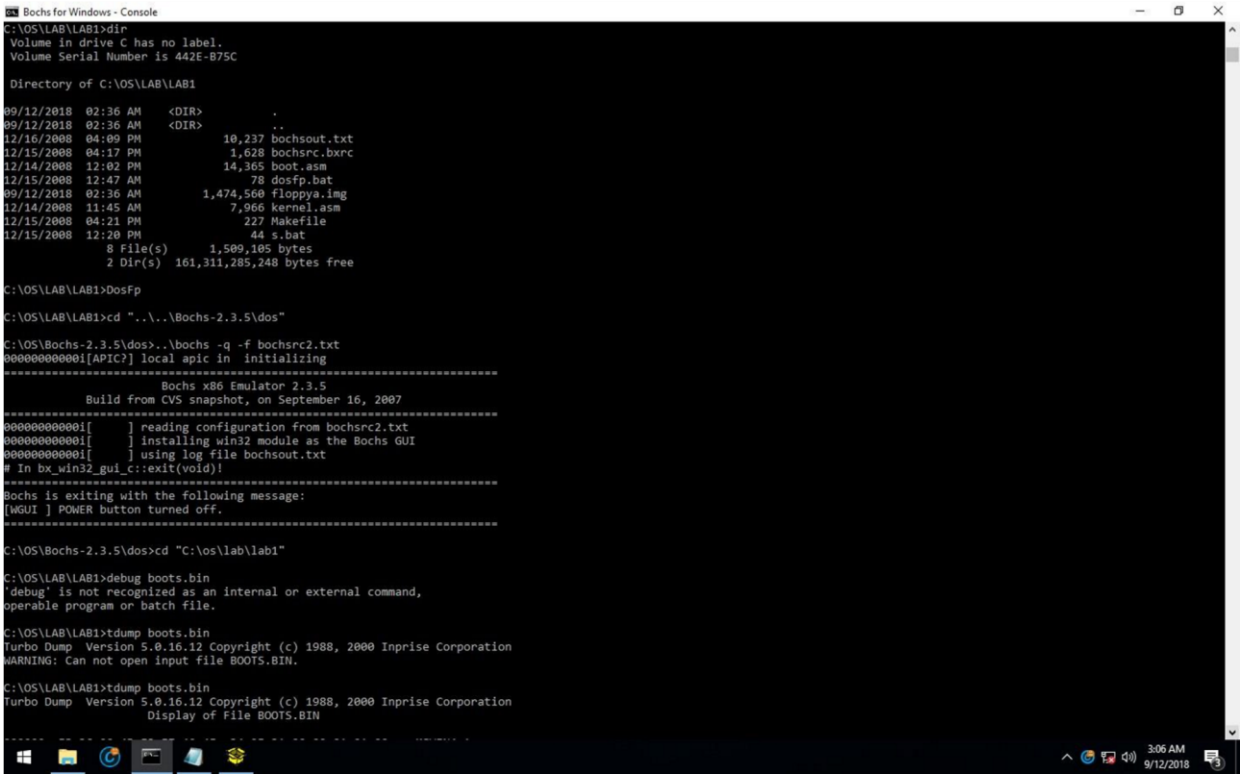
Melihat data dalam boot sector

- Memeriksa data bootsector pada file image floppy dengan mengetikkan „dd if=floppy.img of=boots.bin count=1“

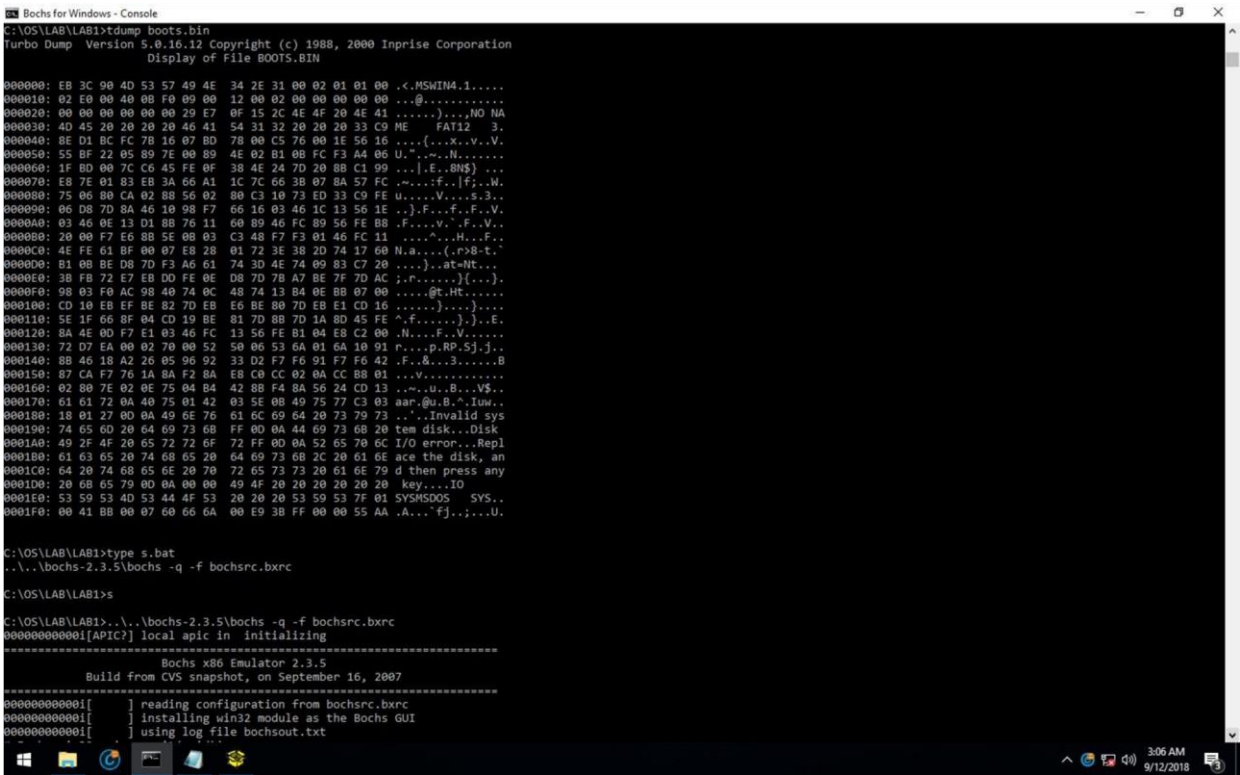
```
Command Prompt
19/09/2018 22:49 512 boot.bin
19/09/2018 21:18 512 boots.bin
16/09/2007 23:22 18.432 bximage.exe
11/09/2017 17:21 10.321.920 c.img
27/02/2007 03:50 342.016 dd.exe
19/09/2018 20:46 1.474.560 dir
15/12/2008 07:47 78 dosfp.bat
19/09/2018 22:51 1.474.560 floppy.img
14/12/2008 18:45 7.966 kernel.asm
15/12/2008 23:21 227 Makefile
15/12/2008 19:20 44 s.bat
31/01/2000 12:00 261.120 tdump.exe
16 File(s) 15.402.714 bytes
2 Dir(s) 447.352.832 bytes free

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
000000000000[APIC?] local apic in initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
000000000000[ ] reading configuration from bochsrc2.txt
000000000000[ ] installing win32 module as the Bochs GUI
000000000000[ ] using log file bochsout.txt
# In bx_win32_gui.c::exit(void)
=====
Bochs is exiting with the following message:
[MGUI ] POWER button turned off.
=====
C:\OS\Bochs-2.3.5\dos>cd "C:\os\lab\lab1"
C:\OS\LAB\LAB1>dd if=floppy.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>
```

- Mengetikkan „debug boots.bin“



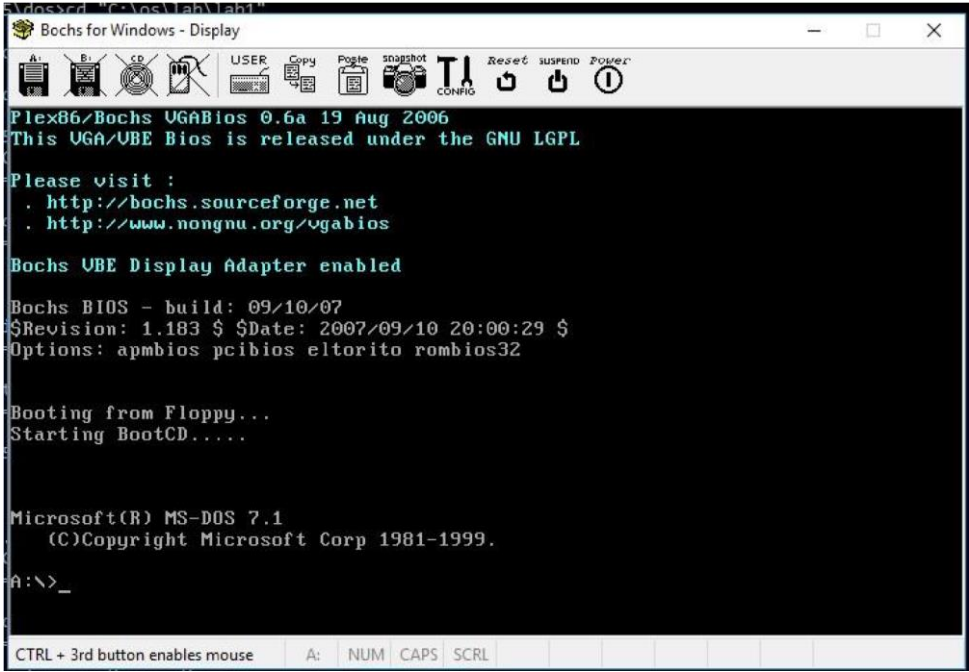
- Mengetikkan „tdump boots.bin“ untuk menampilkan data dalam bootsector file image floppy.img
- S.bat berisi dua baris perintah untuk memanggil PC-simulator „bochs“
- Ketikkan s akan menampilkan windows „bochs for windows – display“ yang sedang melakukan proses booting namun gagal karena tidak menemukan diskboot



- Memformat floppy.img dan menambahkan system file dengan mengetikkan DosFP
- Masukkan format B:/S untuk melakukan proses format

```
Bochs for Windows - Console
C:\OS\LAB\LAB1>
C:\OS\LAB\LAB1>..\..\bochs-2.3.5\bochs -q -f bochsrc.bxrc
000000000000[APIC?] local apic in  initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
000000000000[   ] reading configuration from bochsrc.bxrc
000000000000[   ] installing win32 module as the Bochs GUI
000000000000[   ] using log file bochsout.txt
# In bx_win32_gui_c::exit(void)!
=====
Bochs is exiting with the following message:
[MGUI ] POWER button turned off.
=====
C:\OS\LAB\LAB1>DosFp
C:\OS\LAB\LAB1>cd "..\..\Bochs-2.3.5\dos"
C:\OS\Bochs-2.3.5\dos>..\bochs -q -f bochsrc2.txt
000000000000[APIC?] local apic in  initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
000000000000[   ] reading configuration from bochsrc2.txt
000000000000[   ] installing win32 module as the Bochs GUI
000000000000[   ] using log file bochsout.txt
# In bx_win32_gui_c::exit(void)!
=====
Bochs is exiting with the following message:
[MGUI ] POWER button turned off.
=====
C:\OS\Bochs-2.3.5\dos>cd "C:\os\lab\lab1"
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
000000000000[APIC?] local apic in  initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
000000000000[   ] reading configuration from bochsrc2.txt
000000000000[   ] installing win32 module as the Bochs GUI
000000000000[   ] using log file bochsout.txt
# In bx_win32_gui_c::exit(void)!
=====
Bochs is exiting with the following message:
[MGUI ] POWER button turned off.
=====
```

- Jika sudah berhasil maka akan muncul seperti pada di atas



TUGAS 1. ASCII

**Kode Standar Amerika untuk Pertukaran Informasi** atau *American Standard Code for Information Interchange* (ASCII) merupakan suatu standar internasional dalam kode huruf dan simbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|".

Tabel Kode ASCII

Karakter	Nilai Unicode (heksadesimal)	Nilai ANSI ASCII (desimal)	Keterangan
----------	---------------------------------	----------------------------------	------------



NUL	0000	0	Null (tidak tampak)
SOH	0001	1	Start of heading (tidak tampak)
STX	0002	2	Start of text (tidak tampak)
ETX	0003	3	
EOT	0004	4	End of transmission (tidak tampak)
ENQ	0005	5	Enquiry (tidak tampak)
ACK	0006	6	Acknowledge (tidak tampak)
BEL	0007	7	
BS	0008	8	Menghapus satu karakter di belakang kursor (Backspace)
HT	0009	9	
LF	000A	10	Pergantian baris (Line feed)
VT	000B	11	Tabulasi vertical
FF	000C	12	Pergantian baris (Form feed)
CR	000D	13	Pergantian baris (carriage return)

SO	000E	14	Shift out (tidak tampak)
SI	000F	15	Shift in (tidak tampak)
DLE	0010	16	Data link escape (tidak tampak)
DC1	0011	17	Device control 1 (tidak tampak)
DC2	0012	18	Device control 2 (tidak tampak)
DC3	0013	19	Device control 3 (tidak tampak)
DC4	0014	20	Device control 4 (tidak tampak)
NAK	0015	21	Negative acknowledge (tidak tampak)
SYN	0016	22	Synchronous idle (tidak tampak)
ETB	0017	23	End of transmission block (tidak tampak)
CAN	0018	24	Cancel (tidak tampak)
EM	0019	25	End of medium (tidak tampak)
SUB	001A	26	Substitute (tidak tampak)
ESC	001B	27	Escape (tidak tampak)
FS	001C	28	File separator
GS	001D	29	Group separator

RS	001E	30	Record separator
----	------	----	------------------

US	001F	31	Unit separator
SP	0020	32	Spasi
!	0021	33	Tanda seru (exclamation)
"	0022	34	Tanda kutip dua
#	0023	35	Tanda pagar (kres)
\$	0024	36	Tanda mata uang dolar
%	0025	37	Tanda persen
&	0026	38	Karakter ampersand (&)
„	0027	39	Karakter Apostrof
(	0028	40	Tanda kurung buka
)	0029	41	Tanda kurung tutup
*	002A	42	Karakter asterisk (bintang)
+	002B	43	Tanda tambah (plus)
,	002C	44	Karakter koma
-	002D	45	Karakter hyphen (strip)

.	002E	46	Tanda titik
/	002F	47	Garis miring ( <i>slash</i> )
0	0030	48	Angka nol

1	0031	49	Angka satu
2	0032	50	Angka dua
3	0033	51	Angka tiga
4	0034	52	Angka empat
5	0035	53	Angka lima
6	0036	54	Angka enam
7	0037	55	Angka tujuh
8	0038	56	Angka delapan
9	0039	57	Angka Sembilan
:	003A	58	Tanda titik dua
;	003B	59	Tanda titik koma
<	003C	60	Tanda lebih kecil
=	003D	61	Tanda sama dengan

>	003E	62	Tanda lebih besar
?	003F	63	Tanda Tanya
@	0040	64	A keong (@)
A	0041	65	Huruf latin A capital
B	0042	66	Huruf latin B capital

C	0043	67	Huruf latin C capital
D	0044	68	Huruf latin D capital
E	0045	69	Huruf latin E capital
F	0046	70	Huruf latin F capital
G	0047	71	Huruf latin G capital
H	0048	72	Huruf latin H capital
I	0049	73	Huruf latin I capital
J	004A	74	Huruf latin J capital
K	004B	75	Huruf latin K capital
L	004C	76	Huruf latin L capital
M	004D	77	Huruf latin M capital



N	004E	78	Huruf latin N capital
O	004F	79	Huruf latin O capital
P	0050	80	Huruf latin P capital
Q	0051	81	Huruf latin Q capital
R	0052	82	Huruf latin R capital
S	0053	83	Huruf latin S capital
T	0054	84	Huruf latin T capital

U	0055	85	Huruf latin U capital
V	0056	86	Huruf latin V capital
W	0057	87	Huruf latin W capital
X	0058	88	Huruf latin X capital
Y	0059	89	Huruf latin Y capital
Z	005A	90	Huruf latin Z capital
[	005B	91	Kurung siku kiri
\	005C	92	Garis miring terbalik ( <i>backslash</i> )
]	005D	93	Kurung sikur kanan

^	005E	94	Tanda pangkat
	005F	95	Garis bawah (underscore)
`	0060	96	Tanda petik satu
a	0061	97	Huruf latin a kecil
b	0062	98	Huruf latin b kecil
	0063	99	Huruf latin c kecil
d	0064	100	Huruf latin d kecil
e	0065	101	Huruf latin e kecil
f	0066	102	Huruf latin f kecil

g	0067	103	Huruf latin g kecil
h	0068	104	Huruf latin h kecil
	0069	105	Huruf latin i kecil
j	006A	106	Huruf latin j kecil
k	006B	107	Huruf latin k kecil
l	006C	108	Huruf latin l kecil
	006D	109	Huruf latin m kecil

n	006E	110	Huruf latin n kecil
o	006F	111	Huruf latin o kecil
p	0070	112	Huruf latin p kecil
q	0071	113	Huruf latin q kecil
r	0072	114	Huruf latin r kecil
s	0073	115	Huruf latin s kecil
t	0074	116	Huruf latin t kecil
u	0075	117	Huruf latin u kecil
v	0076	118	Huruf latin v kecil
w	0077	119	Huruf latin w kecil
x	0078	120	Huruf latin x kecil

y	0079	121	Huruf latin y kecil
z	007A	122	Huruf latin z kecil
{	007B	123	Kurung kurawal buka
	007C	124	Garis vertikal (pipa)
}	007D	125	Kurung kurawal tutup

~	007E	126	Karakter gelombang (tilde)
DEL	007F	127	Delete
	0080	128	Dicadangkan
	0081	129	Dicadangkan
	0082	130	Dicadangkan
	0083	131	Dicadangkan
IND	0084	132	Index
NEL	0085	133	Next line
SSA	0086	134	Start of selected area
ESA	0087	135	End of selected area
	0088	136	Character tabulation set
	0089	137	Character tabulation with justification
	008A	138	Line tabulation set

PLD	008B	139	Partial line down
PLU	008C	140	Partial line up
	008D	141	Reverse line feed

SS2	008E	142	Single shift two
SS3	008F	143	Single shift three
DCS	0090	144	Device control string
PU1	0091	145	Private use one
PU2	0092	146	Private use two
STS	0093	147	Set transmit state
CCH	0094	148	Cancel character
MW	0095	149	Message waiting
	0096	150	Start of guarded area
	0097	151	End of guarded area
	0098	152	Start of string
	0099	153	Dicadangkan
	009A	154	Single character introducer
CSI	009B	155	Control sequence introducer
ST	009C	156	String terminator

OSC	009D	157	Operating system command
-----	------	-----	--------------------------



PM	009E	158	Privacy message
APC	009F	158	Application program command
	00A0	160	Spasi yang bukan pemisah kata
¡	00A1	161	Tanda seru terbalik
¢	00A2	162	Tanda sen (Cent)
£	00A3	163	Tanda Poundsterling
¤	00A4	164	Tanda mata uang ( <i>Currency</i> )
¥	00A5	165	Tanda Yen
¦	00A6	166	Garis tegak putus-putus ( <i>broken bar</i> )
§	00A7	167	Section sign
¨	00A8	168	Diaeresis
©	00A9	169	Tanda hak cipta (Copyright)
ª	00AA	170	Feminine ordinal indicator
«	00AB	171	Left-pointing double angle quotation mark
¬	00AC	172	Not sign
	00AD	173	Tanda strip ( <i>hyphen</i> )

®	00AE	174	Tanda merk terdaftar
-	00AF	175	Macron
°	00B0	176	Tanda derajat
±	00B1	177	Tanda kurang lebih (plus-minus)
²	00B2	178	Tanda kuadrat (pangkat dua)
³	00B3	179	Tanda kubik (pangkat tiga)
´	00B4	180	Acute accent
μ	00B5	181	Micro sign
¶	00B6	182	Pilcrow sign
·	00B7	183	Middle dot

2. DAFTAR PERINTAH BAHASA ASSEMBLY X86

1. ACALL (Absolute Call)  
ACALL berfungsi untuk memanggil sub rutin program
2. ADD (Add Immediate Data)  
ADD berfungsi untuk menambah 8 bit data langsung ke dalam isi akumulator dan menyimpan hasilnya pada akumulator.
3. 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.
4. 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 diload dengan a10 – a0 11 bits, untuk membentuk alamat tujuan 16-bit.
5. ANL (logical AND memori ke akumulator)  
ANL berfungsi untuk mengAND-kan isi alamat data dengan isi akumulator.
6. 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.
7. CLR (Clear Accumulator)  
CLR berfungsi untuk mereset data akumulator menjadi 00H.  
Format : CLR A
8. CPL (Complement Accumulator)  
CPL berfungsi untuk mengkomplemen isi akumulator.
9. DA (Decimal Adjust Accumulator)  
DA berfungsi untuk mengatur isi akumulator ke padanan BCD, steleah penambahan dua angka BCD.
10. DEC (Decrement Indirect Address)  
DEC berfungsi untuk mengurangi isi lokasi memori yang ditujukan oleh register R dengan 1, dan hasilnya disimpan pada lokasi tersebut.
11. DIV (Divide Accumulator by B)  
DIV berfungsi untuk membagi isi akumulator dengan isi register B. Akumulator berisi hasil bagi, register B berisi sisa pembagian.
12. 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.
13. INC (Increment Indirect Address)  
INC berfungsi untuk menambahkan isi memori dengan 1 dan menyimpannya pada alamat tersebut.
14. 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.
15. 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.

16. 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.

17. JMP (Jump to sum of Accumulator and Data Pointer)

Instruksi JMP berfungsi untuk memerintahkan loncat kesuatu alamat kode tertentu. Format : JMP alamat kode.

18. 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.

19. 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.

20. 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.

21. 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.

22. LCALL ( Long Call )

LCALL berfungsi untuk memungkinkan panggilan ke subrutin yang berlokasi dimanapun dalam memori program 64K. Operasi LCALL berjalan seperti berikut: · Menambahkan ke dalam konter program sebanyak 3, karena perintahnya adalah perintah 3-byte.

· Menambahkan penunjuk stack sebanyak 1.

· Menyimpan byte yang lebih rendah dari konter program ke dalam stack.

· Menambahkan penunjuk stack.

· Menyimpan byte yang lebih tinggi dari program ke dalam stack. · Me-load konter program dengan alamat tujuan 16-bit.

23. 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 16bit ditentukan secara langsung dalam perintah tersebut. Alamat tujuan ini di-load ke dalam konter program oleh perintah LJMP.

24. MOV ( Move From Memory )

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

25. 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.

26. 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.

27. 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.

28. NOP ( No Operation )

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

29. 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.

30. POP (Pop Stack to Memory)

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

31. 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.

32. RET (Return from subroutine)

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

33. 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.

34. RL (Rotate Accumulator Left)

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

35. 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.

36. 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.

37. 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 bersamasama. Bit 0 akan dirotasi ke carry flag, nilai carry flag akan berpindah ke posisi bit 7. Instruksi ini tidak mempengaruhi flag lain.

38. SETB (set Carry flag)

Instruksi SETB berfungsi untuk menset carry flag.

39. 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“.

40. 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.

41. 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.

42. 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 mengisikan nilai accumulator ke dalam variabel tersebut. Addressing mode yang dapat digunakan adalah: register, direct, atau register indirect.

43. 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.

44. 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.