

Pratikum Sistem Operasi



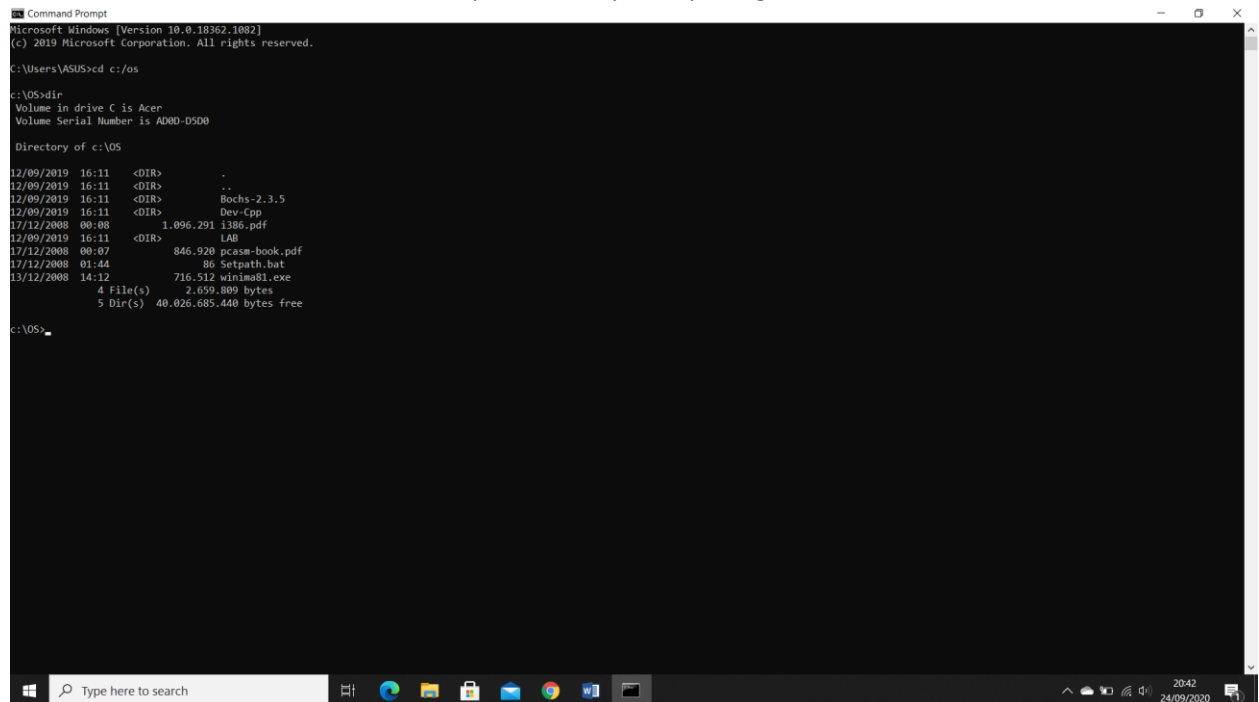
Nama : Mohammad Aziz Nurritanto

NIM : L200190242

Kelas : G

Langkah kerja

Menuju ke direktori kerja. a. Jalankan program command prompt atau cmd. b. Masuk ke direktori kerja 'C:\OS', dengan perintah 'cd os' . c. Masukan perintah dir, untuk melihat isi direktori di dalam folder tersebut. Akan muncul seperti di tampilan pada gambar.



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

C:\Users\ASUS>cd c:/os

C:\OS>dir
Volume in drive C is Acer
Volume Serial Number is AD00-D5D0

Directory of c:\OS

12/09/2019  16:11    <DIR>          .
12/09/2019  16:11    <DIR>          ..
12/09/2019  16:11    <DIR>          Bochs-2.3.5
12/09/2019  16:11    <DIR>          Dev-Cpp
17/12/2008  00:08                1.096.291  1386.pdf
12/09/2019  16:11    <DIR>          LAB
17/12/2008  00:07                846.920  pcasm-book.pdf
17/12/2008  01:44                86 Setpath.bat
13/12/2008  14:12                716.512  winima81.exe
               4 File(s)                2.699.809 bytes
               5 Dir(s)  40.026.685.440 bytes free

C:\OS>
```

d. Jalankan file setpath, untuk menjalankannya ketik 'setpath' tekan File 'setpath.bat' digunakan untuk mengatur lingkungan kerja ('path') selama anda melakukan praktikum, anda harus menjalankan program ini sebelum memulai setiap sesi praktikum anda, untuk menjalankannya ketik 'setpath' tekan . Perhatikan teks yang muncul di layar, seperti ditampilkan pada Gambar 1.1. Untuk melihat script yang terdapat di dalam file 'setpath.bat' dapat digunakan perintah 'type setpath. bat', cobalah. 'PATH' dapat bersisi banyak daftar lokasi, di antara item lokasi di batasi dengan karakter ';' (titik koma). Variabel 'PATH' ini akan digunakan oleh windows untuk mencari lokasi file yang dipanggil oleh 'user'. Urutan pencarian dimulai dari daftar lokasi yang paling awal.

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

C:\Users\ASUS>cd c:/os

C:\OS>dir
Volume in drive C is Acer
Volume Serial Number is AD00-D5D0

Directory of c:\OS

12/09/2019  16:11    <DIR>      .
12/09/2019  16:11    <DIR>      ..
12/09/2019  16:11    <DIR>      Bochs-2.3.5
12/09/2019  16:11    <DIR>      Dev-Cpp
17/12/2008  00:08                1.096.291  i386.pdf
12/09/2019  16:11    <DIR>      LAB
17/12/2008  00:07                846.920  pcasm-book.pdf
17/12/2008  01:44                86 Setpath.bat
13/12/2008  14:12                716.512  winima81.exe
               4 File(s)                2.659.809 bytes
               5 Dir(s)  40.026.685.440 bytes free

C:\OS>setpath
C:\OS>Path=C:\OS\Dev-Cpp\bin;C:\OS\Bochs-2.3.5;c:\OS\Per1;C:\Windows;C:\Windows\System32
C:\OS>
```

pertama kali yang dilakukan Windows adalah mencari file program di dalam direktori kerja saat itu, jika di sana tidak ditemukan file yang dimaksud, selanjutnya Windows akan meneruskan pencarian file di lokasi yang terdaftar pada variabel 'PATH' dimulai dari lokasi 'C:\OS\DevCpp\bin' jika file belum ditemukan pencarian dilanjutkan ke lokasi 'C:\Windows', pencarian terus dilakukan sampai ke lokasi terakhir 'C:\Windows\system32'.

```
Command Prompt
C:\Users\ASUS>cd c:/os

C:\OS>dir
Volume in drive C is Acer
Volume Serial Number is AD00-D5D0

Directory of c:\OS

12/09/2019  16:11    <DIR>      .
12/09/2019  16:11    <DIR>      ..
12/09/2019  16:11    <DIR>      Bochs-2.3.5
12/09/2019  16:11    <DIR>      Dev-Cpp
17/12/2008  00:08                1.096.291  i386.pdf
12/09/2019  16:11    <DIR>      LAB
17/12/2008  00:07                846.920  pcasm-book.pdf
17/12/2008  01:44                86 Setpath.bat
13/12/2008  14:12                716.512  winima81.exe
               4 File(s)                2.659.809 bytes
               5 Dir(s)  40.026.685.440 bytes free

C:\OS>setpath
C:\OS>Path=C:\OS\Dev-Cpp\bin;C:\OS\Bochs-2.3.5;c:\OS\Per1;C:\Windows;C:\Windows\System32
C:\OS>cd lab

C:\OS\LAB>cd lab1

C:\OS\LAB\LAB1>dir
Volume in drive C is Acer
Volume Serial Number is AD00-D5D0

Directory of c:\OS\LAB\LAB1

24/09/2020  19:07    <DIR>      .
24/09/2020  19:07    <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
24/09/2020  19:07                512 boot.bin
16/09/2015  07:51                512 boots.bin
24/09/2020  14:41            10.321.920  c.img
15/12/2008  00:47                78 dosfp.bat
24/09/2020  19:08            1.474.560  floppy.aimg
14/12/2008  11:45                7.966  kernel.asm
15/12/2008  16:21                227 Makefile
15/12/2008  12:20                44 s.bat
               11 File(s)            11.832.047 bytes
               2 Dir(s)  40.028.209.152 bytes free

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

Perhatikan baris menu 'fp.disk: boot', baris ini memerintahkan pada komputer untuk mengerjakan baris perintah di bawah menu tersebut, tetapi setelah baris perintah yang terdapat

di bawah menu 'boot:' dikerjakan. Jadi jika dilihat menu 'fp.disk' maka pertama kali 'make.exe' akan mengerjakan baris perintah yang terdapat di bawah label menu 'boot:'

```
Command Prompt
12/09/2019 16:11 <DIR> Dev-Cpp
17/12/2008 00:08 1.096.291 1386.pdf
12/09/2019 16:11 LAB
17/12/2008 00:07 846.920 pcasm-book.pdf
17/12/2008 01:44 86 Setpath.bat
13/12/2008 14:12 716.512 winima81.exe
4 File(s) 2.659.809 bytes
5 Dir(s) 40.026.685.440 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 is Acer
Volume Serial Number is A000-D5D0

Directory of c:\OS\LAB\LAB1

24/09/2020 19:07 <DIR> .
24/09/2020 19:07 <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
24/09/2020 19:07 512 boot.bin
16/09/2015 07:51 512 boots.bin
24/09/2020 14:41 10.321.920 c.img
15/12/2008 00:47 78 dosfp.bat
24/09/2020 19:08 1.474.568 floppy.a.img
14/12/2008 11:45 7.966 kernel.asm
15/12/2008 16:21 227 Makefile
15/12/2008 12:20 44 s.bat
11 File(s) 11.832.047 bytes
2 Dir(s) 40.028.209.152 bytes free

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

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

Periksa hasil kompilasi dengan memasukan perintah 'dir', sekarang pada direktori kerja terdapat tambahan file baru, yaitu 'boot.bin' dan isinya sudah disalin kedalam bootsector 'floppy.a.img'.

```
Command Prompt - bximage
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 is Acer
Volume Serial Number is A000-D5D0

Directory of c:\OS\LAB\LAB1

24/09/2020 19:07 <DIR> .
24/09/2020 19:07 <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
24/09/2020 19:07 512 boot.bin
16/09/2015 07:51 512 boots.bin
24/09/2020 14:41 10.321.920 c.img
15/12/2008 00:47 78 dosfp.bat
24/09/2020 19:08 1.474.568 floppy.a.img
14/12/2008 11:45 7.966 kernel.asm
15/12/2008 16:21 227 Makefile
15/12/2008 12:20 44 s.bat
11 File(s) 11.832.047 bytes
2 Dir(s) 40.028.209.152 bytes free

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

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

Selanjutnya panggil 'bximage' sehingga ditampilkan window seperti pada gambar.

```
Command Prompt - bximage
24/09/2020 19:07 <DIR> .
24/09/2020 19:07 <DIR> ..
18/09/2019 16:10 10.235 bochsout.txt
15/12/2008 16:17 1.628 bochsrc.bxrc
14/12/2008 12:02 14.365 boot.asm
24/09/2020 19:07 512 boot.bin
16/09/2015 07:51 512 boots.bin
24/09/2020 14:41 10.321.920 c.img
15/12/2008 08:47 78 dosfb.bat
24/09/2020 19:08 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
15/12/2008 12:20 11 File(s) 11.832.047 bytes
2 Dir(s) 40.028.209.152 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
rawrite 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]
```

pilih tipe yang paling banyak digunakan saat ini yaitu tipe floppy dengan kapasitas ‘1.44MB’, ditunjukkan oleh angka 12 Sistem Operasi - Modul Praktikum [1.44]. Selanjutnya tekan , untuk memilih tipe lain, masukan angka yang menunjukkan kapasitas yang anda inginkan (lihat gambar di atas)

```
Command Prompt - bximage
11 File(s) 11.832.047 bytes
2 Dir(s) 40.028.209.152 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
rawrite 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
```

File image floppy ini memiliki tipe Format FAT12, yang memiliki 80 cylinder, 2 head (atas dan bawah), 18 sektor tiap track, jumlah sektor 2880, jumlah byte data tiap sektor 512 byte sehingga total byte $2880 \times 512 = 1.474.560$ byte (1,44M).

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	\$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(88	58	1011000	130	X					
41	29	101001	51)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

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 111 1101 175 125 7D } 111 1110 176 126 7E ~ 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.

Operasi pada register flag CLC : clear carry flag. Menjadikan CF = 0. Sintaks: CLC ;(saja). STC : set carry flag. Menjadikan CF = 1. Sintaks: STC ;(saja). CMC : complement carry flag. Melakukan operasi NOT pada CF. Yang tadinya 0 menjadi 1, dan sebaliknya. CLD : clear direction flag. Menjadikan DF = 0. Sintaks: CLD ;(saja). STD : set direction flag. Menjadikan DF = 1. CLI : clear interrupt flag. Menjadikan IF = 0, sehingga interrupt ke CPU akan di-disable. Biasanya perintah CLI diberikan sebelum menjalankan sebuah proses penting yang riskan gagal bila diganggu. STI : set interrupt flag. Menjadikan IF = 1. Perintah lainnya ORG : origin. Mengatur awal dari program (bagian static data). Analoginya seperti mengatur dimana letak titik (0, 0) pada koordinat Cartesius. sintaks: ORG {alamat awal} Pada program COM (program yang berekstensi .com), harus ditulis "ORG 100h" untuk mengatur alamat mulai dari program pada 0100(hex), karena dari alamat 0000(hex) sampai 00FF(hex) sudah dipesan oleh sistem operasi (DOS). INT : interrupt. Menginterupsi prosesor. Prosesor akan: 1. Membakup data registernya saat itu, 2. Menghentikan apa yang sedang dikerjakannya, 3. Melompat ke bagian interrupt-handler (entah dimana kita tidak tahu, sudah ditentukan BIOS dan DOS), 4. Melakukan interupsi, 5. Mengembalikan data registernya, 6. Meneruskan pekerjaan yang tadi ditunda. sintaks: INT {nomor interupsi} IRET : interrupt-handler return. Kita bisa membuat interrupt-handler sendiri dengan berbagai cara. Perintah IRET adalah perintah yang menandakan bahwa interrupt-handler kita selesai, dan prosesor boleh melanjutkan pekerjaan yang tadi tertunda. CALL : call procedure. Memanggil sebuah prosedur. sintaks: CALL {label nama prosedur} RET : return. Tanda selesai prosedur. Setiap prosedur harus memiliki RET di ujungnya. sintaks: RET ;(saja) HLT : halt. Membuat prosesor menjadi tidak aktif. Prosesor harus mendapat interupsi dari luar atau di-reset supaya aktif kembali. Jadi, jangan gunakan perintah HLT untuk mengakhiri program!! Sintaks: HLT ;(saja). NOP : no operation. Perintah ini memakan 1 byte di memori tetapi tidak menyuruh prosesor melakukan apaapa 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