

**LAPORAN PRAKTIKUM
SISTEM OPERASI
MODUL 3
MENGENAL CARA 'DEBUGGING'
PROGRAM BOOTSTRAP-LOADER**



**Disusun Oleh :
YESY LELY YESTIANA
L200210227
Kelas E**

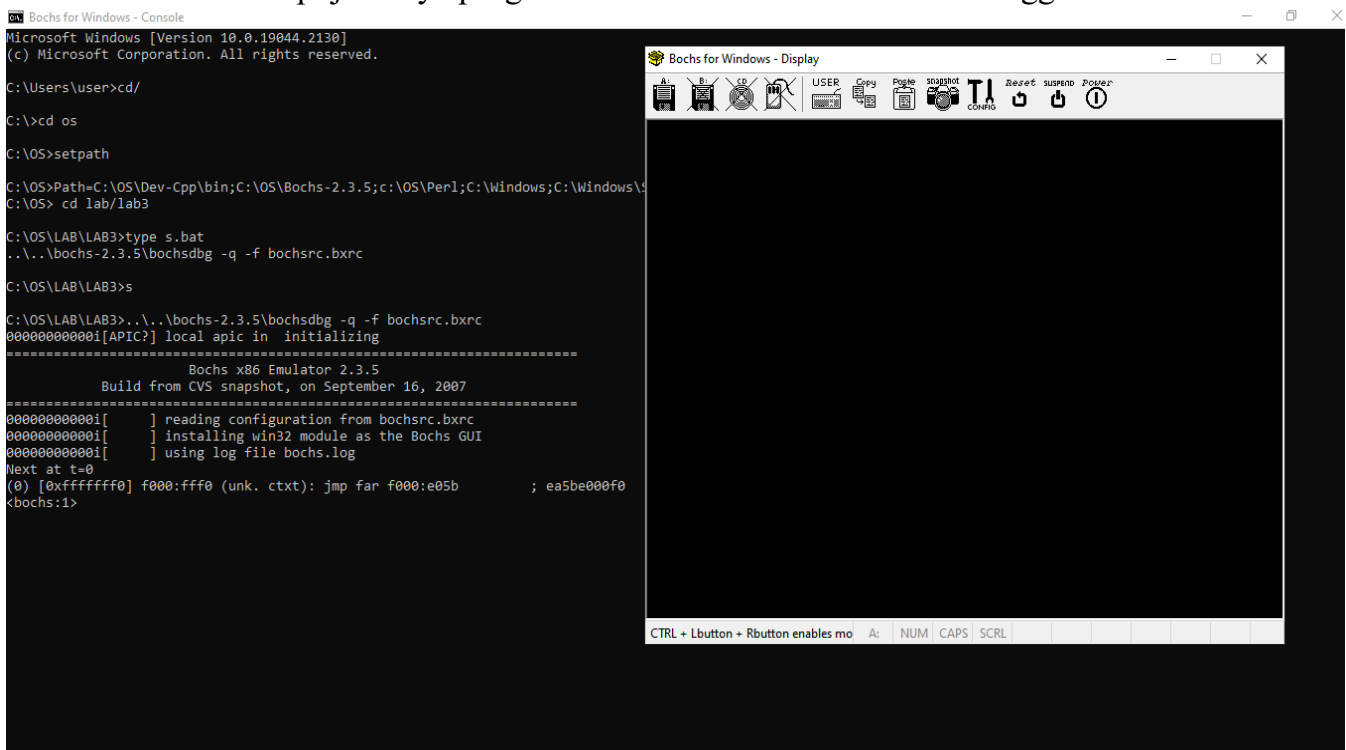
**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS KOMUNIKASI DAN INFORMATIKA
UNIVERSITAS MUHAMMADIYAH SURAKARTA
TAHUN 2022/2023**

Lembar Kerja Praktikum Modul 3

NIM	: L200210227	Nilai praktek	:
Nama	: Yesy Lely Yestiana		
Dosen Pengampu	: Heru Setiya N., ST, M.Kom	Tanda tangan	:
Nama Asisten	: -		
Tanggal Praktikum	: 27/09/2022		

Berikut ini merupakan *screen shoot* dari hasil pemrograman kegiatan modul 3

1. Buka 'cmd' lanjutkan dengan 'cd os', 'setpath', 'cd lab/lab3'.
2. File sudah siap, dilanjut dengan proses 'debugging'. Program diaktifkan yaitu versi 'bochsdbg' dengan menjalankan perintah 'type s.bat'.
3. Masukkan perintah 's'. layar PC-Simulator terlihat gelap tidak ada aktifitas maka tidak ada kesalahan tetapi jalannya program dihentikan oleh 'Bochs' menunggu masukan user.



4. Ketik r lalu enter maka akan muncul 8 byte.

```
Bochs for Windows - Console
C:\>cd os
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/lab3
C:\OS\LAB\LAB3>type s.bat
..\..\bochs-2.3.5\bochsdbg -q -f bochsrc.bxrc
C:\OS\LAB\LAB3>s
C:\OS\LAB\LAB3>..\..\bochs-2.3.5\bochsdbg -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 bochs.log
Next at t=0
(0) [0xffffffff] f000:fff0 (unk. ctxt): jmp far f000:e05b      ; ea5be00f0
<bochs:1> r
rax: 0x00000000:00000000 rcx: 0x00000000:00000000
rdx: 0x00000000:0000f20 rbx: 0x00000000:00000000
rsp: 0x00000000:00000000 rbp: 0x00000000:00000000
rsi: 0x00000000:00000000 rdi: 0x00000000:00000000
r8 : 0x00000000:00000000 r9 : 0x00000000:00000000
r10: 0x00000000:00000000 r11: 0x00000000:00000000
r12: 0x00000000:00000000 r13: 0x00000000:00000000
r14: 0x00000000:00000000 r15: 0x00000000:00000000
rip: 0x00000000:0000fff0
eflags 0x00000002
IOPL=0 id vip vif ac vm rf nt of df if tf sf zf af pf cf
<bochs:2>
```

5. Selanjutnya mengeksekusi perintah dengan ketikan ‘s’ enter kemudian lanjut dengan perintah ‘r’ enter.

```
Bochs for Windows - Console
C:\OS\LAB\LAB3>s
C:\OS\LAB\LAB3>..\..\bochs-2.3.5\bochsdbg -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 bochs.log
Next at t=0
(0) [0xffffffff] f000:fff0 (unk. ctxt): jmp far f000:e05b      ; ea5be00f0
<bochs:1> r
rax: 0x00000000:00000000 rcx: 0x00000000:00000000
rdx: 0x00000000:0000f20 rbx: 0x00000000:00000000
rsp: 0x00000000:00000000 rbp: 0x00000000:00000000
rsi: 0x00000000:00000000 rdi: 0x00000000:00000000
r8 : 0x00000000:00000000 r9 : 0x00000000:00000000
r10: 0x00000000:00000000 r11: 0x00000000:00000000
r12: 0x00000000:00000000 r13: 0x00000000:00000000
r14: 0x00000000:00000000 r15: 0x00000000:00000000
rip: 0x00000000:0000fff0
eflags 0x00000002
IOPL=0 id vip vif ac vm rf nt of df if tf sf zf af pf cf
<bochs:2> s
Next at t=1
(0) [0x000fe05b] f000:e05b (unk. ctxt): xor ax, ax          ; 31c0
<bochs:3> r
rax: 0x00000000:00000000 rcx: 0x00000000:00000000
rdx: 0x00000000:0000f20 rbx: 0x00000000:00000000
rsp: 0x00000000:00000000 rbp: 0x00000000:00000000
rsi: 0x00000000:00000000 rdi: 0x00000000:00000000
r8 : 0x00000000:00000000 r9 : 0x00000000:00000000
r10: 0x00000000:00000000 r11: 0x00000000:00000000
r12: 0x00000000:00000000 r13: 0x00000000:00000000
r14: 0x00000000:00000000 r15: 0x00000000:00000000
rip: 0x00000000:0000e05b
eflags 0x00000002
IOPL=0 id vip vif ac vm rf nt of df if tf sf zf af pf cf
<bochs:4>
```

6. Masukan perintah ‘vb 0:0x7c00’ enter. Perintah tersebut membuat titik pemberhentian pada titik vb 000:7c000 selanjutnya ketik perintah ‘c’ enter.

```
Bochs for Windows - Console
=====
0000000000i[      ] reading configuration from bochsrc.bxrc
0000000000i[      ] installing win32 module as the Bochs GUI
0000000000i[      ] using log file bochs.log
Next at t=0
(0) [0xfffffff0] f000:fff0 (unk. ctxt): jmp far f000:e05b      ; ea5be00f0
<bochs:1> g
rax: 0x00000000:00000000 rcx: 0x00000000:00000000
rdx: 0x00000000:00000f20 rbx: 0x00000000:00000000
rsp: 0x00000000:00000000 rbp: 0x00000000:00000000
rsi: 0x00000000:00000000 rdi: 0x00000000:00000000
r8 : 0x00000000:00000000 r9 : 0x00000000:00000000
r10: 0x00000000:00000000 r11: 0x00000000:00000000
r12: 0x00000000:00000000 r13: 0x00000000:00000000
r14: 0x00000000:00000000 r15: 0x00000000:00000000
rip: 0x00000000:0000fff0
eflags 0x00000002
IOPL=0 id vip vif ac vm rf nt of df if tf sf zf af pf cf
<bochs:2> s
Next at t=1
(0) [0x000fe05b] f000:e05b (unk. ctxt): xor ax, ax          ; 31c0
<bochs:3> r
rax: 0x00000000:00000000 rcx: 0x00000000:00000000
rdx: 0x00000000:00000f20 rbx: 0x00000000:00000000
rsp: 0x00000000:00000000 rbp: 0x00000000:00000000
rsi: 0x00000000:00000000 rdi: 0x00000000:00000000
r8 : 0x00000000:00000000 r9 : 0x00000000:00000000
r10: 0x00000000:00000000 r11: 0x00000000:00000000
r12: 0x00000000:00000000 r13: 0x00000000:00000000
r14: 0x00000000:00000000 r15: 0x00000000:00000000
rip: 0x00000000:0000e05b
eflags 0x00000002
IOPL=0 id vip vif ac vm rf nt of df if tf sf zf af pf cf
<bochs:4> vb 0:0x7C00
<bochs:5> s
Next at t=2
(0) [0x000fe05d] f000:e05d (unk. ctxt): out 0x0d, al      ; e60d
<bochs:6> g
```

7. Ketik perintah ‘s’ enter untuk membandingkan 7 instruksi yang akan dieksekusi oleh PC.

```
Bochs for Windows - Console
=====
0000000000i[      ] reading configuration from bochsrc.bxrc
0000000000i[      ] installing win32 module as the Bochs GUI
0000000000i[      ] using log file bochs.log
Next at t=0
(0) [0xfffffff0] f000:fff0 (unk. ctxt): jmp far f000:e05b      ; ea5be00f0
<bochs:1> g
rax: 0x00000000:00000000 rcx: 0x00000000:00000000
rdx: 0x00000000:00000f20 rbx: 0x00000000:00000000
rsp: 0x00000000:00000000 rbp: 0x00000000:00000000
rsi: 0x00000000:00000000 rdi: 0x00000000:00000000
r8 : 0x00000000:00000000 r9 : 0x00000000:00000000
r10: 0x00000000:00000000 r11: 0x00000000:00000000
r12: 0x00000000:00000000 r13: 0x00000000:00000000
r14: 0x00000000:00000000 r15: 0x00000000:00000000
rip: 0x00000000:0000fff0
eflags 0x00000002
IOPL=0 id vip vif ac vm rf nt of df if tf sf zf af pf cf
<bochs:2> s
Next at t=1
(0) [0x000fe05b] f000:e05b (unk. ctxt): xor ax, ax          ; 31c0
<bochs:3> r
rax: 0x00000000:00000000 rcx: 0x00000000:00000000
rdx: 0x00000000:00000f20 rbx: 0x00000000:00000000
rsp: 0x00000000:00000000 rbp: 0x00000000:00000000
rsi: 0x00000000:00000000 rdi: 0x00000000:00000000
r8 : 0x00000000:00000000 r9 : 0x00000000:00000000
r10: 0x00000000:00000000 r11: 0x00000000:00000000
r12: 0x00000000:00000000 r13: 0x00000000:00000000
r14: 0x00000000:00000000 r15: 0x00000000:00000000
rip: 0x00000000:0000e05b
eflags 0x00000002
IOPL=0 id vip vif ac vm rf nt of df if tf sf zf af pf cf
<bochs:4> vb 0:0x7C00
<bochs:5> s
Next at t=2
(0) [0x000fe05d] f000:e05d (unk. ctxt): out 0x0d, al      ; e60d
<bochs:6> s
Next at t=3
(0) [0x000fe05f] f000:e05f (unk. ctxt): out 0xda, al      ; e6da
<bochs:7> s
Next at t=4
(0) [0x000fe061] f000:e061 (unk. ctxt): mov al, 0xc0      ; b0c0
<bochs:8> g
```

8. Menambahkan 'break-point' yang lain (maksimal 7) dengan memulainya dari prosedur awal sesuai dengan cara sebelumnya sampai menambahkan perintah 'vb 0x0100:0x0000'

The screenshot shows two windows from the Bochs emulator. The 'Bochs for Windows - Console' window displays the command prompt where the user sets the path to the Bochs binaries and runs the debugger. The 'Bochs for Windows - Display' window shows the BIOS boot screen with a break-point set at 0x0100:0x0000.

```

C:\Users\User>cd\
C:\>cd os
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\lab3
C:\OS\LAB\LAB3>s
C:\OS\LAB\LAB3>..\..\bochs-2.3.5\bochsrc -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 bochs.log
text at t=0
(0) [0xfffffff0] f000:fff0 (unk. ctxt): jmp far f000:e05b ; ea5be00f0
<bochs:1> vb 0x0100:0x000
<bochs:2> c
(10264512) Breakpoint 10285608, in 0100:0000 (0x00001000)
text at t=2945013
(0) [0x00001000] 0100:0000 (unk. ctxt): mov ax, 0x0100 ; b80001
<bochs:3>
  
```

The display window shows the BIOS boot screen with the following text:

```

Plex86/Bochs UGABios 0.6a 19 Aug 2006
This UGA/VE Bios is released under the GNU LGPL

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

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...

Loading kernel ver 0.01
.....
..
  
```

9. Selanjutnya teruskan langkah PC-Simulator minimal 10x dengan menggunakan perintah 's' enter. Dan perhatikan perubahan disetiap tampilannya.

The screenshot shows the same two windows as before, but the console window now displays the execution of assembly instructions. The display window shows the BIOS boot screen with the same text as before.

```

00000000000i[ ] reading configuration from bochsrc.bxrc
00000000000i[ ] installing win32 module as the Bochs GUI
00000000000i[ ] using log file bochs.log
text at t=0
(0) [0xfffffff0] f000:fff0 (unk. ctxt): jmp far f000:e05b ; ea5be00f0
<bochs:1> vb 0x0100:0x000
<bochs:2> c
(10264512) Breakpoint 10285608, in 0100:0000 (0x00001000)
text at t=2945013
(0) [0x00001000] 0100:0000 (unk. ctxt): mov ax, 0x0100 ; b80001
<bochs:3> s
text at t=2945014
(0) [0x00001003] 0100:0003 (unk. ctxt): mov ds, ax ; 8ed8
<bochs:4> s
text at t=2945015
(0) [0x00001005] 0100:0005 (unk. ctxt): mov es, ax ; 8ec0
<bochs:5> s
text at t=2945016
(0) [0x00001007] 0100:0007 (unk. ctxt): cli ; fa
<bochs:6> s
text at t=2945017
(0) [0x00001008] 0100:0008 (unk. ctxt): mov ss, ax ; 8ed0
<bochs:7> s
text at t=2945018
(0) [0x0000100a] 0100:000a (unk. ctxt): mov sp, 0xffff ; bcffff
<bochs:8> s
text at t=2945019
(0) [0x0000100d] 0100:000d (unk. ctxt): sti ; fb
<bochs:9> s
text at t=2945020
(0) [0x0000100e] 0100:000e (unk. ctxt): push dx ; 52
<bochs:10> s
text at t=2945021
(0) [0x0000100f] 0100:000f (unk. ctxt): push es ; 06
<bochs:11> s
text at t=2945022
(0) [0x00001010] 0100:0010 (unk. ctxt): xor ax, ax ; 31c0
<bochs:12> s
text at t=2945023
(0) [0x00001012] 0100:0012 (unk. ctxt): mov es, ax ; 8ec0
<bochs:13>
  
```

The display window shows the BIOS boot screen with the following text:

```

Plex86/Bochs UGABios 0.6a 19 Aug 2006
This UGA/VE Bios is released under the GNU LGPL

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

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...

Loading kernel ver 0.01
.....
..
  
```

10. Bandingkan dengan source-code pada program 'kernel.asm'.

```
Bochs for Windows - Console
=====
000000000000[      ] reading configuration from bochsrc.bxrc
000000000000[      ] installing win32 module as the Bochs GUI
000000000000[      ] using log file bochs.log
Next at t=0
(0) [0xffffffff] f000:fff0 (unk. ctxt): jmp far f000:e05b      ; ea5be00f0
<bochs:1> vb 0x0100:0x000
<bochs:2> c
(10264512) Breakpoint 10285608, in 0100:0000 (0x00001000)
Next at t=2945013
(0) [0x00001000] 0100:0000 (unk. ctxt): mov ax, 0x0100      ; b80001
<bochs:3> s
Next at t=2945014
(0) [0x00001003] 0100:0003 (unk. ctxt): mov ds, ax          ; 8ed8
<bochs:4> s
Next at t=2945015
(0) [0x00001005] 0100:0005 (unk. ctxt): mov es, ax          ; 8ec0
<bochs:5> s
Next at t=2945016
(0) [0x00001007] 0100:0007 (unk. ctxt): cli                  ; fa
<bochs:6> s
Next at t=2945017
(0) [0x00001008] 0100:0008 (unk. ctxt): mov ss, ax          ; 8ed0
<bochs:7> s
Next at t=2945018
(0) [0x0000100a] 0100:000a (unk. ctxt): mov sp, 0xffff      ; bcbfff
<bochs:8> s
Next at t=2945019
(0) [0x0000100d] 0100:000d (unk. ctxt): sti                  ; fb
<bochs:9> s
Next at t=2945020
(0) [0x0000100e] 0100:000e (unk. ctxt): push dx             ; 52
<bochs:10> s
Next at t=2945021
(0) [0x0000100f] 0100:000f (unk. ctxt): push es             ; 06
<bochs:11> s
Next at t=2945022
(0) [0x00001010] 0100:0010 (unk. ctxt): xor ax, ax          ; 31c0
<bochs:12> s
Next at t=2945023
(0) [0x00001012] 0100:0012 (unk. ctxt): mov es, ax          ; 8ec0
<bochs:13>
```

```
kernelasm - Notepad
File Edit Format View Help
;
; =====
[org 0x000]
[bits 16]

[SEGMENT .text]

;START #####
mov ax, 0x0100      ;lokasi memori untuk menempatkan kernel
mov ds, ax
mov es, ax

cli                ;set interrupt OFF
mov ss, ax         ;atur stack segment
mov sp, 0xFFFF     ;atur stack pointer maksimum 64k
sti                ;set interrupt ON

push dx
push es
xor ax, ax
mov es, ax
cli
mov word [es:0x21*4], _int0x21 ; setup Interrupt service
mov [es:0x21*4+2], cs         ; untuk menampilkan karakter di layar
sti
pop es
pop dx

mov si, strWelcomeMsg ; Tampilkan informasi proses
mov al, 0x01          ; request service 0x01
int 0x21              ; int 0x21

call _shell           ; call the shell

int 0x19              ; reboot
;END #####
```

TUGAS!!

1. Buatlah table pemetaan memori pada PC selengkap mungkin.

Blok Memori	Alokasi Pemakaian
F 0 0 0 0	ROM BIOS, Diagnostic, BASIC
E 0 0 0 0	ROM program
D 0 0 0 0	ROM program
C 0 0 0 0	Perluasan BIOS untuk hardisk XT
B 0 0 0 0	Monokrom Monitor
A 0 0 0 0	Monitor EGA, VGS, dll
9 0 0 0 0	Daerah kerjapemakai s/d 640 KB
8 0 0 0 0	Daerah kerjapemakai s/d 576 KB
7 0 0 0 0	Daerah kerjapemakai s/d 512 KB
6 0 0 0 0	Daerah kerjapemakai s/d 448 KB
5 0 0 0 0	Daerah kerjapemakai s/d 384 KB
4 0 0 0 0	Daerah kerjapemakai s/d 320 KB
3 0 0 0 0	Daerah kerjapemakai s/d 256 KB
2 0 0 0 0	Daerah kerjapemakai s/d 192 KB
1 0 0 0 0	Daerah kerjapemakai s/d 128 KB
0 0 0 0 0	Daerah kerjapemakai s/d 64 KB

2. Baca buku referensi, jelaskan perbedaan antara mode kerja 'Real-mode' dan mode kerja 'Protect-Mode' pada PC IBM Compatible.

✓ Real-Mode

- Didasarkan pada prosesor 8086 & 8088.
- PC IBM asli menyertakan prosesor 8088 yang dapat menjalankan instruksi 16 bit menggunakan register internal 16 bit dan dapat menangani 1 Mb menggunakan 20 baris alamat.
- Mode instruksi 16 bit 8088 disebut mode real.
- Semua perangkat lunak yang bekerja dalam arsitektur memori 20 bit yang didukungnya.
- Tidak ada multi tasking
- Tidak ada proteksi untuk mencegah 1 program menimpa program lain.

- Semua prosesor memiliki realmode dan sebenarnya computer biasanya menyala dalam realmode.
- Real Mode digunakan oleh apk DOS dan DOS Standar.

✓ Protect-Mode

- Dimulai dengan chip 80286 di IBM AT, protect mode baru diperkenalkan. Ini adalah mode operasi yang jauh lebih kuat dari pada real, mode dan digunakan di semua system operasi multi tasking modern.
- Akses penuh ke semua memori system.
- Kemampuan untuk melakukan banyak tugas, artinya membuat system operasi mengelola eksekusi beberapa program secara bersamaan.
- Dukungan untuk memori virtual, yang memungkinkan system menggunakan hard disk cepat (32-bit) ke memori dan driver 32-bit yang lebih cepat untuk melakukan transfer I/O.
- Setiap program yang sedang berjalan memiliki lokasi memori yang ditetapkan, yang dikindungi dari konflik dengan program lain.
- Jika program mencoba menggunakan alamat memori yang tidak diizinkan, “kesalahan perlindungan” dihasilkan.
- Semua system operasi utama saat ini menggunakan protect mode termasuk Windows 3.x, Windows 9.x, Windows NT, OS / 2 & Linux.
- Semua prosesor dari 286 on dapat menggunakan protect mode