

PRAKTIKUM
SISTEM OPERASI
MODUL 3



Disusun Oleh :

MUHAMMAD

MIFTAHUL HUDA

L200210230

D

TEKNIK INFORMATIKA

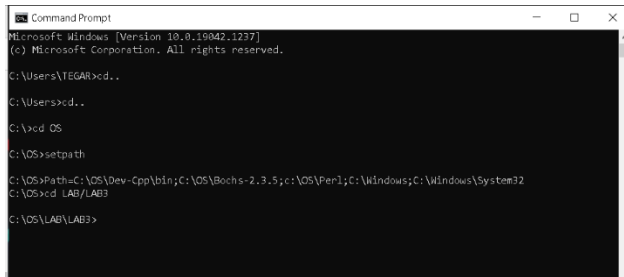
FAKULTAS KOMUNIKASI DAN INFORMATIKA

UNIVERSITAS MUHAMMADIYAH SURAKARTA

2022/2023

Praktik cara Debugging Porgram Bootstrap Loader dan kernel dengan menggunakan program PC-Simulator Bochs :

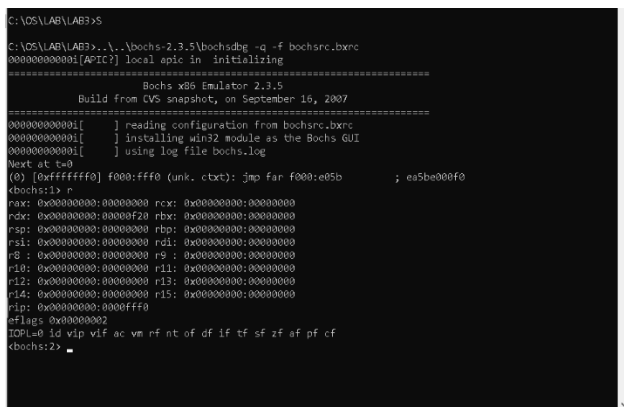
1. Membuka CMD dan masuk ke direktori “CD OS”, “setpath”, dan “cd LAB/LAB3”.



```
Microsoft Windows [Version 10.0.19042.1237]
(c) Microsoft Corporation. All rights reserved.

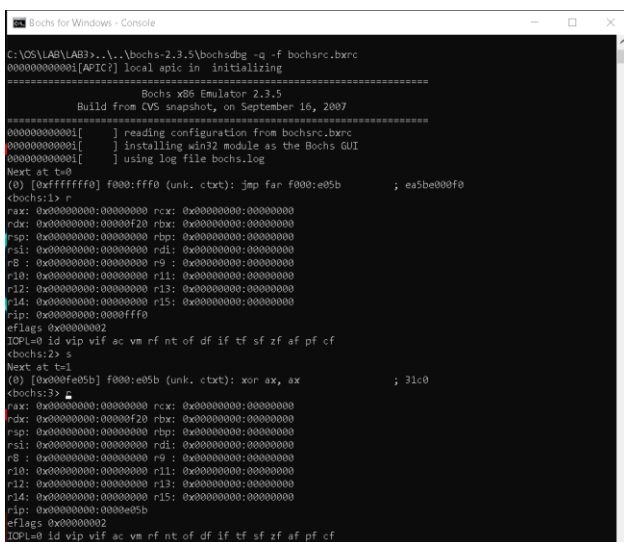
C:\Users\TEGAR>cd..
C:\Users>cd..
C:\>cd OS
C:\OS>setpath
C:\OS>Path=C:\OS\Dev-Cpp\bin;C:\OS\Bochs-2.3.5;c:\OS\Perl;c:\Windows;c:\Windows\System32
C:\OS>cd LAB/LAB3
C:\OS\LAB\LAB3>
```

2. Mulai melakukan debugging dengan memasukkan perintah “S” kemudian “r” di CMD.



```
C:\OS\LAB\LAB3>s
C:\OS\LAB\LAB3>..\..\bochs-2.3.5\bochsrc -q -f bochsrc.bxrc
0000000000000000[APIC?] local apic in initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
0000000000000000[ ] reading configuration from bochsrc.bxrc
0000000000000000[ ] installing win32 module as the Bochs GUI
0000000000000000[ ] 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: 0x00000000:0000f2b0 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
IDPL=0 ld vip vif ac vm nf nt of df if tf sf zf af pf cf
<bochs:2>
```

3. Mengeksekusi perintah tersebut dengan mengetikkan perintah “s” kemudian melanjutkan dengan perintah “r”.



```
Bochs for Windows - Console
C:\OS\LAB\LAB3>..\..\bochs-2.3.5\bochsrc -q -f bochsrc.bxrc
0000000000000000[APIC?] local apic in initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
0000000000000000[ ] reading configuration from bochsrc.bxrc
0000000000000000[ ] installing win32 module as the Bochs GUI
0000000000000000[ ] 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: 0x00000000:0000f2b0 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
IDPL=0 ld vip vif ac vm nf nt of df if tf sf zf af pf cf
<bochs:2> s
Next at t=1
(0) [0x0000e05b] f000:e05b (unk. ctxt): xor ax, ax ; 31c0
<bochs:3> r
rax: 0x00000000:00000000 rcx: 0x00000000:00000000
rdx: 0x00000000:0000f2b0 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
IDPL=0 ld vip vif ac vm nf nt of df if tf sf zf af pf cf
```

4. Melakukan “break point” pada tahapan proses BOOT, dengan memasukkan perintah “vb 0:0x7C00”, kemudian melanjutkan program yang terdapat pada BIOS untuk memeriksa RAM dan peralatan lainnya dengan memasukkan perintah “c”, dan menghentikan proses dengan perintah “q”.

```
kbochs:4> vb 0:0x7C00
kbochs:5> c
(10264512) Breakpoint 10285608, in 0000:7c00 (0x00007c00)
Next at t=2082128
(0) [0x00007c00] 0000:7c00 (unk. ctxt): jmp .+0x003b (0x00007c3e) ; e93b00
kbochs:6> s
Next at t=2082129
(0) [0x00007c3e] 0000:7c3e (unk. ctxt): cli ; fa
kbochs:7> s
Next at t=2082130
(0) [0x00007c3f] 0000:7c3f (unk. ctxt): mov ax, 0x07c0 ; b8:007
kbochs:8> s
Next at t=2082131
(0) [0x00007c42] 0000:7c42 (unk. ctxt): mov ds, ax ; BadB
kbochs:9> s
Next at t=2082132
(0) [0x00007c44] 0000:7c44 (unk. ctxt): mov es, ax ; Sec0
kbochs:10> s
Next at t=2082133
(0) [0x00007c46] 0000:7c46 (unk. ctxt): mov fs, ax ; Bee0
kbochs:11> s
Next at t=2082134
(0) [0x00007c48] 0000:7c48 (unk. ctxt): mov gs, ax ; BeeB
kbochs:12> s
Next at t=2082135
(0) [0x00007c4a] 0000:7c4a (unk. ctxt): mov ax, 0x0000 ; b00000
kbochs:13> q
# In bx_win32_gui_c::exit(void)!
Bochs is exiting. Press ENTER when you're ready to close this window.
```

5. Membuat break-point pada alamat 01.0000 dengan perintah “vb 0x0100:0x0000” dan melakukan secara berulang perintah debugging.

```
Bochs for Windows - Console
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
00000000000000000000[ ] reading configuration from bochssrc.bsrc
00000000000000000000[ ] installing win32 module as the Bochs GUI
00000000000000000000[ ] using log file bochs.log
Next at t=0
(0) [0xffffffff] f000:ffff (unk. ctxt): jmp far f000:e05b ; ea5be00f0
kbochs:1> vb 0x0100:0x0000
kbochs:2> c
(10264512) Breakpoint 10285608, in 0100:0000 (0x00001000)
Next at t=2945013
(0) [0x00001000] 0100:0000 (unk. ctxt): mov ax, 0x0100 ; b00001
kbochs:3> s
Next at t=2945014
(0) [0x00001003] 0100:0003 (unk. ctxt): mov ds, ax ; BadB
kbochs:4> s
Next at t=2945015
(0) [0x00001005] 0100:0005 (unk. ctxt): mov es, ax ; Sec0
kbochs:5> s
Next at t=2945016
(0) [0x00001007] 0100:0007 (unk. ctxt): cli ; fa
kbochs:6> s
Next at t=2945017
(0) [0x00001008] 0100:0008 (unk. ctxt): mov ss, ax ; Bad0
kbochs:7> s
Next at t=2945018
(0) [0x0000100a] 0100:000a (unk. ctxt): mov sp, 0xffff ; bcffff
kbochs:8> s
Next at t=2945019
(0) [0x0000100d] 0100:000d (unk. ctxt): sti ; fb
kbochs:9> s
Next at t=2945020
(0) [0x0000100e] 0100:000e (unk. ctxt): push dx ; 52
kbochs:10> s
Next at t=2945021
(0) [0x0000100f] 0100:000f (unk. ctxt): push es ; 06
kbochs:11> s
Next at t=2945022
(0) [0x00001010] 0100:0010 (unk. ctxt): xor ax, ax ; 31c0
kbochs:12> s
Next at t=2945023
(0) [0x00001012] 0100:0012 (unk. ctxt): mov es, ax ; Sec0
kbochs:13> q
# In bx_win32_gui_c::exit(void)!
Bochs is exiting. Press ENTER when you're ready to close this window.
```

Tugas

1. Tabel pemetaan memori pada PC.

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. Perbedaan mode kerja 'Real-Mode' dan mode kerja 'Protect-Mode'.

A. Real-Mode

Real-Mode adalah sebuah modus di mana prosesor Intel x86 berjalan seolah-olah dirinya adalah sebuah prosesor Intel 8085 atau Intel 8088, meski ia merupakan prosesor Intel 80286 atau lebih tinggi. Karenanya, modus ini juga disebut sebagai modus 8086 (8086 Mode). Dalam modus ini, prosesor hanya dapat mengeksekusi instruksi 16-bit saja dengan menggunakan register internal yang berukuran 16-bit, serta hanya dapat mengakses hanya 1024 KB dari memori karena hanya menggunakan 20-bit jalur bus alamat. Semua program DOS berjalan pada modus ini.

Prosesor yang dirilis setelah 8085, semacam Intel 80286 juga dapat menjalankan instruksi 16-bit, tapi jauh lebih cepat dibandingkan 8085. Dengan kata lain, Intel 80286 benar-benar kompatibel dengan prosesor Intel 8086 yang didesain sebelumnya. Sehingga prosesor Intel 80286 pun dapat menjalankan program-program 16-bit yang didesain untuk 8085 (IBM PC), dengan tentunya kecepatan yang jauh lebih tinggi. Dalam Real-mode, tidak ada proteksi ruang alamat memori, sehingga tidak dapat melakukan multi-tasking. Inilah sebabnya, mengapa program-program DOS bersifat single-tasking. Jika dalam modus real terdapat multi-tasking, maka kemungkinan besar antara dua program yang sedang berjalan, terjadi tabrakan (crash) antara satu dengan lainnya.

B . Protected Mode

Modus terproteksi (protected mode) adalah sebuah modus di mana terdapat proteksi ruang alamat memori yang ditawarkan oleh mikroprosesor untuk digunakan oleh sistem operasi. Modus ini datang dengan mikroprosesor Intel 80286 atau yang lebih tinggi. Karena memiliki proteksi ruang alamat memori, maka dalam modus ini sistem operasi dapat melakukan multitasking.

Prosesor Intel 80286 memang dilengkapi kemampuan masuk ke dalam modus terproteksi, tapi tidak dapat keluar dari modus tersebut tanpa harus mengalami reset (warm boot atau cold boot). Kesalahan ini telah diperbaiki oleh Intel dengan merilis prosesor Intel 80386 yang dapat masuk ke dalam modus terproteksi dan keluar darinya tanpa harus melakukan reset. Inilah sebabnya mengapa Windows 95/Windows 98 dilengkapi dengan modus Restart in MS-DOS Mode, meski sebenarnya sistem operasi tersebut merupakan sistem operasi yang berjalan dalam modus terproteksi.