Introducción a Bochs

Organización del Computador II

David Alejandro González Márquez

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

14.05.2009

¿Porque usar bochs?

El procesador posee intrucciones que no se pueden usar a nivel de usuario.
Por lo tanto, si queremos acceder a estos mecanismos debemos estar en lugar del sistema operativo.

- Utilizar instrucciones de nivel privilegiado
- Acceder a los mecanismos de manejo de memoria
- Cambiar modos del procesador
- Controlar interrupciones

Problemas del bochs

Bochs es un simulador de una computadora por esto nos permite correr instrucción por instrucción.
Pero su versión oficial no esta compilada con esta posibilidad.
Manos a la obra!

- bajar de:
  `http://sourceforge.net/projects/bochs/files/bochs/2.4/`
  el archivo `bochs-2.4.tar.gz`
- descomprimir: `tar -xvvzf bochs-2.4.tar.gz`
- en la carpeta descomprimida hacer:
  - `./configure --enable-debugger --enable-disasm --prefix=/home/<` *usuario* `>/bochs-2.4/`
  - `make`
  - `make install`

Para gastar menos los dedos

Para que podamos usar bochs desde cualquier path, debemos incluirlo en la lista de path del sistema.
Hacemos esto, agregando en el archivo .bashrc la linea que incluye tal path.

- Agregar en el archivo `/home/<` *usuario* `>/.bashrc`

  `export PATH+=":/home/<` *usuario* `>/bochs-2.4/bin/"`

Por último, para que carge los cambios en nuestra consola, corremos:

- `source .bashrc`

Bochs

```
------------------------------
Bochs Configuration: Main Menu
------------------------------
This is the Bochs Configuration Interface, where you can describe the
machine that you want to simulate.  Bochs has already searched for a
configuration file (typically called bochsrc.txt) and loaded it if it
could be found.  When you are satisfied with the configuration, go
ahead and start the simulation.

You can also start bochs with the -q option to skip these menus.

1. Restore factory default configuration
2. Read options from...
3. Edit options
4. Save options to...
5. Restore the Bochs state from...
6. Begin simulation
7. Quit now

Please choose one: [6] 3
```

Bochs: Edit options

```
------------------
Bochs Options Menu
------------------
0. Return to previous menu
1. Logfile options
2. Log options for all devices
3. Log options for individual devices
4. CPU options
5. Memory options
6. Clock \& CMOS options
7. PCI options
8. Bochs Display \& Interface options
9. Keyboard \& Mouse options
10. Disk options
11. Serial / Parallel / USB options
12. Network card options
13. Sound Blaster 16 options
14. Other options

Please choose one: [0] 10
```

Bochs: Edit options: Disk Options

```
------------------
Bochs Disk Options
------------------
0. Return to previous menu
1. First Floppy Drive
2. Second Floppy Drive
3. ATA channel 0
4. First HD/CD on channel 0
5. Second HD/CD on channel 0
6. ATA channel 1
7. First HD/CD on channel 1
8. Second HD/CD on channel 1
9. ATA channel 2
10. First HD/CD on channel 2 (disabled)
11. Second HD/CD on channel 2 (disabled)
12. ATA channel 3
13. First HD/CD on channel 3 (disabled)
14. Second HD/CD on channel 3 (disabled)
15. Boot Options

Please choose one: [0]
```

Bochs: Edit options: Disk Options: First HD/CD on channel 0

```
------------------------
First HD/CD on channel 0
------------------------

Device is enabled: [yes]
Enter type of ATA device, disk or cdrom: [disk]
Enter new filename: [minibootable.img]
Enter mode of ATA device, (flat, concat, etc.): [flat]
Enter number of cylinders: [900]
Enter number of heads: [15]
Enter number of sectors per track: [17]
Enter new model name: [Generic 1234]
Enter bios detection type: [auto]
Enter translation type: [auto]
```

Bochs: Edit options: Disk Options: First Floppy Drive

```
------------------
First Floppy Drive
------------------

What type of floppy drive? [3.5" 1.44M]

Enter new filename, or 'none' for no disk: [a.img]

What type of floppy media? (auto=detect) [1.44M]

Is media inserted in drive? [yes]
```

Bochs: Config file

bochsrc

```
megs: 32
romimage: file=$BXSHARE/BIOS-bochs-latest
vgaromimage: file=$BXSHARE/VGABIOS-lgpl-latest
vga: extension=vbe
floppya: 1_44=a.img, status=inserted
floppyb: 1_44=b.img, status=inserted
ata0-master: type=disk, path=boot.img, cylinders=900, heads=15, spt=17
boot: c
log: bochsout.txt
mouse: enabled=0
ips: 15000000
clock: sync=slowdown
vga_update_interval: 150000
```

A Bochiar!!!

Cuando inicializamos el bochs, lo primero que podemos ejecutar son las instrucciones dentro del BIOS

```
Please choose one: [6] 6
00000000000i[     ] installing x module as the Bochs GUI
00000000000i[     ] using log file bochsout.txt
Next at t=0
(0) [0xfffffff0] f000:fff0 (unk. ctxt): jmp far f000:e05b         ; ea5be000f0
<bochs:1>
```

pero antes veamos que funcionalidades nos da la consola de debugging

Next y Step

- s | step | stepi [count] - ejecuta [count] instrucciones
- n | next | p - ejecuta instrucciones sin entrar a las subrutinas
- c | cont | continue - continua la ejecución
- q | quit | exit - sale del debugger y del emulador
- Ctrl-C Detiene la ejecución y retorna al promt

Registros de uso general

- r | reg | regs | registers - Lista los registros del CPU y sus contenidos

```
<bochs:12> registers
eax: 0x00000000 0
ecx: 0x00000000 0
edx: 0x00000543 1347
ebx: 0x00000000 0
esp: 0x00000000 0
ebp: 0x00000000 0
esi: 0x00000000 0
edi: 0x00000000 0
eip: 0x0000e05d
eflags 0x00000046
id vip vif ac vm rf nt IOPL=0 of df if tf sf ZF af PF cf
```

FPU

- fp | fpu - Muestra el estado de la FPU

```
<bochs:13> fpu
status  word: 0x0000: b c3 TOS0 c2 c1 c0 es sf pe ue oe ze de ie
control word: 0x0040: inf RC\_NEAREST PC\_32 pm um om zm dm im
tag word:     0x5555
operand:      0x0000
fip:          0x00000000
fcs:          0x0000
fdp:          0x00000000
fds:          0x0000
=>FP0 ST0(0):         raw 0x0000:0000000000000000 (0.0000000000) (ZERO)
  FP1 ST1(0):         raw 0x0000:0000000000000000 (0.0000000000) (ZERO)
  FP2 ST2(0):         raw 0x0000:0000000000000000 (0.0000000000) (ZERO)
  FP3 ST3(0):         raw 0x0000:0000000000000000 (0.0000000000) (ZERO)
  FP4 ST4(0):         raw 0x0000:0000000000000000 (0.0000000000) (ZERO)
  FP5 ST5(0):         raw 0x0000:0000000000000000 (0.0000000000) (ZERO)
  FP6 ST6(0):         raw 0x0000:0000000000000000 (0.0000000000) (ZERO)
  FP7 ST7(0):         raw 0x0000:0000000000000000 (0.0000000000) (ZERO)
```

MMX

- mmx - Muesta el estado de los registros de MMX

```
<bochs:3> mmx
MM[0]: 00000000:00000000
MM[1]: 00000000:00000000
MM[2]: 00000000:00000000
MM[3]: 00000000:00000000
MM[4]: 00000000:00000000
MM[5]: 00000000:00000000
MM[6]: 00000000:00000000
MM[7]: 00000000:00000000
```

Memory Dump

- x /nuf [addr] - Muestra el contenido de la dirección [addr]
- xp /nuf [addr] - Muestra el contenido de la dirección fisica [addr] nuf es número que indica cuantos valores se mostrarán, seguido de uno o más de los indicadores de formato.
  - x : hex
  - d : decimal
  - u : sin signo
  - o : octal
  - t : binario
  - c : char
  - s : ascii
  - i : instrucción

  select the size:
  - b : byte
  - h : word = half-word
  - w : dobleword = word

Memory Disassemble

- u | disasm | disassemble [count] [start] [end] - desensambla intrucciones desde la dirección lineal [start] hasta [end] exclusive.
- u | disasm | disassemble switch-mode - Selecciona la sintaxis Intel o AT&T de asembler
- u | disasm | disassemble size = n - Setea el tamaño del segmento a desensamblar

Breakpoints

- p | pb | break | pbreak [addr] - Crea un breakpoint en la dirección física [addr]
- vb | vbreak [seg:offset] - Crea un breakpoint en la dirección virtual [addr]
- lb | lbreak [addr] - Crea un breakpoint en la dirección lineal [addr]
- d | del | delete [n] - Borra el breakpoint número [n]
- bpe [n] - Activa el breakpoint número [n]
- bpd [n] - Desactiva el breakpoint número [n]

Watchs

- watch - Muestra el estado actual de los watchs
- watch stop - Detiene la simulación cuando un watch es encontrado
- watch continue - No detiene la simulación si un wath es encontrado
- watch r | read [addr] - Agrega un watch de lectura en la dirección física [addr]
- watch w | write [addr] - Agrega un watch de escritura en la dirección física [addr]

Infos

- info break - Muestra los Breakpoint creados
- info eflags - Muestra el registro EEFLAGS
- info idt - Muestra el descriptor de interrupciones (idt)
- info ivt - Muestra la tabla de vectores de interrupción
- info gdt - Muestra la tabla global de descriptores (gdt)
- info tss - Muestra el segmento de estado de tarea actual (tss)
- info tab - Muestra la tabla de paginas

Registros de Segmento

- sreg - Muestra los registros de segmento

```
<bochs:5> sreg
cs:s=0xf000, dh=0xff0093ff, dl=0x0000ffff, valid=7
ds:s=0x0000, dh=0x00009300, dl=0x0000ffff, valid=7
ss:s=0x0000, dh=0x00009300, dl=0x0000ffff, valid=7
es:s=0x0000, dh=0x00009300, dl=0x0000ffff, valid=7
fs:s=0x0000, dh=0x00009300, dl=0x0000ffff, valid=7
gs:s=0x0000, dh=0x00009300, dl=0x0000ffff, valid=7
ldtr:s=0x0000, dh=0x00008200, dl=0x0000ffff, valid=1
tr:s=0x0000, dh=0x00008b00, dl=0x0000ffff, valid=1
gdtr:base=0x00000000, limit=0xffff
idtr:base=0x00000000, limit=0xffff
```

Registros de Control

- creg - Muestra los registros de control

```
<bochs:10> creg
CR0=0x60000010: pg CD NW ac wp ne ET ts em mp pe
CR2=page fault laddr=0x00000000
CR3=0x00000000
    PCD=page-level cache disable=0
    PWT=page-level writes transparent=0
CR4=0x00000000: osxsave smx vmx osxmmexcpt osfxsr pce pge mce pae pse de tsd pv
```

Utiles

- calc | ? [expr] - Calcula una expresión
  [expr] puede referenciar a cualquier registro de uso general o registro de segmento, además de poder usar operaciones aritmeticas y lógicas. Se pueden usar también el operador especial ':' para calcular una dirección lineal de memoria, usando la sintaxis segment:offset para modo real o selector:offset en modo protegido

¿Y el BIOS juega en primera?

- Cuando una computadora arranca solo existe el `BIOS` (Basic Input/Output system). Lamentablemente no podemos instalar un sistema operativo en el `BIOS` así que solo podemos usarla...
- El proceso de booteo comienza ejecutando el código del `BIOS`, ubicado en la posición `0xFFFF0`, en modo real. El `BIOS` tiene el código en ROM, que realiza la inicialización (por ejemplo, la placa de video) y una verificación de la máquina (POST). Luego busca algún dispositivo de booteo: Disco Rígido, Floppy, USB, etc...
- Una vez localizado el dispositivo de arranque, carga el primer sector de 512 bytes (excepto en el caso de CDROM, cuyo tamaño es 2048) en la posición de memoria `0x07C00` y salta a esa dirección.
- Ahora la imagen de arranque es la encargada de cargar el kernel y luego pasarle el control.
- Para que una imagen sea de arranque debe ocupar exactamente 512 bytes (excepto en el CDROM), y estar firmada en los últimos dos bytes con `0x55AA`.
- Una imagen de linux de ejemplo: `http://bochs.sourceforge.net/diskimages.html`

¿que podemos hacer?

- Creamos un breakpoint en la posición de memoria física donde comenzará a cargar el bootloader

```
<bochs:1> break 0x07C00
```

- Leemos la posición de memoria donde debería estar la firma del bootloader una vez que carga en memoria principal

```
<bochs:2> x/1x 0x07C00+510
[bochs]:
0x00007dfe <bogus+ 0>: 0x00000000
```

- Continuamos la ejecución, hasta que llegamos al breakpoint

```
<bochs:3> c
(0) Breakpoint 1, 0x00007c00 in ?? ()
Next at t=49462126
(0) [0x00007c00] 0000:7c00 (unk. ctxt): cli ; fa
```

- Nuevamente, podemos leer y notar que el BIOS cargo los 512bytes pertenecientes al bootloader, primer sector de la unidad

```
<bochs:4> x/1x 0x07C00+510
[bochs]:
0x00007dfe <bogus+ 0>: 0x0000aa55
```

Gracias!!!

¿Preguntas?