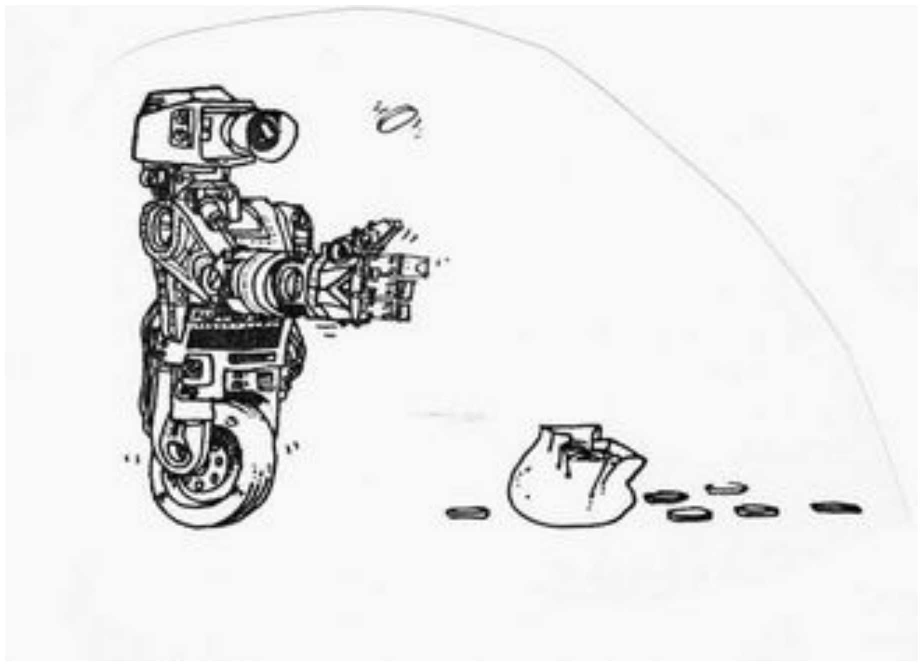


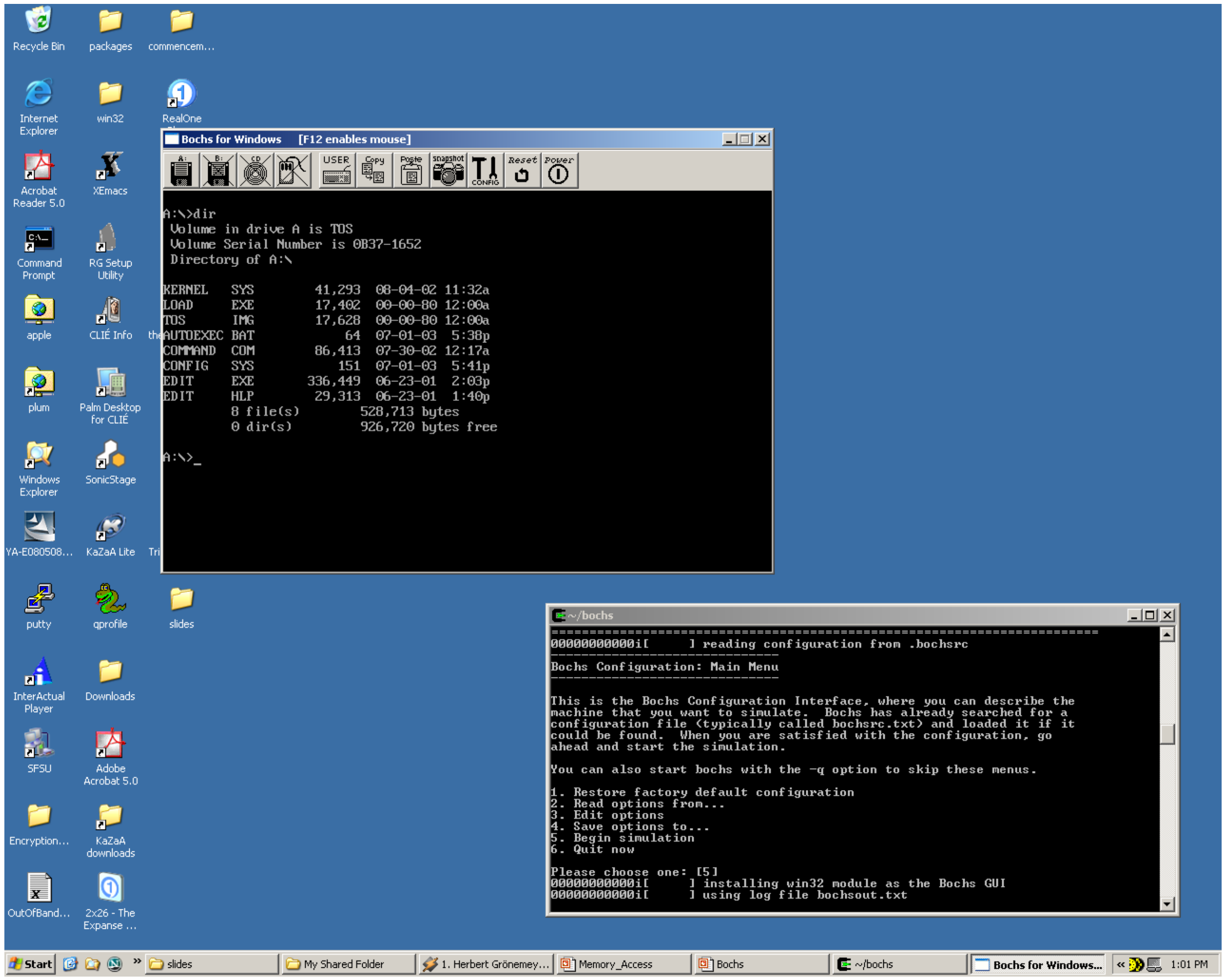
Bochs

Emulation

- TOS runs on regular PCs
- To try a new version of TOS:
  - Compile a new kernel
  - Write the kernel to a floppy
  - Reboot the PC
- A couple of problems:
  - Time consuming!
  - We don’t all have spare PCs (or floppy drives)
- The solution: use an *emulated* PC

Introducing Bochs

- Bochs is an open source PC-Emulator (bochs.sourceforge.net).
- A PC emulator emulates a complete PC on hardware level in software.
- I.e., a PC emulator is a piece of software; not hardware!
- The Bochs window looks just like a PC monitor (there is even a power button).

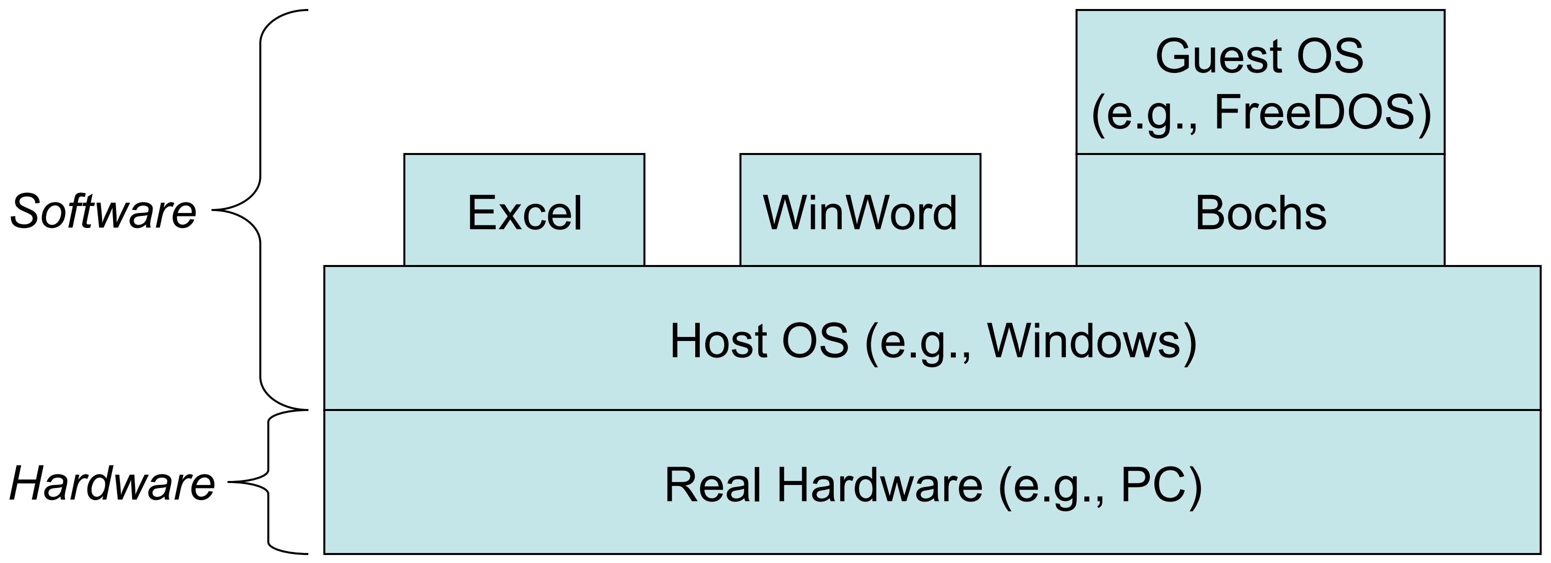


```
A:\>dir
 Volume in drive A is TOS
 Volume Serial Number is 0B37-1652
 Directory of A:\

KERNEL   SYS        41,293  08-04-02 11:32a
LOAD     EXE        17,402  00-00-80 12:00a
TOS      IMG        17,628  00-00-80 12:00a
AUTOEXEC BAT            64  07-01-03  5:38p
COMMAND  COM        86,413  07-30-02 12:17a
CONFIG   SYS           151  07-01-03  5:41p
EDIT     EXE       336,449  06-23-01  2:03p
EDIT     HLP        29,313  06-23-01  1:40p
         8 file(s)        528,713 bytes
         0 dir(s)         926,720 bytes free

A:\>_
```

```
========================================================================
00000000000i[     ] reading configuration from .bochsrc
------------------------------
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. Begin simulation
6. Quit now

Please choose one: [5]
00000000000i[     ] installing win32 module as the Bochs GUI
00000000000i[     ] using log file bochsout.txt
```

Bochs can be started by clicking on the Bochs shortcut and then hitting the <Enter> key in the first window that pops up

Host and Guest Operating System

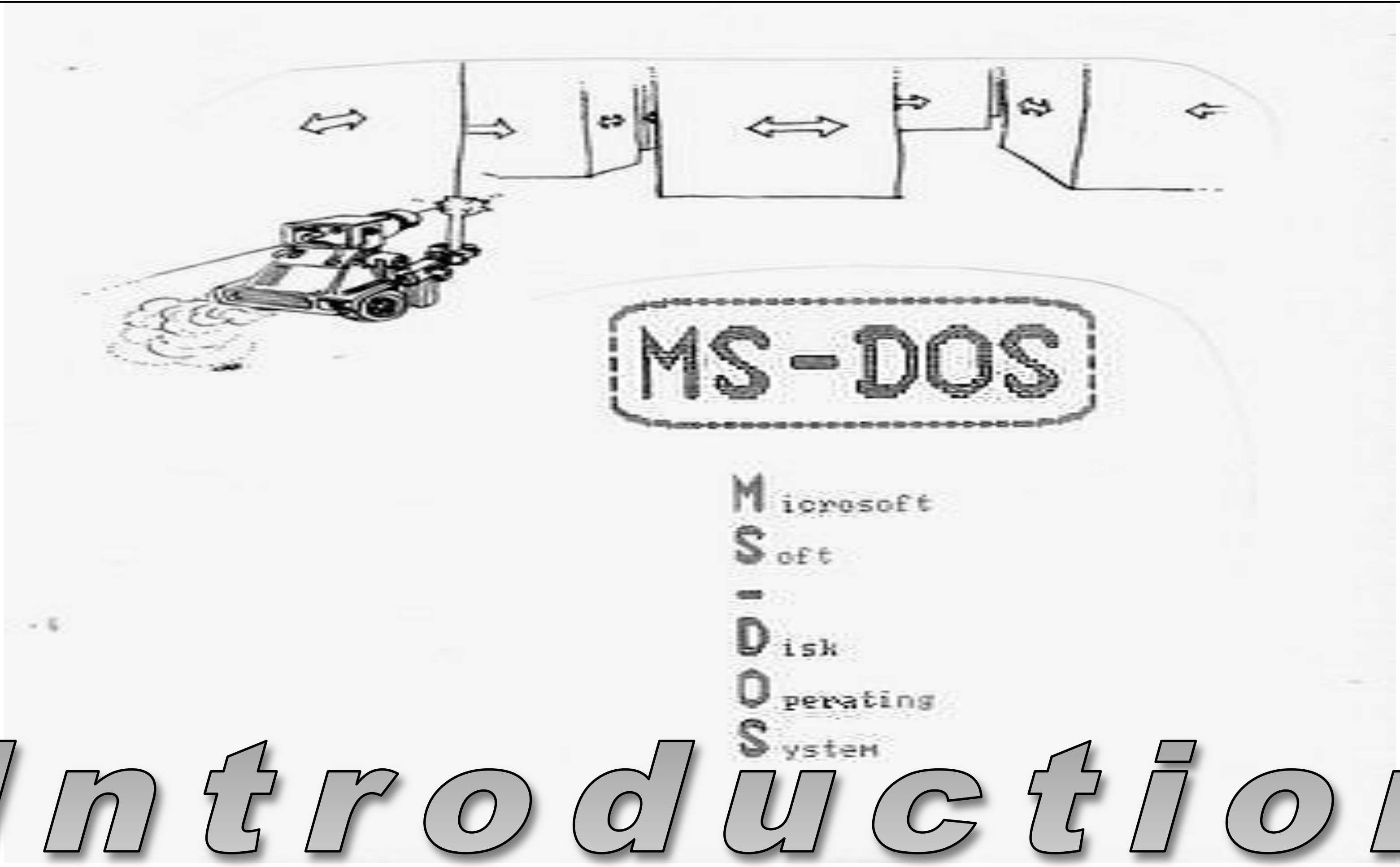


Virtual Hardware

- How does Bochs emulate hardware of the guest OS?
- The ‘virtual’ Hardware is mapped to resources on the Host OS.
- E.g. the floppy drive A: of the guest OS is mapped to a regular file located in the filesystem of the host OS.
- This mapping between virtual and real resources is done with the configuration file `~/.bochsrc` which contains the line:

```
floppya: 1_44 =image/disk_image
```

- This means that the drive A: of the guest OS is mapped to a 1.44 MB file located in `image/disk_image`
- Whenever the guest OS accesses A:, the operation is redirected by Bochs to this file.

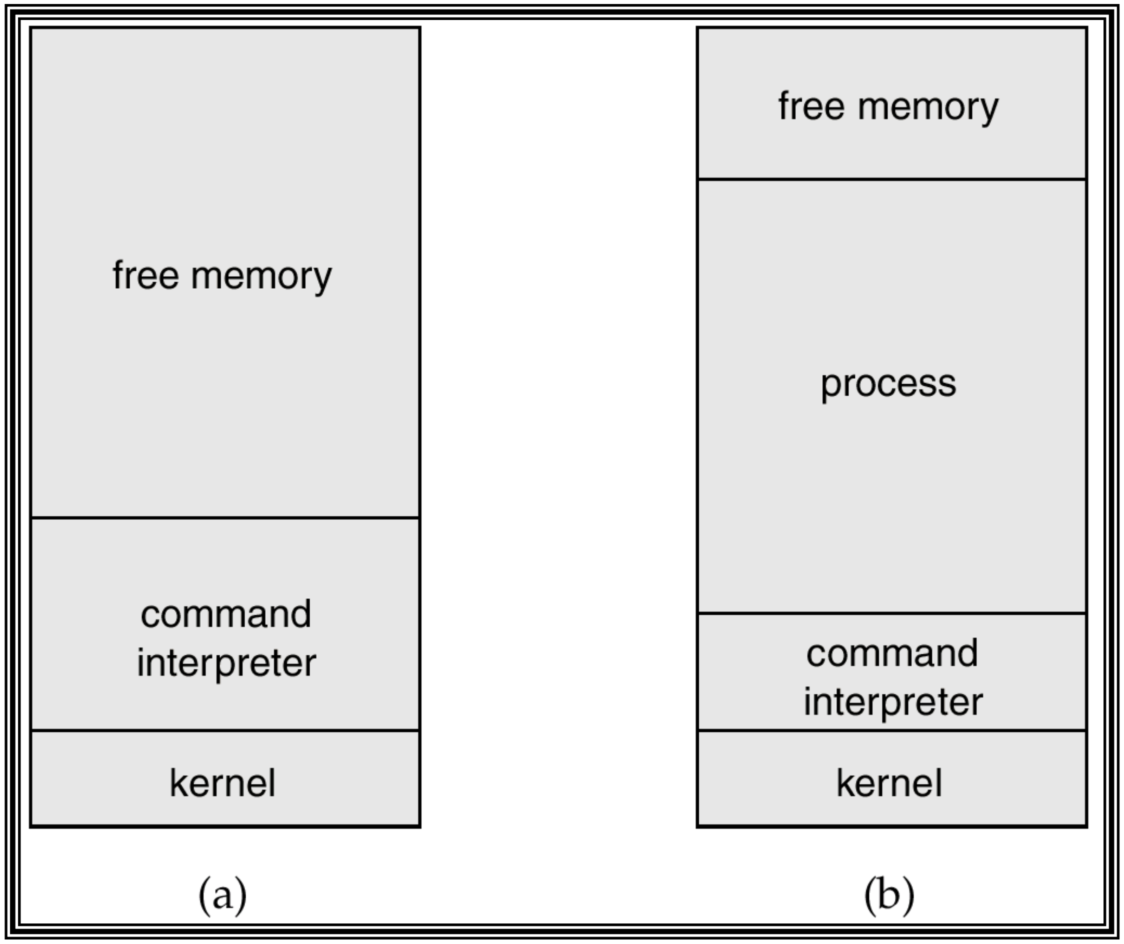


Introduction to MS-DOS

Overview of MS-DOS

- MS-DOS: Microsoft Disk Operating System
- Old operating system implemented by Microsoft for the PC
- Windows is the successor of DOS
- DOS is still "hidden" in windows through the command shell
- MS-DOS – written to provide the most functionality in the least space
  - not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated

MS-DOS Execution



At System Start-up　　Running a Program

DOS Filenames

- Filename have a *name* and an *extension*
- The name can be at most 8 characters long
- The extension can be at most 3 characters long
- Name and extension are separated by a dot, e.g., command.com, autoexec.bat
- The extension indicates the type of the file:
  - .com: command file
  - .exe: executable
  - .bat: batch file; contains a series of DOS commands

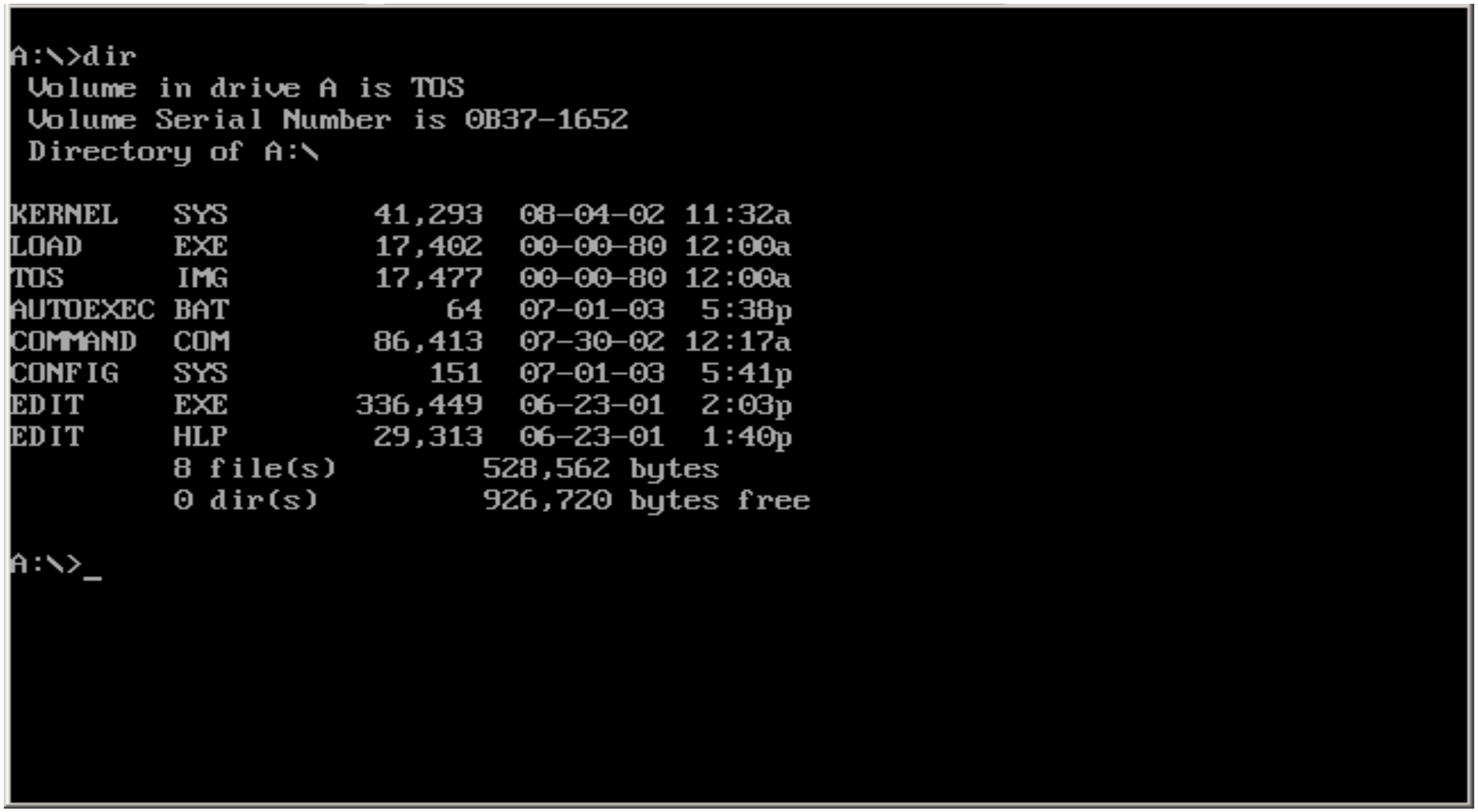
DOS Commands

Command	Description
`copy <from> <to>`	Copies file <from> to file <to>
`echo <message>`	Print <message> to the console
`type <file>`	Prints the contents of <file> to the console
`edit <file>`	Edits the content of <file>
`ren <old> <new>`	Renames <old> to <new>
`del <file>`	Deletes <file>
`md <dir>`	Makes a new directory called <dir>
`dir`	Show all the files contained in the current directory
`rmdir <dir>`	Removes the directory named <dir>
`cd <dir>`	Changes the current directory to <dir>

Examples

- `dir *.bat`
  Show all files of the current directory that end in `.bat`
- `copy autoexec.bat a.old`
  Copy the contents of `autoexec.bat` to `a.old`
- `type autoexec.bat`
  Display the contents of `autoexec.bat`
- `md test`
  Create a directory `test`

Screenshot of DOS

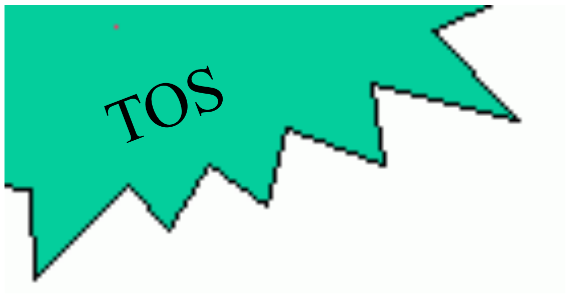


```
A:\>dir
 Volume in drive A is TOS
 Volume Serial Number is 0B37-1652
 Directory of A:\

KERNEL   SYS        41,293  08-04-02 11:32a
LOAD     EXE        17,402  00-00-80 12:00a
TOS      IMG        17,477  00-00-80 12:00a
AUTOEXEC BAT            64  07-01-03  5:38p
COMMAND  COM        86,413  07-30-02 12:17a
CONFIG   SYS           151  07-01-03  5:41p
EDIT     EXE       336,449  06-23-01  2:03p
EDIT     HLP        29,313  06-23-01  1:40p
         8 file(s)        528,562 bytes
         0 dir(s)         926,720 bytes free

A:\>_
```

FreeDOS

- FreeDOS is an Open Source implementation of MS-DOS
- It contains a complete MS-DOS environment
- Available at http://www.freedos.org
- We will use FreeDOS to understand the functionality of a PC Emulator

Conventions



Explains the TOS API.

Assignments. For each assignment you will have to submit a project journal entry.

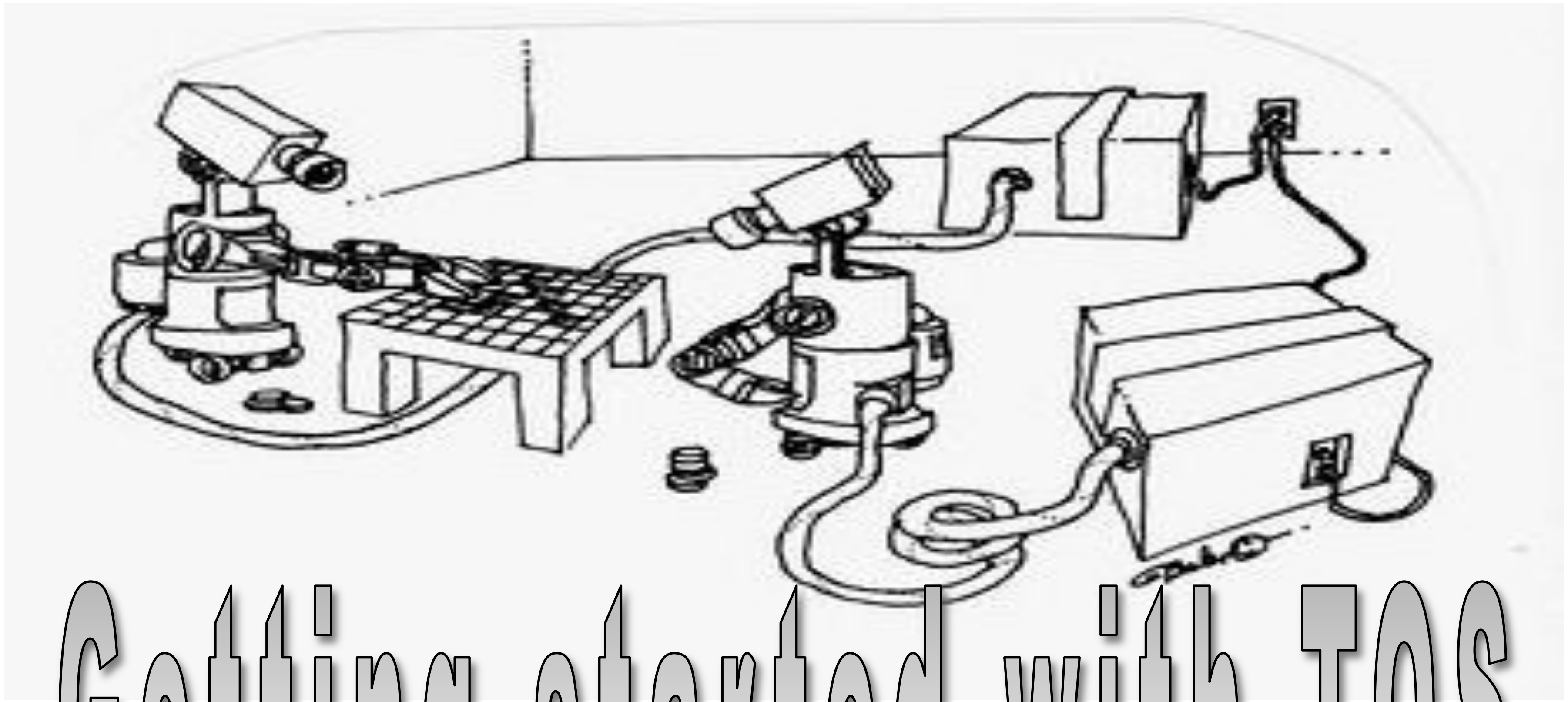
PacMan. A (hopefully) fun project that will be enhanced step-by-step throughout the semester where you will be using your own TOS API.

Assignment 0

- Install Bochs (will be automatically installed as part of the TOS installation)
- Get the FreeDOS disk image from the course web page.
- Run Bochs.
- Run some DOS commands. For example:

```
type autoexec.bat
dir
```

- You will be using Bochs extensively -- make sure you are comfortable using it!

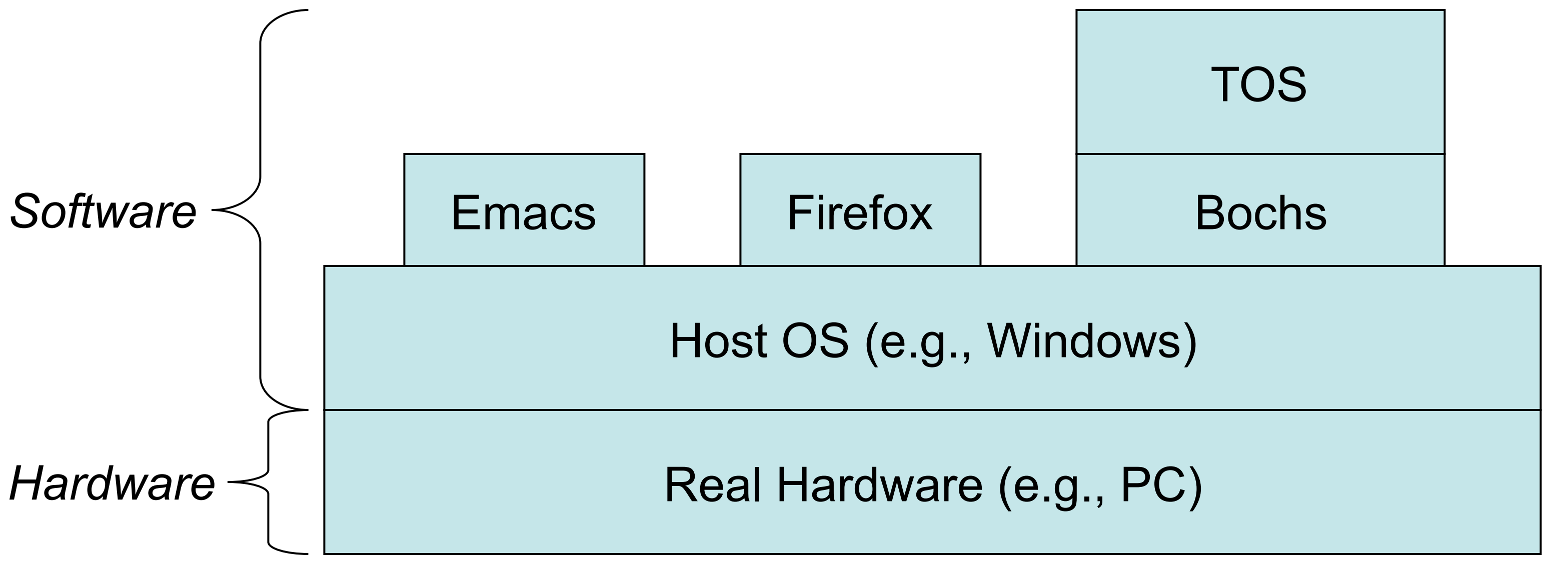


Getting started with TOS

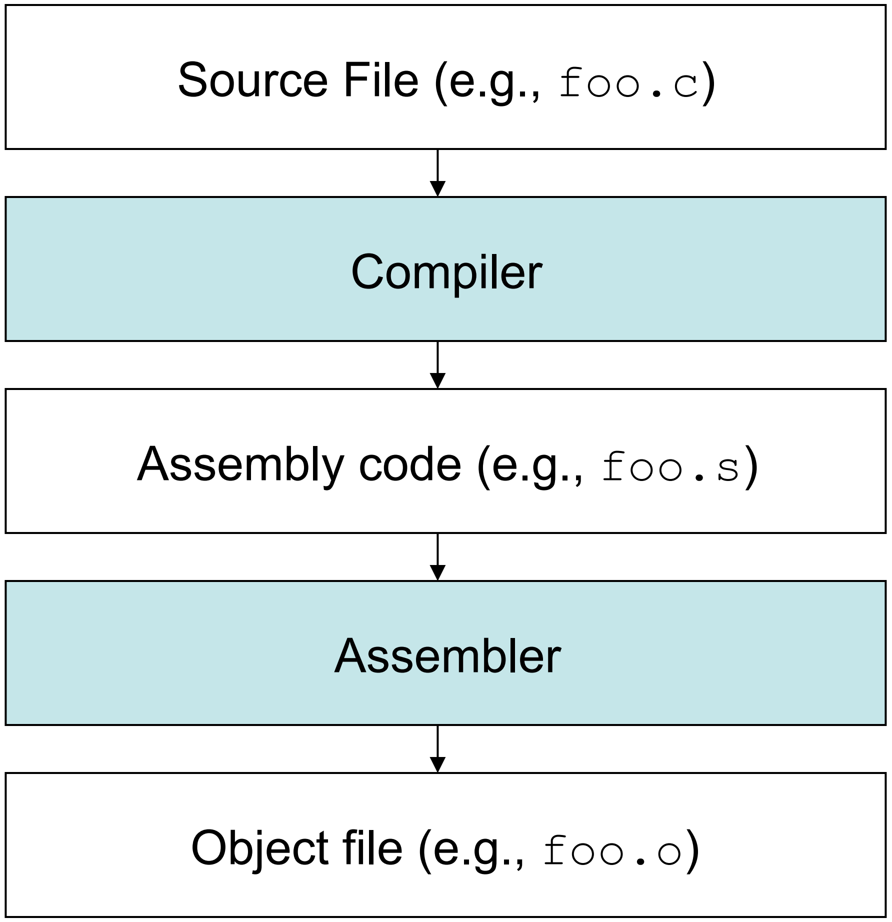
Overview of TOS

- TOS = Train Operating System
  (Train == Training || Model Train ☺)
- An educational operating system running on a PC
- Written in C (99%) and x86 assembly (1%)
- All the files and Makefiles are provided for you
- You just need to implement the core functions.

Running TOS in Bochs

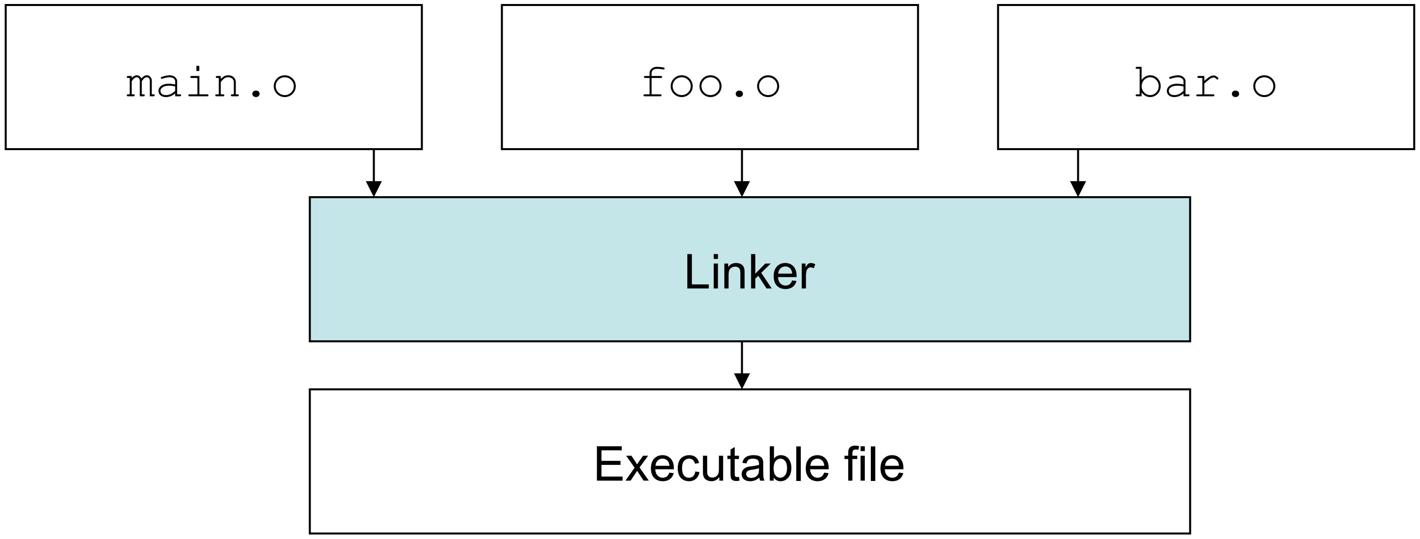


Compilation Process



Source File (e.g., foo.c)
Compiler
Assembly code (e.g., foo.s)
Assembler
Object file (e.g., foo.o)

Compilation Process



- Compiler/assembler/linker usually invoked automatically

  `gcc -v` … -- shows the actual commands

  `gcc -S foo.c` -- run the compiler but not the assembler

Directory structure of TOS

tos

kernel		The main sources of TOS
include		Header file for TOS
test		Test programs
image		Contains the floppy image from which to boot
tools		
	fat	FAT-tools (fatformat, fatdir, fatcopy, fatdel, rawwrite.exe)
	boot	TOS boot loader
	train	Train simulation

Files in ~/tos/kernel

Files	Contents
assert.c	Assert-function. Does not need to be edited.
com.c	COMs interface.
dispatch.c	Dispatcher and scheduler.
intr.c	Interrupt handling.
main.c	Contains main entry point kernel_main()
null.c	Null process.
train.c	Train application.
demo.c	Empty. Does not need to be edited.
inout.c	Low level input/output routines for COM1.
ipc.c	Inter-process communications.
mem.c	Memory access functions.
pacman.c	PacMan implementation.
process.c	Process management.
timer.c	Timer interrupt handling.
keyb.c	Keyboard interface. Does not need to be edited.
shell.c	Mini-shell for typing in commands. Can be extended for own commands.
window.c	Mini-windowing system for text-mode.

Recompiling TOS

- The only files you will be editing are `tos/kernel/*.c`
- Use your preferred editor to make the changes
- Two ways to compile TOS, both from the main tos directory:
  - `make tests` (build a testing kernel)
  - `make` (build a regular kernel)
- For now, always build a test kernel -- we'll build "regular" kernels later

Recompiling TOS

- No need to write or edit Makefiles
- If the build is successful, the new boot image will be located in `tos/image/disk_image`
- Other useful make targets:
  - `make clean` removes all object files and executables
  - `make clean-kernel` removes just kernel-specific object files

Writing a floppy

- The file `tos/image/disk_image` represents the complete 1.44 MB contents of a floppy.
- This file can be transferred to a (real) floppy disk
  - under Linux/MacOS:
    `dd if =tos/image/disk_image of=/dev/fd0`
  - under Windows: use the tool `tos/tools/fat/rawrite.exe` to copy the image. E.g. `rawrite.exe disk.img`
  - Note that rawrite.exe can only handle 8.3 style file names (e.g.: rawrite.exe disk_image will not work)
- You can boot from this floppy on a real PC.
- What you should see on the real PC is exactly the same thing you will see under Bochs.
- As you implement your own OS, it is a good idea to try it on a real PC using the technique explained on this slide.

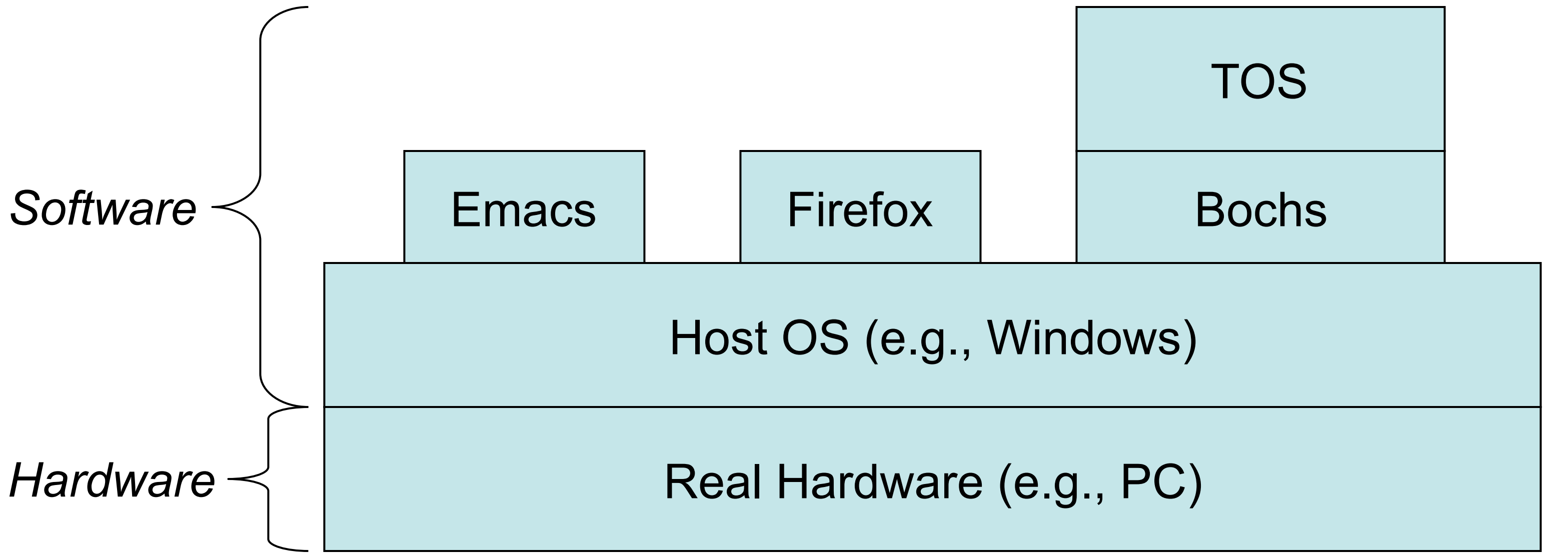
FAT-Tools

- TOS provides tools for manipulating disk images.
- They are called FAT tools because of the name of the DOS filesystem (File Allocation Table)
- Tools (in `tos/tools/fat`)
  - fatdir: displays the content of a directory
  - fatformat: formats the disk image
  - fatcopy: copies files to and from the disk image
  - fatdel: deletes a file on the disk image
- Example:
  - `tos/tools/fat/fatdir tos/image/disk_image /`
- You will not use FAT tools yourself. They are automatically invoked by the TOS Makefile

Some Guidelines

- Only modify C-files in `tos/kernel`
- No need to change Makefiles or C-header files.
- You can (and are encouraged to) look at and understand other files.
- You can <u>not</u> use any C-library functions: no `malloc()`, no `free()` !! (remember, we don’t have an OS yet)

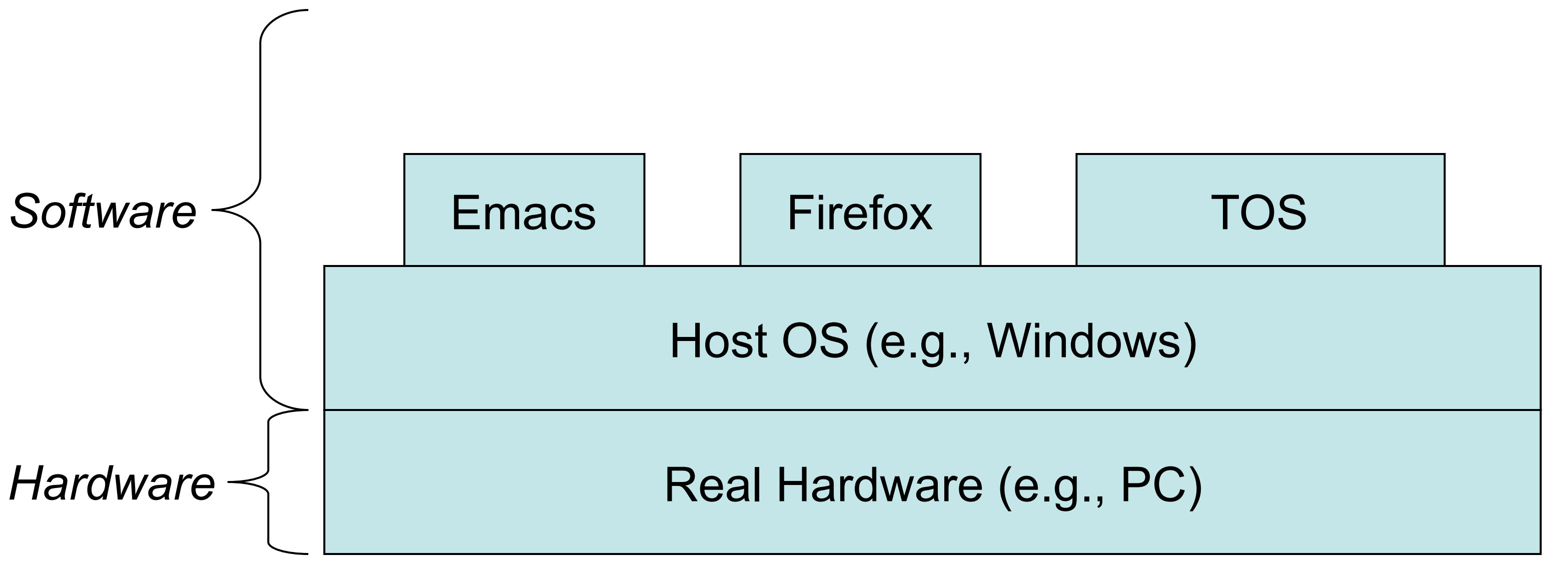
Running TOS (Assignment 2+)



Running TOS

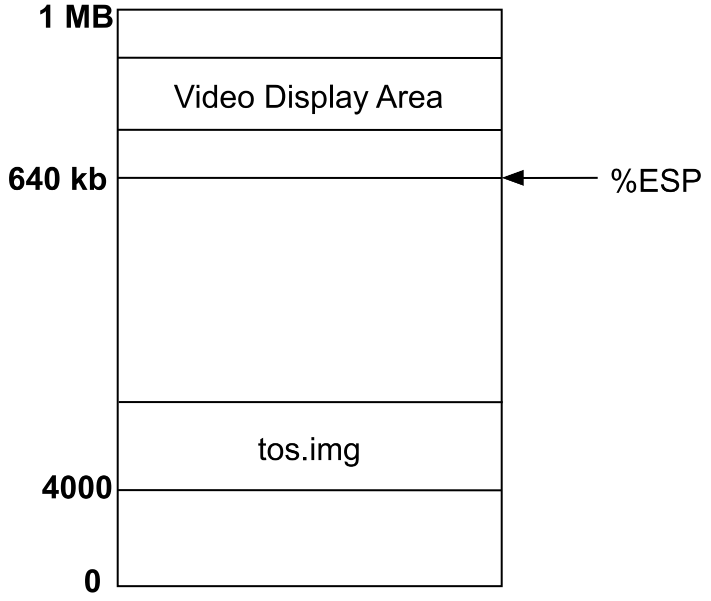
- Do the following to run TOS:
  - Start the Bochs emulator
  - Press <enter> after the menu appears
- The emulation will now start
- Click the Bochs “Power” button to exit
- Click the Bochs “Reset” button to restart

Running TOS (Assignment 1)



TOS Boot Sequence

- Sequence of events during boot:
  - PC is turned on (i.e. Bochs is executed)
  - PC loads the boot sector (the first sector of the floppy disk)
  - The boot-loader loads TOS at address 4000, initializes %ESP just below 640 kB and then jumps to `kernel_main()`
- The entry point of TOS is function `void kernel_main()` in file `tos/kernel/main.c` or `tos/test/run_tests.c`


1 MB
Video Display Area
640 kb
%ESP
tos.img
4000
0