Contents

[1. Von Neumann Interconnect (Architecture) 1](#_Toc468701692)

[2. The Von Neumann Machine 2](#_Toc468701693)

[Elements of Bus Design 5](#_Toc468701694)

[Cleaning Up after Decompression 7](#_Toc468701695)

[Uninstalling Software 9](#_Toc468701697)

[Reference: 10](#_Toc468701698)

# Von Neumann Interconnect (Architecture)

ENIAC The ENIAC (Electronic Numerical Integrator And Computer), designed and constructed at the University of Pennsylvania, was the world's first general purpose electronic digital computer. The project was a response to U.S. needs during World War II. The Army's Ballistics Research Laboratory (BRL), an agency responsible for developing range and trajectory tables for new weapons, was having difficulty supplying these tables accurately and within a reasonable time frame. Without these firing tables, the new weapons and artillery were useless to gunners. The BRL employed more than 200 people who, using desktop calculators, solved the necessary ballistics equations. Preparation of the tables for a single weapon would take one person many hours, even days.

John Mauchly, a professor of electrical engineering at the University of Pennsylvania, and John Eckert, one of his graduate students, proposed to build a general-purpose computer using vacuum tubes for the BRL's application. In 1943, the Army accepted this proposal, and work began on the ENIAC. The resulting machine was enormous, weighing 30 tons, occupying 1500 square feet of floor space, and containing more than 18,000 vacuum tubes. When operating, it consumed 140 kilowatts of power. It was also substantially faster than any electromechanical computer, capable of 5000 additions per second.

The ENIAC was a decimal rather than a binary machine. That is, numbers were represented in decimal form, and arithmetic was performed in the decimal system. Its memory consisted of 20 accumulators, each capable of holding a 10 –digit decimal number. A ring of 10 vacuum tubes represented each digit. At any time, only one vacuum tube was in the ON state, representing one of the 10 digits. The major drawback of the ENIAC was that it had to be programmed manually by setting switches and plugging and unplugging cables.

The ENIAC was completed in 1946, too late to be used in the war effort. Instead, its first task was to perform a series of complex calculations that were used to help determine the feasibility of the hydrogen bomb. The use of the ENIAC for a purpose other than that for which it was built demonstrated its general-purpose nature. The ENIAC continued to operate under BRL management until 1955, when it was disassembled.

## The Von Neumann Machine

The task of entering and altering programs for the ENIAC was extremely tedious. But suppose a program could be represented in a form suitable for storing in memory alongside the data. Then, a computer could get its instructions by reading them from memory, and a program could be set or altered by setting the values of a portion of memory.

This idea, known as the stored-program concept, is usually attributed to the ENIAC designers, most notably the mathematician John von Neumann, who was a consultant on the ENIAC project. Alan Turing developed the idea at about the same time. The first publication of the idea was in a 1945 proposal by von Neumann for a new computer, the EDV AC (Electronic Discrete Variable Computer)

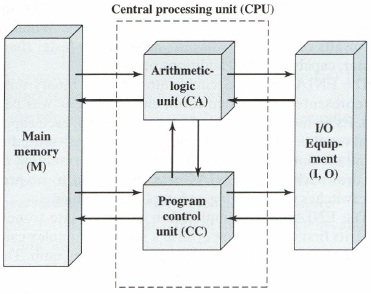
In 1946, von Neumann and his colleagues began the design of a new stored-program computer, referred to as the IAS computer, at the Princeton Institute for Advanced Studies. The IAS computer, although not completed until 1952, is the prototype of all subsequent general-purpose computers.'

• A main memory, which stores both data and instructions

• An arithmetic and logic unit (ALU) capable of operating on binary data

• A control unit, which interprets the instructions in memory and causes them to be executed

• Input/output (I/O) equipment operated by the control unit



This structure was outlined in von Neumann's earlier proposal, which is worth quoting in part at this point [VONN45]:

2.2 First: Since the device is primarily a computer, it will have to perform the elementary operations of arithmetic most frequently. These are addition, subtraction, multiplication, and division. It is therefore reasonable that it should contain specialized organs for just these operations.

It must be observed, however, that while this principle as such is probably sound, the specific way in which it is realized requires close scrutiny. At any rate a central arithmetical part of the device will probably have to exist, and this constitutes the first specific part: CA.

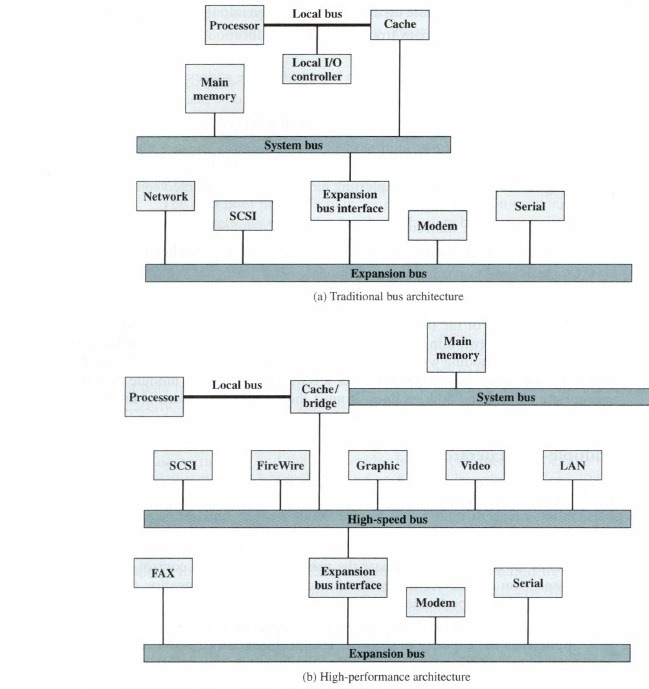
2.3 Second: The logical control of the device, that is, the proper sequencing of its operations, can be most efficiently carried out by a central control organ. If the device is to be elastic, that is, as nearly as possible all purpose, then a distinction must be made between the specific instructions given for and defining a particular problem, and the general control organs that see to it that these instructions-no matter what they are-are carried out.

The former must be stored in some way; the latter are represented by definite operating parts of the device. By the central control we mean this latter function only, and the organs that perform it form the second specific part: cc.

2.4 Third: Any device that is to carry out long and complicated sequences of operations (specifically of calculations) must have a considerable memory ...

The instructions which govern a complicated problem may constitute considerable material, particularly so, if the code is circumstantial (which it is in most arrangements). This material must be remembered.

At any rate, the total memory constitutes the third specific part of the device: M.



It is possible to connect I/O controllers directly onto the system bus. A more efficient solution is to make use of one or more expansion buses for this purpose. An expansion bus interface buffers data transfers between the system bus and the I/O controllers on the expansion bus. This arrangement allows the system to support a wide variety of I/O devices and at the same time insulate memory-to-processor traffic from I/O traffic.

Figure 2. a shows some typical examples of I/O devices that might be attached to the expansion bus. Network connections include local area networks (LANs) such as a 10-Mbps Ethernet and connections to wide area networks (WANs) such as a packet-switching network. SCSI (small computer system interface) is itself a type of bus used to support local disk drives and other peripherals. A serial port could be used to support a printer or scanner.

This traditional bus architecture is reasonably efficient but begins to break down as higher and higher performance is seen in the I/O devices. In response to these growing demands, a common approach taken by industry is to build a high speed bus that is closely integrated with the rest of the system, requiring only a bridge between the processor's bus and the high-speed bus. This arrangement is sometimes known as a mezzanine architecture.

Figure 2. b shows a typical realization of this approach. Again, there is a local bus that connects the processor to a cache controller, which is in turn connected to a system bus that supports main memory. The cache controller is integrated into a bridge, or buffering device, that connects to the high-speed bus. This bus supports connections to high-speed LANs, such as Fast Ethernet at 100 Mbps, video and graphics workstation controllers, as well as interface controllers to local peripheral buses, including SCSI and FireWire. The latter is a high-speed bus arrangement specifically designed to support high-capacity I/O devices. Lower-speed devices are still supported off an expansion bus, with an interface buffering traffic between the expansion bus and the high-speed bus.

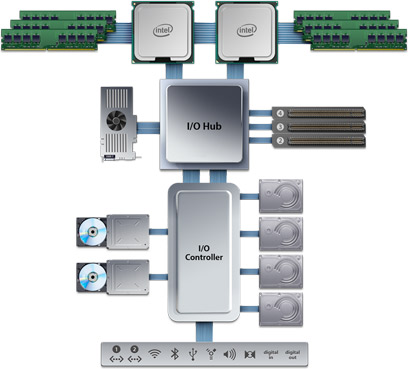
The advantage of this arrangement is that the high-speed bus brings high demand devices into closer integration with the processor and at the same time is independent of the processor. Thus, differences in processor and high-speed bus speeds and signal line definitions are tolerated. Changes in processor architecture do not affect the high-speed bus, and vice versa.

# Elements of Bus Design

Although a variety of different bus implementations exist, there are a few basic parameters or design elements that serve to classify and differentiate buses.

Bus lines can be separated into two generic types: dedicated and multiplexed. A dedicated bus line is permanently assigned either to one function or to a physical subset of computer components.

An example of functional dedication is the use of separate dedicated address and data lines, which is common on many buses. However, it is not essential. For example, address and data information may be transmitted over the same set of lines using an Address Valid control line. At the beginning of a data transfer, the address is placed on the bus and the Address Valid line is activated.



Installing Mac OS X

In this step-by-step exercise, you will install Mac OS X and create a configuration necessary for a typical user. If you have an Apple Mac that meets the hardware requirements but is using an older OS version, you can perform an upgrade during this exercise. If you have the current version of the Mac OS installed on a computer, you may want to download and install a hypervisor, such as the free Sun Virtual Box, create a virtual machine for Mac OS X, and use a Mac OS X DVD to install into the virtual machine.

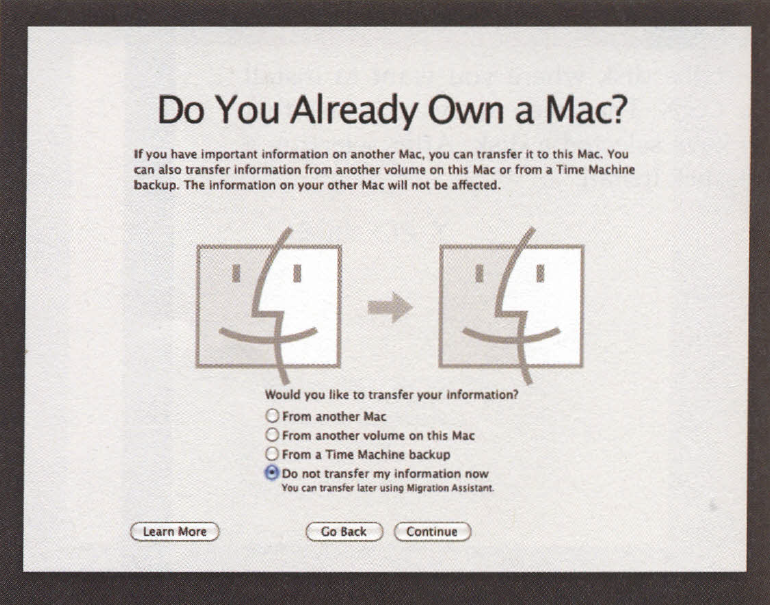
Start this exercise with either Step 1 or Step 2 depending on what you want to do. To complete this exercise, you will need the following:

• An Apple Mac that meets the hardware requirements.

• A preconfigured virtual machine if you want to try that option.

• A user name and password that will allow you to log on to your computer.

• The OS X installation DVD disc.



Installing Mac OS X Programs

In general, new programs arrive on your Mac via one of two avenues: on a CD or DVD, or via an Internet download. The CD method is slightly simpler; see "Performing the Installation" later in this section.

For help installing downloaded programs, on the other hand, read on.

.sit, .zip, .tar, .gz, and .dmg

Programs you download from the Internet generally arrive in a specially encoded, compressed form. (And unless you've changed the settings, they arrive in the Downloads folder on your Dock.)

The downloaded file's name usually has one of these file name extensions:

• .sit indicates a StuffIt file, the standard Macintosh file-compression format of years gone by.

• .zip is the standard Windows compression file format. And because Snow Leopard has a built-in Compress command right in the File menu (and doesn't come with Stuffit Expander), .zip is the new standard Macintosh compression format. It certainly makes life easier for people who have to exchange files with the Windows crowd.

• .tar is short for tape archive, an ancient Unix utility that combines (but doesn't compress) several files into a single icon, for simplicity in sending.

• .gz is short for gzip, a standard Unix compression format.

• .tar.gz or .tgz represents one compressed archive containing several files.

• .dmg is a disk image, described below.

Fortunately, if you use Safari as your Web browser, you don't have to worry about all this, because it automatically unzips and un stuffs them.

If you use some other browser, StuffIt Expander can turn all of them back into usable form when you download a file. (StuffIt Expander doesn't come with Mac OS X, but you can download it for free from, for example, this book's "Missing CD" page at [www.missingmanuals.com](http://www.missingmanuals.com/))

Disk Images (.dmg files)

Once you’ve unzipped a downloaded program, it often takes the form of a disk image file, whose name ends with the letters .dmg (second from top in Figure 5-20). Some files arrive as disk images straight from the Web, too, without having been compressed first.

Disk images are extremely common in Mac OS X. All you have to do is double-click the .dmg icon. After a moment, it magically turns into a disk icon on your desktop, which you can work with just as though it's a real disk.

For example:

• Double-click it to open it. The software you downloaded is inside.

• Remove it from your desktop by dragging it to the Trash (whose icon turns into a big silver (eject) key as you drag), highlighting it and pressing command key + E (the shortcut for Pile-e-Eject), clicking its (eject) button in the Sidebar, or Control-clicking (right clicking) it, and then choosing Eject from the shortcut menu. You’ve still got the original .dmg file you downloaded, so you're not really saying goodbye to the disk image forever.

# Cleaning Up after Decompression

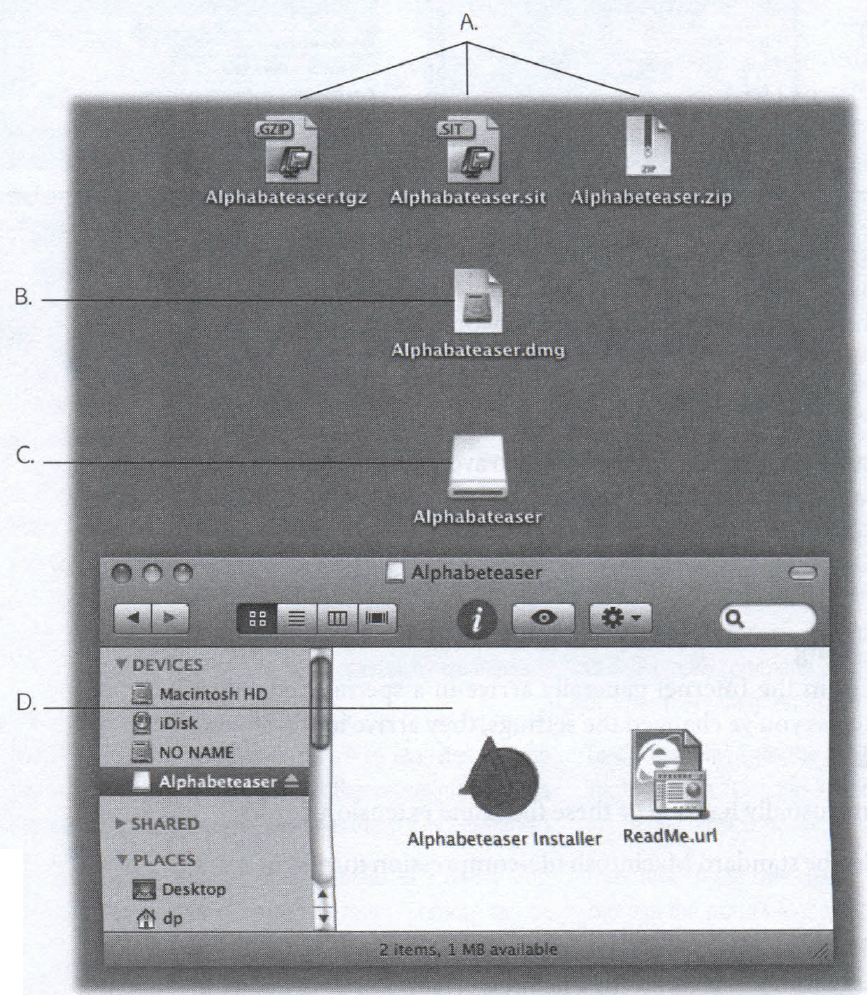
When you've finished unzipping or unstuffing a downloaded file, you may have several icons on your desktop or in the Downloads folder. Some are useful; some you're free to trash:

• The original compressed file. It's safe to throwaway the .sit, .tar, .gz, or .tgz file you originally downloaded (after it's decompressed, of course).

• The .dmg file. Once you've turned it into an actual disk-drive icon, installed the software from it, and "ejected" the disk-drive icon, you can delete the .dmg file.

Keep it only if you think you might need to reinstall the software someday.

• The disk image itself. This final icon, the one that contains the actual software or its installer (third from top in Figure 5-20), doesn't exist as a file on your hard drive. It's a phantom drive, held in memory, that will go away by itself when you log out. So after installing its software, feel free to drag it to the Trash (or highlight it and press SC-E to "eject" it).



Performing the Installation

Once you've got a disk icon on your desktop-either a pseudo-disk from a disk image or a CD or DVD you've inserted-you're ready to install the software. You can install many Mac OS X programs just by dragging their icons or folders to your hard drive (usually the Applications folder). Others offer a traditional installer program that requires you to double-click, read and accept a license agreement, and so on.

In both cases, where you decide to install the new program is suddenly a big issue.

Most of the time, this is where you'll want to install new programs. Putting them in the Applications folder makes them available to anyone who uses the Mac.

Your Home folder.

Suppose you share your Mac with other people, as described in Chapter 12. If that's your situation, you may occasionally want to install a program privately, reserving it for your own use only. In that case, just install or drag it into your Home folder, or a folder inside it. When other people log onto the machine, they won't even know you've installed that new program, since it doesn't show up in the Applications folder.

If you don't have an Administrator account, in fact, this is your only option for installing new programs.

# Uninstalling Software

In Mac OS X, there's generally no Uninstall program, and no Add/Remove Programs window. To uninstall a program, you just drag it (or its folder) from the Applications folder (or wherever it is) to the Trash.

Some programs leave harmless scraps of themselves behind; to check for them, look for preference files or folders bearing the dearly departed program's name in your…

Two Kinds of Programs: Cocoa and Carbon

Mac OS X was supposed to make life simpler. It was supposed to eliminate the confusion and complexity that the old Mac OS had accumulated over the years-and replace it with a smooth, simple, solid system.

Someday, that's exactly what Mac OS X will be. For the moment, however, you're stuck with running two different kinds of programs, each with different characteristics: Cocoa and Carbon.

The explanation involves a little bit of history and a little bit of logic. To take full advantage of Mac OS X's considerable technical benefits, software companies had to write new programs for it from scratch. So what should Apple have done-sent out an email to the authors of the 18,000 existing Mac programs, suggesting that they throw out their programs and rewrite them from the bottom up?

At most big software companies, that suggestion would wind up on the Joke of the Week bulletin board.

Instead, Apple gave software companies a break. It wrote Mac OS X to let programmers and software companies choose precisely how much work they wanted to put into compatibility with the new system. There are two levels:

Update the existing programs (Carbon). If programmers were willing to put some effort into getting with the Mac OS X program, they could simply adapt, or update, their existing software.

The resulting software looks and feels almost like a true Mac OS X program-you get the crash protection, the good looks, the cool-looking graphics, the Save sheets, and so on-but behind the scenes, the bulk of the computer programming is the same as it was in Mac OS 9. These are what Apple calls Carbonized programs, named for the technology (Carbon) that permits them to run on Mac OS X.

To this day, many famous Mac programs have simply been Carbonized: Apple-Works, Photoshop versions before CS3, FileMaker, Microsoft Office, and so on.

Most Carbonized programs don't offer all the features available to Mac OS X, however. In the following pages, you'll discover which Mac OS X goodies you sacrifice when using programs adapted this way.

On the other hand, such software offers a feature that software companies like a lot: Carbon program is a lot easier to write concurrently with a Windows version of the same software. A Cocoa program, by contrast, is almost certainly locked into Mac-only Land.

•Write new programs from scratch (Cocoa). As Mac OS X becomes more popular, more software companies create new programs exclusively for it. The geeks call such programs Cocoa applications. Although they look exactly like Carbonized programs, they feel a little bit more smooth and solid. And they offer a number of special features that Carbonized programs don't offer.

These days, almost all the programs that come on every Mac are true Cocoa applications, including iMovie, iPhoto, iDVD, Safari, iChat, TextEdit, Stickies, Mail, AddressBook, and so on.

# Reference:

William Stallings, 2013, Computer Organization and Architecture Ninth Edition, Chapters 2.1 and 3.4

<http://www.apple.com/macpro/features/processor.html> (last accessed 19.04.2013)

Jane Holcombe, Charles Holcombe, 2012, Survey of operating systems Third Edition, Chapter 9

[Von Neumann Interconnect (Architecture)1](#_Toc356396169)

[The Von Neumann Machine2](#_Toc356396170)

[Elements of Bus Design5](#_Toc356396171)

[Installing Mac OS X6](#_Toc356396172)

[Installing Mac OS X Programs6](#_Toc356396173)

[Disk Images (.dmg files)7](#_Toc356396174)

[Cleaning Up after Decompression7](#_Toc356396175)

[Performing the Installation8](#_Toc356396176)

[Your Home folder.9](#_Toc356396177)

[Uninstalling Software9](#_Toc356396178)

[Two Kinds of Programs: Cocoa and Carbon9](#_Toc356396179)

**No table of figures entries found.**

|  |  |  |
| --- | --- | --- |
| Day | 10.00 | 11.00 |
| Monday | - | Itp |
| Tuesday | Itp | - |

|  |  |  |
| --- | --- | --- |
| 1 |  |  |
| 2 |  | 45 |
| 3 |  | 44 |
| 4 |  | 67 |
| 5 |  | 24 |
|  |  | 225 |
|  | 45% |