Glossary

32-bit clean Said of an application (or other software) that is able to run in an environment where all 32 bits of a memory address are used for addressing.

680x0 See **680x0 microprocessor.**

680x0 application An application that contains code only for a 680x0 microprocessor. See also **fat application** and **PowerPC application.**

680x0-based Macintosh computer Any computer containing a 680x0 central processing unit that runs Macintosh system software. See also **PowerPC processor-based Macintosh computer.**

680x0 compiler Any compiler that produces code that can execute on a 680x0. See also **PowerPC compiler.**

680x0 context block A block of data used by the 68LC040 Emulator to maintain information across mode switches. The structure of this block of data is private.

680x0 microprocessor Any member of the Motorola 68000 family of microprocessors.

680x0 software Any software (that is, application, extension, driver, or other executable code) that consists of code only for a 680x0 microprocessor. See also **680x0 application.**

68LC040 Emulator The part of the system software that allows 680x0 applications and other 680x0 software to execute on PowerPC processor-based Macintosh computers. See also **Mixed Mode Manager.**

A5 world An area of memory in a 680x0 application's partition that contains the QuickDraw global variables, the application global variables, the application parameters, and the jump table—all of which are accessed through the A5 register. See also **mini-A5 world.**

accelerated resource An executable resource consisting of a routine descriptor and PowerPC code that specifically models the behavior of a 680x0 stand-alone code resource. Compare **private resource.**

accelerated system software routine Any Toolbox or Operating System routine that has been rewritten as PowerPC code.

A-line instruction An instruction that is not recognized by a 680x0 microprocessor and that the Trap Manager uses to execute Toolbox and Operating System routines. The first word of an A-line instruction is binary 1010 (hexadecimal A).

ANSI C language dialect The C programming language dialect that adheres to the language defined by the document *American National Standard for Information Systems—Programming Language—C,* ANSI X3.159-1989.

application A file of type `'APPL'` that can be launched by the Process Manager. See also **680x0 application** and **PowerPC application.**

application extension A fragment containing code and data (such as a data-conversion filter, tool, and so forth) that extends the capabilities of an application.

application global variables A set of variables stored in the application partition that are global to the application.

application heap An area of memory in the application heap zone in which memory is dynamically allocated and released on demand.

application parameters Thirty-two bytes of memory in the A5 world of a 680x0 application that are reserved for system use. The first long word is the address of the first QuickDraw global variable.

application partition A partition of memory reserved for use by an application. The application partition consists of free space, along with the application's heap and stack. The application partition for a 680x0 application also contains an A5 world.

A-trap See **A-line instruction.**

backing-store file The file in which the Virtual Memory Manager stores the contents of unneeded pages of memory. See also **file mapping** and **paging file.**

backing volume See **paging device.**

bind To find the referent of an import and place its address in a fragment's table of contents.

bus sizing See **dynamic bus sizing.**

byte smearing The ability of certain members of the 680x0 family of microprocessors to duplicate byte- and word-sized data across all 32 bits of the data bus.

cache See **data cache** or **instruction cache.**

callback routine A routine that is executed as part of the operation of some other routine.

callee A routine that is called by some routine.

caller A routine that calls some routine.

calling conventions A set of conventions that describe the manner in which a particular routine is executed. A routine's calling conventions specify where parameters and function results are passed. For a stack-based routine, the calling conventions determine the structure of the routine's stack frame.

code fragment See **fragment.**

code fragment information record A part of a code fragment resource that provides information about a specific code fragment. There can be more than one code fragment information record in a code fragment resource.

Code Fragment Loader The part of the Macintosh system software that reads containers and loads the fragments they contain into memory. Currently, the application programming interface to the Code Fragment Loader is private. See also **Code Fragment Manager.**

Code Fragment Manager The part of the Macintosh system software that loads fragments into memory and prepares them for execution. See also **Code Fragment Loader** and **fragment.**

code fragment resource A resource of type `'cfrg'` that identifies the instruction set architecture, location, size, and name of an application or import library, as well as version information for import libraries. See also **code fragment information record.**

code patch See **patch.**

code resource See **executable resource.**

code section A section of a fragment that contains executable code. See also **data section.**

code type See **instruction set architecture.**

compile-time library See **definition version.**

Condition Register (CR) A register in the PowerPC processor that holds the result of certain integer and floating-point operations.

connection A link between two fragments.

connection ID A reference number that uniquely identifies a connection. Defined by the `ConnectionID` data type.

container The storage for a fragment. A container is a contiguous chunk of storage that holds a fragment and information describing the location of the parts of the fragment and the format of the container.

context The block of static data (global variables, static variables, and function pointers) associated with one loading of an import library. Each application is loaded into its own context.

context block See **680x0 context block.**

CR See **Condition Register.**

cross-mode call A call to code that is in a different instruction set architecture from the caller's. See also **explicit cross-mode call** and **implicit cross-mode call.**

cross-TOC call A call to code that is in a different fragment from the caller's. A cross-TOC call requires that the Table of Contents Register be changed to the callee's TOC value.

dangling pointer A pointer that no longer points to the correct memory address.

data cache An area of memory internal to some microprocessors (for example, the MC68030 and MC68040 microprocessors) that holds recently accessed data. See also **instruction cache.**

data section A section of a fragment that contains its static data, including the fragment's table of contents. See also **code section.**

de facto C++ standard The current C++ language definition described in the working paper *American National Standard for Information Systems—Programming Language—C++,* ANSI X3J16.

definition function A function that defines the appearance and behavior of some user interface element (for example, a control, list, or window). See also **stub definition function.**

definition resource A resource that contains a definition function. See also **stub definition resource.**

definition version The version of an import library used by the linker to resolve imports in the application (or other fragment) being linked. The definition version defines the external programming interface and data format of the library. Compare **implementation version.**

disk location record A data structure that provides information about the location of a fragment in the data fork of a file on disk. Defined by the `DiskFragment` data type.

drop-in See **application extension.**

dynamically linked library See **import library.**

dynamic bus sizing The ability of certain members of the 680x0 family of microprocessors to allow I/O devices with 8-bit and 16-bit data paths to work with the processor's 32-bit data bus.

emulated application An application whose executable code is not in the instruction set architecture of the CPU. An emulated application relies on an emulator to translate its code into that instruction set. See also **680x0 application.**

emulation The process by which a microprocessor is able to execute code in an instruction set different from its native instruction set. See also **68LC040 Emulator.**

emulation environment The 680x0-compatible environment on PowerPC processor-based Macintosh computers provided by the 68LC040 Emulator and the Mixed Mode Manager.

emulator See **68LC040 Emulator.**

epilog A standard piece of code at the end of a routine that restores any nonvolatile registers saved by the routine's prolog, tears down the routine's stack frame, and returns to the caller. See also **prolog.**

exception An error or other special condition detected by the microprocessor in the course of program execution.

exception code A constant that indicates which kind of exception has occurred.

exception handler Any routine that handles exceptions.

exception information record A data structure that contains information about an exception, such as the exception kind, the machine state at the time of the exception, and so forth. Defined by the `ExceptionInformation` data type.

Exception Manager The part of the Macintosh system software that handles exceptions that occur during the execution of PowerPC applications or other software.

exception stack frame A block of data placed on the stack automatically by the processor when an exception occurs.

executable resource Any resource that contains executable code. See also **accelerated resource** and **private resource.**

explicit cross-mode call A call to code that is in a different instruction set architecture from the caller's, caused by the caller explicitly calling the `CallUniversalProc` function.

export To make a symbol externally visible. Also, a synonym for **exported symbol.**

exported symbol A symbol in a fragment that is visible to some other fragments. See also **import library** and **imported symbol.**

Extended Common Object File Format (XCOFF) A format of executable file generated by some PowerPC compilers. See also **Preferred Executable Format.**

extension See **application extension** and **system extension.**

external code Any block of executable code that is not directly contained in an application or other software.

fake definition resource See **stub definition resource.**

fake handle A handle that was not created by the Memory Manager but is passed to some Memory Manager routine.

fake pointer A pointer that was not created by the Memory Manager but is passed to some Memory Manager routine.

fat Containing or describing code of multiple instruction sets.

fat application An application that contains code of two or more instruction sets. See also **680x0 application** and **PowerPC application.**

fat binary Any piece of executable code (application, code resource, trap, or trap patch) that contains code of multiple instruction sets. See also **fat application, fat patch, fat resource,** and **fat trap.**

fat patch A trap patch that contains executable code in two or more instruction sets.

fat resource A code-bearing resource that contains executable code in two or more instruction sets. A fat resource begins with a fat routine descriptor.

fat routine descriptor A routine descriptor that contains routine records for a routine's code in two or more instruction sets.

fat trap A system software routine that is implemented in two or more instruction sets. In general, the Operating System selects the trap implementation that avoids mode switches. See also **split trap.**

file and directory registry A list of files and directories that the Code Fragment Manager should search when looking for import libraries. See also **ROM registry.**

file mapping The process of using a file's data fork as the virtual memory paging file.

Floating-Point Status and Control Register (FPSCR) A 32-bit PowerPC register used to store the floating-point environment.

FP See **frame pointer.**

FPSCR See **Floating-Point Status and Control Register.**

fragment Any block of executable PowerPC code and its associated data.

fragment initialization block A parameter block passed to a fragment's initialization routine that contains information about the fragment. Defined by the `InitBlock` data type.

fragment location record A data structure that provides information about the location of a fragment. Defined by the `FragmentLocator` data type.

frame See **stack frame** or **switch frame.**

frame pointer (FP) A pointer to the beginning of a stack frame. See also **stack pointer.**

function prototype A declaration of the types of parameters expected by a function and of the type of the result it returns. ANSI C requires function prototypes for all functions you define.

global instantiation The method of allocating an import library's static data in which only one copy of that data is created regardless of how many connections to the library are made. See also **per-context instantiation** and **per-load instantiation.**

global variables See **application global variables, QuickDraw global variables,** and **system global variables.**

glue routine A run-time library routine, usually provided by the development environment, that provides the subroutine linkage between high-level language code and a system routine with an interface protocol different from that of the high-level language.

hard import An imported symbol that must be defined at run time and whose corresponding code or data must therefore be available in an import library on the host machine. Compare **import** and **soft import.**

head patch A patch that, upon completion, jumps to the next patch in the patch daisy chain. Compare **tail patch.**

heap An area of memory in which space is dynamically allocated and released on demand, using the Memory Manager. See also **application heap.**

hybrid environment See **mixed environment.**

implementation version The version of an import library that is connected at load time to the application (or other fragment) being loaded. The implementation version provides the actual executable code and data exported by the library. Compare **definition version.**

implicit cross-mode call A call to code that is in a different instruction set architecture from the caller's, caused by the caller executing a routine descriptor.

import To refer to a symbol located in some other fragment. Also, a synonym for **imported symbol.**

imported symbol A symbol in a fragment that references code or data exported by some other fragment. See also **exported symbol** and **import library.**

import library A shared library that is automatically loaded at run time by the Code Fragment Manager.

initialization block See **fragment initialization block.**

initialization routine A function contained in a fragment that is executed immediately after the fragment has been loaded and prepared. See also **termination routine.**

input/output (I/O) The parts of a computer system that transfer data to or from peripheral devices.

instantiation See **global instantiation, per-context instantiation,** and **per-load instantiation.**

instruction cache An area of memory internal to some microprocessors (for example, the MC68020, MC68030, and MC68040 microprocessors) that holds recently used instructions. See also **data cache.**

instruction set architecture The set of instructions meaningful to a particular microprocessor or to a family of microprocessors.

interface files See **universal interface files.**

interrupt See **exception.**

I/O See **input/output.**

jump table An area of memory in a 680x0 application's A5 world that contains one entry for every externally referenced routine in every code segment of the application. The jump table is the means by which the loading and unloading of segments are implemented.

KB Abbreviation for kilobyte. A kilobyte is 1024 bytes.

leaf procedure A routine that calls no other routines.

library See **import library.**

library directory A directory used by an application or other fragment to store import libraries used by that application or fragment. An application's library directory is specified in the application's code fragment resource.

linkage area The area in a PowerPC stack frame that holds the caller's RTOC value and saved values of the Count Register and Link Register. See also **parameter area.**

Link Register (LR) A register in the PowerPC processor that holds the return address of the currently executing routine.

load directory The directory that contains a fragment being loaded into memory and prepared for execution.

local variable A variable allocated and used only within the current procedure.

location record See **fragment location record.**

lock (1) To prevent a relocatable block from being moved during heap compaction. (2) To temporarily prevent a range of physical memory from being paged out or moved by the Virtual Memory Manager.

low-memory global variables See **system global variables.**

LR See **Link Register.**

machine information record A data structure that contains information about the state of the machine at the time an exception occurs. Defined by the `MachineInformation` data type.

Macintosh Operating System The part of Macintosh system software that manages basic low-level operations such as file reading and writing, memory allocation and deallocation, process execution, and interrupt handling.

Macintosh Programmer's Workshop (MPW) A software development system for the Macintosh family of computers provided by Apple Computer.

Macintosh system software A collection of routines that you can use to simplify your development of Macintosh applications. See also **Macintosh Toolbox** and **Macintosh Operating System.**

Macintosh Toolbox The part of the Macintosh system software that allows you to implement the standard Macintosh user interface in your application or other software.

Macintosh User Interface Toolbox See **Macintosh Toolbox**.

main routine A function contained in a fragment whose use depends on the kind of fragment it is in. For applications, the main routine is the usual entry point. See also **main symbol.**

main symbol A symbol whose use depends on the kind of fragment it is in. For applications, the main symbol refers to the fragment's main routine. See also **main routine.**

MB Abbreviation for megabyte. A megabyte is 1024 kilobytes, or 1,048,576 bytes.

memory location record A data structure that provides information about the location of a fragment in memory. Defined by the `MemFragment` data type.

memory management unit (MMU) Any component that performs address mapping in a Macintosh computer. In Macintosh II computers, it is either the Address Management Unit (AMU) or the Paged Memory Management Unit (PMMU). The MMU function is built into the MC68030 and MC68040 microprocessors.

Memory Manager The part of the Operating System that dynamically allocates and releases memory space in the heap.

mini-A5 world An area of memory created and maintained by the Process Manager for a native PowerPC application. A native application's mini-A5 world contains a pointer to the application's QuickDraw global variables. See also **A5 world.**

mixed environment A process execution environment that supports applications and other software written in more than one instruction set.

Mixed Mode Manager The part of the Macintosh system software that manages the mixed-mode architecture of PowerPC processor-based computers running 680x0-based code (including system software, applications, and stand-alone code modules).

MMU See **memory management unit.**

mode switch The process of switching the execution context between the CPU's native context and an emulator (for example, the 68LC040 Emulator). See also **switch frame.**

MPW See **Macintosh Programmer's Workshop.**

nanokernel The lowest-level part of the system software for PowerPC processor-based Macintosh computers.

native application An application whose executable code is in the instruction set architecture of the CPU. See also **PowerPC application.**

nonvolatile register A register whose contents must be preserved across subroutine calls. If a routine changes the value of a nonvolatile register, it must save the old value on the stack before changing the register and restore that value before returning. See also **saved registers area** and **volatile register.**

opcode See **operation code.**

Operating System See **Macintosh Operating System.**

operation code The part of a machine instruction that encodes the operation to be performed. Often shortened to **opcode.**

page The basic unit of memory used in virtual memory.

paged memory management unit (PMMU) The Motorola MC68851 chip, used in the Macintosh II computer to perform logical-to-physical address translation and paged memory management.

page fault A special kind of bus error caused by an attempt to access data in a page of memory that is not currently resident in RAM.

paging The process of moving data between physical memory and a paging file.

paging device A volume that contains the backing-store file or a paging file.

paging file A file used to store unneeded pages of memory. See also **backing-store file.**

parameter area The area in a PowerPC stack frame that holds the parameters for any routines called by a given routine. See also **linkage area.**

partition A contiguous block of memory reserved for use by the Operating System or by an application. See also **application partition** and **system partition.**

patch Any code used to repair or augment an existing piece of code. In the context of Macintosh system software, a patch repairs or augments a trap. See also **head patch** and **tail patch.**

PC See **program counter.**

PC-relative A form of instruction addressing in which the destination instruction is some number of instructions before or after the current instruction.

PEF See **Preferred Executable Format.**

per-context instantiation The method of allocating an import library's static data in which one copy of that data is created for each separate application using the library. Using this method, a single application may have only one copy of the static data. See also **global instantiation** and **per-load instantiation.**

per-load instantiation The method of allocating an extension's static data in which one copy of that data is created for each separate connection to the extension. Using this method, a single client may have multiple copies of the static data. See also **global instantiation** and **per-context instantiation.**

PMMU See **paged memory management unit.**

PowerPC See **PowerPC microprocessor.**

PowerPC application An application that contains code only for a PowerPC microprocessor. See also **680x0 application** and **fat application.**

PowerPC compiler Any compiler that produces code that can execute on a PowerPC. See also **680x0 compiler.**

PowerPC microprocessor Any member of the family of PowerPC microprocessors. The MPC601 processor is the first PowerPC CPU.

PowerPC Numerics The floating-point environment on PowerPC processor-based Macintosh computers. This environment provides floating-point data types and arithmetic operations, plus some advanced numerical functions (such as logarithmic and trigonometric functions). See also **Standard Apple Numerics Environment.**

PowerPC processor-based Macintosh computer Any computer containing a PowerPC central processing unit that runs Macintosh system software. See also **680x0-based Macintosh computer.**

PowerPC software Any software (that is, application, extension, driver, or other executable code) that consists of code only for a PowerPC microprocessor. See also **PowerPC application.**

Preferred Executable Format (PEF) The format of executable files used for PowerPC applications and other software running on Macintosh computers. See also **Extended Common Object File Format.**

prepare To resolve imports in a fragment to exports in some import library.

private resource Any executable resource whose behavior is defined by your application (or other kind of software) alone. Compare **accelerated resource.**

procedure information A long word that encodes information about a routine's calling conventions, the sizes and locations of the routine's parameters, and the size and location of the routine's result. Defined by the `ProcInfoType` data type.

procedure pointer A reference generated by a compiler when taking the address of a routine. On 680x0-based Macintosh computers, a procedure pointer is the address of the routine's executable code (and is defined by the `ProcPtr` data type). On PowerPC processor-based Macintosh computers, a procedure pointer is the address of the routine's transition vector.

Process Manager The part of the Macintosh Operating System that provides a cooperative multitasking environment by controlling access to shared resources and managing the scheduling, execution, and termination of applications.

processor cache See **data cache** or **instruction cache.**

`ProcInfoType` See **procedure information.**

`ProcPtr` See **procedure pointer.**

program counter (PC) A register in the CPU that contains a pointer to the memory location of the next instruction to be executed.

prolog A standard piece of code at the beginning of a routine that sets up the routine's stack frame and saves any nonvolatile registers used by the routine. See also **epilog.**

prototype See **function prototype.**

QuickDraw global variables A set of variables stored in a 680x0 application's A5 world that contain information used by QuickDraw.

reduced instruction set computer (RISC) A microprocessor in which all machine instructions are uniformly formatted and are processed through the same steps. See also **PowerPC microprocessor.**

Red Zone The area of memory immediately above the address pointed to by the stack pointer. The Red Zone is reserved for temporary use by a function's prolog and as an area to store a leaf routine's nonvolatile registers.

reentrant exception handler An exception handler that can be interrupted while servicing an exception, then service a new exception, and then complete servicing the original exception.

register-based routine A routine that receives its parameters and returns its results, if any, in registers. See also **stack-based routine.**

RISC See **reduced instruction set computer.**

ROM registry A list of the import libraries that are stored in the ROM of a Macintosh computer. See also **file and directory registry.**

routine descriptor A data structure used by the Mixed Mode Manager to execute a routine. A routine descriptor contains one or more routine records. Defined by the `RoutineDescriptor` data type.

routine record A data structure that contains information about a particular routine. A routine record specifies, among other things, a routine's instruction set architecture, the number and size of its parameters, its calling conventions, and its location in memory. Defined by the `RoutineRecord` data type.

RTOC See **Table of Contents Register.**

run-time environment The execution environment provided by the Process Manager and other system software services. The run-time environment dictates how executable code is loaded into memory, where data is stored, and how functions call other functions and system software routines.

run-time library See **implementation version.**

SANE See **Standard Apple Numerics Environment.**

saved registers area The area in a PowerPC stack frame that holds the saved values of the nonvolatile general-purpose and floating-point registers.

section A region of memory occupied by part of a loaded fragment. When a fragment is loaded, it is divided into a code section and one or more copies of the data section. See also **code section** and **data section.**

segment One of several logical divisions of the code of a 680x0 application. Not all segments need to be in memory at the same time.

segment location record A data structure that provides information about the location of a fragment in the resource fork of a file on disk. Defined by the `SegmentedFragment` data type.

Segment Manager The part of the Macintosh Operating System that loads and unloads the code segments of a 680x0 application into and out of memory.

selector-based trap A system software routine that is called by passing a selector code to a single trap macro.

shared library A fragment that exports functions and global variables to other fragments. A shared library is used to resolve imports during linking and also during the loading and preparation of some other fragment. A shared library can be stored in a file of type `'shlb'`. See also **import library.**

smearing See **byte smearing.**

soft import An imported symbol whose corresponding code or data might not be available in any import library on the host machine and which is therefore undefined at run time. Compare **hard import** and **import.**

SP See **stack pointer.**

split trap A system software routine that is implemented as 680x0 code in ROM and as PowerPC code in an import library. Because the PowerPC code is contained directly in the import library, you cannot patch the PowerPC portion of a split trap. Compare **fat trap.**

stack An area of memory in the application partition that is used for temporary storage of data during the operation of an application or other software.

stack-based routine A routine that receives its parameters and returns its results, if any, on the stack. See also **register-based routine.**

stack frame The area of the stack used by a routine for its parameters, return address, local variables, and temporary storage.

stack pointer (SP) A pointer to the top of the stack. See also **frame pointer.**

stale instruction An instruction in the microprocessor's instruction cache whose corresponding values in RAM have changed. You might need to flush the instruction cache to avoid using stale instructions.

Standard Apple Numerics Environment (SANE) The floating-point environment on 680x0-based Macintosh computers and on Apple II computers. This environment provides floating-point data types and arithmetic operations, plus some advanced numerical functions (such as logarithmic and trigonometric functions). See also **PowerPC Numerics.**

static data The variables and other data that persist between calls to a particular function or fragment.

stub definition function Code that dispatches to a definition function contained elsewhere. See also **definition function.**

stub definition resource An executable resource that contains a stub definition function. See also **definition resource.**

subroutine linkage The mechanism by which one routine calls another, possibly passing arguments and receiving a function result.

switch See **mode switch.**

switch frame A stack frame, created by the Mixed Mode Manager during a mode switch, that contains information about the routine to be executed, the state of various registers, and the address of the previous frame.

symbol A name for a discrete element of code or data in a fragment.

system extension A file of type `'INIT'` that contains executable code. System extensions are loaded into memory at system startup time.

system global variables A collection of global variables stored in the system partition.

system heap An area of memory in the system partition reserved for use by the Operating System.

system partition A partition of memory reserved for use by the Operating System.

table of contents (TOC) An area of static data in a fragment that contains a pointer to each routine or data item that is imported from some other fragment, as well as pointers to the fragment's own static data.

Table of Contents Register (RTOC) A processor register that points to the table of contents of the fragment containing the code currently being executed. On the PowerPC processor, the general-purpose register 2 is dedicated to serve as the RTOC.

tail patch A patch that invokes the next patch in the patch daisy chain as a subroutine, guaranteeing that the tail patch regains control after the execution of all subsequent patches. Compare **head patch.**

temporary memory Memory allocated outside an application partition that may be available for occasional short-term use.

termination routine A function contained in a fragment that is executed just before the fragment is unloaded. See also **initialization routine.**

TOC See **table of contents.**

tool See **application extension.**

transition vector An area of static data in a fragment that describes the entry point and TOC address of a routine. See also **procedure pointer.**

trap Any of a large set of Macintosh system software routines accessed via A-line instructions. See also **split trap.**

trap dispatcher The exception handler that deals with the occurrence of A-line instructions, providing the subroutine linkage between the A-line instruction and Macintosh system code.

trap dispatch table A table of entry points to Macintosh system routines that are invoked with A-line instructions.

Trap Manager The part of the Macintosh Operating System that provides the subroutine linkage to most Macintosh system software routines.

trap patch See **patch.**

universal interface files A set of interface files that you can use with both 680x0 compilers and PowerPC compilers.

universal procedure pointer A 680x0 procedure pointer or the address of a routine descriptor.

VBL See **vertical retrace interrupt.**

VBL task A task executed during a vertical retrace interrupt.

vector See **transition vector.**

vertical blanking interrupt (VBL) See **vertical retrace interrupt.**

vertical retrace interrupt An interrupt generated by the video circuitry each time the electron beam of a monitor's display tube returns from the lower-right corner of the screen to the upper-left corner.

virtual memory Addressable memory beyond the limits of the available physical RAM. The Operating System extends the logical address space by allowing unused code and data to be stored on a secondary storage device instead of in physical RAM.

Virtual Memory Manager The part of the Operating System that provides virtual memory.

volatile register A register whose contents need not be preserved across subroutine calls. See also **nonvolatile register.**

weak import See **soft import.**

XCOFF See **Extended Common Object File Format.**