Due December **16 noon**. Please **type** your answers and indicate page # where answer was found. **50** points. Provide answers to **5** chapters, which **must** include 1, 2 and 4 (ie choose 2 from 3, 5, 6, 7, 8).

§1: The Mission

1. Why did Microsoft (MS) abandon the OS/2 effort?
2. What are the 5 market requirements Cutler’s team defined for the new operating system?
3. Because one of the design goals was extensibility, MS borrowed from the Mach OS the design around a kernel & servers. Give the NT name for these two and indicate how (and why) the implementation of the second one is different from Mach.
4. What language(s) is NT written in, and why this choice?
5. NT offers both binary- and source level compatibility with applications other than its own WIN32 API. Define these 2 types of compatibility, and list the applications for each type that NT is compatible with.

§2: System Overview

6. Explain using figure 2-6 how the structure of NT “borrows from both the layered model and the client/server model”.
7. Why are NT servers called protected subsystems? How is this protection accomplished?
8. Name the 2 types of protected subsystems, and group the ones listed in figure 2-6 by type.
9. Although the executive provides API-like services, “it is fundamentally different from the environment subsystems”. In what way?
10. What is the purpose of HAL in figure 2-6?
11. What is a VDM and how is it different from all other subsystems?
12. How big is the virtual address space of each process, and how is it divided up?
13. NT has an NLS API to support different locales. What type of facilities does this API provide?
14. Why is ASCII an inadequate code set, and what new standard (adopted by NT) replaces it?
15. What is an exception, and how is it different from an interrupt?
16. NT enhances the C language to provide for structured exception handling, with the keywords **try** and **except**. Explain how this technique addresses the 3 drawbacks of the common primitive approach of “returning a special value to indicate an exception”.
17. What does a termination handler do, and give an example of its use.

§3: The Object Manager & Object Security

18. Certain data structures in the executive, like process, thread, file, are implemented as objects, whereas others are not. What are the 4 criteria used to decide whether to place the data in objects?
19. Because NT is implemented in C, it is “not an object-oriented system” in the strict sense”; what are 3 of OO constructs that are missing?
20. In what sense is the object model similar to the file model (4 characteristics); in what 2 ways are they different?
21. Benefits of using object handles are indirection, common mechanism for different objects, and gating effect of the object manager. Explain the 3 goals accomplished by this gating.
22. Each object’s structure consists of 2 parts: the header and the body. The latter, variable part, is controlled by their relevant executive components. Who controls the header, and why is the data it stores necessary?
23. What 3 steps must an executive component take when it wishes to define a new object type?
24. What are the 2 reasons for including a type object pointer field in Table 3-2? Give an example of such a static, object-type-specific attribute.
25. To “keep track of” “a multitude of objects”, NT assigns a name to each object. What 3 requirements are satisfied by this global namespace?

26. Explain the example A:\budget\accounts.xls (p64) in terms of object domains and parse methods.

27. Explain the example A:\budget\accounts.xls (p68) in terms of symbolic links.

28. What is object retention (related to Q21), and why are there 2 steps (phases) needed to accomplish this?

29. The parse method mentioned in Q26 is an instance of the methods attribute of the process type object (table 3-4). It is “similar to C++ constructors and destructors, that is, routines that are automatically called when an object is created or destroyed”, but unlike con/de-structors it can be called in other situations also. Custer uses the close method to illustrate why such hooks are necessary for the object manager. Explain.

30. B2 level security is even more stringent than the C2 level supported by NT. Which of the 4 features listed for C2 are not allowed in B2, and what is recognition of compartments?

31. What are ACEs and ACLs, and how does NT decide which ACL to assign to a new object?

§4: Processes and Threads

32. How does Mark Lucovsky, principle architect of the NT process manager, define the goal of this executive component?

33. Why are the object table and virtual address space descriptions of figure 4-2 not listed as part of figure 4-3?

34. Describe APCs and the difference between the 2 types.

35. Explain the significance of the last 2 words in the following quote: “The NT executive views process creation as the creation of an object — nothing more.” (p109)

36. Give an example use of the optional inheritance parameter of table 4-5, i.e. illustrate how this gives the environment subsystems like POSIX or OS/2 “freedom to ... establish an operating system environment for [their client applications] that differs from the NT executive’s native environment.”

§5: Windows and the Protected Subsystems

37. Why is NT sometimes referred to as Hydra, or as a chameleon OS? In this NT is similar and yet different from Mach. Explain.

38. What were the 2 original APIs that the NT design team set out to model?

39. Custer states that it isn’t enough for the NT executive “to implement multiple APIs” but that it should also guarantee that “its underlying execution environment be wholly compatible with the application’s native environment”. Name some of the things that are implied by the term environment.

40. To provide the execution environment of Q39, the NT designers finally decided on a client/server model. Briefly explain what was wrong wit the other 2 options they considered.

41. The client/server model has 3 benefits: clean separation of mechanisms from presentation of these mechanisms to applications; any number of subsystems can coexist; and it provides for robustness and security. The latter is done by replacing links to shared DLLs by LPC messages. Explain why this is more secure, and yet doesn’t require recoding OS/2 and 16-bit windows programs (figure 5-6).

42. What 2 reasons make an API call in a client/server model potentially slower than in traditional (layered) OSs?

43. Performance issues were addressed by highly optimizing the code of Q42, and by using private DLLs wherever possible. What are the latter?

44. NT’s environment subsystems not only interact with client applications but also with one another. Briefly describe 2 types of such interaction.

45. What are consoles?

46. What was the “number-one priority” in evolving the Windows3.0 API into a new Win32 API? In what 2 ways, however, do the Win32 API functions differ from their 16-bit counterparts?

47. Explain the difference between figures 5-14 & 5-15, and what prompted these changes.

48. Because the Win32 GDI was implemented as a protected subsystem, its API routines were split into client-side (private) DLLs and server implementations (global) (cf Q43). Because of the context-switching
performance penalty of the latter, Whitmer & Diamond use 2 optimization techniques. Briefly describe them. Are these totally transparent to programmers? Why (not)?

49. Traditional MS-DOS programs are written in assembly language, and “assume that they have free access to memory, devices, and so forth.” Yet the MS-DOS subsystem allows users to run such programs without affecting the rest of the OS; how did Bharati and Hastings accomplish this (i.e. how come these programs can run even on a MIPS machine, and how come they don’t affect other processes)?

50. Multiple MS-DOS applications can run, each in its own VDM (cf figure 5-17). Yet memory usage remains fairly low. How is this accomplished?

51. Why are multiple MS-DOS applications run in their own VDM, but multiple 16-bit Windows applications in a single WOW?

52. 16-bit Windows applications multitask both preemptively and non-preemptively, depending on the perspective. Explain.

53. Threads within a single process can communicate easily, because they share the same address space. What are 2 approaches for communication between 2 threads running in different processes, and which option is used in NT. What is it called, and why is it called that?

54. The LPC facility can pass messages 3 different ways, “each designed for a different situation” (p156). What are those 3 situations?

55. Copying a message between the two service queues in figure 5-22 involves an intermediate location (message queue). Why?

56. What is a section object, and how is it used for communication?

57. What is a callback mechanism, and how is NT’s implementation superior to other OSs (2 reasons).

58. Quick LPC is optimized in 3 ways to speed up message passing: name them. What is the tradeoff?

§6: The Virtual Memory Manager

59. What are the five tasks of a virtual memory system?

60. What is the difference between reserving and committing virtual memory?

61. Explain how Win32 applications can “conveniently perform random I/O (in addition to sequential I/O) to large files” by using mapped files. Phrase your answer in terms of sections and views.

62. Why is the VM manager more than any other part of NT subject to hardware differences among platforms?

63. Explain how figure 6-7 works. Why is this an example of lazy evaluation?

64. What is the difference between figures 6-9 and 6-10? (done in hardware-software? In what order?)

65. What strategy does NT use for fetch, placement, and replacement policies?

66. What is the meaning of transitional page entries, as in figure 6-13 or 6-10?

67. In what order does the VM manager look for a page frame to service a page fault? (5 possibilities).

68. Because NT can do SMP, multiple threads must be prevented from accessing the page frame database simultaneously. This is done with a spin lock, thus “giving preference to fast page faults over increased parallelism in the pager”. Explain.

§7: The Kernel

69. What 2 means are used by the kernel to “shield the NT executive from an eventual ‘dumping ground fate’”? Another way of describing this, is the principle of separating __________.

70. What are the 5 main tasks of the kernel?

71. Name 3 ways the kernel is different from the rest of the executive.

72. In the 6-state model of figure 7-3, explain the purpose of the states standby and transition.

73. By tweaking the priority level, the kernel’s dispatcher avoids penalizing I/O-bound threads. Explain how this works (affects both I/O- and CPU-bound threads).

74. What is the purpose of the idle threads?

75. Are the 32 ready queues of figure 7-4 truly FIFO? When are they not?

76. In what sense does the trap handler act as a “switchboard”?

77. What is masking an interrupt, and why is it done?

78. Interrupt objects allow device drivers to register their ISR with a particular level of interrupt in the IDT. List 4 advantages of using this approach.
79. Explain the marked part of the next quote using figure 7-9: “activating the dispatcher by using a software interrupt is a way to defer dispatching until conditions are right”.

80. What is meant with “DPC routines execute ‘under the covers’”? How can a specific thread be targeted for interruption instead?

81. What is the reason for the int 2Eh instruction?

82. “The way in which the kernel implements system services allows new services to be added dynamically to the OS in future releases”. Explain how this is done.

83. What mechanism does NT use to guarantee mutual exclusion rather than simply disabling interrupts? (this works only if NT is run on a single processor)

84. To guarantee ME on a multiprocessor system, NT uses a spin lock. Why is it called that? How is it implemented?

§8: The IO System

85. What 2 reasons does Lister give for the I/O area to be considered one of the more “sordid” ones?

86. What are the 9 design goals of NT’s I/O system?

87. What are the 3 functions of the I/O manager?

88. What are the 2 challenges in designing an I/O model?

89. NT programs manipulate files using file handles which refer to an executive file object. Why use objects?

90. NT defines “driver” rather broadly. Explain what is included.

91. What is the difference between synchronous and asynchronous IO, and what is the benefit of the latter?

92. What is mapped I/O?

93. Device interrupts:
   a. are handled by ISRs in 2 steps. Explain.
   b. are implemented using DPCs. What is the advantage?

94. What are the 2 differences between a DPC and an APC?

95. What is “reusing an IRP”? Give an example.

96. a. Using the file handle to wait for the completion of an asynchronous I/O operation can cause problems if there is more than 1 outstanding request. Explain
   b. What 2 solutions exist to this problem?

97. What are the 9 standard components of a driver?

98. What are the 3 advantages to the I/O manager of using objects to record information about drivers?

99. a. How do DOS and OS/2 prevent the problem of Figure 8-24?
   b. Why does this not work with SMP?

100. What 2 means does a device driver in NT have to help recover from power failure?