SOLUTIONS MANUAL

OPERATING SYSTEMS
NINTH EDITION

CHAPTERS 1–9

WILLIAM STALLINGS

NOTICE

This manual contains solutions to the review questions and homework problems in *Operating Systems, Ninth Edition*. If you spot an error in a solution or in the wording of a problem, I would greatly appreciate it if you would forward the information via email to wllmst@me.net. An errata sheet for this manual, if needed, is available at http://www.box.net/shared/fa8a0oyxxl . File name is S-OS9e-mmyy.

W.S.

TABLE OF CONTENTS

CHAPTER 1 COMPUTER SYSTEM OVERVIEW

ANSWERS TO QUESTIONS

1.1 A processor, which controls the operation of the computer and performs its data processing functions ; a **main memory**, which stores both data and instructions; **I/O modules**, which move data between the computer and its external environment; and the system bus, which provides for communication among processors, main memory, and I/O modules.

1.2 **User-visible registers:** Enable the machine- or assembly-language programmer to minimize main memory references by optimizing register use. For high-level languages, an optimizing compiler will attempt to make intelligent choices of which variables to assign to registers and which to main memory locations. Some high-level languages, such as C, allow the programmer to suggest to the compiler which variables should be held in registers. **Control and status registers:** Used by the processor to control the operation of the processor and by privileged, operating system routines to control the execution of programs.

1.3 These actions fall into four categories: **Processor-memory:** Data may be transferred from processor to memory or from memory to processor. **Processor-I/O:** Data may be transferred to or from a peripheral device by transferring between the processor and an I/O module. **Data processing:** The processor may perform some arithmetic or logic operation on data. **Control:** An instruction may specify that the sequence of execution be altered.

1.4 An interrupt is a mechanism by which other modules (I/O, memory) may interrupt the normal sequencing of the processor.

1.5 Two approaches can be taken to dealing with multiple interrupts. The first is to disable interrupts while an interrupt is being processed. A second approach is to define priorities for interrupts and to allow an interrupt of higher priority to cause a lower-priority interrupt handler to be interrupted.

1.6 The three key characteristics of memory are cost, capacity, and access time.

1.7 Cache memory is a memory that is smaller and faster than main memory and that is interposed between the processor and main memory. The cache acts as a buffer for recently used memory locations.

1.8 A multicore computer is a special case of a multiprocessor, in which all of the processors are on a single chip.

1.9 **Spatial locality** refers to the tendency of execution to involve a number of memory locations that are clustered. **Temporal locality** refers to the tendency for a processor to access memory locations that have been used recently.

1.10 **Spatial locality** is generally exploited by using larger cache blocks and by incorporating prefetching mechanisms (fetching items of anticipated use) into the cache control logic. **Temporal locality** is exploited by keeping recently used instruction and data values in cache memory and by exploiting a cache hierarchy.

ANSWERS TO PROBLEMS

1.1 Memory (contents in hex): 300: 3005; 301: 5940; 302: 7006
Step 1: 3005 → IR; **Step 2:** 3 → AC
Step 3: 5940 → IR; **Step 4:** 3 + 2 = 5 → AC
Step 5: 7006 → IR; **Step 6:** AC → Device 6

1.2
1. a. The PC contains 300, the address of the first instruction. This value is loaded in to the MAR.
b. The value in location 300 (which is the instruction with the value 1940 in hexadecimal) is loaded into the MBR, and the PC is incremented. These two steps can be done in parallel.
c. The value in the MBR is loaded into the IR.
2. a. The address portion of the IR (940) is loaded into the MAR.
b. The value in location 940 is loaded into the MBR.
c. The value in the MBR is loaded into the AC.
3. a. The value in the PC (301) is loaded in to the MAR.
b. The value in location 301 (which is the instruction with the value 5941) is loaded into the MBR, and the PC is incremented.
c. The value in the MBR is loaded into the IR.
4. a. The address portion of the IR (941) is loaded into the MAR.
b. The value in location 941 is loaded into the MBR.
c. The old value of the AC and the value of location MBR are added and the result is stored in the AC.

5. a. The value in the PC (302) is loaded in to the MAR.
b. The value in location 302 (which is the instruction with the value 2941) is loaded into the MBR, and the PC is incremented.
c. The value in the MBR is loaded into the IR.
6. a. The address portion of the IR (941) is loaded into the MAR.
b. The value in the AC is loaded into the MBR.
c. The value in the MBR is stored in location 941.

1.3 a. 2^{24} = 16 MBytes
b. (1) If the local address bus is 32 bits, the whole address can be transferred at once and decoded in memory. However, since the data bus is only 16 bits, it will require 2 cycles to fetch a 32-bit instruction or operand.
(2) The 16 bits of the address placed on the address bus can't access the whole memory. Thus a more complex memory interface control is needed to latch the first part of the address and then the second part (since the microprocessor will end in two steps). For a 32-bit address, one may assume the first half will decode to access a "row" in memory, while the second half is sent later to access a "column" in memory. In addition to the two-step address operation, the microprocessor will need 2 cycles to fetch the 32 bit instruction/operand.
c. The program counter must be at least 24 bits. Typically, a 32-bit microprocessor will have a 32-bit external address bus and a 32-bit program counter, unless on-chip segment registers are used that may work with a smaller program counter. If the instruction register is to contain the whole instruction, it will have to be 32-bits long; if it will contain only the op code (called the op code register) then it will have to be 8 bits long.

1.4 In cases **(a)** and **(b)**, the microprocessor will be able to access 2^{16} = 64K bytes; the only difference is that with an 8-bit memory each access will transfer a byte, while with a 16-bit memory an access may transfer a byte or a 16-byte word. For case **(c)**, separate input and output instructions are needed, whose execution will generate separate "I/O signals" (different from the "memory signals" generated with the execution of memory-type instructions); at a minimum, one additional output pin will be required to carry this new signal. For case **(d)**, it can support 2^8 = 256 input and 2^8 = 256 output byte ports and the same number of input and output 16-bit ports; in either case, the distinction between an input and an output port is defined by the different signal that the executed input or output instruction generated.

1.5 Clock cycle = $\frac{1}{8\text{ MHz}} = 125\text{ ns}$

Bus cycle = 4×125 ns = 500 ns
2 bytes transferred every 500 ns; thus transfer rate = 4 MBytes/sec

Doubling the frequency may mean adopting a new chip manufacturing technology (assuming each instructions will have the same number of clock cycles); doubling the external data bus means wider (maybe newer) on-chip data bus drivers/latches and modifications to the bus control logic. In the first case, the speed of the memory chips will also need to double (roughly) not to slow down the microprocessor; in the second case, the "word length" of the memory will have to double to be able to send/receive 32-bit quantities.

1.6 **a.** Input from the Teletype is stored in INPR. The INPR will only accept data from the Teletype when FGI=0. When data arrives, it is stored in INPR, and FGI is set to 1. The CPU periodically checks FGI. If FGI =1, the CPU transfers the contents of INPR to the AC and sets FGI to 0.

When the CPU has data to send to the Teletype, it checks FGO. If FGO = 0, the CPU must wait. If FGO = 1, the CPU transfers the contents of the AC to OUTR and sets FGO to 0. The Teletype sets FGI to 1 after the word is printed.

b. The process described in **(a)** is very wasteful. The CPU, which is much faster than the Teletype, must repeatedly check FGI and FGO. If interrupts are used, the Teletype can issue an interrupt to the CPU whenever it is ready to accept or send data. The IEN register can be set by the CPU (under programmer control)

1.7 If a processor is held up in attempting to read or write memory, usually no damage occurs except a slight loss of time. However, a DMA transfer may be to or from a device that is receiving or sending data in a stream (e.g., disk or tape), and cannot be stopped. Thus, if the DMA module is held up (denied continuing access to main memory), data will be lost.

1.8 Let us ignore data read/write operations and assume the processor only fetches instructions. Then the processor needs access to main memory once every microsecond. The DMA module is transferring characters at a rate of 1200 characters per second, or one every 833 μs. The DMA therefore "steals" every 833rd cycle. This slows down the processor approximately $\frac{1}{833} \times 100\% = 0.12\%$

1.9 a. The processor can only devote 5% of its time to I/O. Thus the maximum I/O instruction execution rate is $10^6 \times 0.05 = 50{,}000$ instructions per second. The I/O transfer rate is therefore 25,000 words/second.

b. The number of machine cycles available for DMA control is

$$10^6(0.05 \times 5 + 0.95 \times 2) = 2.15 \times 10^6$$

If we assume that the DMA module can use all of these cycles, and ignore any setup or status-checking time, then this value is the maximum I/O transfer rate.

1.10 a. A reference to the first instruction is immediately followed by a reference to the second.

b. The ten accesses to `a[i]` within the inner for loop which occur within a short interval of time.

1.11 Define

C_i = Average cost per bit, memory level i

S_i = Size of memory level i

T_i = Time to access a word in memory level i

H_i = Probability that a word is in memory i and in no higher-level memory

B_i = Time to transfer a block of data from memory level (i + 1) to memory level i

Let cache be memory level 1; main memory, memory level 2; and so on, for a total of N levels of memory. Then

$$C_s = \frac{\sum_{i=1}^{N} C_i S_i}{\sum_{i=1}^{N} S_i}$$

The derivation of T_s is more complicated. We begin with the result from probability theory that:

$$\text{Expected Value of } x = \sum_{i=1}^{N} i \Pr[x = 1]$$

We can write:

$$T_s = \sum_{i=1}^{N} T_i H_i$$

We need to realize that if a word is in M_1 (cache), it is read immediately. If it is in M_2 but not M_1, then a block of data is transferred from M_2 to M_1 and then read. Thus:

$$T_2 = B_1 + T_1$$

Further

$$T_3 = B_2 + T_2 = B_1 + B_2 + T_1$$

Generalizing:

$$T_i = \sum_{j=1}^{i-1} B_j + T_1$$

So

$$T_s = \sum_{i=2}^{N} \sum_{j=1}^{i-1} \left(B_j H_i \right) + T_1 \sum_{i=1}^{N} H_i$$

But $$\sum_{i=1}^{N} H_i = 1$$

Finally

$$T_s = \sum_{i=2}^{N} \sum_{j=1}^{i-1} \left(B_j H_i \right) + T_1$$

1.12 **a.** Cost = $C_m \times 8 \times 10^6 = 8 \times 10^3$ ¢ = \$80

b. Cost = $C_c \times 8 \times 10^6 = 8 \times 10^4$ ¢ = \$800

c. From Equation 1.1 : $1.1 \times T_1 = T_1 + (1 - H)T_2$

$(0.1)(100) = (1 - H)(1200)$

$H = 1190/1200$

1.13 There are three cases to consider:

Location of referenced word	Probability	Total time for access in ns
In cache	0.9	20
Not in cache, but in main memory	(0.1)(0.6) = 0.06	60 + 20 = 80
Not in cache or main memory	(0.1)(0.4) = 0.04	12ms + 60 + 20 = 12,000,080

So the average access time would be:

Avg = (0.9)(20) + (0.06)(80) + (0.04)(12000080) = 480026 ns

1.14 Yes, if the stack is only used to hold the return address. If the stack is also used to pass parameters, then the scheme will work only if it is the control unit that removes parameters, rather than machine instructions. In the latter case, the processor would need both a parameter and the PC on top of the stack at the same time.