Contact me in order to access the whole complete document. Email: solution9159@gmail.com
WhatsApp: https://wa.me/message/2H3BV2L5TTSUF1 Telegram: https://t.me/solutionmanual

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 Multiple interrupts may be serviced by assigning different priorities to interrupts arising from different sources. This enables a higher-priority interrupt to be serviced first when multiple requests arrive simultaneously; it also allows a higher-priority interrupt to preempt a lower-priority interrupt. For example, suppose a system has assigned a higher priority to a communication line and a lower priority to a magnetic disk. When two simultaneous requests arrive, the computer services the communication line. Similarly, if some disk operations are

ongoing when a request for the communication line arrives, the state of the disk is put in a stack and the communication line operations are catered to.

1.6 The characteristics observed while going up the memory hierarchy are **a.** increasing cost per bit, **b.** decreasing capacity, **c.** decreasing access time, and **d.** increasing frequency of access to the memory by the processor.

1.7 The main trade-offs for determining the cache size are the speed and the cost of the cache.

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: 3007; 301: 4880; 302: 2881;
Step 1: 3007 → IR **Step 2**: 6 → AC
Step 3: 4880 → IR **Step 4**: 6 SUB 5 (0110 – 0101 = 0001) = 1 → AC
Step 5: 2881 → IR **Step 6**: AC → Memory Location 881

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. Number of bits for memory address is 64 – 4×8 = 32. Hence, the maximum addressable memory capacity is 2^{32} = 4 GBytes.
b. The address buses must be ideally 64 bits so that the whole address can be transferred at once and decoded in the memory without requiring any additional memory control logic.
In case of 64-bit data buses, the whole instruction or operand can be transferred in one cycle. 32-bit data buses will require 2 fetch cycles and 16-bit data buses will require 4 fetch cycles. Hence, system speed will be reduced for lesser capacity buses.
c. If the IR is to contain only the opcode, it should contain 32 bits. However, if it contains the whole instruction, it should contain 64 bits.

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 = 1/16 MHz = 62500 ps = 62.5 ns
Bus cycle = 4 × 62.5 ns = 250 ns
4 bytes transferred every 250 ns; thus transfer rate = 16 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 64-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 1350 characters per second, or one every 740 μs. The DMA therefore "steals" every 740th cycle. This slows down the processor approximately $\frac{1}{740} \times 100\% = 0.14\%$.

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 Let the three memory hierarchies be M1, M2, and M3.
Let us define the following parameters:
T_s = average system access time
T_1, T_2, and T_3 = access time of M1, M2, and M3 respectively.
$h1$, $h2$ = hit ratios of memories M1 and M2.
Cs = average cost per bit of combined memory.
C_1, C_2, and C_3 = cost per bit of M1, M2, and M3 respectively.

Extension of Equation (1.1) to 3-level memory hierarchy:
We can say that a word is found in M1 with a probability $h1$.
So the word is not found in M1 with a probability $(1 - h1)$.
Or in other words, memory M2 is accessed with a probability $(1 - h1)$.
As the hit ratio of M2 is $h2$, the word is found in M2 with a probability $(1 - h1)h2$.
So, memory M3 is accessed with a probability $(1 - h1)\ (1 - h2)$.
If we multiply the probabilities of access with the access times and sum up, we will get the average system access time.
Hence, $T_S = h1\ .T_1 \ + (1 - h1)h2.T_2 \ + \ (1 - h1)\ (1 - h2).T_3$

Extension of Equation (1.2) to 3-level memory hierarchy:
Average cost = Total cost/Total size of memory

$$= \frac{C_1S_1 + C_2S_2 + C_3S_3}{S_1 + S_2 + S_3}$$

1.12 **a.** Cost of 1 GByte of main memory = $C_m \times 8 \times 10^9$ = \$64,000

b. Cost of 1 MByte of cache memory = $C_c \times 8 \times 10^6$ ¢ = \$400

c. From Equation 1.1:

$2 \times T_c = T_c + (1 - H)T_m$

$2 \times 120 = (1 - H) \times 1500$

H = 1260/1500 = 0.84

1.13 There are three cases to consider:

Location of the preferred word	Probability	Total time for access in ns
In cache	0.85	25
Not in cache, but in main memory	(1 − 0.85) × 0.8 = 0.12	25 + 100 = 125
Neither in cache nor in main memory	(1 − 0.85) × (1 − 0.8) = 0.03	10 ms + 100 + 25 = 1,000,125

Average Access Time = 0.85 × 25 + 0.12 × 125 + 0.03 × 1000125
= 21.25 + 15 + 30003.75
= 30040 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.

CHAPTER 2 OPERATING SYSTEM OVERVIEW

ANSWERS TO QUESTIONS

2.1 **Convenience:** An operating system makes a computer more convenient to use. **Efficiency:** An operating system allows the computer system resources to be used in an efficient manner. **Ability to evolve:** An operating system should be constructed in such a way as to permit the effective development, testing, and introduction of new system functions without interfering with service.

2.2 The kernel is a portion of the operating system that includes the most heavily used portions of software. Generally, the kernel is maintained permanently in main memory. The kernel runs in a privileged mode and responds to calls from processes and interrupts from devices.

2.3 Multiprogramming is a mode of operation that provides for the interleaved execution of two or more computer programs by a single processor.

2.4 A process is a program in execution. A process is controlled and scheduled by the operating system.

2.5 The **execution context,** or **process state,** is the internal data by which the operating system is able to supervise and control the process. This internal information is separated from the process, because the operating system has information not permitted to the process. The context includes all of the information that the operating system needs to manage the process and that the processor needs to execute the process properly. The context includes the contents of the various processor registers, such as the program counter and data registers. It also includes information of use to the operating system, such as the priority of the process and whether the process is waiting for the completion of a particular I/O event.

2.6 **Process isolation:** The operating system must prevent independent processes from interfering with each other's memory, both data and instructions. **Automatic allocation and management:** Programs