# SOLUTIONS MANUAL

## COMPUTER ORGANIZATION AND ARCHITECTURE: DESIGNING FOR PERFORMANCE ELEVENTH EDITION

CHAPTERS 1–11

WILLIAM STALLINGS

# NOTICE

**This manual contains solutions to the review questions and homework problems in *Computer Organization and Architecture, Eleventh 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/q4a7bmmtyc . File name is S-COA11e-mmyy**

**W.S.**

## TABLE OF CONTENTS

# CHAPTER 1 BASIC CONCEPTS AND COMPUTER EVOLUTION

## ANSWERS TO QUESTIONS

**1.1** **Computer architecture** refers to those attributes of a system visible to a programmer or, put another way, those attributes that have a direct impact on the logical execution of a program. **Computer organization** refers to the operational units and their interconnections that realize the architectural specifications. Examples of architectural attributes include the instruction set, the number of bits used to represent various data types (e.g., numbers, characters), I/O mechanisms, and techniques for addressing memory. Organizational attributes include those hardware details transparent to the programmer, such as control signals; interfaces between the computer and peripherals; and the memory technology used.

**1.2** Computer structure refers to the way in which the components of a computer are interrelated. Computer function refers to the operation of each individual component as part of the structure.

**1.3** Data processing; data storage; data movement; and control.

**1.4** **Central processing unit (CPU):** Controls the operation of the computer and performs its data processing functions; often simply referred to as processor.
**Main memory:** Stores data.
**I/O:** Moves data between the computer and its external environment.
**System interconnection:** Some mechanism that provides for communication among CPU, main memory, and I/O. A common example of system interconnection is by means of a system bus, consisting of a number of conducting wires to which all the other components attach.

**1.5** **Control unit:** Controls the operation of the CPU and hence the computer
**Arithmetic and logic unit (ALU):** Performs the computer's data processing functions
**Registers:** Provides storage internal to the CPU

**CPU interconnection:** Some mechanism that provides for communication among the control unit, ALU, and registers

**1.6** In a stored program computer, programs are represented in a form suitable for storing in memory alongside the data. The computer gets its instructions by reading them from memory, and a program can be set or altered by setting the values of a portion of memory.

**1.7** Moore observed that the number of transistors that could be put on a single chip was doubling every year and correctly predicted that this pace would continue into the near future.

**1.8** **Similar or identical instruction set:** In many cases, the same set of machine instructions is supported on all members of the family. Thus, a program that executes on one machine will also execute on any other. **Similar or identical operating system:** The same basic operating system is available for all family members. **Increasing speed:** The rate of instruction execution increases in going from lower to higher family members. **Increasing Number of I/O ports:** In going from lower to higher family members. **Increasing memory size:** In going from lower to higher family members. **Increasing cost:** In going from lower to higher family members.

**1.9** In a microprocessor, all of the components of the CPU are on a single chip.

# ANSWERS TO PROBLEMS

**2.1 a**

| Location | Instruction/Value | Comments |
|---|---|---|
| 0 | <> | Constant (N) [initialized to some value] |
| 1 | 1 | Constant; Integer value = 1 |
| 2 | 2 | Constant; Integer value = 2 |
| 3 | 0 | Variable Y (initialized to integer zero); Sum(Y) |
| 4L | LOAD M(0 | N → AC |
| 4R | ADD M(1) | AC + 1 → AC |
| 5L | MUL M(0) | N(N+1) → AC |
| 5R | DIV M(2) | AC/2 → AC |
| 6L | STOR M(3) | AC → Y; saving the Sum in variable Y |
| 6R | JUMP M(6,20:39) | Done; HALT |

**b.**

| Location | Instruction/Value | Comments |
|---|---|---|
| 0 | <> | Constant (N) [initialized to some value] |
| 1 | 1 | Constant (loop counter increment) |
| 2 | 1 | Variable i (loop index value; current) |
| 3 | 1 | Variable Y = Sum of X values (Initialized to One) |
| 4L | LOAD M(0 | N → AC (the max limit) |
| 4R | SUB M(2) | Compute N–i → AC |
| 5L | JUMP + M(6,0:19) | Check AC > 0 ? [i < N] |
| 5R | JUMP + M(5,20:39) | i=N; done so HALT |
| 6L | LOAD M(2) | i<N so continue; Get loop counter i |
| 6R | ADD M(1) | i+1 in AC |
| 7L | STOR M(2) | AC → i |
| 8R | ADD M(3) | i + Y in AC |
| 8L | STOR M(3) | AC → Y |
| 8R | JUMP M(4,0:19) | Continue at instruction located at address 4L |

**2.2 a.**

| Opcode | Operand |
|---|---|
| 00000001 | 000000000010 |

**b.** First, the CPU must make access memory to fetch the instruction. The instruction contains the address of the data we want to load. During the execute phase accesses memory to load the data value located at that address for a total of two trips to memory.

**2.3** To read a value from memory, the CPU puts the address of the value it wants into the MAR. The CPU then asserts the Read control line to memory and places the address on the address bus. Memory places the contents of the memory location passed on the data bus. This data is then transferred to the MBR. To write a value to memory, the CPU puts the address of the value it wants to write into the MAR. The CPU also places the data it wants to write into the MBR. The CPU then asserts the Write control line to memory and places the address on the address bus and the data on the data bus. Memory transfers the data on the data bus into the corresponding memory location.

**2.4**

| Address | Contents |
|---|---|
| 08A | LOAD M(0FA) |
| | STOR M(0FB) |
| 08B | LOAD M(0FA) |
| | JUMP +M(08D) |
| 08C | LOAD –M(0FA) |
| | STOR M(0FB) |
| 08D | |

This program will store the absolute value of content at memory location 0FA into memory location 0FB.

**2.5** All data paths to/from MBR are 40 bits. All data paths to/from MAR are 12 bits. Paths to/from AC are 40 bits. Paths to/from MQ are 40 bits.

**2.6** The purpose is to increase performance. When an address is presented to a memory module, there is some time delay before the read or write operation can be performed. While this is happening, an address can be presented to the other module. For a series of requests for successive words, the maximum rate is doubled.

**2.7** The discrepancy can be explained by noting that other system components aside from clock speed make a big difference in overall system speed. In particular, memory systems and advances in I/O processing contribute to the performance ratio. A system is only as fast as its slowest link. In recent years, the bottlenecks have been the performance of memory modules and bus speed.

**2.8** As noted in the answer to Problem 2.7, even though the Intel machine may have a faster clock speed (2.4 GHz vs. 1.2 GHz), that does not necessarily mean the system will perform faster. Different systems are not comparable on clock speed. Other factors such as the system components (memory, buses, architecture) and the instruction sets must also be taken into account. A more accurate measure is to run both systems on a benchmark. Benchmark programs exist for certain tasks, such as running office applications, performing floating-point operations, graphics operations, and so on. The systems can be compared to each other on how long they take to complete these tasks. According to Apple Computer, the G4 is comparable or better than a higher-clock speed Pentium on many benchmarks.

**2.9** This representation is wasteful because to represent a single decimal digit from 0 through 9 we need to have ten tubes. If we could have an arbitrary number of these tubes ON at the same time, then those same tubes could be treated as binary bits. With ten bits, we can represent

$2^{10}$ patterns, or 1024 patterns. For integers, these patterns could be used to represent the numbers from 0 through 1023.

**1.10** **a.** No. These programs are never considered to be embedded because they are not an integral component of a larger system.

**b.** Yes, regardless of what the disk drive is used for. The software (firmware, actually) within the disk drive controls the HDA (head disk assembly) hardware and is hard realtime as well.

**c.** No, because that computer may be a general-purpose computer that is not part of a larger system.

**d.** No. People often say that PDAs are embedded because they are very small and constrained and because PDA OS and application software is kept in non-volatile memory, but PDAs parallel the desktop systems used to run office productivity applications, and no special hardware is being controlled.

**e.** Yes. The firmware in the cell phone is controlling the radio hardware.

**f.** Yes. These computers were generally some of the most powerful computers available when the system was built, are located in a large computer room occupying almost one whole floor of a building, and may be hundreds of meters away from the radar hardware. However, the software running in these computers controls the radar hardware; therefore, the computers are an integral component of a larger system.

**g.** If the FMS is not connected to the avionics and is used only for logistics computations, a function readily performed on a laptop, then the FMS is clearly not embedded.

**h.** Yes, both in the simulator, and in the thing being tested in the HIL simulator. Hardware is being controlled on both sides.

**i.** Yes. In this case the "system" is the combination of the pacemaker and the person's heart.

**j.** Yes. It is part of a larger system, the engine, and it is directly monitoring and controlling the engine through special hardware.