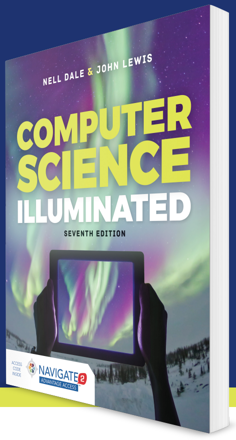

CHAPTER 6
EXERCISES AND ANSWERS

Computer Science Illuminated, Seventh Edition

Nell Dale, PhD; John Lewis, PhD

Answers are in blue.

For Exercises 1–15, mark the answers true and false as follows:

A. True
B. False

1. Arithmetic can be performed in the instruction register.
   B
2. Arithmetic can be performed in the A register.
   A
3. Arithmetic can be performed in the accumulator.
   A (This is another name for the A register.)
4. LDBA 0x008B,i loads the byte 008B into register A.
   A
5. ADDA 0x008B,i adds the contents of 008B to the A register.
   B
6. The program counter and the instruction register are two names for the same place.
   B
7. The A register and the accumulator are two names for the same place.
   A
8. The instruction register is 3 bytes long.
   A
9. The program counter is 3 bytes long.
   B
10. The branch instruction, BR, branches to the location specified in the operand specifier.
    A
11. The instruction specifier is one byte long.
    B
12. If the data to be loaded into the accumulator is stored in the operand, the instruction specifier ends with 000.
    A
13. If the data in the accumulator is to be stored in the place named in the operand, the instruction specifier ends with 000.
    B
14. All Pep/9 instructions use 3 bytes.
    B
15. At least one branching instruction is required in a loop.
    A

Given the following state of memory (in hexadecimal), answer Exercises 16–20 by matching the problem to the solution shown.

```
0001  A2
0002  11
0003  00
0004  FF
```

A. A2 11
B. A2 12
C. 00 02
D. 11 00
E. 00 FF

16. What are the contents of the A register after the execution of this instruction?
    C1 00 01
    A
17. What are the contents of the A register after the execution of this instruction?
    C1 00 02
    D
18. What are the contents of the A register after the execution of the following two instructions?
    C0 00 01
    60 00 01
    C
19. What are the contents of the A register after the execution of the following two instructions?
    C1 00 01
    60 00 01
    B
20. What are the contents of location 0001 after the execution of the following two instructions?
    C1 00 03
    E0 00 01
    E

Exercises 21–60 are programs or short answer questions.

21. What does it mean when we say that a computer is a *programmable* device?

Programmable means that data and instructions are logically the same and are stored in the same place. The consequence of this fact is that the program the computer executes is not wired into the hardware, but entered from outside.

22. List five operations that any machine language must include.
There must be machine-language instructions to store, retrieve, and process data; to input data; and to output data. These instructions mirror the operations of the von Neumann machine.

23. How many low-level tasks can each machine-language instruction perform?
A machine-language instruction can perform only one low-level task.

24. What is a virtual machine? Discuss this definition in terms of the Pep/9 computer.
A virtual machine is a hypothetical machine designed to illustrate important features of a real computer. The Pep/9 computer is a virtual machine designed to illustrate the features of the von Neumann architecture. It has instructions to store, retrieve, and process data, as well as instructions to input and output data.

25. How many bits does an instruction take in Pep/9?
The instructions in Pep/9 are variable in length. Some instructions that do not require operands are stored in only 8 bits (1 byte). Instructions that use operands use 24 bits (3 bytes).

26. Describe the features of the Pep/9 CPU that we covered in this chapter.
There is one register for arithmetic and logical operations: the A register (the accumulator). There is a program counter that contains the address of the next instruction to be executed, and the instruction register contains the instruction being executed. An operand may be immediate (stored in the operand specifier) or direct (stored in the memory location named in the operand specifier).

27. Where is the data (operand) if the address mode specifier is:
A. 000?
B. 001?
A. In the operand specifier
B. In the memory location named in the operand specifier

28. We discussed two mode specifiers. How many are there?
8

29. Distinguish between the IR (instruction register) and the PC (program counter).
The IR contains an instruction (the one being executed); the PC contains an address (the address of the next instruction to be executed).

30. How many bits are required to address the Pep/9 memory?
The Pep/9 memory contains 65,536 bytes, so 16 bits are required to address each one.

31. How many more cells could be added to memory without having to change the instruction format? Justify your answer.
None. The operand specifier could not address more than the current memory.

32. Some Pep/9 instructions are unary, taking only one byte. Other instructions require 3 bytes. Given the instructions that we have covered in this chapter, would it be useful to define instructions that require only 2 bytes?
The instructions we have covered, other than the `Stop` instruction, use the operand specifier of the instruction. The operand specifier is two bytes long, so three bytes are required for the instruction: the instruction specifier and the operand specifier. Therefore, two-byte instructions would not be useful.

33. What is the result of executing the following instructions?
0001 **D0 00 48**
0004 **F1 FC 16**
The character "H" is written on the screen (the first instruction loads the ASCII value of the character into the accumulator and the second instruction writes it to the output device).

34. What is the result of executing the following instructions?
0001 **D0 00 07**
0004 **70 00 02**
The value 5 is stored in the accumulator (the first instruction loads 7 and the second instruction subtracts 2).

35. Write a program in Pep/9 machine code to output the word "go".
D0 00 67 F1 FC 16 D0 00 6F F1 FC 16 zz
Explanation: Load the ASCII code for "g" (67), print character, load the ASCII code for "o" (6F), print character.

36. Write a program in Pep/9 assembly language to output the word "go".
```
LDBA 0x0067,i
STBA 0xFC16,d
LDBA 0x006F,i
STBA 0xFC16,d
.END
```

37. Write a program in Pep/9 machine code to output the word "home".
D0 00 68 F1 FC 16 D0 00 6F F1 FC 16 D0 00 6D F1 FC 16 D0 00 65 F1 FC 16 zz

38. Write a program in Pep/9 assembly language to output the word "home".
```
LDBA 0x0068,i
STBA 0xFC16,d
LDBA 0x006F,i
STBA 0xFC16,d
LDBA 0x006D,i
STBA 0xFC16,d
LDBA 0x0065,i
STBA 0xFC16,d
END
```

39. Explain how input and output work in Pep/9.
Pep/9 uses the principle of memory-mapped I/O, which wires devices to specific, fixed memory addresses. The Pep/9 input device is at address FC15 and the output device is at address FC16. To read a character, load it from the input device into the accumulator. To write a character, load the ASCII character value into the accumulator, then write it to the output device.

40. Distinguish among the Pep/9 menu options *Assemble, Load*, and *Execute (run)*.
Assemble translates the assembly language program into machine code. *Load* puts the program into memory ready to be executed. *Execute* runs the loaded program.

41. The following program seems to run, but does strange things with certain input values. Can you find the bug?

```
        BR   Main
sum:    .WORD         0x0000
num1:   .BLOCK        1
num2:   .BLOCK        1
num3:   .BLOCK        1
Main:   LDWA                 sum,d
        DECI          num1,d
        DECI          num2,d
        DECI          num3,d
        ADDA                 num3,d
        ADDA                 num2,d
        ADDA                 num1,d
        STWA                 sum,d
        DECO                 sum,d
        STOP
        .END
```

One byte of storage is set up for each input value. If the value that is read is greater than one byte, the excess spills over to the byte above, giving the wrong answer.

42. Correct the code in Exercise 41 and run the test plan outlined in the Chapter.
Here is the corrected program (allocating 2 bytes per number):

```
        BR   Main
sum:    .WORD         0x0000
num1:   .BLOCK        2
num2:   .BLOCK        2
num3:   .BLOCK        2
Main:   LDWA                 sum,d
        DECI          num1,d
        DECI          num2,d
        DECI          num3,d
        ADDA                 num3,d
        ADDA                 num2,d
        ADDA                 num1,d
        STWA                 sum,d
        DECO                 sum,d
        STOP
        .END
```

43. What is the purpose of the .ASCII pseudo-operation?
The .ASCII pseudo-operation lets the programmer represent a string of ASCII characters, such as "Hello", with one step, avoiding the need to load each character using a separate instruction.

44. Write a pseudocode algorithm that reads in three values and writes out the result of subtracting the second value from the sum of the first and the third values.

```
Read num1
Read num2
Read num3
Load num1
Add num3
Sub num2
Store in answer
Write answer
```

45. Implement the algorithm in Exercise 44 as a Pep/9 assembly-language program.

```
BR     main
sum:   .WORD         0x0000
num1:         .BLOCK        2
num2:  .BLOCK        2
num3:         .BLOCK        2
answer:   .BLOCK    2
main:  DECI num1,d
       DECI num2,d
       DECI num3,d
       LDWA  num1,d
       ADDA num3,d
       SUBA num2,d
       STWA answer,d
       DECO answer,d
       STOP
       .END
```

46. Write and implement a test plan for the program in Exercise 45.

Reason for Test Case	*Input Values*	*Expected Output*	*Observed Output*
Assumption: Input values are no greater than $21^5 – 1$ or less than $–21^5$.			
Input three positive numbers	4, 6, 1	–1	–1
Input three negative numbers	–4, –6, –1	1	1
Input mixed numbers	4, 6, –1	–3	–3
	4, –6, 1 4, –6, –1 –4, 6, 1 –4, 6, –1 –4, –6, 1	11 –9 –9 –11 3	11 –9 –9 –11 –3
Large numbers	32767, –1, +1 32767, 1, –1	Overflows 32767	Overflows

47. Design with pseudocode, then implement, an assembly-language program that uses a loop to read four values and print their sum.

Pseudocode

```
Set count to 0
Set sum to 0
WHILE count < 4
    Read num
    Add num to sum
    Set count to count + 1
Write sum
```

Program

```
          BR         Read  ;branch to location Read
sum:      WORD 0x0000             ;set word to zero
count: .  WORD 0x0000             ;set up a two byte block for
                                  count
```

```
limit:    WORD 0x0004  ;set up a block for value 4
number:.BLOCK 2        ;set up word for value read
Read:     LDWA   sum,d
          DECI  number,d
          ADDA            number,d
          STWA   sum,d              ;store accumulator in sum
          LDWA   count,d            ;load a copy of limit in accumulator
          ADDA             1,i
          STWA   count,d            ;store contents in count
          CPWA            limit,d   ;compare accumulator to 0
          BREQ   Quit        ;branch to Quit if count is 4
          BR      Read           ;go back to read in another number
Quit:     DECO   sum,d
          STOP
          .END
```

48. Is the test plan for a machine-language program valid for the same solution written in assembly language? Explain your answer.
A data-coverage plan is written without looking at the code, so the same test plan would be valid. A code-coverage plan looks at the code, but because there is a one-to-one relationship between a machine-code instruction and an assembly-language instruction, the same test plan would be valid.

49. Distinguish between the pseudo-operations `.BLOCK` and `.WORD`.
The pseudo-operation `.BLOCK` takes a decimal argument and sets aside that many bytes of storage and set them to zero. A `.WORD` pseudo-operation takes a decimal argument and generates one word of storage with the decimal value stored in it.

50. Distinguish between assembly-language pseudo-operations and mnemonic instructions.
Pseudocode instructions are instructions to the assembler; mnemonic instructions are to be translated by the assembler.

51. Distinguish between test plans based on code coverage and data coverage.
A code-coverage test plan is based on examining and covering the statements in the code. A data-coverage test plan is based on the input data to the program.

52. What button on the Pep/9 console must be pushed for keyboard input?
Terminal I/O

53. Write the Pep/9 assembly-language statement for the following instructions.
    A. Branch to location Branch1 if the accumulator is zero.
    `BREQ Branch1`
    B. Branch to location Branch1 if the accumulator is negative.
    `BRLT Branch1`
    C. Branch to location Branch1 if the accumulator is negative and to Branch2 if accumulator is not negative.
    `BRLT Branch1`
    `Branch2:`
    `That is, go to location Branch1 if the accumulator is negative. If the accumulator is not negative, the next instruction is executed, so it must be labeled Branch2.`

54. Write a pseudocode algorithm to read in a name and write a "Good morning" message.

```
Write "Enter a Name"
Read name
Write "Good morning", name
```

55. Write a pseudocode algorithm to get three integers from the user and print them in numeric order.

```
Write "Enter three integer values"
Read first, second, third
IF (first < second)
    IF (second < third)
        Write first, second, third
    ELSE
        IF (first < third)
            Write first, third, second
        ELSE
            Write third, first, second
ELSE
    IF (first < third)
        Write second, first, third
    ELSE
        IF (second < third)
            Write second, third, first
        ELSE
            Write third, second, first
```

56. Enclose the design in Exercise 55 within a loop that reads in the three values until the user enters the first value of the trio as negative.

```
Write "Enter three values; a negative value stops the processing"
Read first, second, third
WHILE (first > 0)
    // Process the three values
    Write "Enter three values; a negative value stops the processing"
    Read first, second, third
```

57. Rewrite the algorithm in Exercise 56 so that the user has to enter only one negative value to stop (that is, the second and third values are not entered).

```
Write "Enter three values; a negative value stops the processing"
Read first
WHILE (first > 0)
    Read second, third
    // Process the three values
    Write "Enter three values; a negative value stops the processing"
    Read first
```

58. Distinguish between pseudocode and pseudo-operations.
    Pseudocode is a language for expressing algorithms. Pseudo-operations are instructions to a translating system.
59. What are the constructs that pseudocode must be able to express?
    Variables, assignment, input/output, repetition, selection (if-then, if-then-else)
60. Distinguish between the looping construct and the selection construct.
    A looping construct repeats an action as long as a condition is true. A selection construct determines if a condition is true and does one action if it is and another action if it is not.