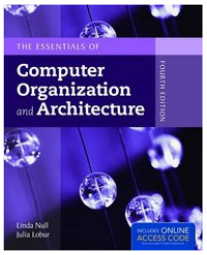


The Essentials of Computer Organization and Architecture
4th Edition

Linda Null and Julia Lobur
Jones and Bartlett Publishers © 2015

Chapter 5 Instructor's Manual

Chapter Objectives

Chapter 5, A Closer Look at Instruction Set Architectures, provides a closer look at instruction set architectures, including instruction formats, instruction types, endianess, and addressing modes. Instruction-level pipelining is introduced as well. Real-world ISAs (including Intel, MIPS, and Java) are discussed to reinforce the concepts presented in the chapter.

This chapter should be covered after Chapter 4.

Lectures should focus on the following points:

- **Instruction formats.** Important issues here include instruction length, little versus big endian, register usage (and the use of stacks), and expanding opcodes.
- **Addressing.** Although addressing is an instruction design issue, there are many aspects to consider, the most important of which is addressing modes.
- **Instruction-level pipelining.** The fetch-decode-execute cycle can be overlapped, resulting in faster execution time. Resource conflicts, conditional branching, and data dependencies can slow this process down.
- **Real-world examples of ISAs.** Intel, MIPS, and the Java virtual machine are all presented to reinforce the concepts of the chapter.

Required Lecture Time

The important concepts in Chapter 5 can typically be covered in 4 lecture hours. However, if a teacher wants the students to have a mastery of all topics in Chapter 5, 6 lecture hours are more reasonable. If lecture time is limited, we suggest that the focus be on instruction formats

(including big versus little endian, use of registers, and instruction length) and addressing (with considerable attention paid to the various modes).

Lecture Tips

It is important to motivate the little versus big endian debate. Many students comprehend the concept, but don't see the significance. The various software applications listed in this chapter will help the instructor make this point.

Students often have difficulty understanding stack machines. It is important to emphasize that not ALL instructions on stack machine have zero operands (push and pop must have operands), but rather, the instructions that allow for operands are limited.

The concept of expanding opcodes is sometimes difficult for students as well. We suggest going over a small example in detail.

When covering the addressing modes, we suggest that instructors include many examples, as this is one concept that tends to be easier to understand through illustration.

Students often confuse instruction-level pipelining with other types of pipelining. It is important to stress that there are many types of pipelining, but in this chapter only pipelining the fetch-decode-execute cycle is addressed.

Although the real-life examples can be left for students to read, these case studies provide the instructor with a means to tie the concepts from this chapter together, as well as a method for motivating study of the concepts from this chapter.

Answers to Exercises

1. Assume you have a byte-addressable machine that uses 32-bit integers and you are storing the hex value 3456 at address 0.
   - ♦a) Show how this is stored on a big endian machine.
   - ♦b) Show how this is stored on a little endian machine.
   - c) If you wanted to increase the hex value to 123456, which byte assignment would be more efficient, big or little endian? Explain your answer.

Ans.

a and b.

Address ⟶	**00**	**01**	**10**	**11**
Big Endian	00	00	34	56
Little Endian	56	34	00	00

c) Little endian is more efficient because the additional information simply needs to be appended. With big endian, the "34" and "56" would need to shift to maintain the correct byte ordering.

---

2. Show how the following values would be stored by byte-addressable machines with 32-bit words, using little endian and then big endian format. Assume each value starts at address 0x10. Draw a diagram of memory for each, placing the appropriate values in the correct (and labeled) memory locations.

   a) 0x456789A1
   b) 0x0000058A
   c) 0x14148888

Ans.

a)

Address ⟶	**0x10**	**0x11**	**0x12**	**0x13**
Big Endian	45	67	89	A1
Little Endian	A1	89	67	45

b)

Address ⟶	**0x10**	**0x11**	**0x12**	**0x13**
Big Endian	00	00	05	8A
Little Endian	8A	05	00	00

c)

Address ⟶	**0x10**	**0x11**	**0x12**	**0x13**
Big Endian	14	14	88	88
Little Endian	88	88	14	14

---

3. Fill in the following table to show how the given integers are represented, assuming 16-bits are used to store values and the machine uses 2's complement notation.

Integer	Binary	Hex	4 Byte Big Endian (hex value as seen in memory)	4 Byte Little Endian (hex value as seen in memory)
28				
2216				
-18675				
-12				
31456				

Ans.

Integer	Binary	Hex	4 Byte Big Endian (hex value as seen in memory)	4 Byte Little Endian (hex value as seen in memory)
28	0000000000011100	001C	001C	1C00
2216	0000100010101000	08A8	08A8	A808
-18675	1011011100001101	B70D	B70D	0DB7
-12	1111111111110100	FFF4	FFF4	F4FF
31456	0111101011100000	7AE0	7AE0	E07A

---

4. Assume a computer that has 32-bit integers. Show how each of the following values would be stored sequentially in memory, starting at address 0x100, assuming each address holds one byte. Be sure to extend each value to the appropriate number of bits. You will need to add more rows (addresses) to store all five values.

a) 0xAB123456
b) 0x2BF876
c) 0x8B0A1
d) 0x1
e) 0xFEDC1234

Byte Order

Address	Big Endian	Little Endian
0x100		
0x101		
0x102		
0x103		
0x104		
0x105		
0x106		
0x107		

...

Ans.

a) 0xAB123456
b) 0x2BF876
c) 0x8B0A1
d) 0x1
e) 0xFEDC1234

Byte Order

Address	Big Endian	Little Endian
0x100	AB	56
0x101	12	34
0x102	34	12
0x103	56	AB
0x104	00	76
0x105	2B	F8
0x106	F8	2B
0x107	76	00
0x108	00	A1
0x109	08	B0
0x10A	B0	08
0x10B	A1	00
0x10C	00	01
0x10D	00	00
0x10E	00	00
0x10F	01	00
0x110	FE	34
0x111	DC	12
0x112	12	DC
0x113	34	FE

---

5. Consider a 32-bit hexadecimal number stored in memory as follows:

Address	Value
100	2A
101	C2
102	08
103	1B

a) If the machine is big endian and uses 2's complement representation for integers, write the 32-bit integer number stored at address 100 (you may write the number in hex).
b) If the machine is big endian and the number is an IEEE single-precision floating point value, is the number positive or negative?
c) If the machine is big endian and the number is an IEEE single-precision floating point value, determine the decimal equivalent of the number stored at address 100 (you may leave your answer in scientific notation form, as a number times a power of two).
d) If the machine is little endian and uses 2's complement representation for integers, write the 32-bit integer number stored at address 100 (you may write the number in hex).
e) If the machine is little endian and the number is an IEEE single-precision floating point value, is the number positive or negative?
f) If the machine is little endian and the number is an IEEE single-precision floating point value, determine the decimal equivalent of the number stored at address 100 (you may leave your answer in scientific notation form, as a number times a power of two).

Ans.

a) 2AC2081B
b) 2A implies first bit is zero so it is positive
c) IEEE single precision has 1 sign bit, 8 exp bits (bias of 127), and 23 bits for the significand. So we have:
2AC2081B = 0010 1010 1100 0010 0000 1000 0001 1011
which can be written: $+1.10000100000100000011011 \times 2^{-42}$

d) 1B08C22A
e) 1B implies first bit is zero so it is positive
f) IEEE single precision has 1 sign bit, 8 exp bits (bias of 127), and 23 bits for the significand. So we have:
1B08C22A = 0001 1011 0000 1000 1100 0010 0010 1010
which can be written: $+1.00010001100001000101010 \times 2^{-73}$

---

6. The first two bytes of a 2M x 16 main memory have the following hex values:

Byte 0 is FE
Byte 1 is 01

If these bytes hold a 16-bit two's complement integer, what is its actual decimal value if:

- a) memory is big endian?
- b) memory is little endian?

Ans.

a) 0xFE01 = 1111 1110 0000 0001_2 = -511_{10}

b) 0x01FE = 0000 0001 1111 1110_2 = 510_{10}

---

7. What kinds of problems do you think endian-ness can cause if you wished to transfer data from a big endian machine to a little endian machine? Explain.

Ans.

If the machines receiving the data uses different endian-ness than the machine sending the data, the values can be misinterpreted. For example, the value from Problem 3, sent as the value -511 on a big endian machine, would be read as the value 510 on a little endian machine.

---

8. The Population Studies Institute monitors the population of the United States. In 2008, this institute wrote a program to create files of the numbers representing the various states, as well as the total population of the U.S. This program, which runs on a Motorola processor, projects the population based on various rules, such as the average number of births and deaths per year. The Institute runs the program and then ships the output files to state agencies so the data values can be used as input into various applications. However, one Pennsylvania agency, running all Intel machines, encountered difficulties, as indicated by the following problem. When the 32-bit unsigned integer 0x1D2F37E8 (representing the overall U.S. population predication for 2013) is used as input, and the agency's program simply outputs this input value, the U.S. population forecast for 2013 is far too large. Can you help this Pennsylvania agency by explaining what might be going wrong? (Hint: They are run on different processors.)

Ans.

Intel and Motorola use different endian-ness. If the program on the Intel machine does not adjust for this, then the integer is interpreted incorrectly.

---

9. There are reasons for machine designers to want all instructions to be the same length. Why is this not a good idea on a stack machine?

Ans.

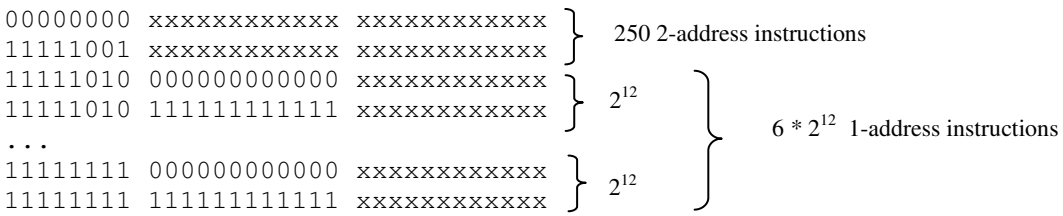
The only instructions on a stack machine that need to address memory are push and pop. So an operand field is required, which implies the instruction field must be divided into an opcode and an operand. However, the other instructions need not access memory and can

thus consist of only the opcode. To make them all the same length, these instructions would need to be "artificially lengthened".

♦10. A computer has 32-bit instructions and 12-bit addresses. Suppose there are 250 2-address instructions. How many 1-address instructions can be formulated? Explain your answer.

Ans.

There are 250 2-address instructions. There are only a total of 256 2-address instructions allowed if we have 32-bit instructions (two addresses take up 24 bits, leaving only 8 bits for the opcode). Looking at the 8 bit opcode, assume bit patterns 00000000 (0) through 11111001 (249) are used for the 250 two-address instructions. Then there are 6 bit patterns left for one address instructions. However, each one of these can use the remaining 12 bits gained from having only one operand, so we have $6 * 2^{12}$.



---

11. Convert the following expressions from infix to reverse Polish (postfix) notation.
    a) (8 – 6) / 2
    b) (2 + 3) * 8 / 10
    c) (5 × (4 + 3) × 2 – 6)

Ans.

a) 8 6 – 2 /

b) 2 3 + 8 * 10 /

c) 5 4 3 + × 2 × 6 –

---

12. Convert the following expressions from infix to reverse Polish (postfix) notation.
♦a) X * Y + W * Z + V * U
    b) W * X + W * (U * V + Z)
    c) (W * (X + Y * (U * V)))/(U * (X + Y))

Ans.

a) X Y * W Z * V U * + +

b) W X * W U V * Z + * +

c) W X Y U V * * + * U X Y + * /

---

13. Convert the following expressions from reverse Polish notation to infix notation.
    a) 12 8 3 1 + – /
    b) 5 2 + 2 × 1 + 2 ×
    c) 3 5 7 + 2 1 – × 1 + +

Ans.

    a) 12 / (8 – (3 + 1))
    b) (((5 + 2) × 2) + 1) × 2
    c) 3 + (((5 + 7) × (2 – 1)) + 1)

---

14. Convert the following expressions from reverse Polish notation to infix notation.
    a) W X Y Z - + *
    b) U V W X Y Z + * + * +
    c) X Y Z + V W - * Z + +

Ans.

    a) W * (X + Y - Z)
    b) U + (V * (W + (X * (Y + Z))))
    c) X + ((Y + Z) * (V - W) + Z)

---

15. Explain how a stack is used to evaluate the RPN expressions from Exercise 11.

Ans.

    a) 12 8 3 1 + – /
    12, 8, 3 and 1 are pushed onto the stack. The plus operator adds 3 + 1, pops them from the stack, and pushes 4. The minus operator takes 8 – 4, pops them from the stack, and pushes 4. The divide operator takes 12/4, pops them from the stack, and pushes 3.
    b) 5 2 + 2 × 1 + 2 ×
    5 and 2 are pushed onto the stack. The plus operator adds 5+2, pops them from the stack, and pushes 7. 2 is pushed, and then the times operator takes 7 X 2, pops them from the stack, and pushes 14. 1 is pushed, and then the plus operator adds 14 + 1, pops them, and pushes 15. 2 is pushed, and then the times operators multiples 15 by 2, pops them, and pushes 30.
    c) 3 5 7 + 2 1 – × 1 + +
    3, 5 and 7 are pushed. The plus operator adds 5 + 7, pops them, and pushes 12. Then 2 and 1 are pushed. The minus operator subtracts 1 from 2, pops them, and pushes 1. The times operator multiplies 12 by 1, pops them, and pushes 12. The 1 is pushed, then the plus operator adds 12 plus 1, pops them, and pushes 13. The plus operator them adds 3 + 13, pops them, and pushes 16.

---

16. a) Write the following expression in postfix (Reverse Polish) notation. Remember the rules of precedence for arithmetic operators!

$$X = \frac{A - B + C * (D * E - F)}{G + H * K}$$

b) Write a program to evaluate the above arithmetic statement using a stack organized computer with zero-address instructions (so only pop and push can access memory).

Ans.

a) A B – C D E * F – * + G H K * + /

b) The program would be:

```
Push A
Push B
Subtract
Push C
Push D
Push E
Mult
Push F
Subtract
Mult
Add
Push G
Push H
Push K
Mult
Add
Div
Pop X
```

---

17. a) In a computer instruction format, the instruction length is 11 bits and the size of an address field is 4 bits. Is it possible to have:

5 2-address instructions
45 1-address instructions
32 0-address instructions

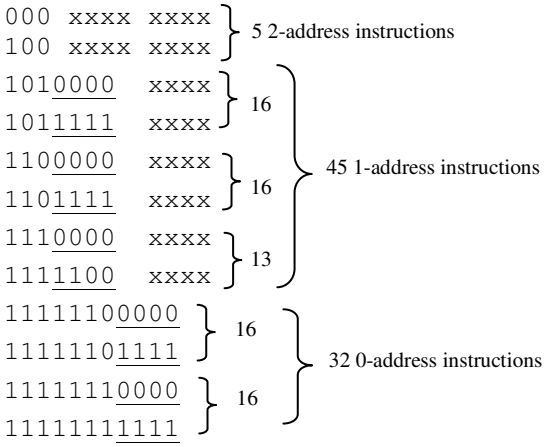
using the specified format? Justify your answer.

b) Assume that a computer architect has already designed 6 two-address and 24 zero-address instructions using the instruction format above. What is the maximum number of one-address instructions that can be added to the instruction set?

Ans.

a) Yes. The 2-address instructions could be represented 000xxxxxxxx through 100xxxxxxxx (using 000 through 100 for opcodes). The 1-address instructions could use 1010000 through 1011111 (16), 1100000 through 1101111 (16), and 1110000 through 1111100

(13 more, for a total of 45). The 0-address instructions could use 1111110000 through 1111110<u>1111</u> (16), and 11111110000 through 1111111<u>1111</u> (16). So we have:



b) Assume the two-address instructions use bit patterns 000 xxxx xxxx through 101 xxxx xxxx. Assume also that the zero-address instructions are of the format 11111101000 through 11111101111 (8), 11111110000 through 11111110111 (8), and 11111111000 through 11111111111 (8) (These constitute the last 16 binary numbers possible with 11 bits). Then all instructions beginning with 110 (1100000 xxxx through 1101111 xxxx) could be one address instructions (16). In addition, 1110000 xxxx through 1111101 xxxx could be one address instructions, giving us 14 more, for a total of 30 1-address instructions.

---

18. Suppose a computer has an instruction format with space for an opcode and either three register values or one register value and an address. What are the various instruction formats that could be used for an ADD instruction on this machine?

Ans.

Format 1: ADD R1, R2, R3 where R1 is the destination and R2 and R3 are sources. We could also use R3 as the destination and R1 and R2 as sources
Format 2: ADD R1, Addr where Addr holds the contents of memory we wish to add to the value in R1

---

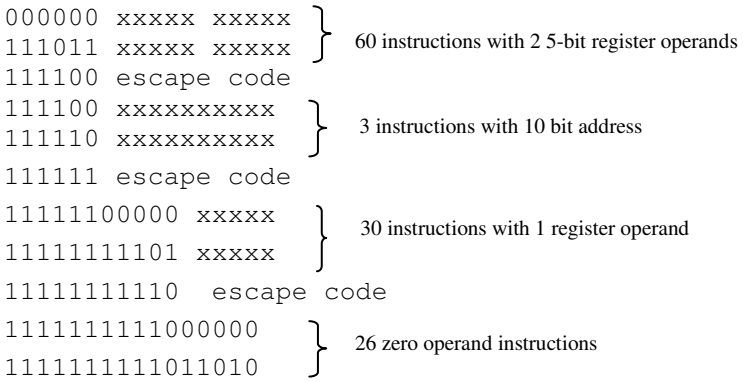
19. Given 16-bit instructions, is it possible to use expanding opcodes to allow the following to be encoded assuming we have a total of 32 registers? If so, show the encoding. If not, explain why it is not possible.

60 instructions with 2 register operands
30 instructions with 1 register operand
3 instructions with one 10-bit address

26 instructions with zero operands

Ans.

Yes.



---

20. What is the difference between using direct and indirect addressing? Give an example.

Ans.

Direct addressing provides the actual memory address of the operand in the instruction, whereas indirect addressing provides, as part of the instruction, a pointer to a memory location. For example, the instruction Load X interpreted using direct addressing would go to memory location X and load the value found there. Using indirect addressing, memory location X would be used as the effective address of what should actually be loaded. So if a value of 200 were found at location X, the value *located* at address 200 would be loaded.

♦21. Suppose we have the instruction Load 1000. Given memory and register R1 contain the values below:

Memory

0x1000	0x1400
…	
0x1100	0x400
…	
0x1200	0x1000
…	
0x1300	0x1100
…	
0x1400	0x1300

R1

0x200

and assuming R1 is implied in the indexed addressing mode, determine the actual value loaded into the accumulator and fill in the table below:

Mode	Value Loaded into AC
Immediate	
Direct	
Indirect	
Indexed	

Ans.

Mode	Value
Immediate	1000
Direct	1400
Indirect	1300
Indexed	1000

---

22. Suppose we have the instruction Load 500. Given memory and register R1 contain the values below:

0x100	0x600
…	
0x400	0x300
…	
0x500	0x100
…	
0x600	0x500
…	
0x700	0x800

R1: 0x200

and assuming R1 is implied in the indexed addressing mode, determine the actual value loaded into the accumulator and fill in the table below:

Mode	Value Loaded into AC
Immediate	
Direct	
Indirect	
Indexed	

Ans.

Mode	Value
Immediate	500
Direct	100
Indirect	600
Indexed	800

---

23. A nonpipelined system takes 200ns to process a task. The same task can be processed in a 5-segment pipeline with a clock cycle of 40ns. Determine the speedup ratio of the pipeline for 200 tasks. What is the maximum speedup that could be achieved with the pipeline unit over the nonpipelined unit?

Ans.

SpeedUp = (200ns x 200)/((5+200-1)(40ns)) = 40000/8160 = 4.9019

Max SpeedUp = 5

---

24. A nonpipelined system takes 100ns to process a task. The same task can be processed in a 5-segment pipeline with a clock cycle of 20ns. Determine the speedup ratio of the pipeline

for 100 tasks. What is the theoretical speedup that could be achieved with the pipeline system over a nonpipelined system?

Ans.

SpeedUp = (100ns x 100)/((5+100-1)(20ns)) = 10000/2080 = 4.8

Max SpeedUp = 5

---

25. Assuming the same stages as in Example 5.11, explain the potential pipeline hazards (if any) in each of the following code segments.
    a) X = R2 + Y
       R4 = R2 + X

    b) R1 = R2 + X
       X = R3 + Y
       Z = R1 + X

Ans.

a) The problem is a resource conflict at time 4, as both instructions need to access memory.

Time Period →	1	2	3	4	5
X=R2+Y	fetch instruction	decode	fetch Y	Add & store in X	
R4=R2+X		fetch instruction	decode	fetch X*	

b) The problem is a resource conflict at time 3 and a data dependency at time 5.

Time Period →	1	2	3	4	5	6
R1 = R2 + X	fetch inst	decode	fetch X	Add & store in R1		
X = R3 + Y		fetch instr	decode	fetch Y	Add and store in X	
Z = R1 + X			fetch instr*	decode	fetch X*	

---

26. Write code to implement the expression: A= (B + C) * (D + E) on 3-, 2-, 1- and 0-address machines. In accordance with programming language practice, computing the expression should not change the values of its operands.

Ans.

Add R1, B, C
Add R2, D, E
Mult A, R1, R2

3-address machine

Load R1, B
Add R1, C
Load R2, E
Add R2, E
Mult R2, R1
Store A, R2

2-address machine

Load B
Add C
Store Temp
Load D
Add E
Mult Temp
Store A

1-address machine

Push B
Push C
Add
Push D
Push E
Add
Mult
Store A

0-address machine

♦27. A digital computer has a memory unit with 24 bits per word. The instruction set consists of 150 different operations. All instructions have an operation code part (opcode) and an address part (allowing for only one address). Each instruction is stored in one word of memory.

a) How many bits are needed for the opcode?
b) How many bits are left for the address part of the instruction?
c) What is the maximum allowable size for memory?
d) What is the largest unsigned binary number that can be accommodated in one word of memory?

Ans.

a) 150 instructions implies 2^8 (2^7 will only give us 128 instructions), or 8 bits for the opcode.
b) 24 - 8 = 16
c) 2^{16} or 32M
d) 24 1's or 2^{24} -1

---

28. The memory unit of a computer has 256K words of 32 bits each. The computer has an instruction format with 4 fields: an opcode field; a mode field to specify 1 of 7 addressing modes; a register address field to specify one of 60 registers; and a memory address field. Assume an instruction is 32 bits long. Answer the following:

a) How large must the mode field be?
b) How large must the register field be?
c) How large must the address field be?
d) How large is the opcode field?

Ans.

a) We need to identify 1 of 7 items, so there must be 3 bits ($2^3 = 8$)
b) 60 registers implies 6 bits ($2^6 = 64$)
c) $256K = 2^8 2^{10} = 2^{18}$, or 18 bits
d) 32 - (3 + 6 + 18) = 5 bits

---

29. Suppose an instruction takes four cycles to execute in a nonpipelined CPU: one cycle to fetch the instruction, one cycle to decode the instruction, one cycle to perform the ALU operation, and one cycle to store the result. In a CPU with a 4-stage pipeline, that instruction still takes four cycles to execute, so how can we say the pipeline speeds up the execution of the program?

Ans.

For one instruction, there is no speedup. The speedup comes with the *parallel* execution of multiple instructions. While the first instruction is decoding, the second can be fetched;

while the first instruction is performing the ALU instruction, the second can be decoding, and the third can be fetched, etc.

---

30. Pick an architecture (other than those covered in this chapter). Do research to find out how your architecture approaches the concepts introduced in this chapter, as was done for Intel, MIPS, and Java.

Ans.

No answer given.

True or False

1. Most computers typically fall into one of three types of CPU organization: (1) general register organization; (2) single accumulator organization; or (3) stack organization.

2. The advantage of zero-address instruction computers is that they have short programs; the disadvantage is that the instructions require many bits, making them very long.

3. An instruction takes less time to execute on a processor using an instruction pipeline than on a processor without an instruction pipeline.

4. The term “endian” refers to an architecture’s byte ordering.

5. Stack architectures have good code density and a simple model for evaluation of expressions, but do not allow random access, which can cause a problem with the generation of efficient code.

6. Most architectures today are accumulator-based.

7. Fixed-length instruction format typically results in better performance than variable length instruction format.

8. Expanding opcodes make instruction decoding much easier than when it is not used.

9. Instruction set orthogonality refers to the characteristic in an instruction set architecture where each instruction has a “backup” instruction that performs the same operation.

10. The effective address of an operand is the value of its actual address in memory.

11. Resource conflicts occur in a pipeline when there are multiple instructions that require the same resource.

12. Data dependencies occur in a pipeline when multiple instructions need the CPU.

Ans.

1. True 2. False 3. False 4. True 5. True 6. False 7. True
8. False 9. False 10. True 11. True 12. False
