

Fill in the Blanks and Short Answer

1. Define a structure named **ThreeInts** containing three unsigned doublewords named **field1**, **field2**, and **field3**. The fields should be uninitialized.
2. Using the **ThreeInts** structure from the previous question, declare a variable named **first** of type **ThreeInts**:
3. Using the **ThreeInts** structure from the previous question, write a statement that assigns the value 10h to **field1**:
4. Using the **ThreeInts** structure from the previous question, write a statement that creates a variable named **pThree** that contains the offset of **ThreeInts**.

Short Programming Problems

Example 1

```
POINT STRUCT
    X WORD ?
    Y WORD ?
POINT ENDS

POLY STRUCT
    points POINT 10 DUP(<>)
    numPoints WORD ?
POLY ENDS
```

5. Using Example 1, declare a variable named **aPoint** of type POINT that initializes X to 10 and Y to 20:
6. Using Example 1, declare a variable named **aPoly** of type POLY that uses default structure values.
7. Using Example 1, set the first element of the **aPoly.points** array to X=1, Y=2.
8. Using Example 1, set the second element of the **aPoly.points** array to X=3, Y=4.
9. Using Example 1, write a sequence of statements that initialize all X values in the **aPoly.points** array to random integers between 0 and 99. Use an indexed operand to access the **points** array.
10. Write a sequence of statements that first set ESI to the offset of **aPoly**. Next, use ESI as an indirect operand to set the **numPoints** field to 2.
11. Declare an array of two POLY structures named **myArray**. Use default initialization for both.

Example 2

```
RECT STRUCT
    Left    WORD ?
    Top     WORD ?
    Right   WORD ?
    Bottom  WORD ?
RECT ENDS

AnyData UNION
    mTitle BYTE 10 DUP(?)
    mRect  RECT  <>
```

Chapter 10

```
AnyData ENDS
```

12. Using Example 2, declare a variable named **myData** of type **AnyData**.
13. Using Example 2, move the letter "A" to the first byte of the **mTitle** field.
14. Using Example 2, move 10 to the **Left** field within the **mRect** field of **myData**.
15. Using Example 2, use the **SIZEOF** operator to set **ECX** to the size of the **mTitle** field in the **AnyData** union:

Example 3

```
POINT STRUCT
    X WORD ?
    Y WORD ?
POINT ENDS

Rectangle STRUCT
    upperLeft POINT <>
    lowerRight POINT <>
Rectangle ENDS
```

16. Using Example 3, declare a **Rectangle** variable named **myRect**. In the same declaration, initialize its **upperLeft** field to (10,20) and its **lowerRight** field to (50,60).
17. Using Example 3, declare an array of three Rectangles named **myArray** and initialize the rectangles to the same coordinates as the variable in the previous question.
18. Using indirect addressing with **ESI**, assign a value of 80 to the X-coordinate of the upper left corner of the third rectangle in **myArray** (from the previous question).

(The following questions are not related to Example 3.)

19. Create a macro named **putChar** that writes a single character, passed as a macro argument, to standard output. Be sure to push and pop any registers modified by the macro.
20. Create a macro named **mMove16** that moves any 16-bit memory operand to any other 16-bit memory operand. Save and restore any registers modified by the macro. Syntax: **mMove destination,source**
21. Create a macro named **mMult16** that multiplies any two signed 16-bit memory operands and produces a 16-bit product. Syntax: **mMult16 product,op1,op2**
22. Create a macro named **mWriteHex** that writes the value of a 32-bit memory operand to standard output in hexadecimal format.
23. Using the **mWriteHex** macro from the previous question, write a statement that invokes the macro, using **ESI** as an indirect operand.
24. Create a macro named **mWriteHexConst** that lets the caller pass an integer constant argument. The integer is written to standard output in hexadecimal format. All labels must be local.
25. Create a macro named **LongLoop** that is not constrained by the **LOOP** instruction's limited -128 to +127 byte range. All labels must be local.
26. Write a macro named **mShowValue** that writes a literal string to standard output, followed by the contents of an integer variable in unsigned decimal. Push and pop any registers modified by the macro. All labels must be local. Following is a sample call:

```
.data
```

Chapter 10

```
salary DWORD 52100
.code
mShowValue "Salary is equal to: ", salary
```

27. Write a sequence of statements that exits a macro if the **count** argument is blank.
28. Write a sequence of statements that displays an error message on the console during assembly if the first macro argument named **count** is identical to "ECX". Make the comparison case-insensitive.
29. Write a macro named **myMac** that has a single parameter named **first**, which is given a default argument initializer of 0. The macro should move **first** to the EAX register.
30. The **mWrite** macro shown in Chapter 10 displays a string literal on the console. A sample call is:

```
mWrite "Hello there"
```

Show how to call **mWrite** and pass it a string that contains both ordinary characters and carriage-return/linefeed bytes (0Dh, 0Ah).

31. Use the REPEAT directive to define an array named **myArray** that contains 50 16-bit unsigned words, with initial values based on the following sequence: {5,7,9,...,103}
32. Use the FOR directive to create five uninitialized DWORD variables in which each variable name is a member of the following list: monday,tuesday,wednesday,thursday,friday.
33. Write down the contents of the first five rows (in hexadecimal) of the table generated by the following statements:

```
count = 0
FORC digit,<0123456789ABCDEF>
    BYTE count,"&digit"
    count = count + 1
ENDM
```

Multiple-Choice

(Each of the following questions has only one correct answer.)

34. What data definitions will be created by the following directives?

```
FOR colorVal,<red,green,blue,turquoise,purple,orange>
    BYTE "&colorVal",0
ENDM
```

 - a. six strings, each equal to "&colorVal", followed by a null byte.
 - b. six null-terminated strings, each containing the name of a different color
 - c. six variables, each having a label whose name is a color
 - d. this statement will cause a syntax error
35. Which of the following statements permits assembly if the symbol **Win32** has been defined?
 - a. IF Win32 == 1
 - b. IF DEFINED Win32
 - c. IFDEF Win32
 - d. IFNDEF Win32

Chapter 10

36. Which of the following statements permits assembly if **arg1** is exactly the same as **arg2**? (Assume that a case-sensitive comparison is used.)
- a. IFEQUAL <arg1>,<arg2>
 - b. IFIDN <arg1>,<arg2>
 - c. IF arg1 EQ arg2
 - d. IFEQ arg1,arg2