* Packed Arrays

***One Dimensional***

module tb;  
        bit [7:0]         m\_data;         // A vector or 1D packed array

initial begin  
                // 1. Assign a value to the vector  
                m\_data = 8'hA2;

// 2. Iterate through each bit of the vector and print value  
                for (int i = 0; i < $size(m\_data); i++) begin  
                        $  
                end  
        end  
endmodule

***Multi-dimensional***

module tb;  
  bit [2:0][3:0][7:0]         m\_data;         // An MDA, 12 bytes

  initial begin  
        // 1. Assign a value to the MDA  
      m\_data[0] = 32'hface\_cafe;  
      m\_data[1] = 32'h1234\_5678;  
      m\_data[2] = 32'hc0de\_fade;

      // m\_data gets a packed value  
      $display ("m\_data = 0x%0h", m\_data);

      // 2. Iterate through each segment of the MDA and print value  
      foreach (m\_data[i]) begin  
        $display ("m\_data[%0d] = 0x%0h", i, m\_data[i]);  
        foreach (m\_data[i][j]) begin  
          $display ("m\_data[%0d][%0d] = 0x%0h", i, j, m\_data[i][j]);  
        end  
      end  
   end  
endmodule

* Unpacked Arrays

***One Dimensional***

module tb;  
   byte stack [8];         // depth = 8, 1 byte wide variable

  initial begin

  // Assign random values to each slot of the stack  
        foreach (stack[i]) begin  
          stack[i] = $random;  
          $display ("Assign 0x%0h to index %0d", stack[i], i);  
        end

  // Print contents of the stack  
    $display ("stack = %p", stack);  
  end  
Endmodule

***2 Dimensional***

module tb;  
  byte stack [2][4];         // 2 rows, 4 cols

initial begin  
                // Assign random values to each slot of the stack  
                foreach (stack[i])  
          foreach (stack[i][j]) begin  
            stack[i][j] = $random;  
            $display ("stack[%0d][%0d] = 0x%0h", i, j, stack[i][j]);  
                        end

// Print contents of the stack  
                $display ("stack = %p", stack);  
        end  
endmodule

* Packed + Unpacked Arrays

module tb;  
  bit [3:0][7:0]         stack [2][4]; // 2 rows, 4 cols

  initial begin  
    // Assign random values to each slot of the stack  
    foreach (stack[i])  
      foreach (stack[i][j]) begin  
        stack[i][j] = $random;  
        $display ("stack[%0d][%0d] = 0x%0h", i, j, stack[i][j]);

      end

     // Print contents of the stack  
     $display ("stack = %p", stack);

     // Print content of a given index  
     $display("stack[0][0][2] = 0x%0h", stack[0][0][2]);  
   end  
endmodule

* Unpacked Structure with typedef

module tb;  
  // Create a structure called "st\_fruit"  
  // which to store the fruit's name, count and expiry date in days.  
  // Note: this structure declaration can also be placed outside the module  
  typedef struct {  
    string fruit;  
              int    count;  
          byte          expiry;  
   } st\_fruit;

    initial begin  
      // st\_fruit is a data type, so we need to declare a variable of this data type  
      st\_fruit fruit1 = '{"apple", 4, 15};  
      st\_fruit fruit2;

      // Display the structure variable  
      $display ("fruit1 = %p fruit2 = %p", fruit1, fruit2);

      // Assign one structure variable to another and print  
      // Note that contents of this variable is copied into the other  
           fruit2 = fruit1;  
      $display ("fruit1 = %p fruit2 = %p", fruit1, fruit2);

      // Change fruit1 to see if fruit2 is affected  
      fruit1.fruit = "orange";  
      $display ("fruit1 = %p fruit2 = %p", fruit1, fruit2);  
    end  
endmodule

* Packed Structure

module tb;  
  // Create a structure called "st\_fruit"  
  // which to store the fruit's name, count and expiry date in days.  
  // Note: this structure declaration can also be placed outside the module  
  typedef struct {  
    string fruit;  
    int    count;  
    byte          expiry;  
   } st\_fruit;

  initial begin  
    // st\_fruit is a data type, so we need to declare a variable of this data type  
    st\_fruit fruit1 = '{"apple", 4, 15};  
    st\_fruit fruit2;

        // Display the structure variable  
    $display ("fruit1 = %p fruit2 = %p", fruit1, fruit2);

    // Assign one structure variable to another and print

    // Note that contents of this variable is copied into the other  
    fruit2 = fruit1;  
    $display ("fruit1 = %p fruit2 = %p", fruit1, fruit2);

    // Change fruit1 to see if fruit2 is affected  
    fruit1.fruit = "orange";  
    $display ("fruit1 = %p fruit2 = %p", fruit1, fruit2);  
  end  
Endmodule