State Machines – Implementation

• A Jump Table is a possible implementation not mentioned in the book
• Traditionally, we have been told that a switch-case statement is implemented by the compiler with a “hidden” jump table for time efficiency
• That may or may not be what the compiler does
• However, we can implement jump tables directly and know that we are getting the time efficiency
State Machines - Implementation

• Jump Tables are a time and memory tradeoff
• If you have enough memory to build the table you can get fast access to the state code
• If you are too memory constrained, you may need to implement a binary search of the state values to get faster access to the state code than a linear “if, else if, ... else” implementation
• First level “if else” checks the middle state value and subdivides the search to one of two halves of the range of state values, etc.
State Machines - Implementation

• A Jump Table can be implemented using an array of function pointers
  – The array index is the case value for interpreting state
  – The contents at that index is the function to execute

• The state machine controller
  – Uses the state value to calculate the index into the table (A one to one mapping is commonly used)
  – Invokes the function in that entry with the required arguments
State Machines - Implementation

• A switch – case statement has a “fixed” mapping from each constant case value to the state code
• If you implement a jump table yourself, you can do a “late binding” from case value to state code
• The contents of the array can be written at any time during the execution of the code, so you can “override” the previous code for any state by writing a new function pointer into the array
• This may be useful for SW upgrades or bug fixes
State Machines - Implementation

• If you want to run the same state machine on multiple instances of the object that it models
  – Define a struct to contain one instance of the data that must be maintained by the state machine
  – Declare a pointer to this struct in the argument list for each function in the jump table
  – Have the state machine controller (the switch part of the jump table) pass the address of the struct for the appropriate instance to each function along with any other required input arguments
  – Have each function use/update the struct’s data