1. **Output:** “The value of i is: 7”
   Looking at the MIPS code, we can see that the instructions in “main:” are setting up for the loop section. We begin by initializing the registers $s3, $s4, $s5, and $s6 to 0, 1, 0, and starting address of array “save”, respectively in that order. We then continue into the loop where we are traversing through the elements of save and incrementing s3 for every loop. The condition for this loop is once an element of save does not equal 0, we will exit this loop. If we look at the array save, we notice that the first non-zero element is at index 6 (start at 0). This means that we must traverse the array 7 times thus making $s3 equal 7. In the Exit: region, we load the address of message1 in $a0 (argument register). We then perform a syscall to print out the contents in $a0, “\n The value of i is: “. We repeat this except we store the value of $s3 inside $a0. As a result, ‘7’ is printed which yields the output, “**The value of i is: 7**”.

2. **Answer:** e
   The order **‘jyix**’ will produce the worst/slowest execution time. This is because we are exploiting the least amount of spatial locality when accessing memory in this order. Spatial locality is present when we perform row-order access. In this code, the potential variables that can be utilized to perform successive reference to the L1 cache are y and j. Having the variable ‘y’ in the inner loop would result in the best results because both arrays c and b rely on ‘y’ to reference successive elements in the array. At the same time, having ‘j’ in the inner loop would also exploit some spatial locality because the array ‘a’ relies on ‘j’ to perform row order access. Knowing this, four of the five potential answer choices are now eliminated leaving us with our answer and slowest execution time order, ‘**jyix**’.