```
int binarySearch(int array[], int size, int key) {

  int first = 0, last = size-1;
  while (first <= last) {
     int mid = (first + last) / 2;  // find the middle element.
     if (key > array[mid])  // not in the lower half
        first = mid + 1;
     else if (key < array[mid]) // not in the upper half
        last = mid - 1;
     else
        return mid;   // search succeeds
  }
  return -1;    // search fails
}
```

Language help: to divide the value in ebp by two and save the result in ebp, you can use the instruction "shr ebp, 1"

Example 1:
Given a search key **12** and a sorted array as follows:

Array Elements	2	6	12	17	24	29	53	64	80	91
Index	0	1	2	3	4	5	6	7	8	9

See how first, last, and array[mid] are changing during the execution (note ***mid*** values are decided using ***first*** and ***last*** in the previous rows):

	First	last	mid	array[mid]
Before the while loop	0	9		
At the end of the 1st iteration	0	3	4	24 (>12)
At the end of the 2nd iteration	2	3	1	6 (<12)
At the end of the 3rd iteration			2	**12** (==12)
Search succeeded and 2 is returned				

Example 2:
Given a different search key **85** and the same array:

See how first, last, and array[mid] are changing during the execution (note ***mid*** values are decided using ***first*** and ***last*** in the previous rows):

	first	last	mid	array[mid]
Before the while loop	0	9		
After the 1st iteration	5	9	4	24 (<85)
After the 2nd iteration	8	9	7	64 (<85)
After the 3rd iteration	9	9	8	80 (<85)
After the 4th iteration	**10**	**9**	9	91 (>85)
first>last, → while loop ends → search failed → -1 is returned				