Bryan Nonni

CSC 3320

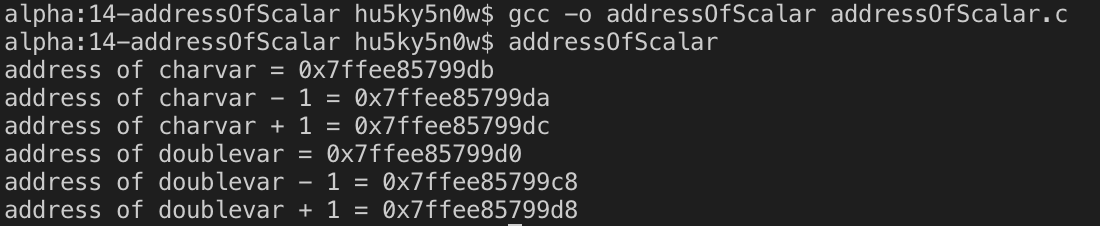
Lab 8 – PostLab

Due at 11:59 pm on Monday, Apr. 15, 2019

**Part 1:**



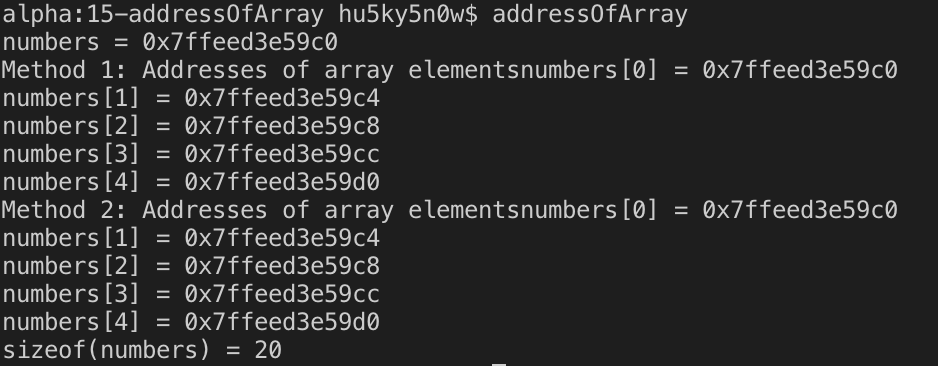
**Part 2:**

**1)** 

2)   
#include <stdio.h>  
int main(){  
// intialize a char variable, print its address and the next address  
char charvar = 'a';  
printf("address of charvar = %p\n", (void \*)(&charvar));  
printf("address of charvar - 1 = %p\n", (void \*)(&charvar - 1));  
printf("address of charvar + 1 = %p\n", (void \*)(&charvar + 1));  
// intialize an int variable, print its address and the next address  
double doublevar = 1.0;  
printf("address of doublevar = %p\n", (void \*)(&doublevar));  
printf("address of doublevar - 1 = %p\n", (void \*)(&doublevar - 1));  
printf("address of doublevar + 1 = %p\n", (void \*)(&doublevar + 1));  
return 0;  
}

3. Because a double is allocated 8 bytes of memory when initialized.

**Part 3:**

1) 

2) They’re the same because when a fixed array is used, the fixed array will be implicitly converted into a pointer that points to the first element of the array.

3) They’re the same because in both cases, “numbers” points to the same memory address, so either way of incrementing will increase the position of the numbers memory address.

4) printf("lengthof (numbers) = %lu\n", sizeof(numbers)/sizeof(numbers[0]));

5) No, because when an array if passed as an argument of a function call, it becomes a pointer to the array. Because of this fact, we have to pass the length of the array as an additional argument in the function call.