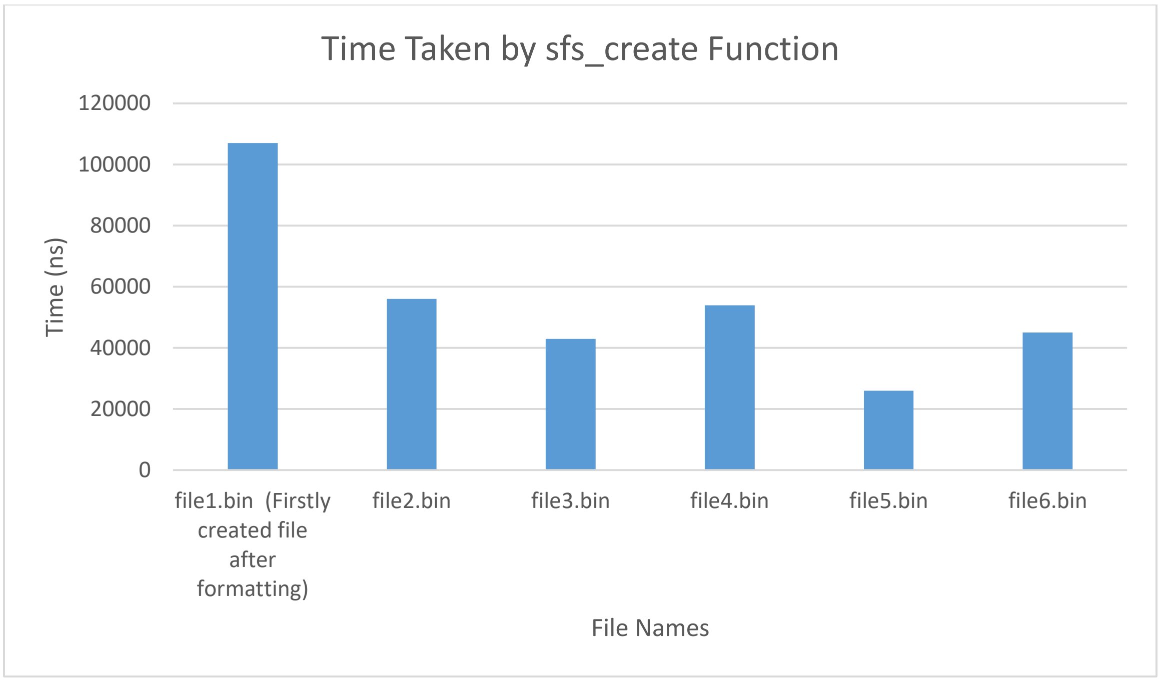
CS342

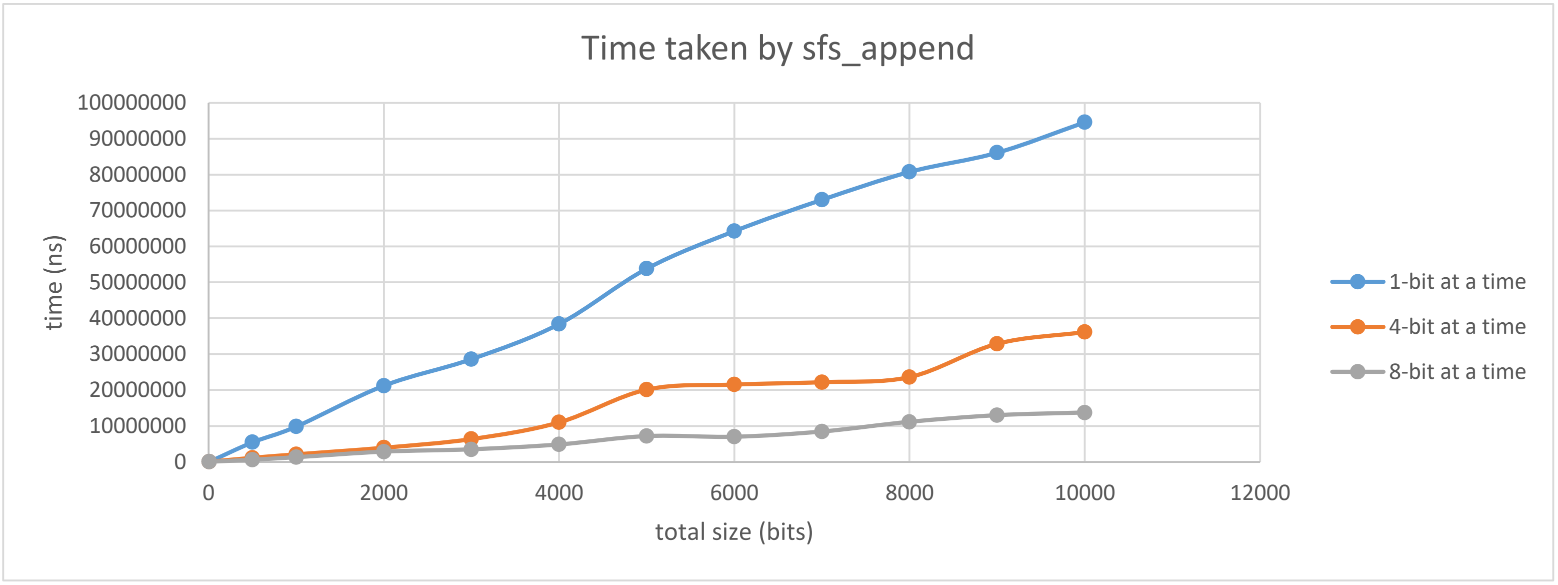
Project 04

Radman Lotfiazar - 21600450

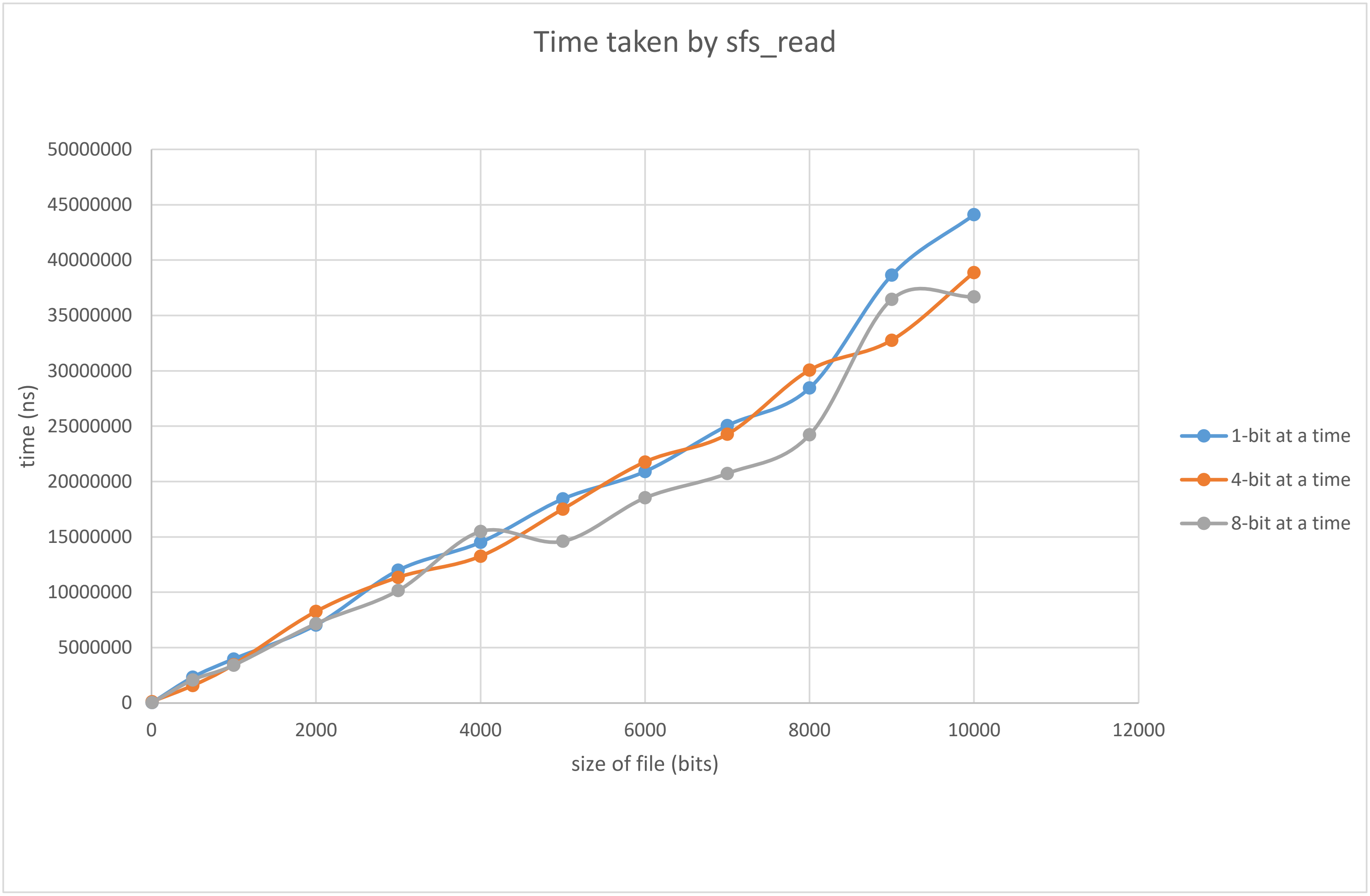
Turan Mert Duran - 21601418



This table indicates the creating a file time. As you can see the first time which application has at first formatting the virtual disc and after that starts to creating file takes the highest time compare to other files which is created after that. For creating file from 2 to 6 since virtual disc is already formatted takes less time and almost they are in the same time. The difference in the time for creating file from 2 to 6 is because of system and CPU state which leads to takes a little bit less or more time. Therefore, we can conclude that creating first file takes the highest time among other creating files which is happening after that because application have to format the virtual disk and after that creating the file. However, for other creating file after the first time takes less time and almost they are same.



This graph indicates the time for appending. Appending occurs in different sizes from 0 to 10000 bits. Furthermore, in sfs_append function we are able to indicates the size of bit which will append. Therefore, done our experiment in three different sizes from 1 to 8 bits. As you can see if we want to append 1 bit each time it takes the highest amount of time because we have to append to file equal to the size. for instance, if we want to have a file with size of 1000bits we have to append a file 1000 time because each time we are able to append 1bits. However, for 4bits and 8bits at a time it takes less time because we append to file more bits each time. Furthermore, you are able to see that 8-bit at a time has the best performance and takes the less time. However, the problem for 8 and 4 bits at a time is you have to choose the size which are multiply by 8 and 4 respectively. Therefore, it is better to use append with bigger bits at a time because you access append function less than lower bits at a time.



This graph indicates the time take by reading file with different size. in our experiment we read the files with different size with different bits at a time. As you are able see the number of bits which sfs_function is reading a file at a time is not important and it takes the same time. The difference between 1, 4 and 8 bits at a time in different points of graph is because of system and CPU state which leads to difference in time slightly. Furthermore, if you compare the time which is taken by sfs_append and sfs_read you will find out for 1 bit at a time append function takes higher time. Append takes more time because in that function we have to check the next block for availability. However, in read function we are not check availability of any block. Therefore, for 1 bit at a time the time which is taking by append is considerably less than read.

In the Superblock we are storing the number of blocks, size of virtual disk and block size. Superblock is initialized in the create_format_vdisk function. In the initialization function of Superblock we are getting size of virtual disk and initialize each variables. In for Bitmap blocks we are creating an integer array (bitMap) with 1024 index. Since each integer is 4 bytes, we have 4096 bytes for each Bitmap blocks which is the same as our fix size for each blocks. Our each bit block represents 4096 x 8 = 32768 blocks totally. We acted our int bitmap array as bitArray. Implemented its' bit changing methods. We have totally control on 4 x 32768 = 131072 blocks. As the assignment indicates each bit of Bitmap indicates a block. Therefore, in each of integer index of array there is an integer with 8 digits. Which each digit represents one bit of Bitmap. Bitmap the same Superblock is initialized in the create_format_vdisk function. In initBitMap function we are initializing each of integer index based on the number of blocks in the virtual disk. Furthermore, bits 1 to 13 will be initialized 1 which means blocks 1 to 13 are filled by file system structure blocks. For controlling the bit of Bitmap there are 3 different functions setBit, clearBit and readBit. All of them getting integer index and a integer array as parameter. If the index which is stated in the parameter was empty (0) setBit will make it 1 and if it was filled (1) clreaBit will make it empty. There are two different types of file system blocks Directory Block and FCB block. All arrays in fcbBlock and dirBlock structs are 32 because based on the assignment each block takes 32 entries of each type (FCB or Directory). Hence, each index of arrays indicates one entry. In each directory block there are 32 directory entry which each directory entry has 2 variable file name and index of FCB. For the name of file, we decide to there is a 2 dimension character array. There are 32-character array which each of them takes 110 characters. And there is a integer array with size of 32 for each directory entry. By this implementation we are able to indicate 32 files which have a file name with maximum 110 characters and store an index node of FCB in one blocks. By initiDirBlocks function in create_format_vdisk function we are assigning directory blocks into the specified locations (blocks from 5 to 8). For FCB blocks we have a Boolean and 3 different integer arrays which all of them has size of 32. Each index of the arrays indicates one entry of FCB. Therefore, we are able to indicate 32 entry in each blocks. There is a initFcbBlock function in create_format_vdisk function which is assigning blocks from 9 to 12 to the FCBs. For initializing the index of arrays at the create virtual disk we decied to be -1 in order to finding out easily.