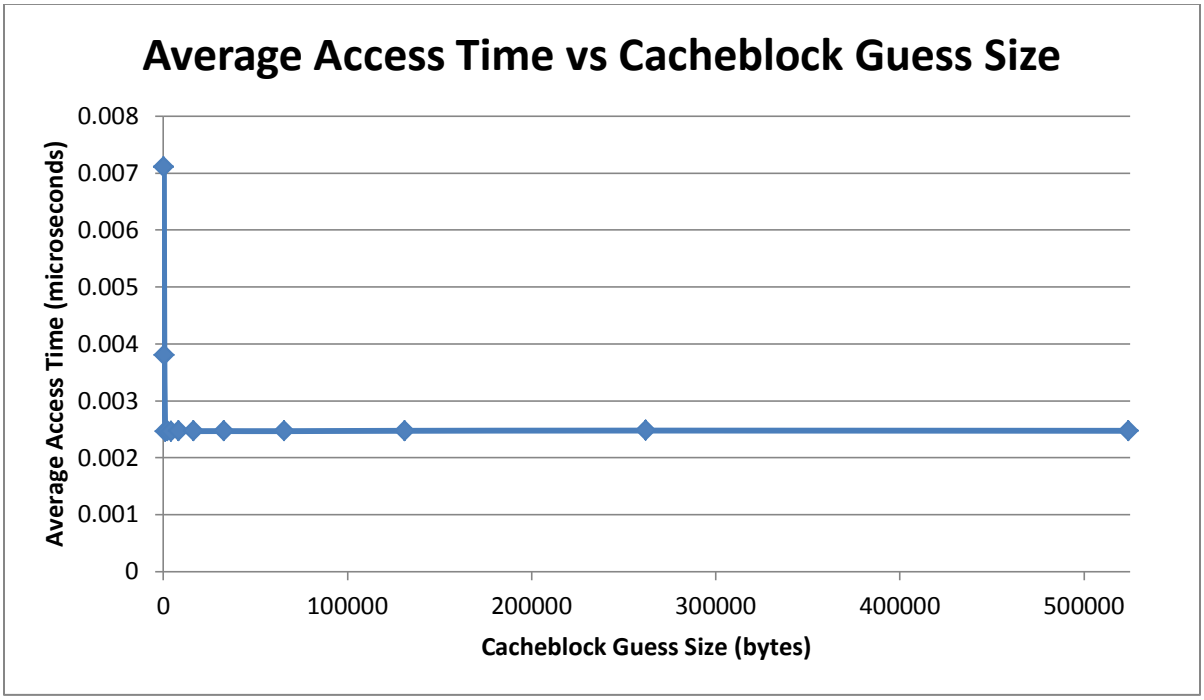
cacheblock and cachesize Readme
By: John Cierpial

The cacheblock.c program is designed to measure the cache block size of a system's processor. First, the following integers are created in the register: a (as described by the programming assignment instructions), ARRAY_SIZE representing the size of the array that the program will read from (the value of 1024 was chosen as each integer would be 4 bytes long, making the array sized 4KB), numIterations (chosen to be an arbitrarily large number so there would be little to no interference in measurement times), SOME_ACCESS_SIZE (1048576 bytes or 1MB) which represents the first "guess" at the cache block size. The integer variables dummy, i, and times were created for purposes of array access and loop control variables as defined in the assignment details. timeTaken1 and timeTaken2 were designated as float variables to give more precise and accurate representation of time (the values of 1000 for both are a number chosen at random and have no bearing on the outcome of the program). Next, a is allocated an array of around 4 million integers as dictated by the instructions. Next, the program enters a while loop based on comparison of present and previous access times. Many test runs indicate that the "jump" from a constant average time to a non-constant average time is 1.5%. Therefore, 1.6 was chosen to provide some leeway in calculation error. The program then sets timeTaken1 (or previous time) to be equal to the current run (timeTaken2). The access size (or cache block size guess) is then halved (1MB to 512KB on the first run) and a timer is started to measure the amount of time it takes to access an array numIterations times. After that, the timer is stopped. timeTaken2 is calculated in microseconds and divided by numIterations to get the average time taken for one access of the array. The following line then prints the cache block guessed size and the average time taken for that guess size. This loop continues until there is greater than a 1.6% difference between the two time values (timeTaken2 and timeTaken1).

The cachesize.c program is designed to measure the cache size of a system's processor. The structure and logic behind this program is very similar to cacheblock.c. The main difference is that an integer memSize is introduced to represent the 16MB integer size used in sbrk as well as in calculations for numIterations. The cacheBlockSize variable is defined as 64 bytes as to also calculate the numIterations variable and because it is the cache alignment size for L1 cache. The ARRAY_SIZE variable is initialized to a value of 200 so that it is neither too small or too large and is of optimal size. The same reasoning is applied to the assignment of 10 for the SOME_ACCESS_SIZE variable. The while loop works on the same conditions as cacheblock.c except the breakpoint is when the time difference between timeTaken2 and timeTaken1 is greater than 2.3% as calculations and multiple executions of the program revealed that was approximately the "jump" expected when the cache size boundary was hit. Within the while loop, the ARRAY_SIZE variable is doubled each time the loop is run. This is also why the numIterations calculation ensues on the following line. numIterations is calculated to be a different value on each run of the loop so that as the ARRAY_SIZE increases, the numIterations decreases allowing for more calculations at a faster rate. It doesn't matter whether the numIterations variable remains constant because the program is finding the *average* time. As long as a reasonable size for numIterations is used each time, the average time will not be that much different whether there are 1,000,000 iterations run or 1000. Following that, the timer is started, the array is accessed, and the time is ended. timeTaken2 is measured in microseconds and divided by numIterations to get the average time. The loop then prints what the cache access size was and how long in microseconds it took on average. This loop continues until the ration of timeTaken2 to timeTaken1 is greater than 2.3%.

cacheblock and cachesize Readme
By: John Cierpial

When running cacheblock.c on the iLab machines, the following outputs were produced:
Guess: 524288 bytes: 0.002473 microseconds.
Guess: 262144 bytes: 0.002477 microseconds.
Guess: 131072 bytes: 0.002473 microseconds.
Guess: 65536 bytes: 0.002467 microseconds.
Guess: 32768 bytes: 0.002468 microseconds.
Guess: 16384 bytes: 0.002467 microseconds.
Guess: 8192 bytes: 0.002468 microseconds.
Guess: 4096 bytes: 0.002463 microseconds.
Guess: 2048 bytes: 0.002473 microseconds.
Guess: 1024 bytes: 0.002464 microseconds.
Guess: 512 bytes: 0.003806 microseconds.
Guess: 256 bytes: 0.007106 microseconds.
Plotted on a graph starting at 256 bytes and working up:



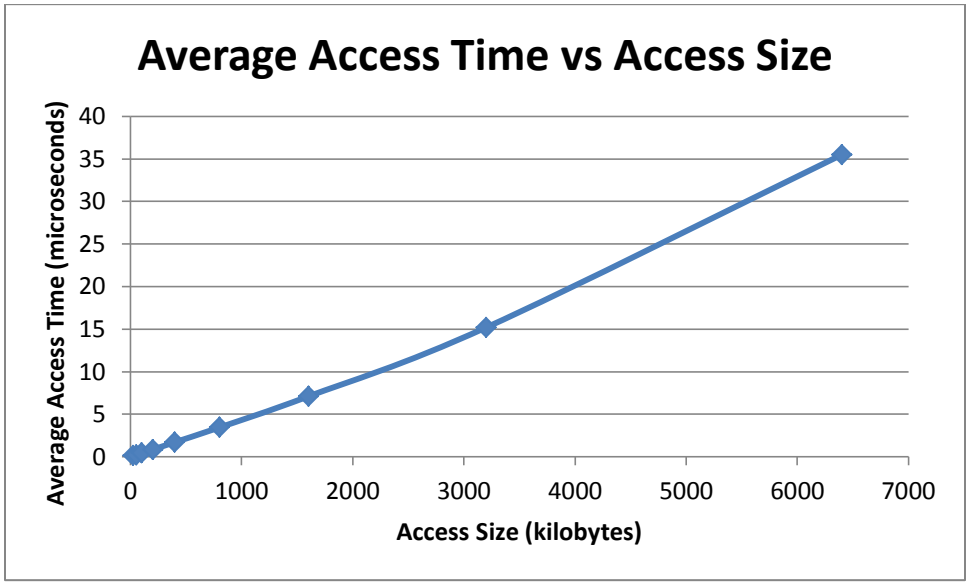
As you can see, the cacheblock guess size jumps to a very large number so it's hard to definitively see from the chart. However, the chart combined with the data above makes it clear the average time is no longer constant around 512 bytes. The cache block size for the processors on the iLab machines are 256 bytes each for L2 and L3 cache and 64 bytes for L1 cache. It appears to me that my program is measuring all 3 caches, summing to 576 bytes. With my program noting a change in average running time at 512 bytes, I would say my program to measure cache block size is reasonably accurate.

While running cachesize.c on the iLab machines, the following outputs were produced:
The access time for access size 25 kilobytes is 0.117557 microseconds
The access time for access size 50 kilobytes is 0.220183 microseconds
The access time for access size 100 kilobytes is 0.429448 microseconds

cacheblock and cachesize Readme
By: John Cierpial

The access time for access size 200 kilobytes is 0.851852 microseconds
The access time for access size 400 kilobytes is 1.725000 microseconds
The access time for access size 800 kilobytes is 3.450000 microseconds
The access time for access size 1600 kilobytes is 7.100000 microseconds
The access time for access size 3200 kilobytes is 15.200000 microseconds
The access time for access size 6400 kilobytes is 35.500000 microseconds

Plotted on a graph:



This graph combined with the data output by the program indicates that while the average access time increases the larger the access size, there is a noticeable jump at 6400 kilobytes. This indicates that the cache size of the processor is approximately 6400 kilobytes. This is confirmed when running cat /proc/cpuinfo on the iLab machines. The cache size printed there is 6144 kilobytes, which I assume to be the L3 cache size because of its large size. Therefore, I would again say my cachesize.c program calculates a reasonable approximation of the actual processor cachesize.