Module 6 (Lectures 25-31) Cache memory

1. Compute the total size of the cache directory given the following design details: 2-way set associative, write through cache of size 64 KBytes, with block size of 16 Bytes, page size of 4 KBytes, main memory size of 1 GBytes and address size of 32 bits.
2. The hardware cache used in a given computer system has the following properties: Cache size: 16 Kbytes; Block size: 64 Bytes; Placement: 4-way set associative; Write policies: Write back, Write allocate; Replacement policy: LRU. Show how a 32 bit address will be viewed by the cache hardware in terms of offset, index and tag bits. A program running on this computer uses an array **A** declared as float A[2048], where A[0] is located at address 0x1000.
   (a) What is the cache index of array element A[1024]?
   (b) Name any other array element other than the neighbours of A[1024] that has the same cache index.
3. The hardware cache used in a given computer system has the following properties: Cache size: 4 KBytes; Block size: 16 Bytes; Placement: Direct mapping; Write policies: Write back, Write allocate. Consider the following code fragment:
   ```
   float X[1024], Y[1024];
   int i;
   for (i=0; i<1024; i++) Y[i] = X[i];
   ```
   Assume that the start addresses of the arrays X and Y are 0xA000 and 0xB000 and that the size of float data is 4 Bytes. Ignore all memory references other than those to the elements of arrays X and Y. What is the hit ratio of this program fragment if the cache starts off cold?
4. Consider the DAXPY loop shown below. Recall that the loop can be unrolled to perform the computation of more than one loop iteration each time through the newly unrolled loop. Study the effects of increasing the degree of loop unrolling. You can do this by writing a loop including the DAXPY loop and measuring the execution time of both the original loop and unrolled versions of the loop
   ```
   Loop: double a, X[16384], Y[16384];
         for (i = 0 ; i < 16384; i++)
               Y[i] = a * X[i] + Y[i] ;
   ```
5. Implementing Matrix Multiplication: Consider the problem of multiplying two 1024x1024 double float matrices. This can be done using the triply nested *for* loops that you are familiar with from a course on matrices. Implement a program for this matrix multiplication; the values of the matrix elements are unimportant, so you can initialize them to contain arbitrary values. Experiment with this program on any 2 different machines that you have access to. Try and improve your program using techniques we have discussed, such as loop interchange and blocking. Measure the execution time of your program and produce the fastest program that you can.