COMPUTER ARCHITECTURE

ASSIGNMENT-2: CACHES

Name: Umang Barbhaya

Roll No: M20CS017

Problem-1

RISC - V code for the following C-code

```
.data
a: .word 2048

.text
li a0,1024 #size=1024
li x21,2 #step=2
li x20,4 #reps=10
li x22,1 #option=1

jal store

li a0,10
ecall

store:

la x2,a
add x3,x2,a0
slli x5,x21,2

go1:
beq x22,x0,go
lb x6,0(x2)
addi x6,x6,1
sb x6,0(x2)

j go2
go:
sb x0,0(x2)
go2:
add x2,x2,x5
blt x2,x3,go1
addi x20,x20,-1
bgtz x20, store
jr ra
```

Answer the following common questions first:

- **How big is your cache block?**
    - 8-bytes is the Cache block size
- **How many consecutive accesses (taking into account the step size) fit within a single block?**
    - 2
      a[j] being fetched for 1st time will result in cache miss, but there is an increment operator, at this time there will be a cache hit.
      During 2nd step, single block can't store the consecutive memory. So it can access only the second element and have to flush the first element
      Therefore, alternative access at steps of 2 will be there.
- **How much data fits in the WHOLE cache?**
    - 4*8 = 32 bytes of data
- **How far apart in memory are blocks that map to the same set (and could create conflicts)?**
- **What is your cache's associativity?**
    - Cache associativity is 1 because of Direct Mapping
- **Where in the cache does a particular block map to?**
    - Since it the direct Mapping, particular block will map to fixed or definite cache line.

Setting-1

Answer the following

- **When considering why a specific access is a miss or hit: Have you accessed this piece of data before? If so, is it still in the cache or not?**
    - If no data available in cache a miss will occur and when task gets complete blocks from cache are flushed out. Cache becomes full then LRU scheme is used. After the above setting 1st access leads to a miss and after that the rest 3 block out of the total 4 access. The entire array will be stored in cache after the 1st repetition.
- **What combination of parameters is producing the hit rate you observe? Write your answer in the form "[parameter A], [parameter B]" where the two parameters complete the following response: "Because [parameter A] in bytes is exactly equal to [parameter B] in bytes." Note: Don't forget that 'cache size' is a valid parameter that you implicitly set by choosing the block size and the # of blocks.**
    - The [cache size] and [step size] are equal which is 32 bytes. This combination leads to 0 hit rate as indexes of block becomes identical and gets overwritten. [Block Size], [Step Size], [No. of Block], [Total Memory access]
- **What is our hit rate if we increase Rep Count arbitrarily? Write your answer as a decimal (e.g. "1.0" if the HR is 100%).**
    - Hit rate is 0. Because of 8 bytes of the step size, how many rep counts we increase the hit rate is always going to be 0. Only the tag will be changed and same block will be accessed by CPU always.
- **How could we modify one program parameter get the highest possible hit rate? Write your answer in the form "[parameter], [value]" where [parameter] is the program parameter you want to change and [value] is the value you want to change it to. Note: We don't care if we access the same array elements. Just give us a program parameter**

modification that would increase the hit rate. However, do make sure that your proposed value is valid.
  - Changing step size to 1 will increase the Hit rate with 50% or 0.5

Setting-2

Answer the following

- **How many memory accesses are there per iteration of the inner loop (not the one involving Rep Count)?**
  - Per Iteration there are 2 memory accesses. i.e.: a[j]+=1
- **What is the repeating hit/miss pattern? Write your answer in the form "MMHHMH" and so on, where your response is the shortest pattern that gets repeated.**
  - MHHH is the pattern. In 4 Access there are 3 hits and there is 1 miss. Hit Rate = 0.75
- **Keeping everything else the same, what does our hit rate approach as Rep Count goes to infinity? Try it out by changing the appropriate program parameter and letting the code run! Write your answer as a decimal.**
  - Below are the different values of Rep count and its corresponding hit rate

Rep Count	**Hit Rate**
1	0.75
2	0.875
3	0.9166
100	0.9975
1000	0.9998

Setting-3

Answer the following

- **What is the hit rate of our L1 cache? Our L2 cache? Overall? Write your answer in the form "[L1 HR], [L2 HR], [Overall HR]" where each hit rate is a decimal rounded to two places.**
  - [L1 HR] =0.50, [L2 HR] = 0, [Overall HR] = 0.25
- **How many accesses do we have to the L1 cache total? How many of them are misses? Write your answer in the form "[# of L1 accesses], [# of L1 misses]".**
  - [# of L1 accesses] =32, [# of L1 misses] = 16
- **How many accesses do we have to the L2 cache total? HINT: Think about how this relates to the L1 cache (think about what the L1 cache has to do in order to make us access the L2 cache)?**
  - L1 cache has 16 misses because of which 16 accesses are there in L2 cache.
- **What program parameter would allow us to increase our L2 hit rate, but keep our L1 hit rate the same?**
  - L2 hit rate can be increased when we increase the number of repeat time. When rep time is 2, the L2 hit rate obtained was 0.5. Similarly when rep time is 4, the L2 hit rate obtained was 0.75. Therefore, we can conclude that L2 cache is huge.
- **Do our L1 and L2 hit rates decrease (-), stay the same (=), or increase (+) as we (1) increase the number of blocks in L1, or (2) increase the L1 block size? Write your answer in the form "[1_L1], [1_L2], [2_L1], [2_L2]" (e.g. if I thought L1 will stay the same for both modifications while L2 will decrease for the first and increase for the second, I would answer "=, -, =, +").**

- [1_L1] -> ‘+’, [1_L2] -> ‘-’, [2_L1] -> ‘+’ , [2_L2] ->‘-’
- Hit rate of L1 increases since the number of block increases which will lead to decrease in the hit rate of L2 cache
- Miss in L2 is there since cache miss has occurred in L1 cache

Problem-2

Code discussed in the class is given below

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<sys/time.h>
#include<cstdlib>
#define drand48 1.0/RAND_MAX * rand
#define N 1000
#define index(i,j) (i+(N*j))
int matMult(float *c, float *a, float *b)
{
        int i,j,k;

        for(k=0;k<N;k++)
        {
                for(j=0;j<N;j++)
                {
                        for(i=0;i<N;i++)
                        {
                                c[index(i,j)] += a[index(i,k)]*b[index(k,j)];
                        }
                }
        }
}
int main()
{
        float *a,*b,*c; //WHERE WILL THIS ALLOCATE? STACK
        a=(float*)malloc(N*N*sizeof(float)); //WHERE WILL THIS ALLOCATE? HEAP
        b=(float*)malloc(N*N*sizeof(float)); //WHERE WILL THIS ALLOCATE? HEAP
        c=(float*)malloc(N*N*sizeof(float)); //WHERE WILL THIS ALLOCATE? HEAP
        int i;
        struct timeval start, end;
        for(i=0; i<N*N; i++)
        {
                a[i]=drand48()*2-1;
                b[i]=drand48()*2-1;
                c[i]=0;
        }
gettimeofday(&start, NULL);
matMult(c,a,b);
gettimeofday(&end, NULL);
double elapsed_time=(end.tv_sec - start.tv_sec)+ 1.0e-6*(end.tv_usec - start.tv_usec);
printf("Elapsed Time is %lf\n",elapsed_time);
double gflops = 2e-9*N*N*N/elapsed_time;
printf("GFLOPS is %lf\n",gflops);
return 0;
}
```

- **Which 2 ordering performs best for these 1000-by-1000 matrices? Write your answer in the form "[Ordering1],[Ordering2]" (e.g. "ijk, ikj").**

ORDER	**ELAPSED TIME**	**GFLOPS**
i j k	26.929970	0.074102
i k j	102.598737	0.019493
j i k	31.891978	0.062712
k i j	84.491261	0.023671
j k i	17.397515	0.114959
k j i	**17.390182**	**0.115007**

  - [j k i], [k j i] are the 2 orderings that performs best for this 1000-by-1000 matrices\

- **Explain why this ordering works best.**
  - For ordering [j k i]
    - j is the outermost loop, so in c[index(i,j)] += a[index(i,k)]*b[index(k,j)]; , effect of j will be minimum.
    - k and i will be used maximum number of times than j, since j is present in both the rhs side and twice in lhs i.e.: matrices c and b, because of which cache utilization will be more. i.e.: j will be incremented only after the completion of i and k.
  - For ordering [k j i]
    - Over here incrementing of k will be minimum. On completion of i and j then only k will be incremented.
    - Again the cache utilization will be more since k Is present in the both rhs side i.e.: matrices a and b
- **Which two ordering performs the worst?**
  - [i k j] and [k i j] are the two orderings that performs the worst with highest elapsed time and lowest GFLOPS.

Setting-1: Changing the size of Arrays

- Fix the block size to be 20, and run your code with n equal to 100, 1000, 2000, 5000, and 10000. Record your output in exercise3.txt.

  Answer:

n	**Testing naïve transpose(ms)**	**Testing transpose with blocking (ms)**
100	0.054	0.075
1000	10.634	6.45
2000	110.343	19.33
5000	550.643	102.211
10000	2124.234	489.533

Question-1

- **At what point does cache blocked version of transpose become faster than the non-cache blocked version?**
  - At n=1000 the cache blocked version becomes faster than the non-cache blocked version

Question-2

- **Why does cache blocking require the matrix to be certain size before It outputs the non-cache blocked code?**
  - When n has small value, the cache block can store all the matrix element at same time, and hence naïve transpose has an advantage over cache blocking

- When n becomes large cache can’t accommodate entire matrix and naïve approach becomes expensive. With blocked code, the matrix small parts are close to each other and they can be accessed frequently in such cases the blocked code becomes faster and hence cache blocking requires the matrix to be of certain size since blocked cache needs locality.

Setting-2: Changing Block size

- **Fix n to be 10000, and run your code with block size equal to 50, 100, 500, 1000, 5000. Record you output in exercise3.txt.**

Blocksize	**Testing naïve transpose(ms)**	**Testing transpose with blocking (ms)**
50	2220.532	490.56
100	1452.843	495.78
500	1632.246	305.74
1000	1853.266	400.89
5000	2042.833	1045.85

Question-3

- **How does performance change as block size increases? Why is this the case?**
  - Less time is taken by Transpose with blocking then the naïve transpose.’
  - Also from the above table it is evident that as blocking increases execution time also increases and locality of reference is the reason for the same.
  - Because of increased block size, elements of same block gets stored in the different cache line. This results in the cache miss. Which is the reason for increased execution time.