

CS 2021: Practice Exam 3

Spring 2022

University of Minnesota

Exam period: 20 minutes

Points available: 40

Problem 1 (10 pts): Below is an initial memory/cache configuration along with several memory load operations. Indicate whether these load operations result in cache hits or misses and show the state of the cache after these loads complete.

MAIN MEMORY				
Addr	Addr Bits			Value
10	000	10	000	10
14	000	10	100	11
18	000	11	000	12
1C	000	11	100	13
20	001	00	000	20
24	001	00	100	21
28	001	01	000	22
2C	001	01	100	23
30	001	10	000	100
34	001	10	100	101
38	001	11	000	102
3C	001	11	100	103
40	010	00	000	200
44	010	00	100	201
48	010	01	000	202
4C	010	01	100	203

	Tag	Set	Offset	

DIRECT-MAPPED Cache, 8-byte lines 4 Sets, 8-bit Address = 3-bit tag					
INITIAL CACHE STATE					
Set	V	Tag	Blocks/Line		
			0-3	4-7	
00	1	010	200	201	
01	1	001	22	23	
10	1	000	10	11	
11	0	-	-		

HITS OR MISSES?	
OPEARTION	HIT/MISS?

1. Load 0x48	
2. Load 0x4C	
3. Load 0x24	

FINAL CACHE STATE					
Set	V	Tag	Blocks/Line		Changed?
			0-3	4-7	
00					
01					
10					
11					

Problem 3 (15 pts): Nearby is the definition for `base_scalvec()` which scales a vector by multiplying each element by a number. Write an optimized version of this function in the space provided. Mention in comments why you performed certain transformations.

```

1 int vget(vector_t vec, int idx){
2     return vec.data[idx];
3 }
4 void vset(vector_t vec, int idx, int x){
5     vec.data[idx] = x;
6 }
7 void base_scalevec(vector_t *vec, int *scale){
8     for(int i=0; i < vec->len; i++){
9         int cur = vget(*vec,i);
10        int new = cur * (*scale);
11        vset(*vec,i,new);
12    }
13 }

```

Problem 4 (10 pts): Examine the two functions below which add elements of a row or column vector to all corresponding rows or columns of a matrix. Consider the benchmark timing of these two provided.

1. Explain which of these two functions is faster and **why**.
2. Suggest a way to increase the speed of the slower function with only moderate changes to the code.

```

1 // add given row to each row of mat
2 void matrix_addrow_vec(matrix_t mat,
3     vector_t row) {
4     for(int i=0; i<mat.rows; i++){
5         for(int j=0; j<mat.cols; j++){
6             int elij = MGET(mat,i,j);
7             int vecj = VGET(row,j);
8             MSET(mat,i,j, elij + vecj);
9         }
10    }
11 }
12 // add given col to each column of mat
13 void matrix_addcol_vec(matrix_t mat,
14     vector_t col) {
15     for(int j=0; j<mat.cols; j++){
16         for(int i=0; i<mat.rows; i++){
17             int elij = MGET(mat,i,j);
18             int veci = VGET(col,i);
19             MSET(mat,i,j, elij + veci);
20         }
21    }
22 }
23 // BENCHMARK TIMING:
24 //   SIZE      addrow      addcol
25 //   512 2.9040e-03 5.5230e-03
26 //  1024 5.9290e-03 1.3160e-02
27 //  2048 1.3809e-02 9.9269e-02
28 //  4096 5.0853e-02 3.6760e-01
29 //  8192 2.0867e-01 1.4719e+00

```