Question 6: *Cache in While You Can* (17 points, 26 Minutes)

Consider a single 4KiB cache with 512B blocks and a write-back policy. Assume a 32-bit address space.

a) If the cache were direct-mapped,

of ~~rows~~ sets? __________   # of offset bits? __________

b) If the cache were 4-way set associative,

# of tag bits? __________   # of index bits? __________   # of bits per cache ~~slot~~ line? __________

Consider an array of the following `location` structs:

```
typedef struct {
      ... // some undefined number of other struct members
      int visited;
      int danger;
} location;
location locs[NUM_LOCS];
```

Here's a piece of code that counts the number of places we've visited. Assume this gets executed somewhere in the middle of our program, that `count` is held in a register, and the size of the array is greater than 4 KiB.

```
for(int i = 0; i < NUM_LOCS; i++)
      if(locs[i].visited) count++;
```

c) What's the fewest possible number of bytes written to main memory? ____________

d) What's the greatest possible number of bytes written to main memory? ____________

Now consider if we store the `visited` and `danger` information in individual arrays instead:

```
int visited[NUM_LOCS];
int danger[NUM_LOCS];
```

e) This way, the cache can exploit better ____________________ for the above task.

We can expect a ____________________ (higher or lower) miss rate

because of the change in the number of ____________________(type of cache miss) misses.

Consider the following code with `NUM_LOCS > 2^10`.

```
for(int i = 0; i < NUM_LOCS; i++)
      if(visited[i] && danger[i] > 5) count++;
```

Two memory accesses are made per iteration: one into `visited`, the other into `danger`. Assume that the cache has no valid blocks initially. **You are told that in the worst case, the cache has a miss rate of 100%.** Consider each of the following possible changes to the cache individually.

f) Mark each as **E**, if it eliminates the chances of this worst-case scenario miss rate, **R** if it reduces the chances, or **N** if it's not helpful.

- More sets, same block size, same associativity ______
- Double associativity, half block size, same total cache size ______
- Everything stays the same but use a write-through policy instead ______

3) "*Cache*, money. Dollar bills, y'all." (24 min, 15 pts)

Suppose we have a standard 32-bit byte-addressed MIPS machine, a single direct-mapped 32KiB cache, a write-through policy, and a 16B block size.

a) Give the T:I:O breakup. ________________

b) How many bits are there per ~~row~~ line on the cache? ____

Use the C code below and the description of the cache above to answer the questions that follow it. Suppose that the only memory accesses are accesses and stores to arrays and that all memory accesses in the code are valid. Assume `A` starts on a block boundary (byte 0 of `A` in byte 0 of block).

```
#define NUM_INTS 32
#define OFFSET   8192    // 8192 = 2^13

int rand(int x, int y); // returns a random integer in the range [x, y)

int main(){
      int A[NUM_INTS + OFFSET];                // Assume A starts on a block boundary

      // START LOOP 1
      for ( int count = 0 ; count < NUM_INTS ; count += 1 ) { // count by 1s
            A[count]             = count;        // ACCESS #1
            A[count + OFFSET] = count+count; // ACCESS #2
      }
      // END LOOP 1

      // START LOOP 2
      for ( int count = 0 ; count < NUM_INTS ; count += 4 ) { // count by 4s now
            for ( int r = 0 ; r < 4 ; r++ ) {                 // …but do it 4 times
                  printf("%d", A[rand(count, count+4)]);
            }
      }
      // END LOOP 2
}
```

c) Hit rate for Loop 1? _____________ What types of misses are there? ______________________

d) Hit rate for Loop 2? _____________ What types of misses are there? ______________________

Questions (e), (f), and (g) below are three independent variations on the original code & settings.

e) If the cache were 2-way set associative, what would be the hit rate for Loop 2? ________________
(assume the standard LRU replacement policy)

f) If instead we removed the line labeled `ACCESS #2`, what would be the hit rate for Loop 2? ________

g) Instead, what's the smallest we could shrink `OFFSET` to maximize our Loop 2 hit rate? __________
(assume we still need to maintain the same functionality)

M2) *Cache Money, y'all* (10 pts, 20 min)

This C code runs on a 32-bit MIPS machine with 4 GiB of memory and a single L1 cache. Vectors **`A,B`** live in different places of memory, are of equal size (**`n`** is a power of 2 and a [natural number] multiple of the cache size), block aligned. The size of the cache is C, a power of 2 (and always bigger than the block size, obviously).

`// sizeof(uint8_t) = 1`
`SwapLeft(uint8_t *A, uint8_t *B, int n) {`
`   uint8_t tmp;`
`   for (int i = 0; i < n; i++) {`
`      tmp = A[i];`
`      A[i] = B[i];`
`      B[i] = tmp;`
`   }`
`}`
	`// sizeof(uint8_t) = 1`
	`SwapRight(uint8_t *A, uint8_t *B, int n) {`
	`   uint8_t tmpA, tmpB;`
	`   for (int i = 0; i < n; i++) {`
	`      ____________________________`
	`      ____________________________`
	`      ____________________________`
	`      ____________________________`
	`   }`
	`}`

Let's first just consider the **`SwapLeft`** code for parts (a) and (b).

a) If the cache is **direct mapped** and the *best* hit:miss ratio is "H:1", what is the block size in bytes? _____

b) What is the *worst* hit:miss *ratio*? _____:_____

c) Fill in the code for **`SwapRight`** so that it does the same thing as **`SwapLeft`** but improves the (b) hit:miss ratio. You may not need all the blanks.

d) If the block size (in bytes) is *a*, what is the *worst* hit:miss ratio for **`SwapRight`**? _______:_______

e) We next change the cache to be **2-way set-associative**, and let's go back to just considering **`SwapLeft`**. What is the **worst** hit:miss ratio for **`SwapLeft`** with the following replacement policies? The cache size is *C* (bytes), the block size is *a* (bytes), LRU = Least Recently Used, MRU = Most Recently Used.

LRU and **an empty cache**	**MRU** and **a full cache**
_______:_______	_______:_______

d) Ash Ketchum has six slots in his party, each of which can hold a single Pokémon. Additionally, Ash has access to a PC (personal computer) which holds the rest of the Pokémon he owns. Essentially, his party acts as a "cache" for accesses to the PC (the "memory").

i. Each slot in Ash's party can hold any Pokémon. What kind of cache is this analogous to? (Circle one)

Set-associative   Write-back   Fully Associative   Direct Mapped   Write-through

ii. Ash's party exploits _____________ locality but not _____________ locality.

Explain in one sentence:

SID: ________________

Question 4: Caches (11 pts)

We have a 64 KiB address space and two different caches. Both are 1 KiB, direct-mapped caches with random replacement and write-back policies. **Cache X** uses 64 B blocks and **Cache Y** uses 256 B blocks.

a) Calculate the TIO address breakdown for **Cache X**:

Tag	Index	Offset

b) During some part of a running program, **Cache Y**'s management bits are as shown below. Four options for the next two memory accesses are given (R = read, W = write). Circle the option that results in data from the cache being *written to memory*.

Line ~~Slot~~	Valid	Dirty	Tag
00	0	0	1000 01
01	1	1	0101 01
10	1	0	1110 00
11	0	0	0000 11

(1) R 0x4C00, W 0x5C00

(2) W 0x5500, W 0x7A00

(3) W 0x2300, R 0x0F00

(4) R 0x3000, R 0x3000

c) The code snippet below loops through a character array. Give the value of LEAP that results in a Hit Rate of 15/16 for **Cache Y**.

```
#define ARRAY_SIZE 8192
char string[ARRAY_SIZE];            // &string = 0x8000
for(i = 0; I < ARRAY_SIZE; i += LEAP) {
    string[i] |= 0x20;              // to lower
}
```

[ ]

d) For the loop shown in part (c), let LEAP = 64. Circle ONE of the following changes that increases the hit rate of **Cache X**:

Increase Block Size   Increase Cache Size   Add a L2$ Increase LEAP

e) For the following cache access parameters, calculate the AMAT. ~~All miss and hit rates are local to that cache level.~~ Please simplify and include units.

L1$ Hit Time	L1$ **Miss** Rate	~~L2$ Hit Time~~	~~L2$ **Hit** Rate~~	MEM Hit Time
2 ns	40%	~~20 ns~~	~~95%~~	400 ns

[ ]

SID: ________________

Question 8: Caches (10 pts)

We are using a 20-bit byte addressed machine. We have two options for caches: **Cache A** is fully associative and **Cache B** is 4-way set associative. Both caches have a capacity of 16 KiB and 16 B blocks.

a) Calculate the TIO address breakdown for **Cache A**:

Tag	Index	Offset
		4

b) Below is the initial state of **one set** (four ~~slots~~ lines) in **Cache B**. Each slot holds 2 LRU bits, with `0b00` being the most recently used and `0b11` being the least recently used. Circle ONE option below for two memory accesses that result in the final LRU bits shown and ***only one block replacement***/eviction.

Index: 1001 1110

~~Slot~~ Line	Tag	Initial LRU bits		Final LRU bits
0	0110 1010	00	→	10
1	0000 0001	10	→	00
2	0101 0101	01	→	11
3	1010 1100	11	→	01

(1) 0x019D0, 0xAD9D0 (2) 0xAC9E0, 0x129E0

(3) 0xAD9D0, 0x019D0 (4) 0x129E0, 0xAC9E0

c) For the code given below, calculate the hit rate for **Cache B** assuming that it starts cold.

```
#define ARRAY_SIZE 8192
int int_arr[ARRAY_SIZE];                    // &int_arr = 0x80000
for (int i = 0; i < ARRAY_SIZE / 2; i++) {
    int_arr[i] *= int_arr[i + ARRAY_SIZE / 2];
}
```

d) For each of the proposed changes below, write **U** for "increase", **N** for "no change", or **D** for "decrease" to indicate the effect on the hit rate of **Cache B** for the loop shown in part (c):

Direct-mapped ______ Increase cache size ______

Double ARRAY_SIZE ______ Random block replacement ______

e) Calculate the AMAT for ~~a multi-level cache given~~ the following values. Don't forget units!

HT = Hit Time, MR = Miss Rate, ~~GMR = Global Miss Rate~~

L1$ HT	L1$ MR	~~L2$ HT~~	~~GMR~~	MEM HT
4 ns	20%	~~25 ns~~	~~5%~~	500 ns