Name: ID:

BLG322E – Quiz 2

Duration: 30 Minutes

A computer system includes **64 KB (kilobyte) main memory** and a **cache memory that can hold 256 B (byte)** data. Data transfers between main and cache memories are performed using **blocks of 16 bytes**. The cache control unit uses set associative mapping technique where each set contains two frames **(2-way set associative)**. In necessary cases **FIFO** is used as a replacement technique. Assume that **Frame i is older than Frame i+1 in each set**. Cache memory is used only for data, not for instructions.

The CPU runs the piece of pseudo code given below.

```
for (int i = 0; i<8; i++) {:
        B[i] = A[i];
}
```

~~Ten~~ Eight elements of array A are copied to array B. Each element is one byte, and each iteration of the program involves accessing data at the byte level, indicating that the **program processes data byte by byte (data is accessed in bytes).** The starting addresses of the arrays are given below:

- **A: $\$$0031A, B: $\$$00520**

Assume that the cache memory is full; but arrays A and B are not in the cache at the beginning of the program. For write operations, the **Simple Write Back (SWB)** with **Write Allocate (WA)** method is used.

a) **(10 pts)** Into which fields does the cache controller divide the physical address? Specify field names and bit widths (Tag, Set Index, Block Offset/Word bits).

Field	**Width (bits)**	**Explanation**
Block Offset	4 bits	16 B block $\Rightarrow \log_2 16 = 4$
Set Index	3 bits	Cache holds 256 B / 16 B = 16 blocks $\rightarrow$ 16 / 2 (ways) = 8 sets $\Rightarrow \log_2 8 = 3$
Tag	9 bits	16-bit physical address – (4 + 3) = 9

b) **(40 pts)** Which set/frames of cache memory does the cache control unit place arrays A and B into? Indicate all blocks of arrays.

Physical address should be in hex, and its binary form to show controller's division of the address into fields. Set and frame indices are in decimal.

Example:
A0 -> A[i]-A[j]: $\$$xxxx *(in hex)*, bbb..|bb..|bb.. *(in binary)*, set X *(in decimal)*, frame 0 *(in decimal)*
A1 -> A[j+1]-A[k]: $\$$xxxx, bbb..|bb..|bb.., set X, frame 1
…

A0 -> A[0]…A[5]: $\$$0031A, 000000011 | 001 | 1010, Set 1, Frame 0 (older)
A1 -> A[6]…A[7]: $\$$00320, 000000011 | 010 | 0000, Set 2, Frame 1 (younger)
B0 -> B[0]…B[7]: $\$$00520, 000000101 | 010 | 0000, Set 2, Frame 0 (older)

c) **(30 pts)** How many read misses, read hits, write misses, write hits, write-to-main-memory operations, and block transfers occur during the run of the given loop?

Parameter	**Answer**
Number of **Read misses:**	2
Number of **Read hits:**	6
Number of **Write misses:**	1
Number of **Write hits:**	7
Number of **Block transfers for main->cache**	3
Number of **Block transfers for cache->main**	3

d) **(30 pts)** Explain the reason for each block transfer (both of main->cache and cache->main) during the run of the given program.

During the loop, three block replacements occur, each triggering two memory transfers due to the Simple Write-Back (SWB) policy.

1. A[0] access causes a read miss, fetching block $0031A into set 1, frame 0. The previous block in that frame is written back to memory.

2. B[0] access causes a write miss (write-allocate), bringing in block $00520 into set 2, frame 0. The old block is evicted and written back.

3. A[6] access causes another read miss, fetching block $00320 into set 2, frame 1, evicting and writing back the existing block.

Each miss results in 1 memory → cache and 1 cache → memory transfer, totaling 3 loads and 3 write-backs. After that, all accesses hit cached blocks, so no more transfers occur.