**Following problems needs to be solved manually and executed in Para cache**

1. Consider a direct mapped cache of size 16 bytes with block size 4 bytes. The size of main memory is 256 bytes. Find Number of bits in tag, index and offset. The processor generates requests as follows

1,4,8,5,14,11,13,38,9,B,4,2B,5,6,9,11.

Find hit rate and miss rate.

1. Consider a direct mapped cache of size 16 KB with block size 256 bytes. The size of main memory is 128 KB. Find Number of bits in tag. Randomly generate 10 addresses and find hit rate and miss rate.

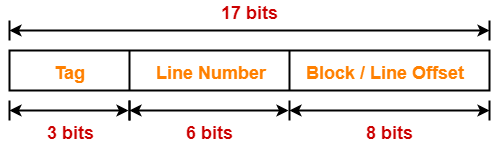
**Solution:**

**Given-**

**Cache memory size = 16 KB**

**Block size = Frame size = Line size = 256 bytes**

**Main memory size = 128 KB = 2^17**

****

1. A computer system uses 16-bit memory addresses. It has a 2K-byte cache organized in a direct-mapped manner with 64 bytes per cache block. Assume that the size of each memory word is 1 byte.

a) Calculate the number of bits in each of the Tag, Block, and Word fields of the memory address.

(b) When a program is executed, the processor reads data sequentially from the following word addresses: **128, 144, 2176, 2180, 128, 2176**

All the above addresses are shown in decimal values. Assume that the cache is initially empty. For each of the above addresses, indicate whether the cache access will result in a hit or a miss.

**Solution:**

**Access # 1: The cache is initially empty. Therefore, all the cache blocks are invalid**

**Address = (128)10 = (0000000010000000)2**

**For this address, *Tag* = 00000, *Block* = 00010, *Word* = 000000**

**Since the cache is empty before this access, this will be a cache miss**

**After this access, Tag field for cache block 00010 is set to 00000**

**Access # 2: Address = (144)10 = (0000000010010000)2**

**For this address, *Tag* = 00000, *Block* = 00010, *Word* = 010000**

**Since tag field for cache block 00010 is 00000 before this access, this will be a cache hit (because address tag = block tag)**

**Access # 3: Address = (2176)10 = (0000100010000000)2**

**For this address, *Tag* = 00001, *Block* = 00010, *Word* = 000000**

**Since tag field for cache block 00010 is 00000 before this access, this will be a cache miss**

**(address tag ≠ block tag)**

**After this access, Tag field for cache block 00010 is set to 00001**

**Access # 4:**

**Address = (2180)10 = (0000100010000100)2**

**For this address, *Tag* = 00001, *Block* = 00010, *Word* = 000100**

**Since tag field for cache block 00010 is 00001 before this access, this will be a cache hit (address tag = block tag)**

**Access # 5:**

**Address = (128)10 = (0000000010000000)2**

**For this address, *Tag* = 00000, *Block* = 00010, *Word* = 000000**

**Since tag field for cache block 00010 is 00001 before this access, this will be a cache miss**

**(address tag ≠ block tag)**

**After this access, Tag field for cache block 00010 is set to 00000**

**Access # 6:**

**Address = (2176)10 = (0000100010000000)2**

**For this address, *Tag* = 00001, *Block* = 00010, *Word* = 000000**

**Since tag field for cache block 00010 is 00000 before this access, this will be a cache miss**

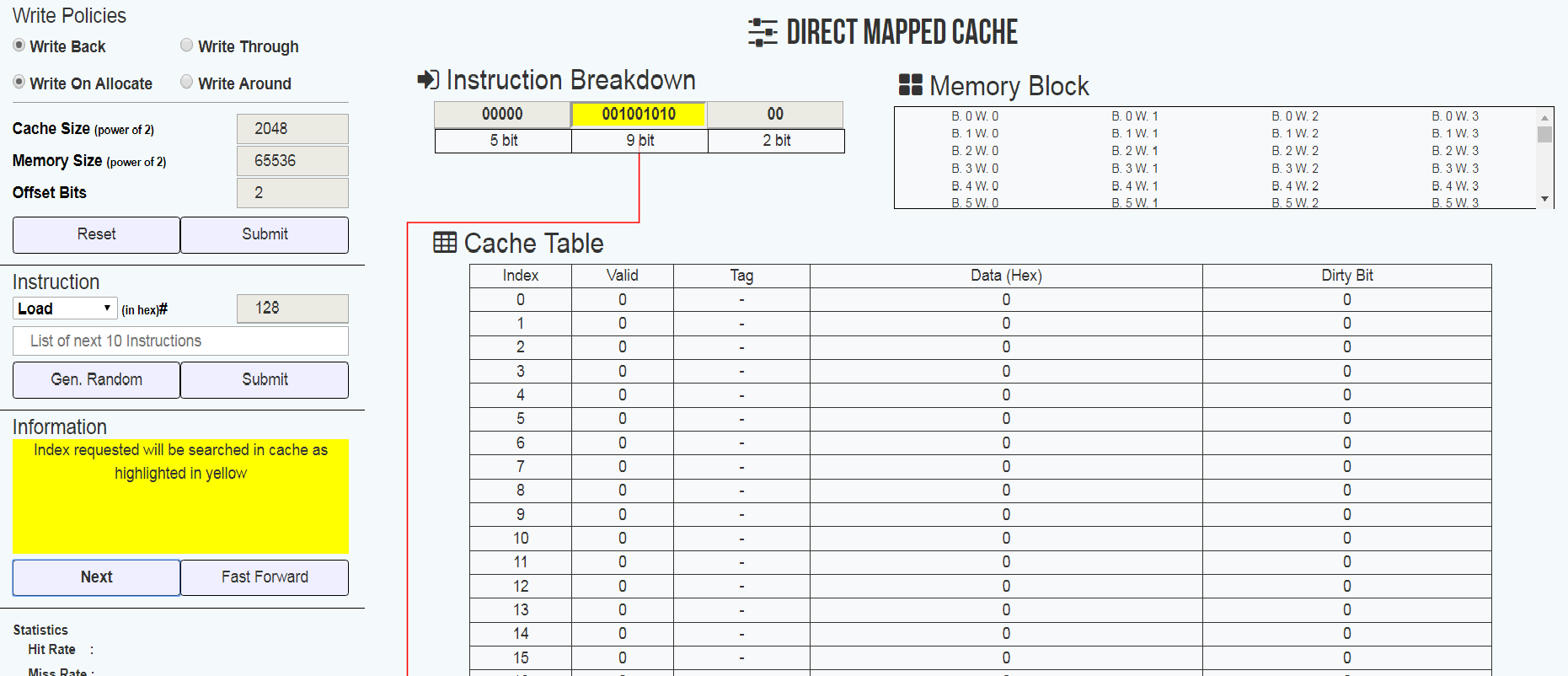
**(address tag ≠ block tag)**

**After this access, Tag field for cache block 00010 is set to 00001**

**Cache hit rate = Number of hits / Number of accesses**

**= 2/6**

**= 0.333**



1. Consider a 2-way set associative cache of size 16 KB with block size 256 bytes. The size of main memory is 128 KB. Find Number of bits in tag.

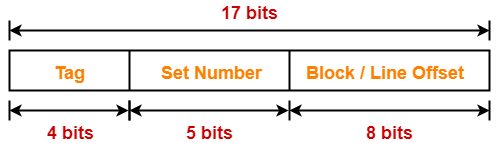
Randomly generate 10 addresses and find hit rate and miss rate.

**Solution:**

**Cache memory size = 16 KB**

**Block size = Frame size = Line size = 256 bytes**

**Main memory size = 128 KB = 2^17**

****

5 .Consider a main memory having 64 byte capacity and cache memory of 8 bytes initially empty

The following addresses are generated by the CPU.All values in hexadecimal

Clearly label data that is replaced in cache lines

Show the cache memory table and filled data in the cache lines of block size 1 byte

LRU Policy is used.

The cache is mapped as

a)Direct Mapped

b)Two way set Associative

c)Four Way Set associative

d)Fully Associative