1a.

Sequential Access: Memory is organized into units of data called records. Records are accessed in a linear fashion. Stored addressing information is used to separate records and assist in the retrieval process. Time to access a record is highly variable. Tape units eg: Magnetic tapes are sequential access.

Direct Access: As with Sequential Access, Direct Access involves a shared read-write mechanism. Records have a unique address based on the physical location. Access is accomplished by direct access to reach a general vicinity plus sequential searching, counting, or waiting to reach the final location. Disk units eg: magnetic disk, CD-ROM, are direct access.

Random Access: Each addressable location in memory has a unique, physically wired- in addressing mechanism. The time to access a given location is constant. Thus, any location can be selected at random and directly addressed and accessed. Main memory and some cache systems eg: DRAM, are random access.

1b.

Memory access time(latency): For random-access memory, it is defined as the time to perform a read or write operation. The time starts from the instant that an address is presented to the memory to the instant tat data have been stored or made available for use. For non-random-access memory, access time is the time it takes to position the read-write mechanism at the desired location.

Memory cycle time: The time includes the access time plus any additional time required before the next access an commence. Additional time may be required for transients to die out on signal lines or to regenerate data if they are read destructively. Concerned with the system bus,not the processor.

Transfer Rate: This is the rate at which data can be transferred into or out of a memory unit. For random-access memory it is equal to 1/(cycle time). For non-random-access memory:

$$Tn = TA + \frac{n}{R}$$

where *Tn* = Average time to read or write *n* bits, *TA* = Average access time, *n* = Number of bits, *R* = Transfer rate, in bits per second(bps)

2.

Performance depends more on memory access pattern than on cache size. More precisely, if the program is mainly sequential, cache size is not a big deal. If there are quite a lot of random access (ex. when associative containers are actively used), cache size really matters. The latency of looking up a higher size cache for a hit is slower than looking up a smaller size cache. Increasing the cache size doesn't equate to increase in overall performance. Even tough having a higher cache size, if the hit ratio is low then it takes more time to access data at higher memory levels. The larger the cache, the larger the number of gates involved in addressing the cache. The result is that large caches tend to be slightly slower than small ones—even when built with the same integrated circuit technology and put in the same place on chip and circuit board.

3.

Given 4-way set associativity where k =4.

Each line is 8 bytes long, hence no of bits to represent the word is 2^3 = 8 bytes

Main memory is 64MB and it address size is 26 bits, 2^{26} = 64MB

We know that Cache size is = 8KB = 2^{13}

We can write k*8*v = 2^{13} , where v is size of set

v = 2^8 and it requires 8 bits for representation. So set size is 8 bits.

Now we have s+w = 26 and w =3 so s = 23

We know that tag size is s-d = 23 - 8 = 15 bits

Tag = 15 bits
Set = 8 bits
Word = 3 bits

4.

I am not submitting my solution for question 4 since its optional.

5.

Given k = 2

No of bits to represent block = 4 = 4*16 bits

Cache size = 4096 = 4k words

Number of set is 2^d = $4k/8 = 2^9$, hence, d = 9

Given block size is four 16 bits

$2^w = 4 \times 16 bits = 2^3$ *bytes*, hence, w = 3

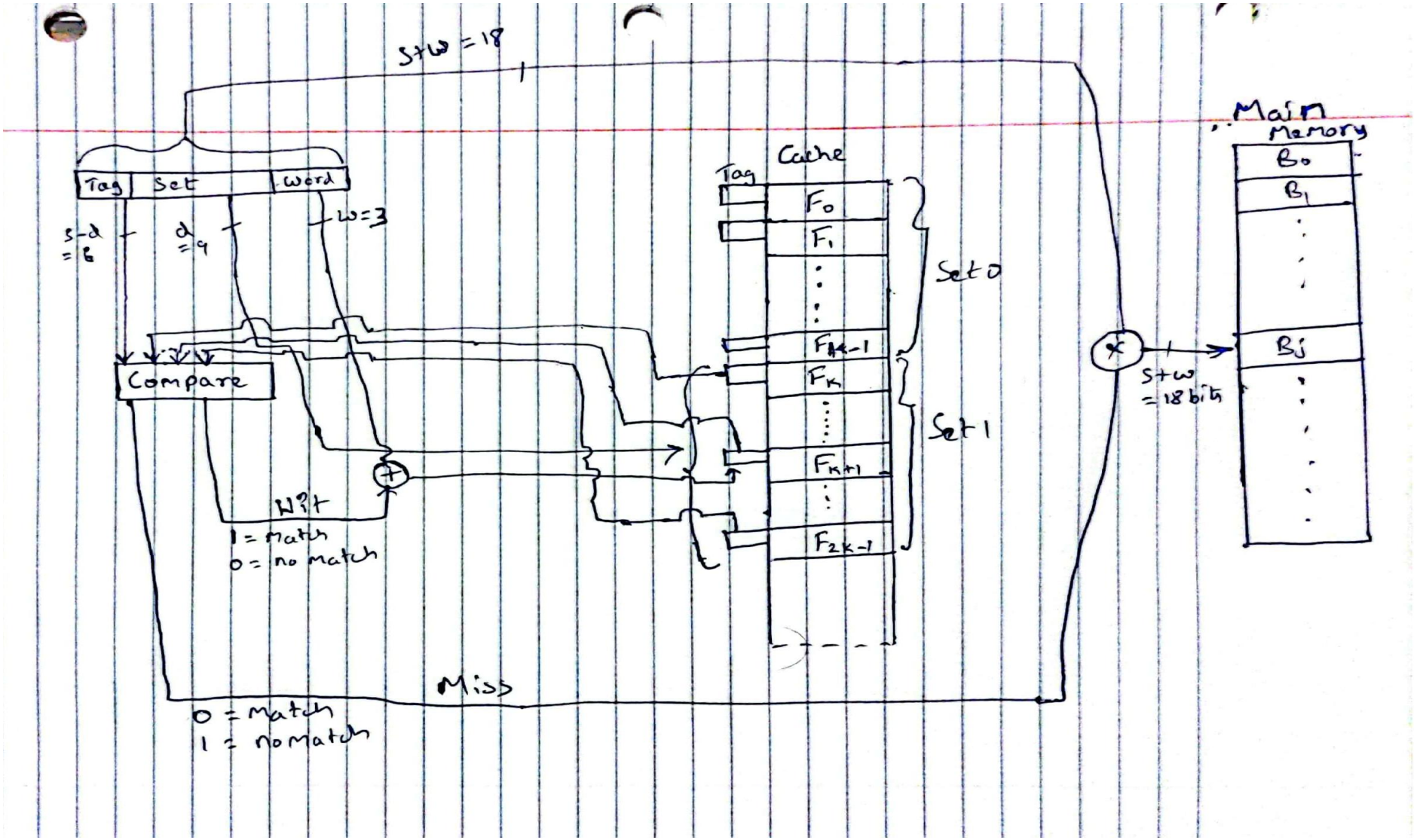
Given cache-able memory $64k \times 32 bits$

$64k \times 32 bits = 2^{18} bytes$

Hence s+w = 18 and w=3, Size of tag = s-d = 18 − 9 − 3 = 6

Tag = 6
Set = 9
Word = 3

The design for the same will be as follows


s+w = 18
Main Memory
Tag
Set
word
s-d = 6
d = 9
w = 3
Cache
Tag
F0
F1
FK-1
FK
FK+1
F2K-1
Set 0
Set 1
Compare
Hit
1 = match
0 = no match
Miss
0 = match
1 = no match
B0
B1
Bj
s+w = 18 bits

6a.

Address format is as follows

Tag	Line	Word
20	6	6

number of addressable units is = 2^{32} bytes = 4GB
number of blocks in main memory = $2^s = 2^{32-6} = 2^{26}$, where 6 is no of bits used to represent word
number of lines = $2^r = 2^{32-20-6} = 2^6 = 64$
Size of Tag is = 20 given

6b.

Address format is as follows

Tag	Word
26	6

number of addressable units is = 2^{32} bytes = 4GB
number of blocks in main memory = $2^s = 2^{32-6} = 2^{26}$, where 6 is no of bits used to represent word
number of lines = not present
Size of Tag is = $2^{32-6} = 2^{26}$, Tag is 26 bits long.

6c.

Address format is as follows

Tag	Set	Word
9	17	6

number of addressable units is 2^{32} bytes = 4GB
number of blocks in main memory is = $2^s = 2^{32-6}$, as w = 6 given
number of lines in set = *k* = 4 given
number of sets in cache = v = $2^d = 2^{32-6-9} = 2^{17}$ as d = s - tagbits
number of lines in cache = $k \times sets = 4 \times 2^{17} = 2^{19}$
size of tag = 9 given.