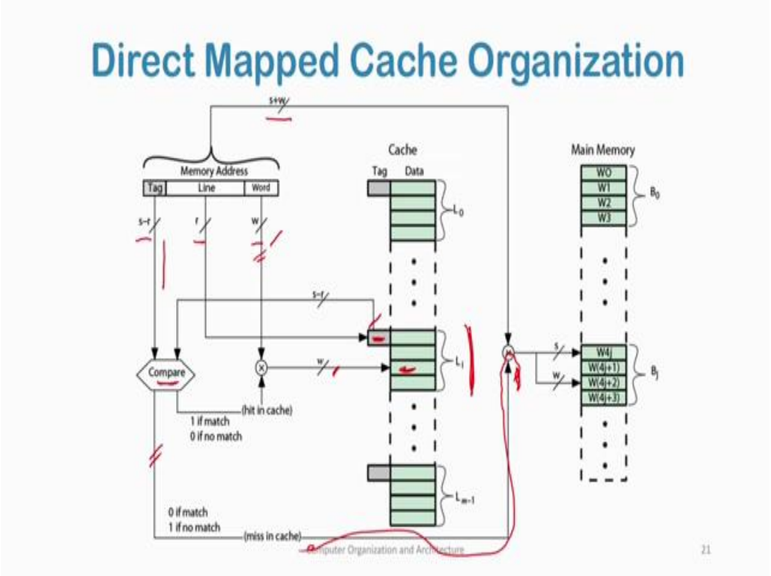
(Refer Slide Time: 17:19)



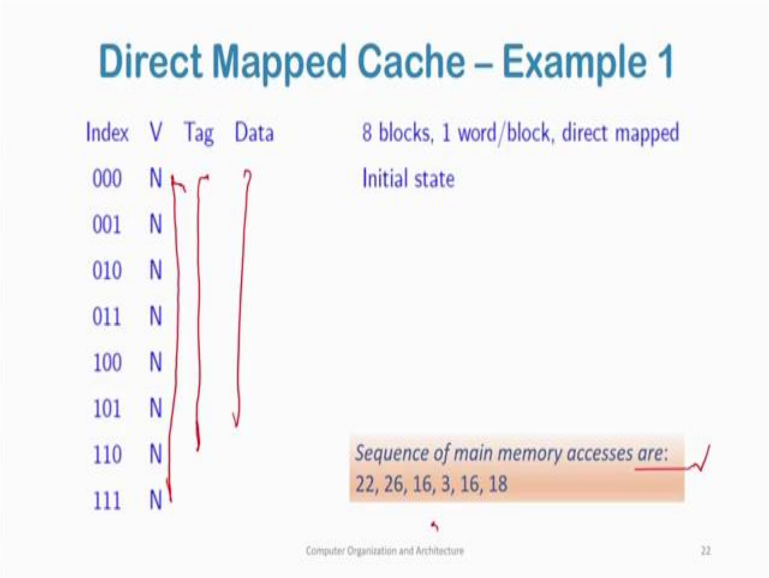
Direct Mapped Cache Organization

So, this figure shows the organization of a direct mapped cache. We see that the memory address consists of s plus w bits. The tag is s minus r bits long. The cache line index, the cache index the cache is indexed by an r length r bit length quantity and the each word within a particular block or line is identified by the by this word offset here by this word offset ok.

So, to identify whether a particular a particular line is in cache or not what do we do? We first match the; we first come to the line. We come to the line which is identified by identified by these r bits and then we compare the tag field. This is the tag field within the cache, we compare the tag field with the s minus r main memory bits. If this comparison says is if this comparison is 1, we have a match and a hit in cache. When we have a hit in cache, we read the word we read the corresponding word in the cache and we retrieve it ok.

So, the corresponding word is identified by these least significant w bits. And so, this identifies which word within this block or line which word within this line is in cache ok. Now if there is a miss; that means, the tag in the in the particular cache line in the tag in that particular cache line does not match with the main memory address tag; that is these s minus r when we have a mismatch here, we have a cache miss and then we go to the main memory. We go to the main memory and find the particular block in main memory containing the word and then retrieve it into the cache.

(Refer Slide Time: 19:49)



Direct Mapped Cache – Example 1

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

8 blocks, 1 word/block, direct mapped

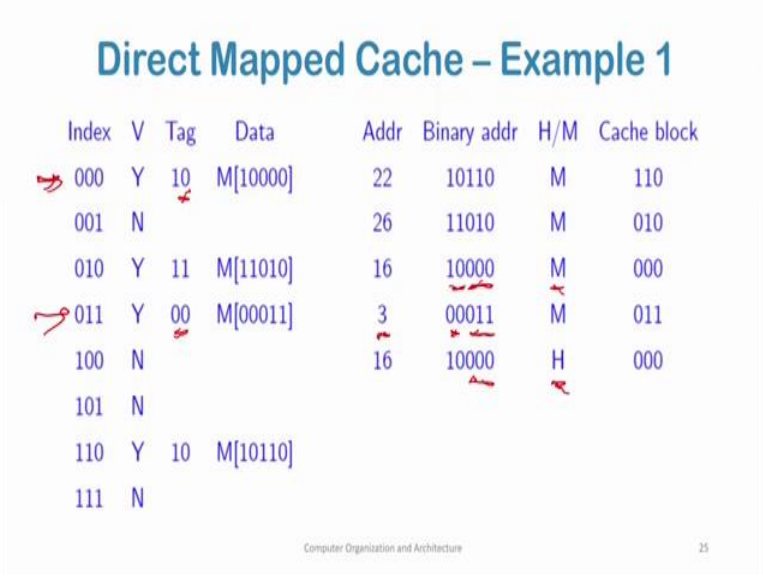
Initial state

Sequence of main memory accesses are:
22, 26, 16, 3, 16, 18

Computer Organization and Architecture 22

Now we take an example of a very simple example of a direct mapped cache. For this cache we only have 8 blocks or 8 lines in the cache. We have 1 word per block so every word is a block. We have a direct mapped cache and the initial state is all blank. So, we see that all the valid bits are N; that means, nothing has been accessed, the tag field is empty, data is empty and there is nothing in the cache basically. And we have let us say we have the sequence of memory accesses 22, 26, 16, 3, 16, 18.

(Refer Slide Time: 20:36)



Direct Mapped Cache – Example 1

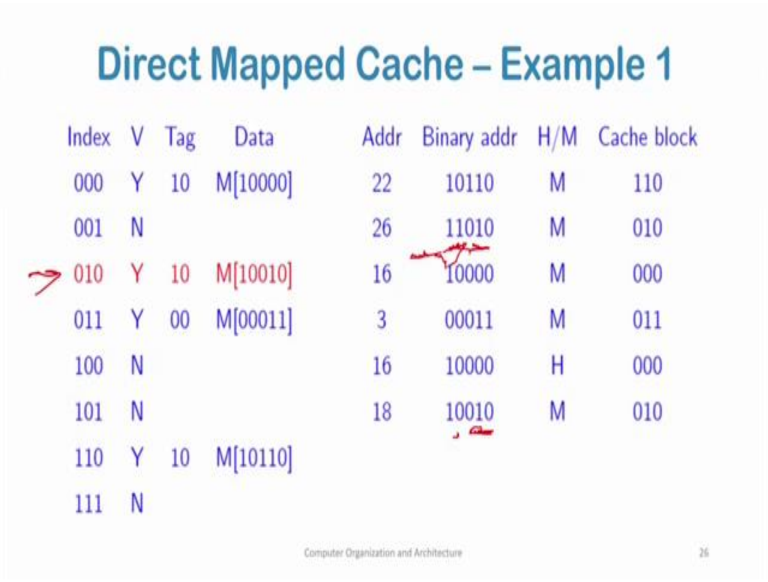
Index	V	Tag	Data	Addr	Binary addr	H/M	Cache block
000	Y	10	M[10000]	22	10110	M	110
001	N			26	11010	M	010
010	Y	11	M[11010]	16	10000	M	000
011	Y	00	M[00011]	3	00011	M	011
100	N			16	10000	H	000
101	N						
110	Y	10	M[10110]				
111	N						

Computer Organization and Architecture 25

So, when the first address 22 is accessed, the corresponding binary address of 22 is 10110. We have 8 lines in cache. So, the least 3 significant bits identify the cache line, the 2 most significant bits become the tag bits. We have a miss in cache because the cache is initially empty. We would retrieve the, we retrieve the cache we retrieve it from the main memory and put it at line put it at line 110 and the tag is 10 as we see. So, when the next address 26 is accessed, we again have a miss. The corresponding binary addresses is the corresponding binary address is 010. So, we put it at line 010 with the tag 11 which is the most 2 significant bits right and we take it from memory and put it in this cache line.

Next 3 memory accesses we access first we access 16. So, the line index is 000, the tag field is 10, we have a miss in cache right we have a miss in cache and we put it back into the memory. Next we access 3. So, the line number is 011, we again have a miss in cache and the tag is 00. Next we access 16 again.

(Refer Slide Time: 22:32)



Direct Mapped Cache – Example 1

Index	V	Tag	Data
000	Y	10	M[10000]
001	N		
010	Y	10	M[10010]
011	Y	00	M[00011]
100	N		
101	N		
110	Y	10	M[10110]
111	N		

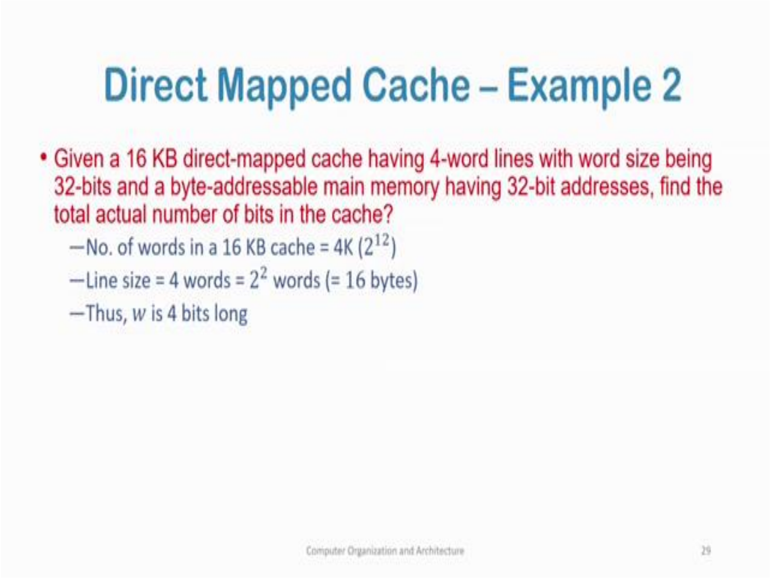
Addr	Binary addr	H/M	Cache block
22	10110	M	110
26	11010	M	010
16	10000	M	000
3	00011	M	011
16	10000	H	000
18	10010	M	010

Computer Organization and Architecture

26

Now, 16 is already there in the cache, 16 is already there in the cache. We have a hit, we have a hit right after that when we access 18 we see that the line is 010. So, when we have 010 we already had 010 which is 26, 26 previously in this position we had we had 010; that means, the tag was 11, there was a mismatch in the tag. And therefore, there was a miss. There was a mismatch in the tag and therefore, there was a miss. We replaced the cache block 26 with 18 and put it back into the cache with the new tag and word.

(Refer Slide Time: 23:16)



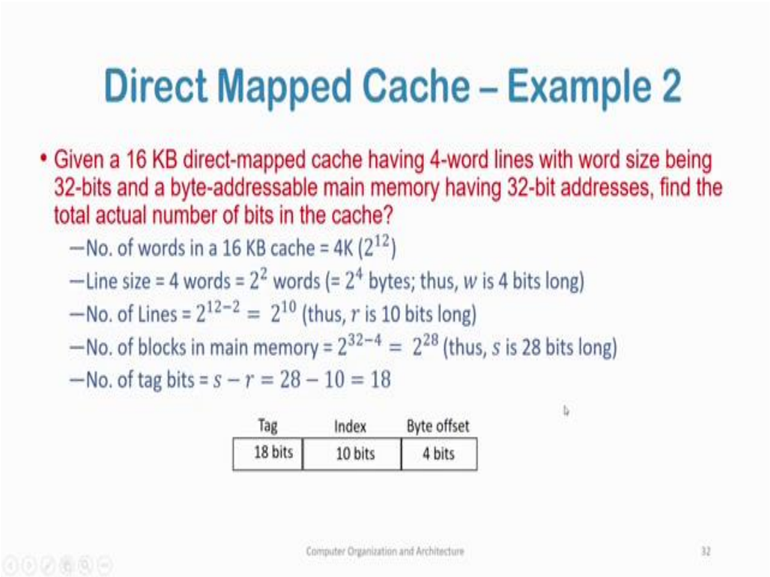
Direct Mapped Cache – Example 2

- Given a 16 KB direct-mapped cache having 4-word lines with word size being 32-bits and a byte-addressable main memory having 32-bit addresses, find the total actual number of bits in the cache?
  - —No. of words in a 16 KB cache = 4K (2^{12})
  - —Line size = 4 words = 2^2 words (= 16 bytes)
  - —Thus, w is 4 bits long

Computer Organization and Architecture 29

We come to a second example: given, a 16 KB direct mapped cache, having 4-word blocks with word size being 32 bits. A byte addressable main memory having 32 bit addresses we need to find the total actual number of bits in the cache. The first important thing to remember here is that line size is given by the number of words in each cache line. However, the number of bits in each line is given by the word itself along with the number of bits represented by the tag as well as the valid bit.

(Refer Slide Time: 24:39)



Direct Mapped Cache – Example 2

- Given a 16 KB direct-mapped cache having 4-word lines with word size being 32-bits and a byte-addressable main memory having 32-bit addresses, find the total actual number of bits in the cache?
  - —No. of words in a 16 KB cache = 4K (2^{12})
  - —Line size = 4 words = 2^2 words (= 2^4 bytes; thus, w is 4 bits long)
  - —No. of Lines = $2^{12-2} = 2^{10}$ (thus, r is 10 bits long)
  - —No. of blocks in main memory = $2^{32-4} = 2^{28}$ (thus, s is 28 bits long)
  - —No. of tag bits = $s - r = 28 - 10 = 18$

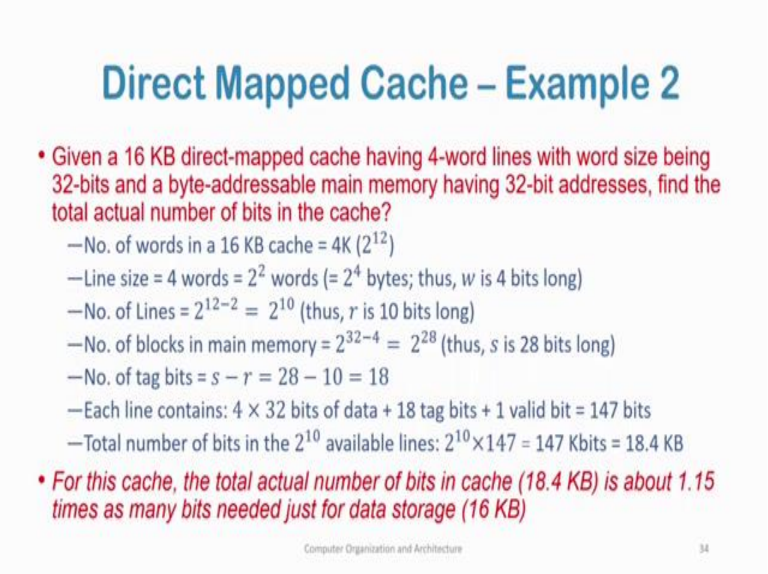
Tag	Index	Byte offset
18 bits	10 bits	4 bits

Computer Organization and Architecture 32

So firstly, the number of words in the cache is given by in the in this in the in the 16 KB cache is 4K. Why? Because each word is 32 bits, so we have a 4 byte word and we have a 16 KB cache. So, we have 4K words in the cache. Line size equals to 4 words. So, each line contains 4 words; that means 2 to the power 2 words. So, w in this case is given by is 4 bits long. So, number of lines in the cache is given by the number of words divided by the number the number of words divided by the number of words in each line. So, total number of words divided by the number of words in each line, therefore, it is equal to 2 to the power twelve divided by 2 equals to the power 10. So, in our case we need 10 bits to address each line in the cache.

Number of blocks in main memory is given by 32. So, at the total number of blocks in main memory this total number of words in main memory is 2 to the power 32. Each block each block contains 2 to the power 4, 2 to the power 4 bytes. So, the number of blocks in main memory is equal to the total number of bytes divided by the number of bytes in each block and that is equals to 2 to the power 32 divided by 2 to the power 4 equals 2 to the power 28. So, therefore, we have s equals 2 to the power s is given by 2 to the power sorry 28 bits because we have 2 to the power 28 blocks in main memory. Therefore, the number of tag, so the number of tag bits in each line is s minus r equals to 28 minus 10 equals to 18 ok.

(Refer Slide Time: 26:20)



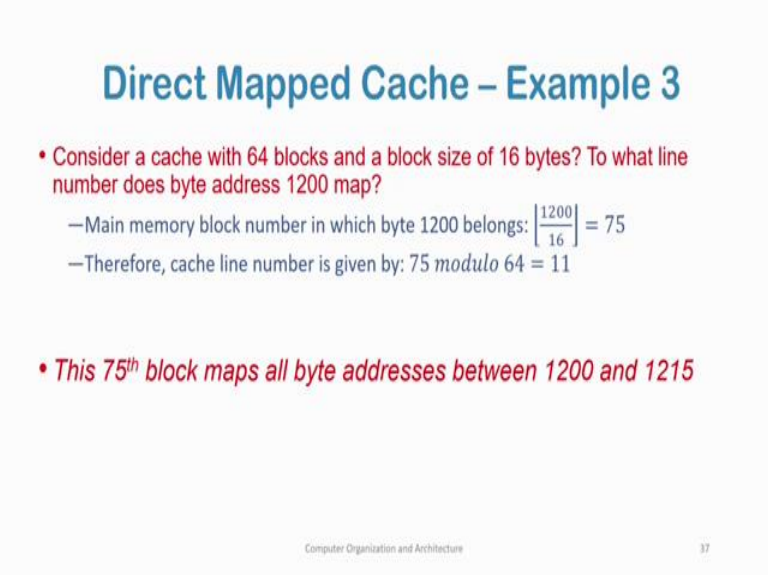
Direct Mapped Cache – Example 2

- Given a 16 KB direct-mapped cache having 4-word lines with word size being 32-bits and a byte-addressable main memory having 32-bit addresses, find the total actual number of bits in the cache?
  - —No. of words in a 16 KB cache = 4K (2^{12})
  - —Line size = 4 words = 2^2 words (= 2^4 bytes; thus, w is 4 bits long)
  - —No. of Lines = $2^{12-2} = 2^{10}$ (thus, r is 10 bits long)
  - —No. of blocks in main memory = $2^{32-4} = 2^{28}$ (thus, s is 28 bits long)
  - —No. of tag bits = $s - r = 28 - 10 = 18$
  - —Each line contains: 4×32 bits of data + 18 tag bits + 1 valid bit = 147 bits
  - —Total number of bits in the 2^{10} available lines: $2^{10} \times 147$ = 147 Kbits = 18.4 KB
- *For this cache, the total actual number of bits in cache (18.4 KB) is about 1.15 times as many bits needed just for data storage (16 KB)*

Computer Organization and Architecture 34

Now, each line therefore, contains 4 into 32 bits of data. So, we have 4 words each containing 32 bits. So, we have 4 in to 32 bits of data plus we have 18 tag bits plus 1 valid bit. So, we have one 47 bits of data in each line. So, the total number of bits in the 2 to the power 10 available lines is given by 2 to the power 10 into 147 equals to 147K bits equals to 18.4 kilobytes. Hence for this cache the total actual number of bits in cache which is 18.4KB is about 1.5 times as many bits needed for data storage which is 16 kilobytes.

(Refer Slide Time: 27:14)



Direct Mapped Cache – Example 3

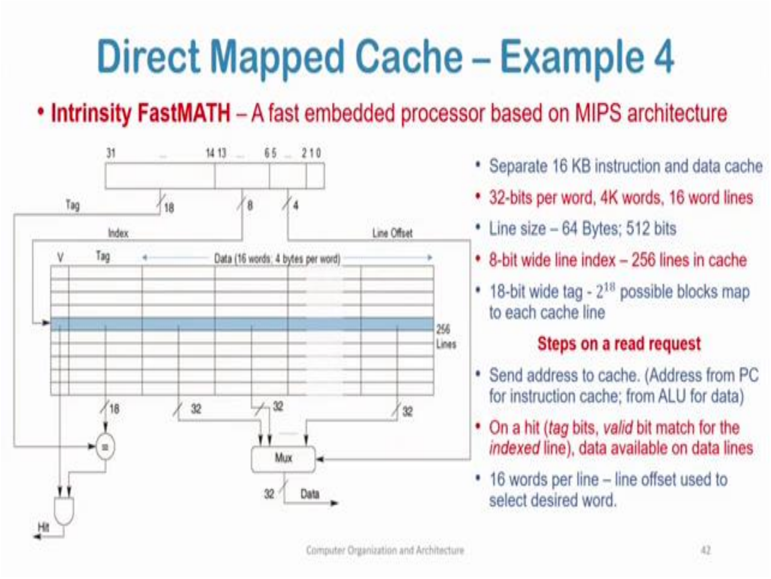
- Consider a cache with 64 blocks and a block size of 16 bytes? To what line number does byte address 1200 map?
  - —Main memory block number in which byte 1200 belongs: $\left\lfloor \frac{1200}{16} \right\rfloor = 75$
  - —Therefore, cache line number is given by: $75 \; modulo \; 64 = 11$
- *This 75th block maps all byte addresses between 1200 and 1215*

Computer Organization and Architecture 37

Now we take a third example: Consider a cache with 64 blocks and a block size of 16 bytes. To what line number does byte address 1200 map? So, the main memory block number in which byte 1200 belongs is given by 1200 divided by 16. Why? We have 16 bytes in each block and the block id is 1200. So, to which block number will this is will this byte 1200 belong it is given by 1200 divided by 16 which is 75.

Now, therefore, the cache line number is given by 75 modulo 64. Why because we have 64 lines in the cache. So, blocks have been mistakenly said we have 64 lines in the cache and therefore, the cache line number is given by 75 modulo 64 which is 11. So, this 75th block of the main memory or this 11th line in the cache will contain all addresses between 1200 and between 1200 and 1215. The line in the cache may so, the 75th block rather of the main memory will contain 1200 to 1215 addresses.

(Refer Slide Time: 28:54)



Direct Mapped Cache – Example 4

- **Intrinsity FastMATH** – A fast embedded processor based on MIPS architecture

- Separate 16 KB instruction and data cache
- 32-bits per word, 4K words, 16 word lines
- Line size – 64 Bytes; 512 bits
- 8-bit wide line index – 256 lines in cache
- 18-bit wide tag - 2^{18} possible blocks map to each cache line

Steps on a read request

- Send address to cache. (Address from PC for instruction cache; from ALU for data)
- On a hit (*tag* bits, *valid* bit match for the *indexed* line), data available on data lines
- 16 words per line – line offset used to select desired word.

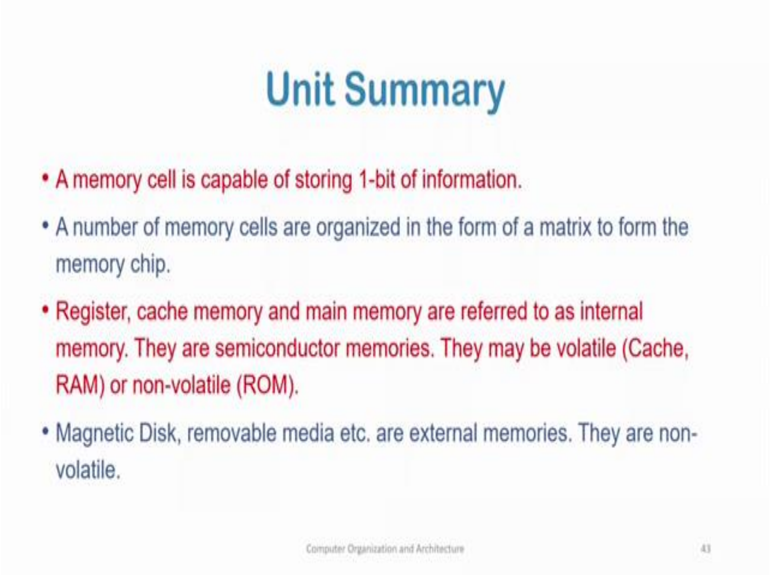
Computer Organization and Architecture 42

As a fourth and last example we take the example of a real word processor which uses direct mapped cache. So, we take the example of in Intrinsity FastMATH processor which is a fast embedded processor based on MIPS architecture. The direct mapped cache organization of this of this processor is shown in the figure here. So, the cache uses separate 16 KB instruction and data caches there is the mem the organization has 16 KB instruction and data caches separate. We have 32 bits per word. So, therefore, we have 4 byte words. We have 4Kwords in the cache and we have 16 word lines. So, each line contains 16 words. So, line size is 64 bytes. So, 16 words, each containing 4 bytes is 64 bytes and 64 bytes or 512 bits. We have a 8 bit wide line index. So, therefore, we have 256 lines in the cache. We have a 18 bit wide tag field. So, 2 to the power 18 possible blocks can map to each cache line ok.

So, we have 2 to the power 18 possible blocks that can map to each cache line. What are the steps to further read request on this? We send the address to cache, either the instruction cache or the data cache. Addresses are sent from the PC for the instruction cache and from the ALU for the data cache. On a hit, that means, the tag bits and valid bits match the tag bits and the valid bits match. On a hit when we have the tag bits and the valid bits matching, the data is made available on the data lines. We have 16 words per line; that means, a line offset 16 words per line.

So, we need to identify which word in the line is required. So, what we have? We have a line offset which is used to select which word in the line is desired by the memory. So, this line offset is used as a selector in a 16 cross 1 mux and we have a 4 bit line access because we have 16 words in the line. And based on this selection mechanism from the mux we get the required data.

(Refer Slide Time: 32:00).



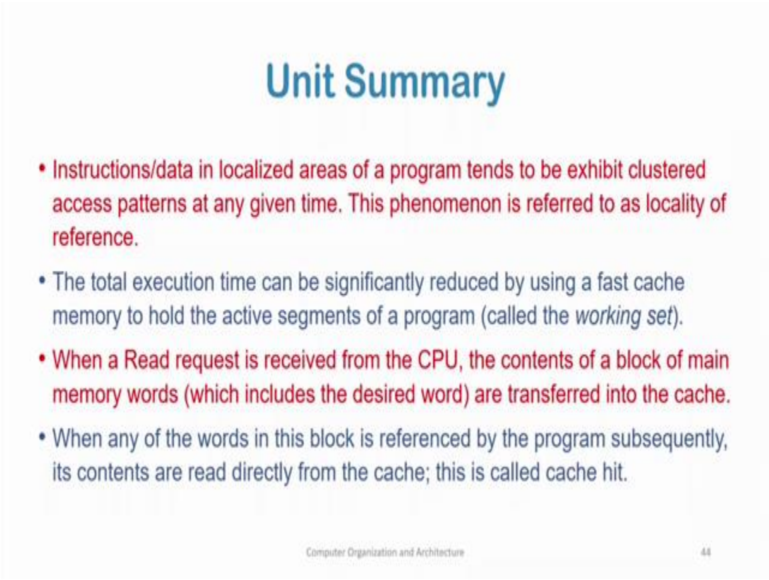
Unit Summary

- A memory cell is capable of storing 1-bit of information.
- A number of memory cells are organized in the form of a matrix to form the memory chip.
- Register, cache memory and main memory are referred to as internal memory. They are semiconductor memories. They may be volatile (Cache, RAM) or non-volatile (ROM).
- Magnetic Disk, removable media etc. are external memories. They are non-volatile.

Computer Organization and Architecture 43

With this we come to the end of this unit the summary. In summary what we studied in the unit is as follows. A main memory cell is capable of storing 1-bit of information. A number of memory cells are organized in the form of a matrix to form the memory chip, Register, cache memory and main memory are referred to as internal or inboard memory. These are semiconductor memories. They may be volatile, for example, for caches and RAMs or non-volatile in case of ROM. Magnetic disk removable media etcetera are external memories. They are non-volatile.

(Refer Slide Time: 32:46)



Unit Summary

- Instructions/data in localized areas of a program tends to be exhibit clustered access patterns at any given time. This phenomenon is referred to as locality of reference.
- The total execution time can be significantly reduced by using a fast cache memory to hold the active segments of a program (called the *working set*).
- When a Read request is received from the CPU, the contents of a block of main memory words (which includes the desired word) are transferred into the cache.
- When any of the words in this block is referenced by the program subsequently, its contents are read directly from the cache; this is called cache hit.

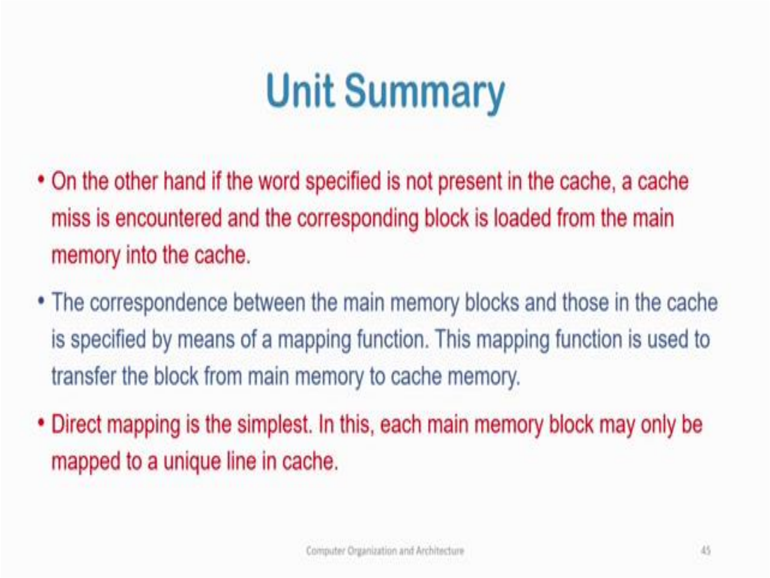
Computer Organization and Architecture 44

Instruction slash data in localized area of a program tends to exhibit clustered access patterns at any given time. This phenomenon is referred to as the locality of reference. The total execution time can be significantly reduced by using a fast cache. So, the total execution time of a program can be significantly reduced by using a fast cache memory to hold active segments of a program which is called the working set in OS parlance.

So, basically, because the memory is slower than the processor, we can to make things faster the processor has to wait for data to come from the memory to make things faster we can put a fast cache, which will hold active segments of the memory. We can only hold the active segment or not the whole program because this fast cache memory is very expensive and we cannot have a very large cache. Also when the when the size of the memory tends to grow the access times also tend to grow.

When a read request is received from the CPU, the contents of a block of main memory are transferred to the cache which includes the desired word. When and we were and we bring a block instead of a single word from the main memory to take advantage of the locality of reference. Due to which subsequent accesses may be near the vicinity of this memory word and therefore, subsequent accesses may result in hits. When any of the words in this block is referenced by the program subsequently its contents are read directly from the cache and this is called cache hit.

(Refer Slide Time: 34:54)



Unit Summary

- On the other hand if the word specified is not present in the cache, a cache miss is encountered and the corresponding block is loaded from the main memory into the cache.
- The correspondence between the main memory blocks and those in the cache is specified by means of a mapping function. This mapping function is used to transfer the block from main memory to cache memory.
- Direct mapping is the simplest. In this, each main memory block may only be mapped to a unique line in cache.

Computer Organization and Architecture 45

On the other hand, if the word specified is not present in a cache, a cache miss is encountered and the corresponding block is loaded from the main into the cache. The correspondence between the main memory blocks and those of the cache is specified by means of a mapping function. This mapping function is used to transfer the block from main memory to cache memory. Direct mapping is the simplest. In this each memory block can only be mapped to a unique line in the cache. There are other more complex forms of mapping as fully associative mapping and set associative mapping which we will study later.

With this we end the unit 1 of the module memory system.