Control Unit

The control unit of a processor performs two tasks:

- It causes the processor to step through a series of micro-operations in the proper sequence, based on the program being executed
- It generates the control signals that cause each micro-operation to be executed

These control signals cause the opening and closing of logic gates, resulting in the transfer of data to and from registers and the operation of the ALU. Their are broad categories are

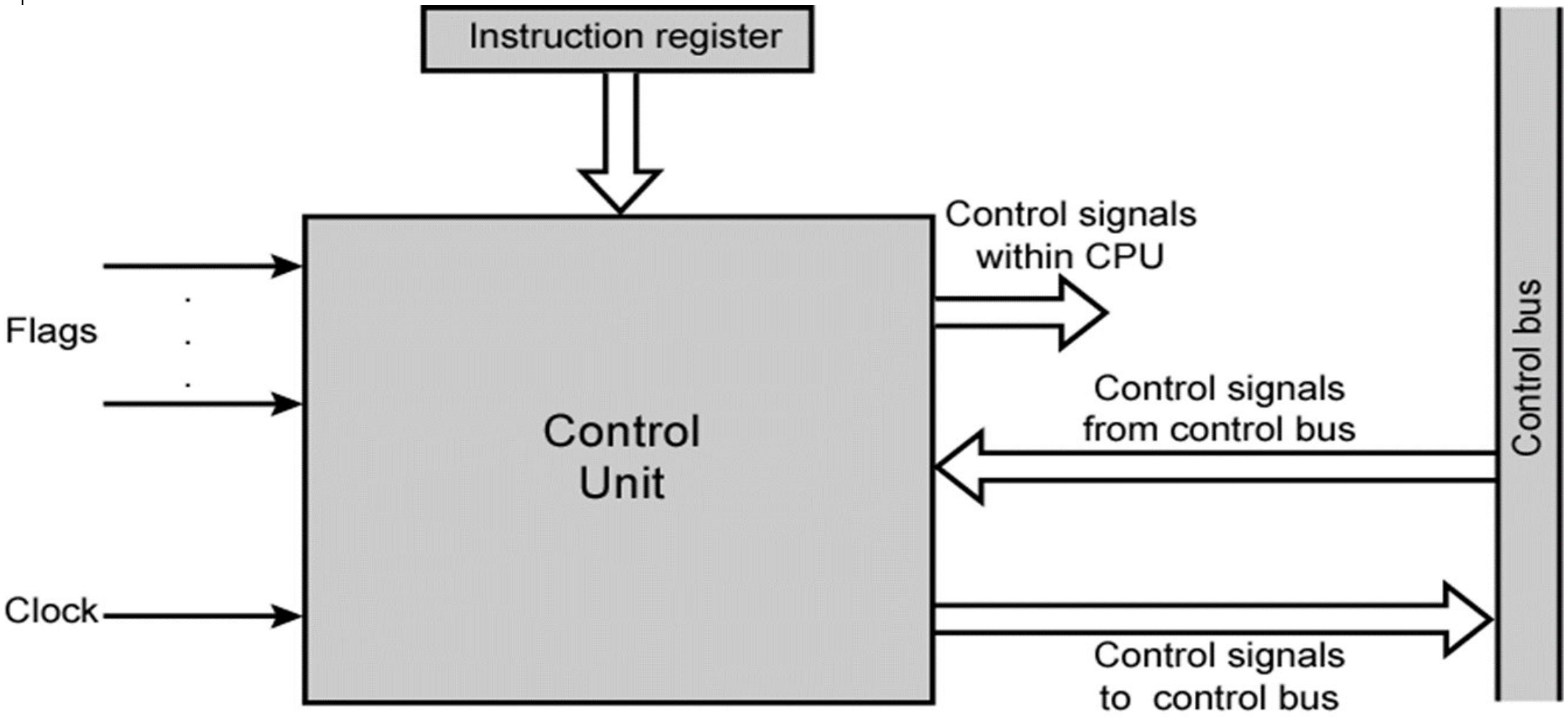
1. Hardwired
2. Micro-programmed

Control Unit (Hardwired)

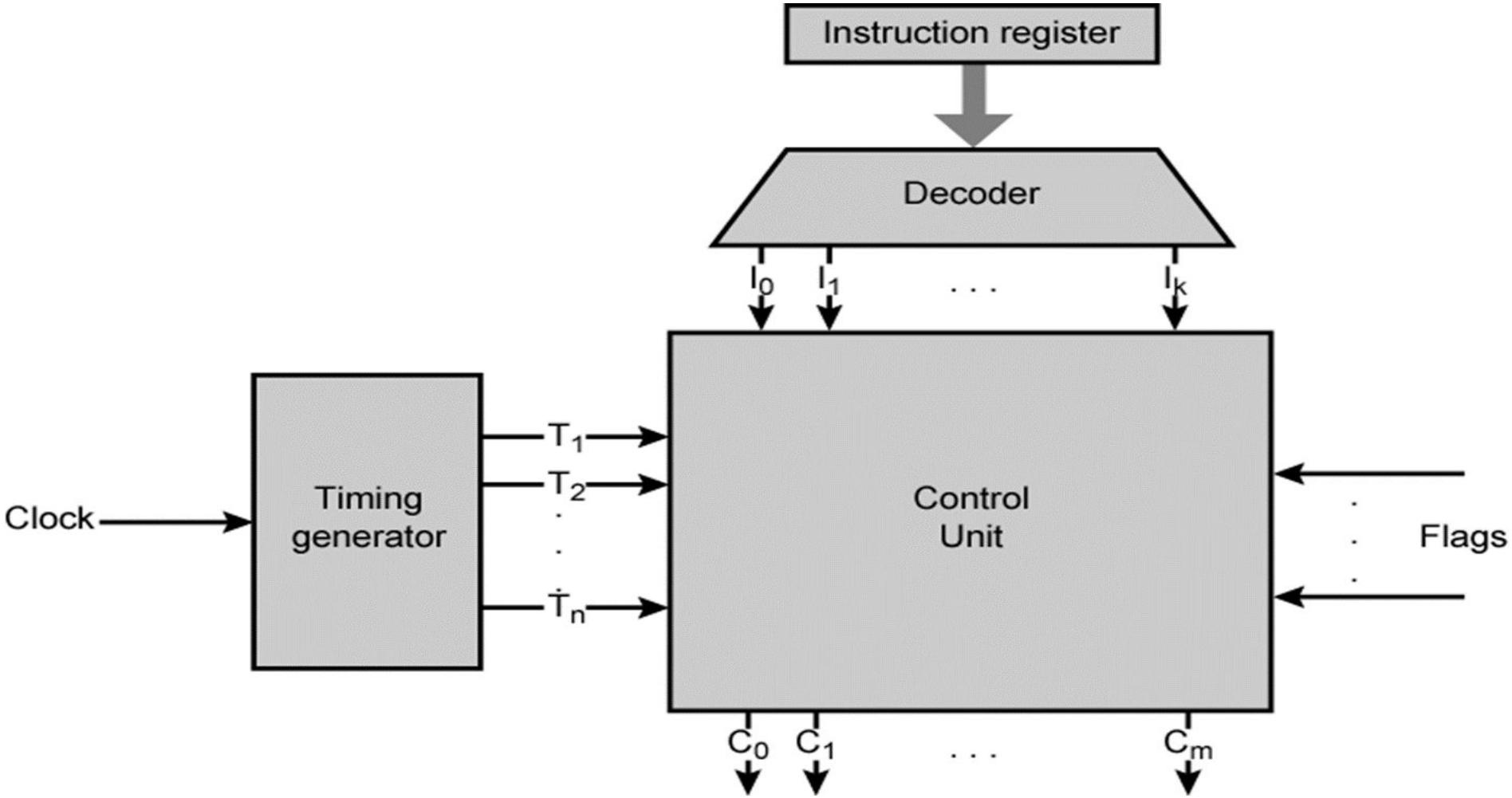
Hardwired implementation of control unit –

- the control unit is a combinatorial circuit.
- the input logic signals to this combinational circuit are governed by the current machine instruction
- the output of this combinational circuit is now a set of output control signals.

Control Unit (Hardwired)



Control Unit (Hardwired) with Decoded Input



Problems With Hard Wired Designs

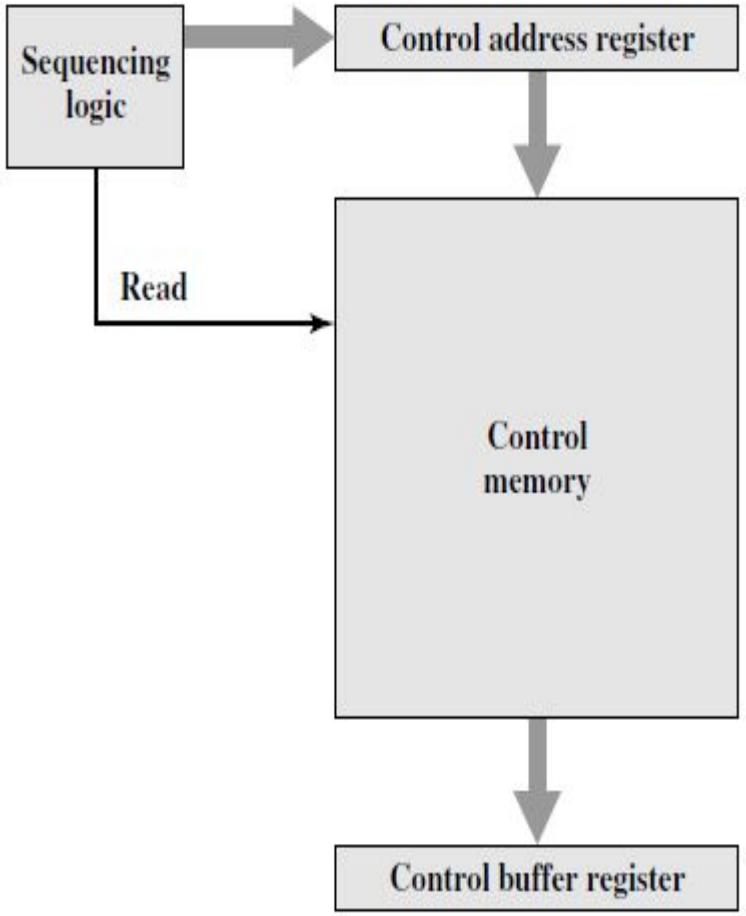
- Complex sequencing & micro-operation logic
- Difficult to design and test
- Inflexible design
- Difficult to add new instructions

Control Unit (Micro-programmed)

- 2nd category - **micro-programmed control**

- Uses sequences of micro-instructions to control complex instructions
- Sequence of instructions is a micro-program
- This introduces a microprogramming language
- Also called as firmware
- Firmware is midway between hardware and software

Microprogrammed Control Unit - Organization



- control memory stores the set of microinstructions
- control address register contains the address of next instruction to be read
- the sequencing logic loads the control address register and issues a read command
- when a microinstruction is read from the control memory, it is transferred to the control buffer register

Microinstruction Types

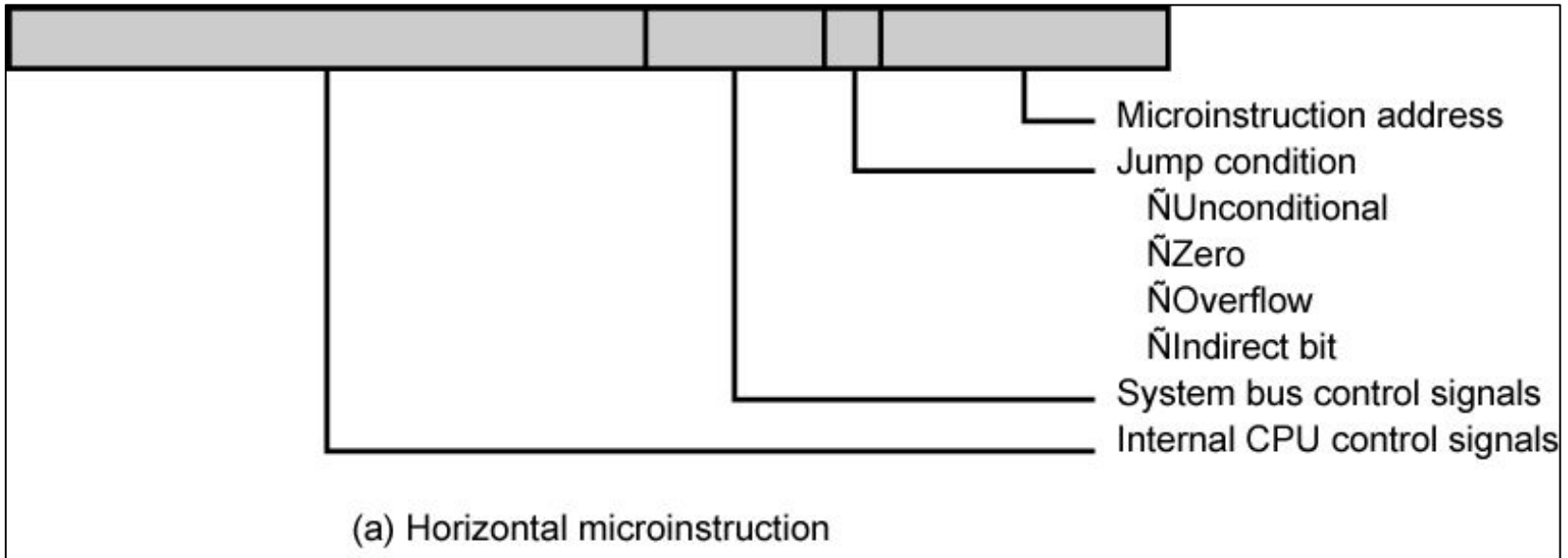
Vertical micro-programming

- Width is narrow
- n control signals encoded into $\log_2$ n bits
- Limited ability to express parallelism
- Considerable encoding of control information requires external memory word decoder to identify the exact control line being manipulated

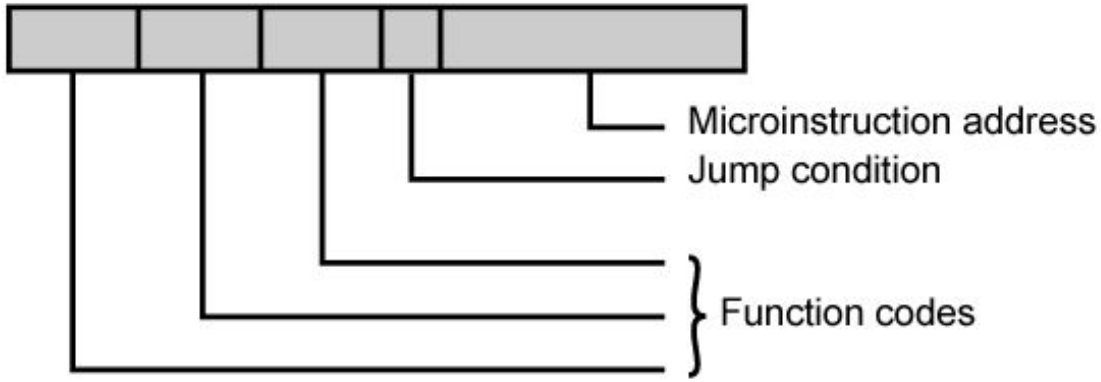
Horizontal micro-programming

- Wide memory word
- High degree of parallel operations possible
- Little encoding of control information

Microinstruction formats

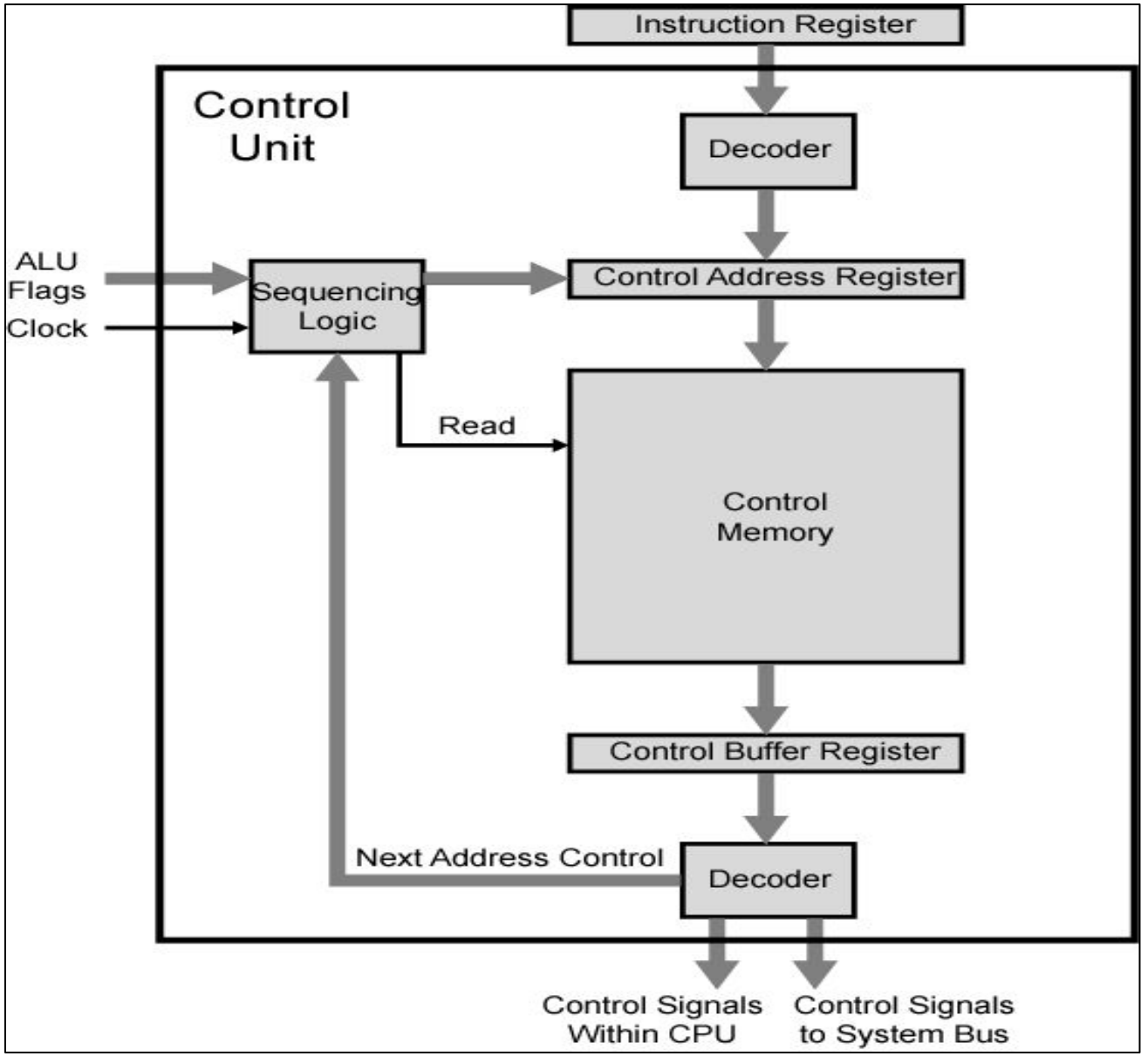


(a) Horizontal microinstruction



(b) Vertical microinstruction

Functioning of Microprogrammed Control Unit



Functioning of Microprogrammed Control Unit

1. To execute a microinstruction, the sequencing logic unit issues a READ command to the control memory
2. The word whose address is specified in the control address register is read into the control buffer register
3. The content of the control buffer register generates control signals and next address information for the sequencing logic unit
4. The sequencing logic unit loads a new address into the control address register based on the next-address information from the control buffed register and the ALU flags

All this happens in one clock pulse.

Functioning of Microprogrammed Control Unit contd…

The figure shows 2 modules named decoders.

The upper decoder translates the opcode of the IR into a control memory address.

The lower decoder is not used for horizontal microinstructions but is used for vertical microinstructions.

Micro-programmed Control Unit

Advantages and Disadvantages of Microprogramming –

- Simplifies design of control unit
- Cheaper
- Less error-prone
- Slower

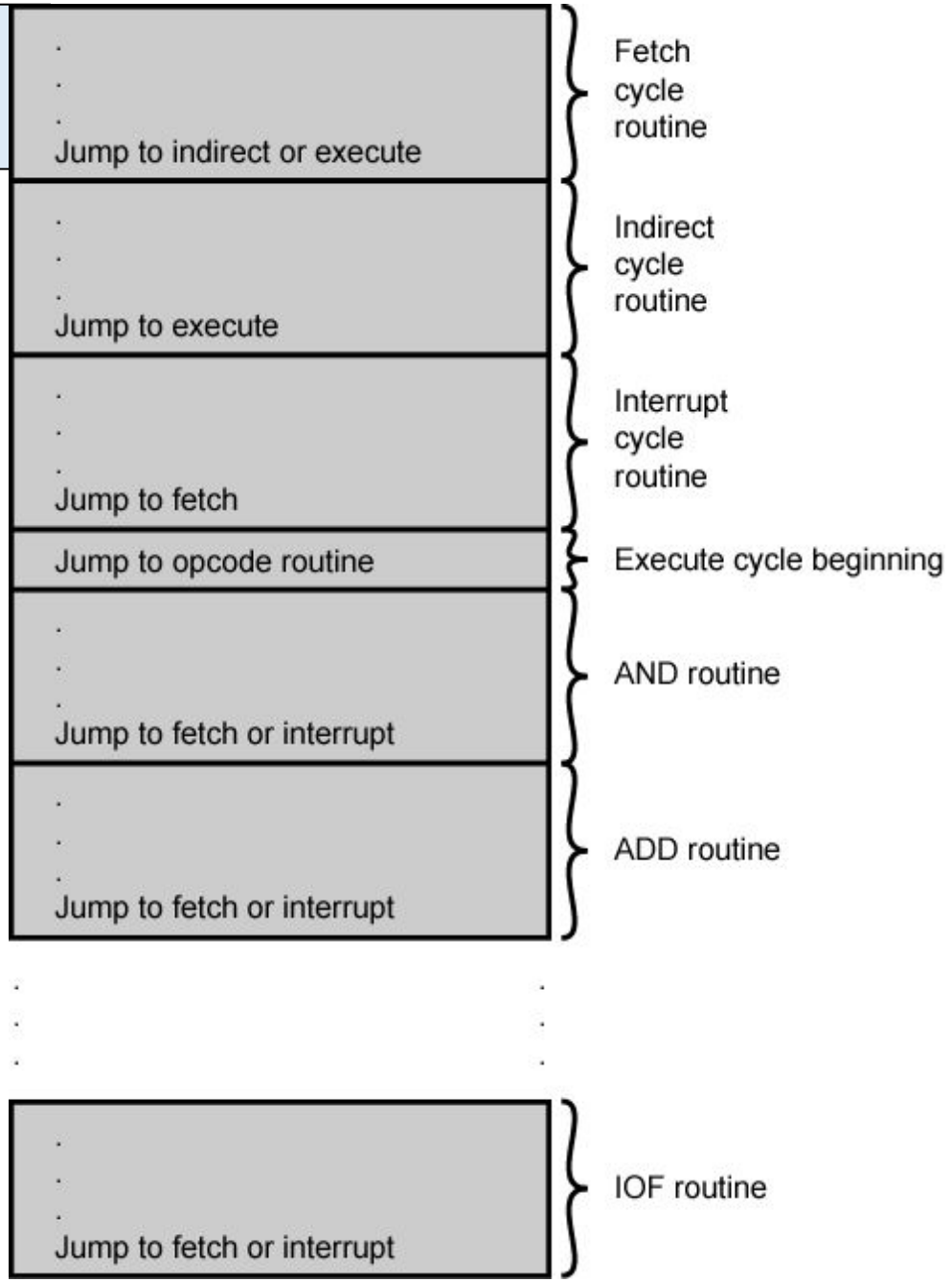
Tasks performed –

- **Microinstruction sequencing**: get the next instruction from the control memory
- **Microinstruction execution**: generate the control signals needed to execute the microinstruction

Both these tasks are considered together, because both affect the format of the microinstruction and the timing of the control unit.

Organization of Control Memory

- defines the sequence of micro-operations to be performed during each cycle - fetch, indirect, execute, interrupt.
- specifies the sequence of all these cycles



Microinstruction Sequencing

Design Concerns:

1. size of microinstruction - to minimize the size of control memory
2. address generation time - fast execution of microinstruction

   In executing a microprogram, the address of the next microinstruction to be executed is in one of these categories:

   - Determined by instruction register - occurs only once, just after an instruction is fetched
   - Next sequential address - most common in most designs
   - Branch - conditional and unconditional, are necessary part of a microprogram. One out of three or four microinstruction could be a branch.

Sequencing Techniques

A control memory address must be generated for the next instruction should be based on - current microinstruction

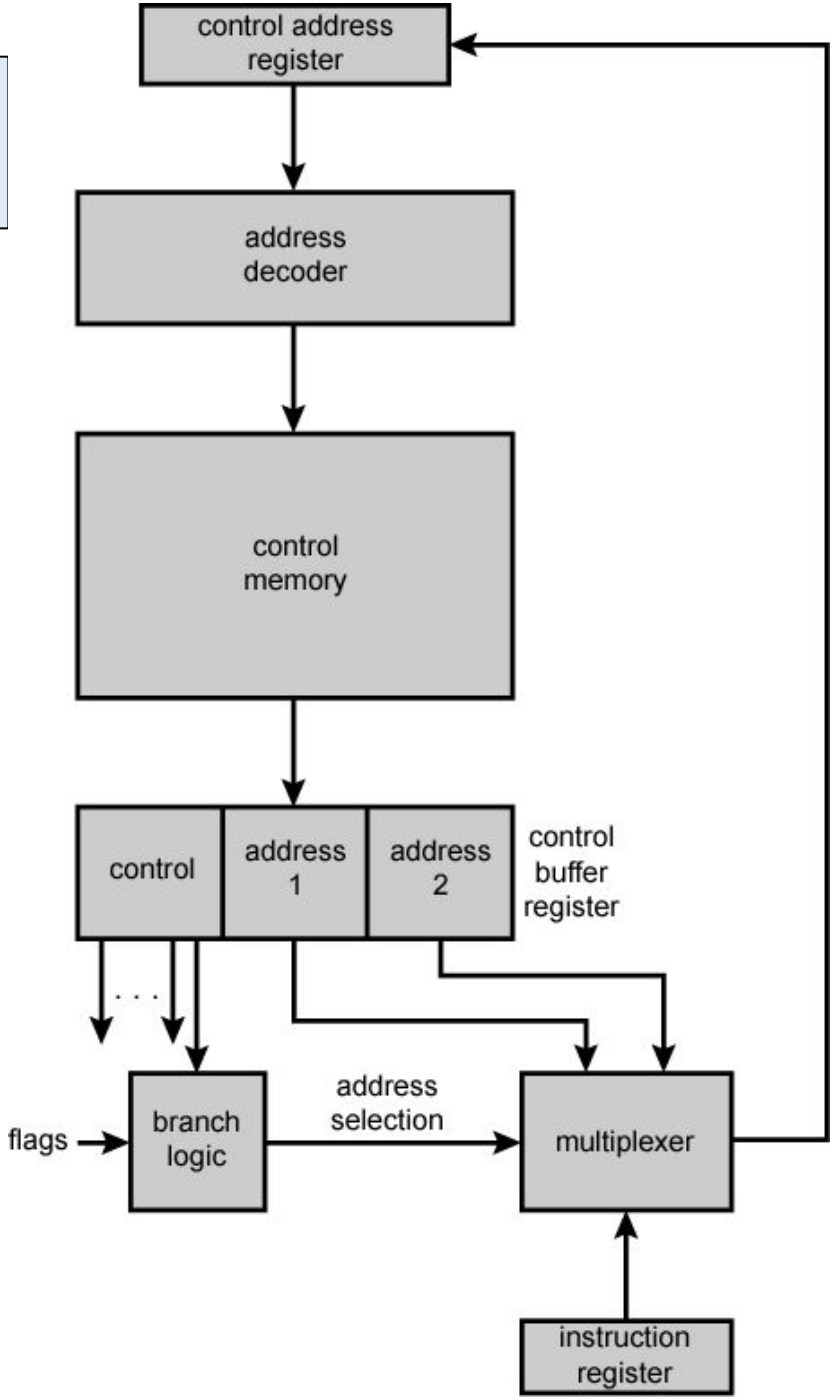
- condition flags
- contents of instruction register

Hence, 3 general categories for branch control logic are given below

1. two address fields
2. single address field
3. variable format

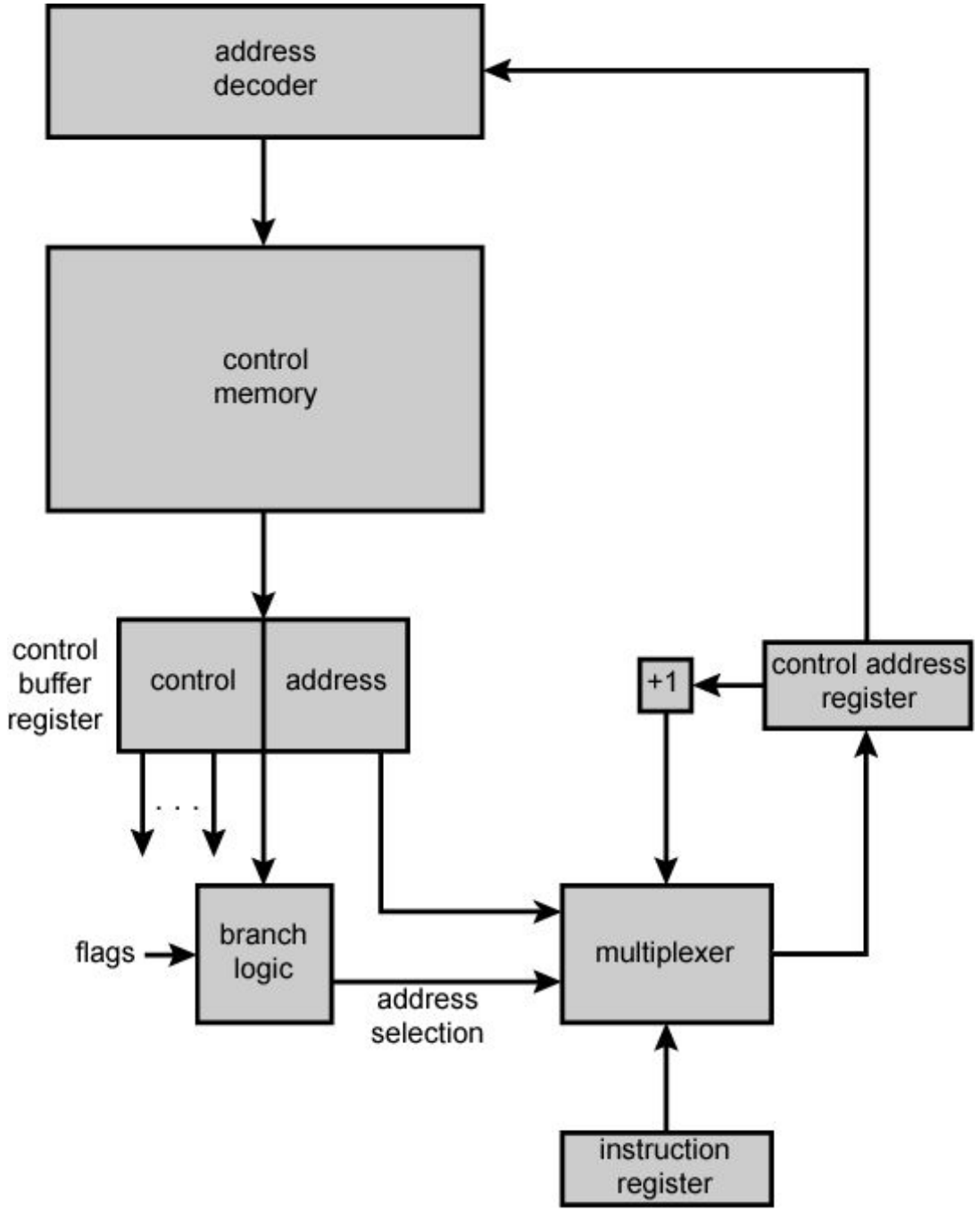
Branch Control Logic: Two Address Fields

- simplest approach - provide two address fields in each microinstruction
- multiplexer serves as a destination for both address fields plus the instruction register
- based on the address selection input, multiplexer transmits either the opcode or one of the two addresses to the control address register (CAR)
- the CAR is subsequently decoded to produce the next microinstruction address
- the branch logic module provides the address selection signals. Inputs to this module are the control unit flags plus the control portion of the microinstruction
- though simple this approach needs more bits in the microinstruction than other approaches



Branch Control Logic: Single Address Field

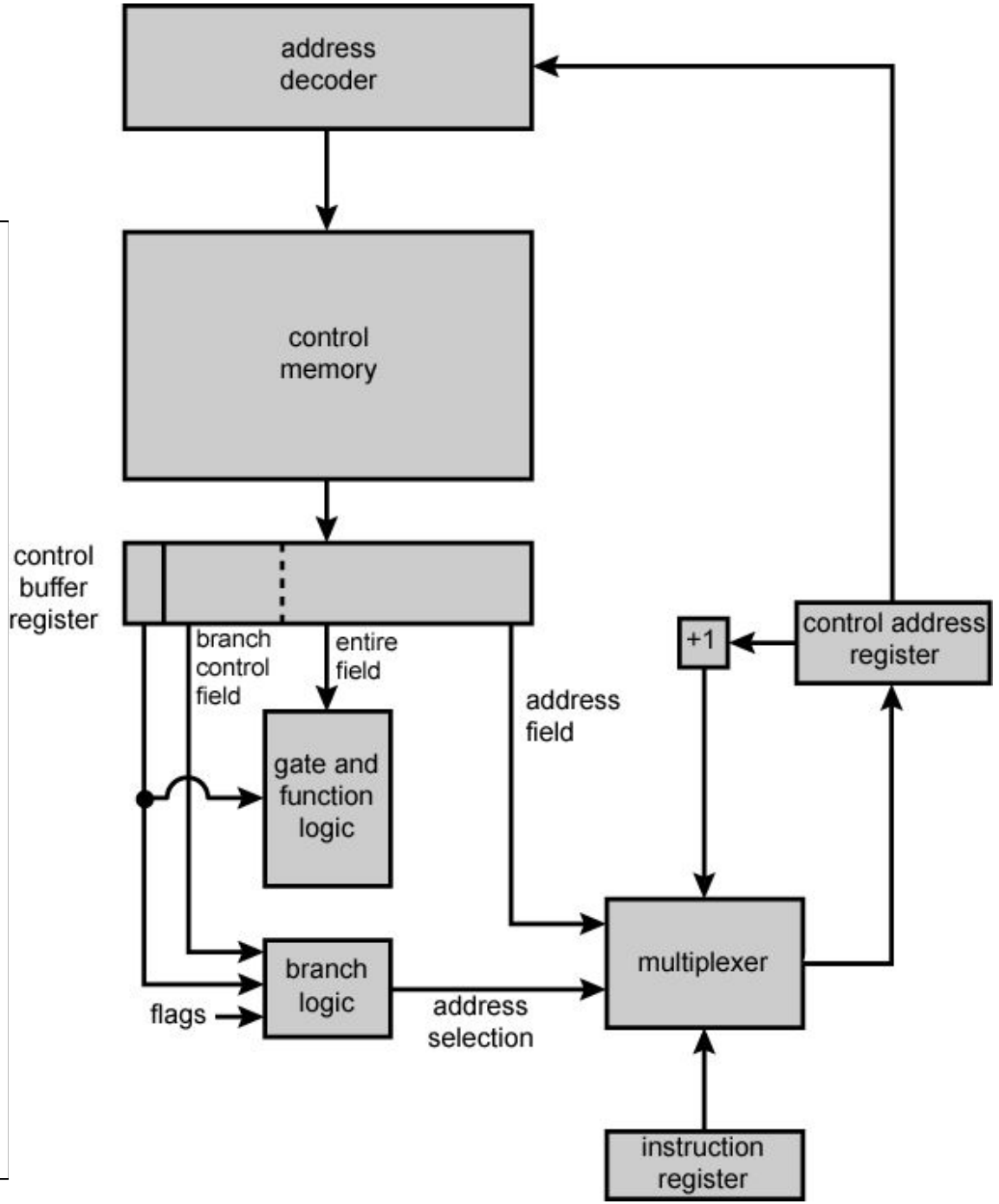
- a common approach is to have a single address field
- the options for next address are
  - address field
  - instruction register code
  - next sequential address
- the address selection signals determine which option is selected
- this approach reduces the number of address field to one
- thus, there is some inefficiency in the microinstruction coding scheme



Branch Control Logic: Variable Format

- this approach provides for two entirely different microinstruction formats
- one bit designates which format is being used
- in one format, the remaining bits are used to activate control signals. The next address is either the next sequential address or an address derived from the instruction register
- in other format, some bits drive the branch logic module and the remaining bits provide the address. Here either a conditional or unconditional branch is being specified

One disadvantage is that one entire cycle is consumed with each branch microinstruction. With the other approaches, address generation occurs as part of the same cycle as control signal generation, minimizing control memory accesses.



Address Generation

- explicit techniques - address is explicitly available in the microinstruction
- implicit techniques - require additional logic to generate the address

Explicit	Implicit
Two-field	Mapping
Unconditional Branch	Addition
Conditional branch	Residual control

Microinstruction Execution

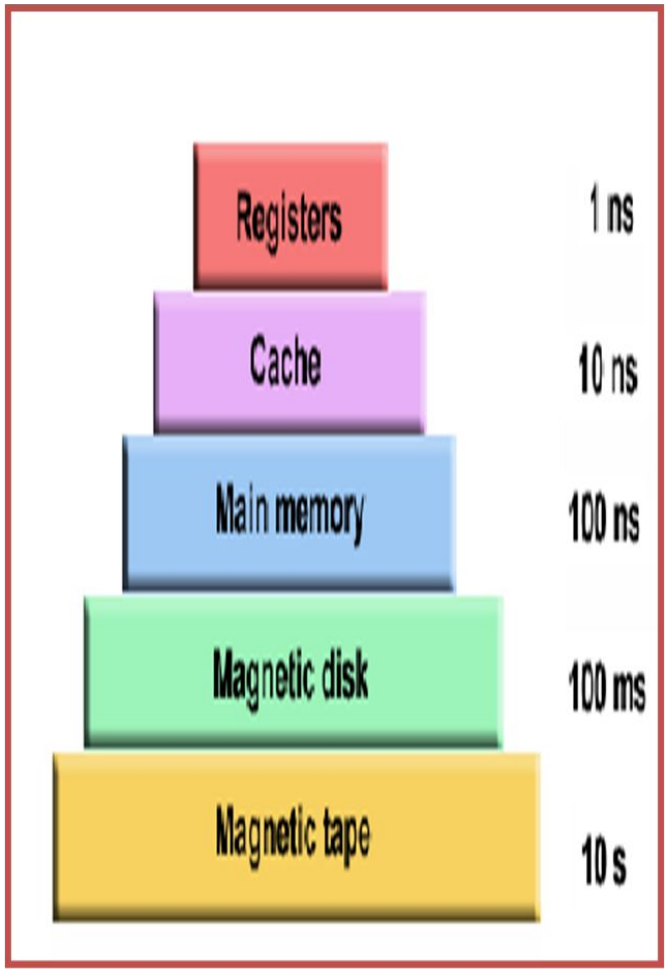
- The cycle is the basic event
- Each cycle is made up of two events
  - Fetch
    - Determined by generation of microinstruction address
  - Execute
    - effect is to generate control signals,
    - some control points internal to processor,
    - rest go to external control bus or other interface

Key Characteristics of Computer Memory Systems

Location Internal (e.g. processor registers, cache, main memory) External (e.g. optical disks, magnetic disks, tapes) **Capacity** Number of words Number of bytes **Unit of Transfer** Word Block **Access Method** Sequential Direct Random Associative	**Performance** Access time Cycle time Transfer rate **Physical Type** Semiconductor Magnetic Optical Magneto-optical **Physical Characteristics** Volatile/nonvolatile Erasable/nonerasable **Organization** Memory modules

Characteristics of Memory Systems: Location

- **Internal**
  - Processor Registers
  - Cache Memory
  - Main Memory
- **External**
  - Optical Disks
  - Magnetic Tape
  - Magnetic Disks



Location: Refers to whether memory is internal and external to the computer.
Internal memory is often equated with main memory Processor requires its own local memory, in the form of registers. Cache is another form of internal memory.
External memory consists of peripheral storage devices that are accessible to the processor via I/O controllers

Characteristics of Memory Systems: Capacity

Capacity: Memory is typically expressed in terms of bytes

- Number of words (Word size)
  - The natural unit of organization
  - Common length-8,16 or 32 bits
  - Internal memory described in words
- Number of Bytes
  - 1 byte is 8 bits
  - External memory described in bytes

Characteristics of Memory Systems :Unit of transfer

- **For Internal memory**

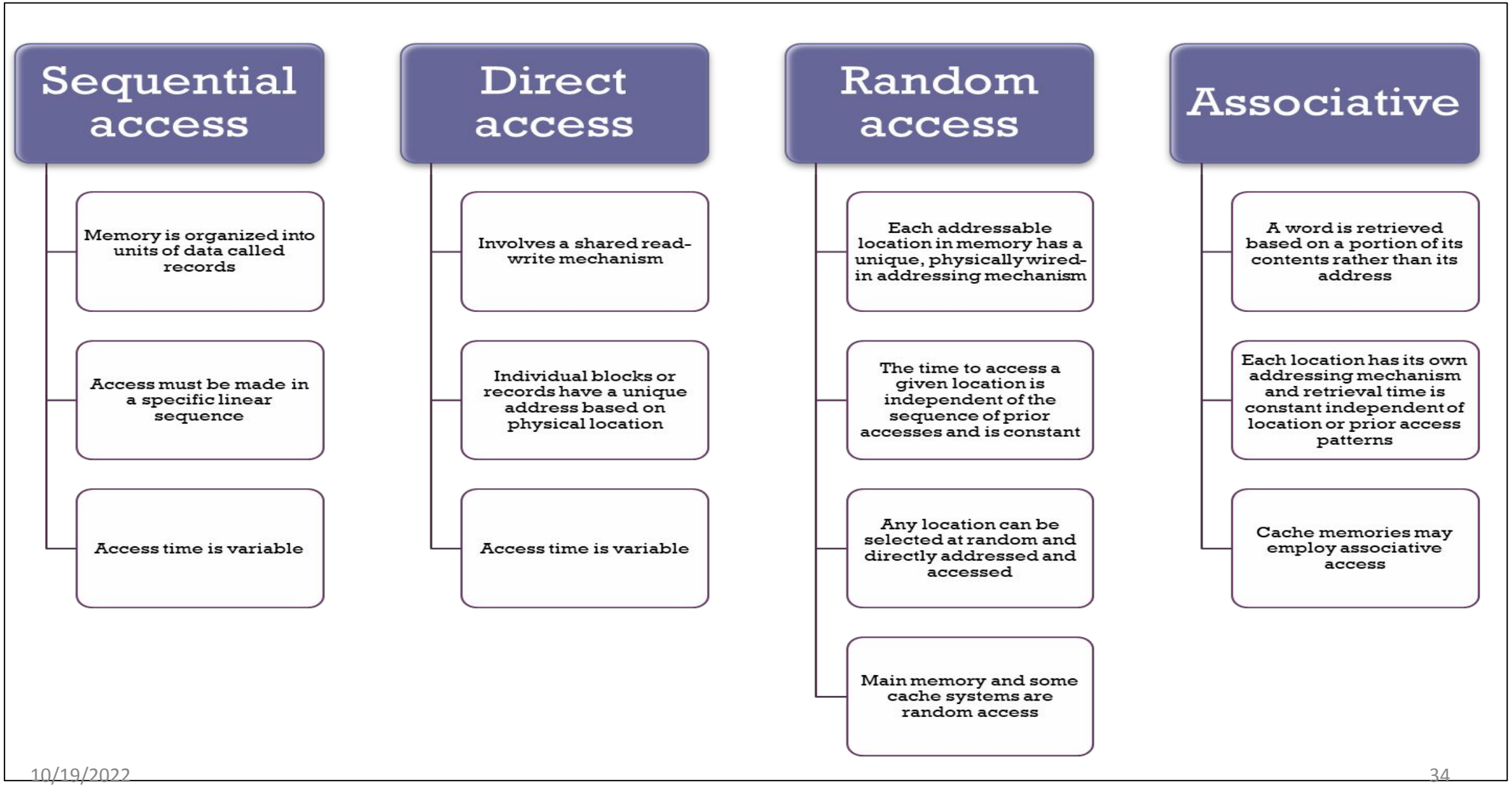
–Unit of transfer is equal to the number of electrical lines into and out of the memory module. 8086 processor has 16 bit unit of transfer. This may be equal to the word length, but is often larger, such as 64, 128, or 256 bits.

–Unit of transfer: For main memory, this is **the number of bits read out of or written into memory at a time**

- **For External Memory**

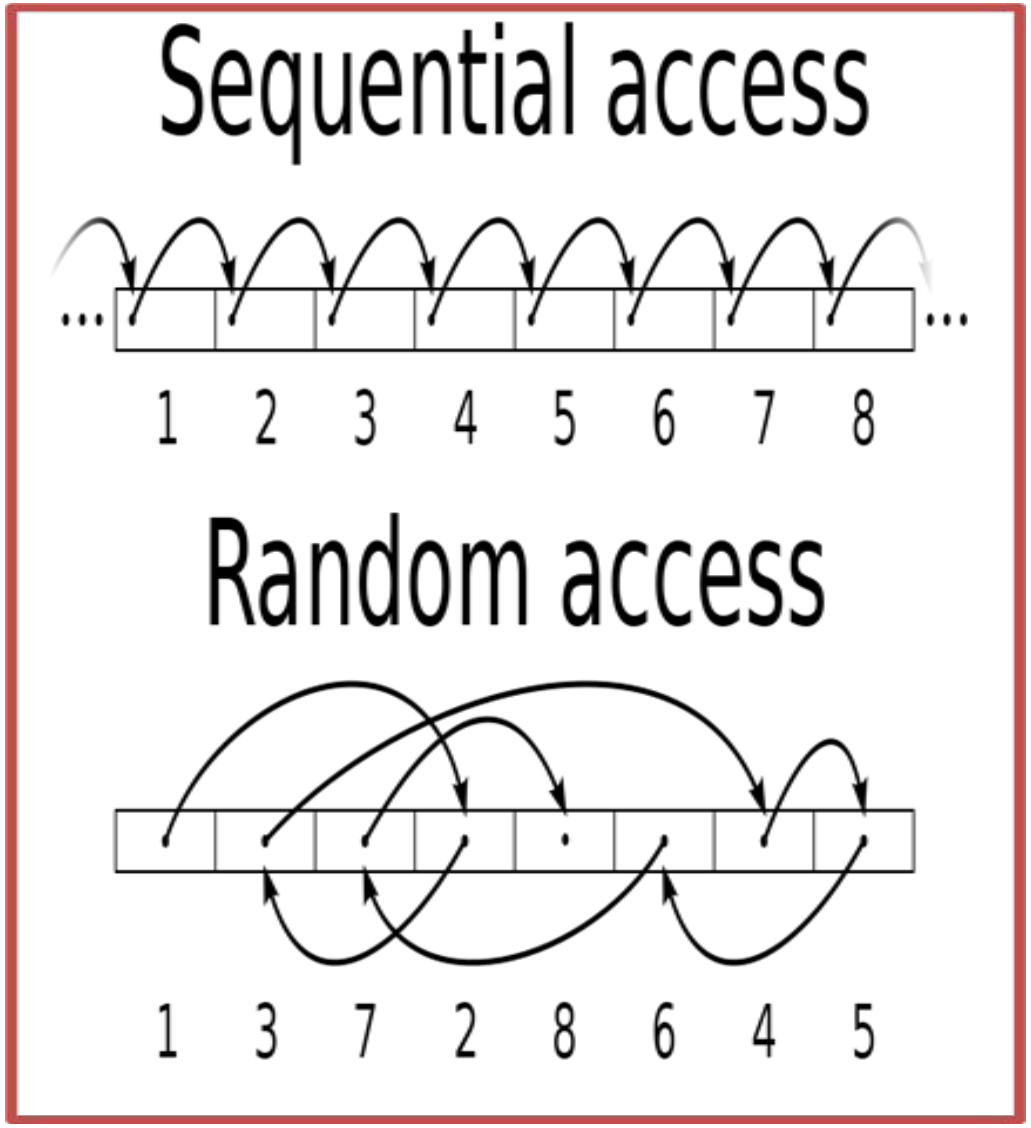
–Usually a block which is much larger than a word

Characteristics of Memory Systems :Access Methods



Characteristics of Memory Systems :Access Methods

- **Sequential**
  - Start at the beginning and read through in order
  - Access time depends on location of data and previous location.The time to access an arbitrary record is highly variable
  - e.g. **Tape**
- **Direct**
  - Individual blocks have unique address
  - Access is by jumping to vicinity plus sequential search
  - Access time depends on location and previous location
  - e.g. **Disk**



Characteristics of Memory Systems :Access Methods

- **Random**
  - Individual addresses identify locations exactly
  - Access time is independent of location or previous access
  - e.g. **RAM**
- **Associative**
  - Data is located by a comparison with contents of a portion of the store.
  - Access time is independent of location or previous access.
  - e.g. **Cache**

Characteristics of Memory Systems :Performance

Access time (latency)

- Time between "Requesting data and getting it".
- Random-access memory: **Predictable**
  = 1/cycle time
- Non-random-access memory: Not Predictable

=Tn=Ta+(N/R)

Where Tn- Average time to read and write N bits

Ta-Average access time, N-No. of bits, R-Transfer rate

- **Memory cycle time**
  - This concept is primarily applied to random-access memory (RAM)
  - Access time + Time required before a second access can commence
- **Transfer rate:**
  - Rate at which data can be transferred into or out of a memory unit

Characteristics of Memory Systems :Performance

The two most important characteristics of memory

Three performance parameters are used:

Access time (latency)

- For random-access memory it is the time it takes to perform a read or write operation
- For non-random-access memory it is the time it takes to position the read-write mechanism at the desired location

Memory cycle time

- Access time plus any additional time required before second access can 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

- 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)

Characteristics of Memory Systems :Physical Type and Physical characteristics

- Semiconductor
- Magnetic surface
- Optical
- Volatile
- Non Volatile

Memory Type	Category	Erasure	Write Mechanism	Volatility
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile
Programmable ROM (PROM)			Electrically	
Erasable PROM (EPROM)	Read-mostly memory	UV light, chip-level		
Electrically Erasable PROM (EEPROM)		Electrically, byte-level		
Flash memory		Electrically, block-level		

Memory Hierarchy

Design constraints on a computer's memory can be summed up by three questions:

- How much, how fast, how expensive

There is a trade-off among capacity, access time, and cost

- Faster access time, greater cost per bit
- Greater capacity, smaller cost per bit
- Greater capacity, slower access time

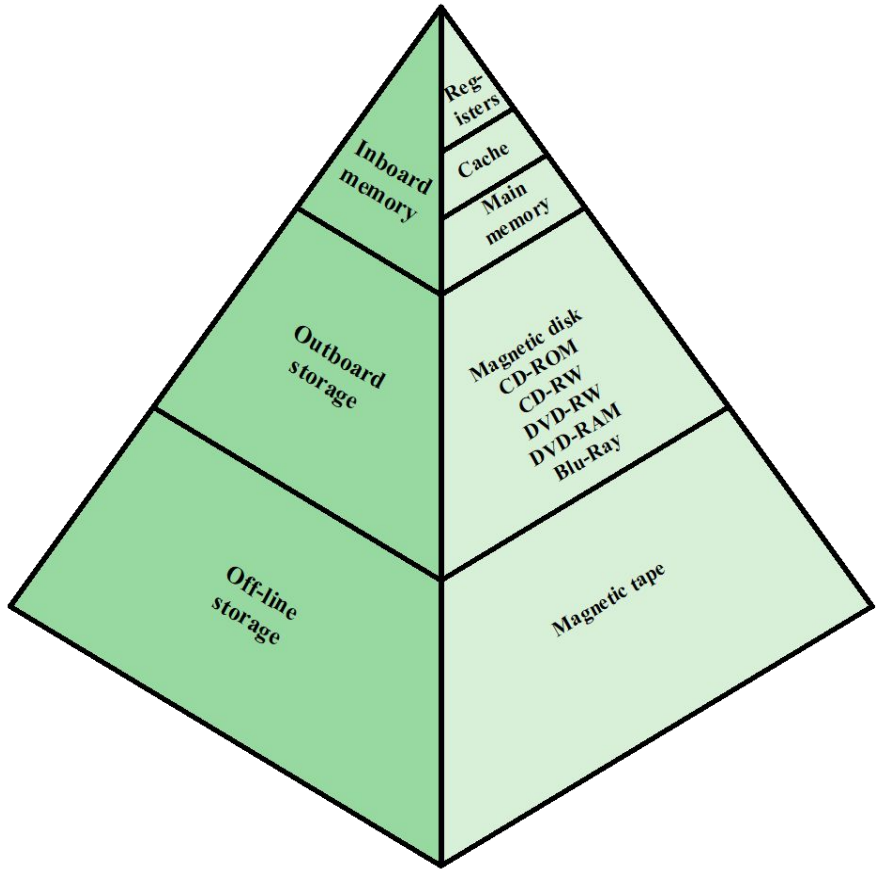
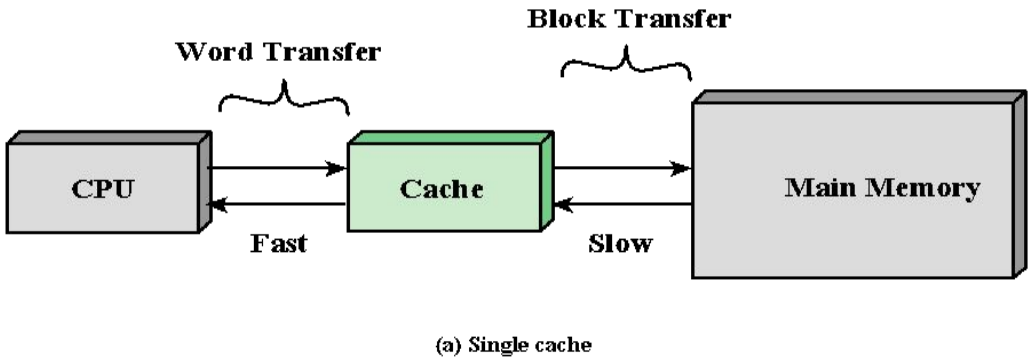


Figure 4.1 The Memory Hierarchy

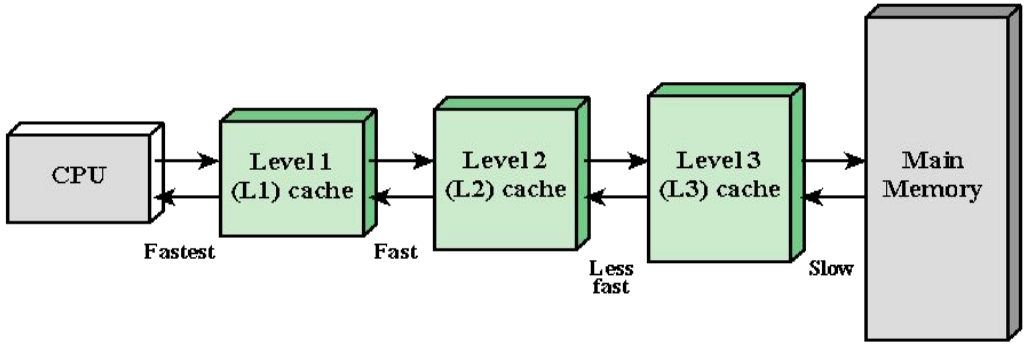
Cache Memory Principles

The concept is illustrated in Figure 4.3a. There is a relatively large and slow main memory together with a smaller, faster cache memory. The cache contains a copy of portions of main memory.

Figure 4.3b depicts the use of multiple levels of cache. The L2 cache is slower and typically larger than the L1 cache, and the L3 cache is slower and typically larger than the L2 cache.



(a) Single cache



(b) Three-level cache organization

Figure 4.3 Cache and Main Memory

Cache/ Main Memory Structure

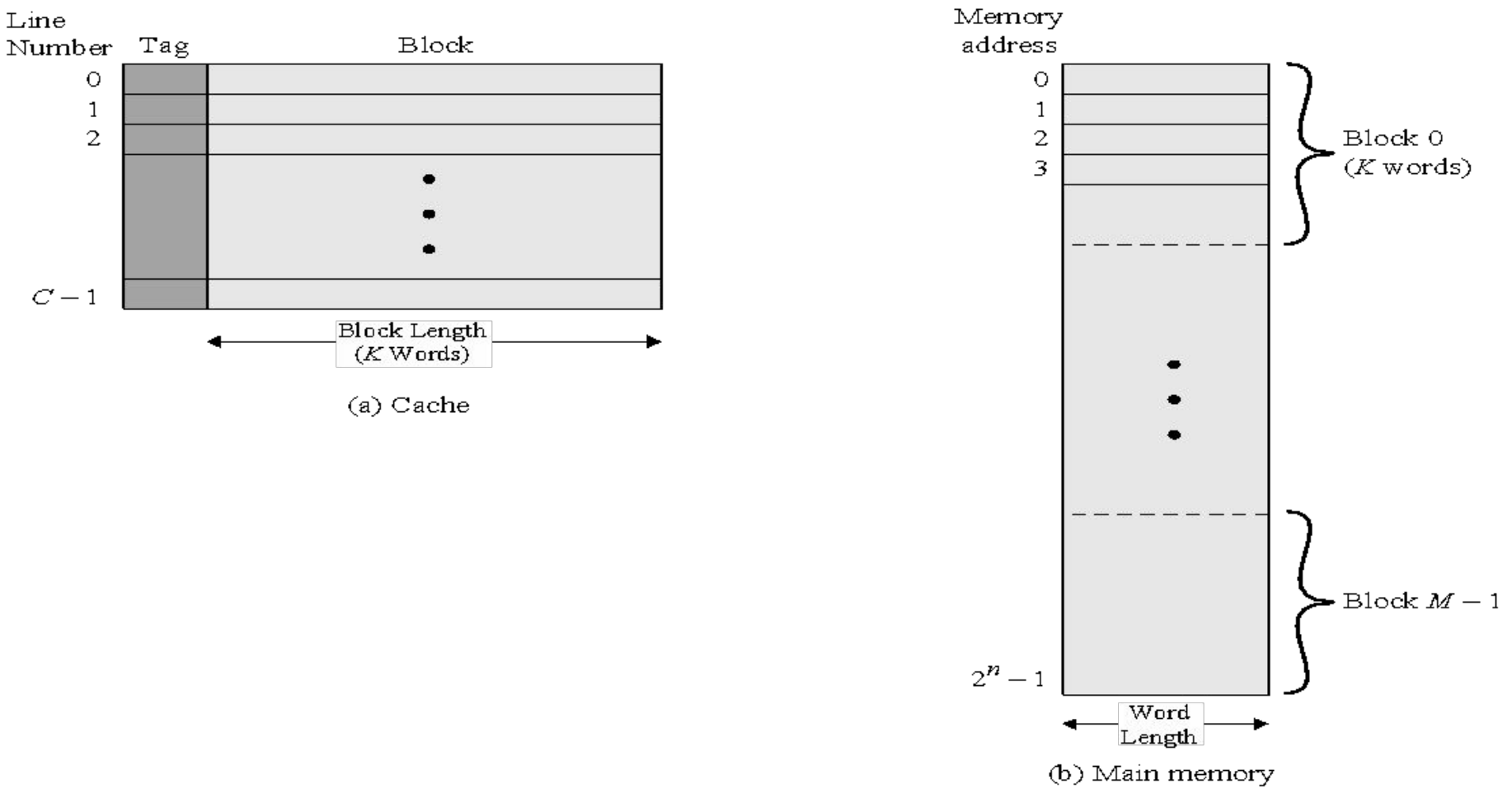


Figure 4.4 Cache/Main-Memory Structure

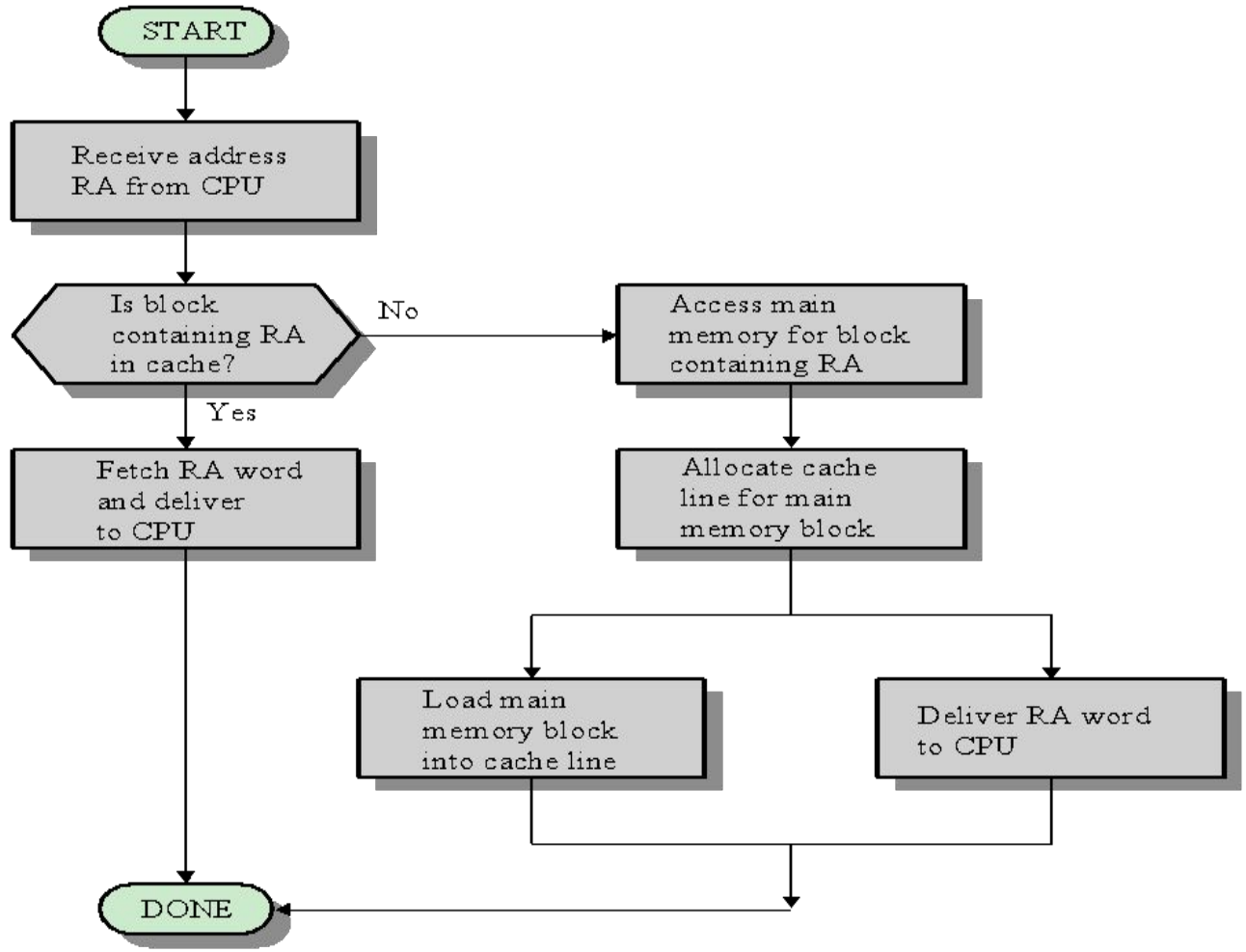


Figure 4.5 Cache Read Operation

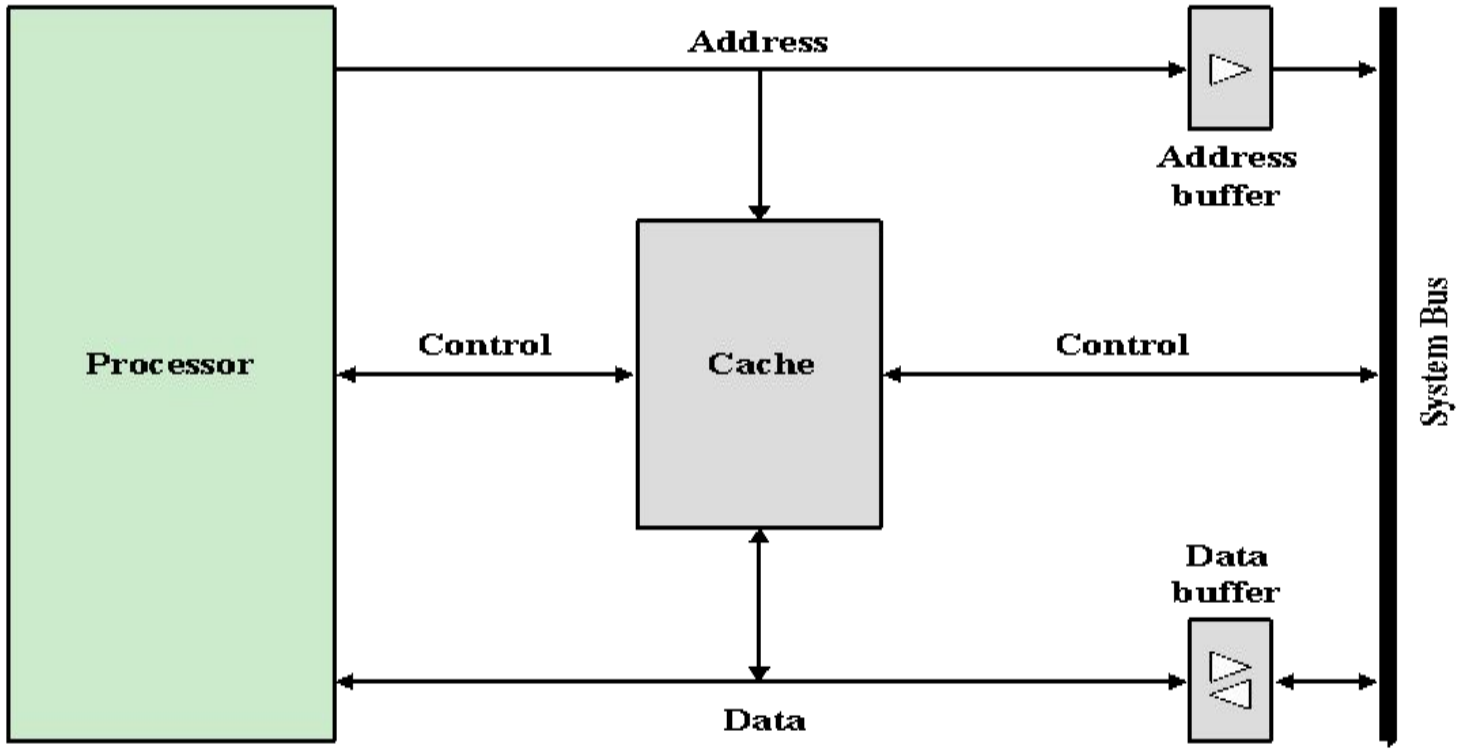


Figure 4.6 Typical Cache Organization

Elements of Cache Design

Cache Addresses	**Write Policy**
Logical	Write through
Physical	Write back
Cache Size	**Line Size**
Mapping Function	**Number of caches**
Direct	Single or two level
Associative	Unified or split
Set Associative	
Replacement Algorithm	
Least recently used (LRU)	
First in first out (FIFO)	
Least frequently used (LFU)	
Random	

Table 4.2
Elements of Cache Design

Caches

Cache Addresses

Virtual Memory

- Virtual memory
  - Facility that allows programs to address memory from a logical point of view, without regard to the amount of main memory physically available
  - When used, the address fields of machine instructions contain virtual addresses
  - For reads to and writes from main memory, a hardware memory management unit (MMU) translates each virtual address into a physical address in main memory

Cache Address

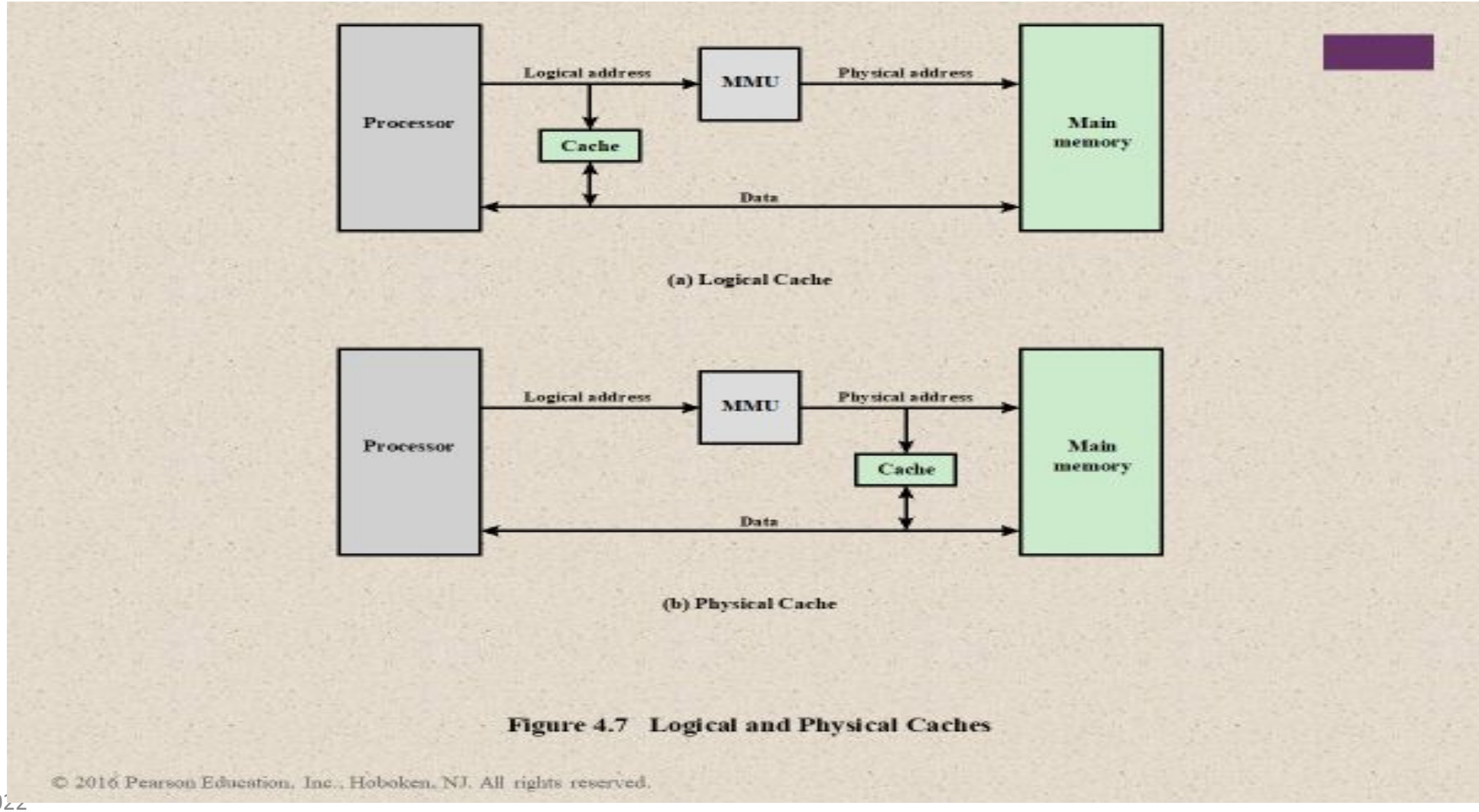


Figure 4.7 Logical and Physical Caches

Mapping Function

Mapping Function

- Because there are fewer cache lines than main memory blocks, an algorithm is needed for mapping main memory blocks into cache lines
- Three techniques can be used:

Direct

- The simplest technique
- Maps each block of main memory into only one possible cache line

Associative

- Permits each main memory block to be loaded into any line of the cache
- The cache control logic interprets a memory address simply as a Tag and a Word field
- To determine whether a block is in the cache, the cache control logic must simultaneously examine every line's Tag for a match

Set Associative

- A compromise that exhibits the strengths of both the direct and associative approaches while reducing their disadvantages

Direct Memory Mapping Function

- Cache of 64kByte
- Cache block of 4 bytes
  - i.e. cache is 16k (2^{14}) lines of 4 bytes
- 16MBytes main memory
- 24 bit address
  - (2^{24}=16M)

- Each block of main memory maps to only one cache line
  - i.e. if a block is in cache, it must be in one specific place
- Address is in two parts
- Least Significant w bits identify unique word
- Most Significant s bits specify one memory block
- The MSBs are split into a cache line field r and a tag of s-r (most significant)

Direct Memory Mapping Function

Direct Mapping
Address Structure

Tag s-r	Line or Slot r	Word w
8	14	2

- 24 bit address
- 2 bit word identifier (4 byte block)
- 22 bit block identifier
  - 8 bit tag (=22-14)
  - 14 bit slot or line
- No two blocks in the same line have the same Tag field
- Check contents of cache by finding line and checking Tag

Direct Memory Mapping Function

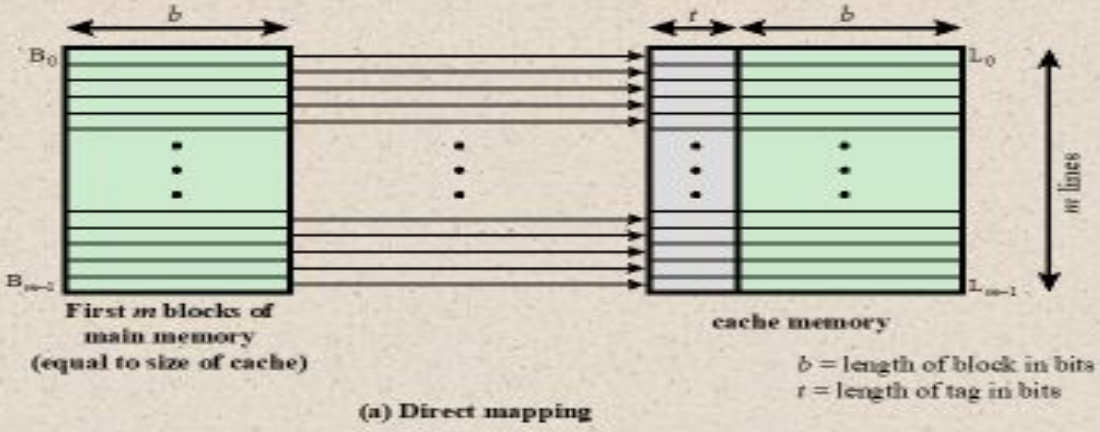


Figure 4.8 Mapping From Main Memory to Cache: Direct and Associative

Direct Memory Mapping Function

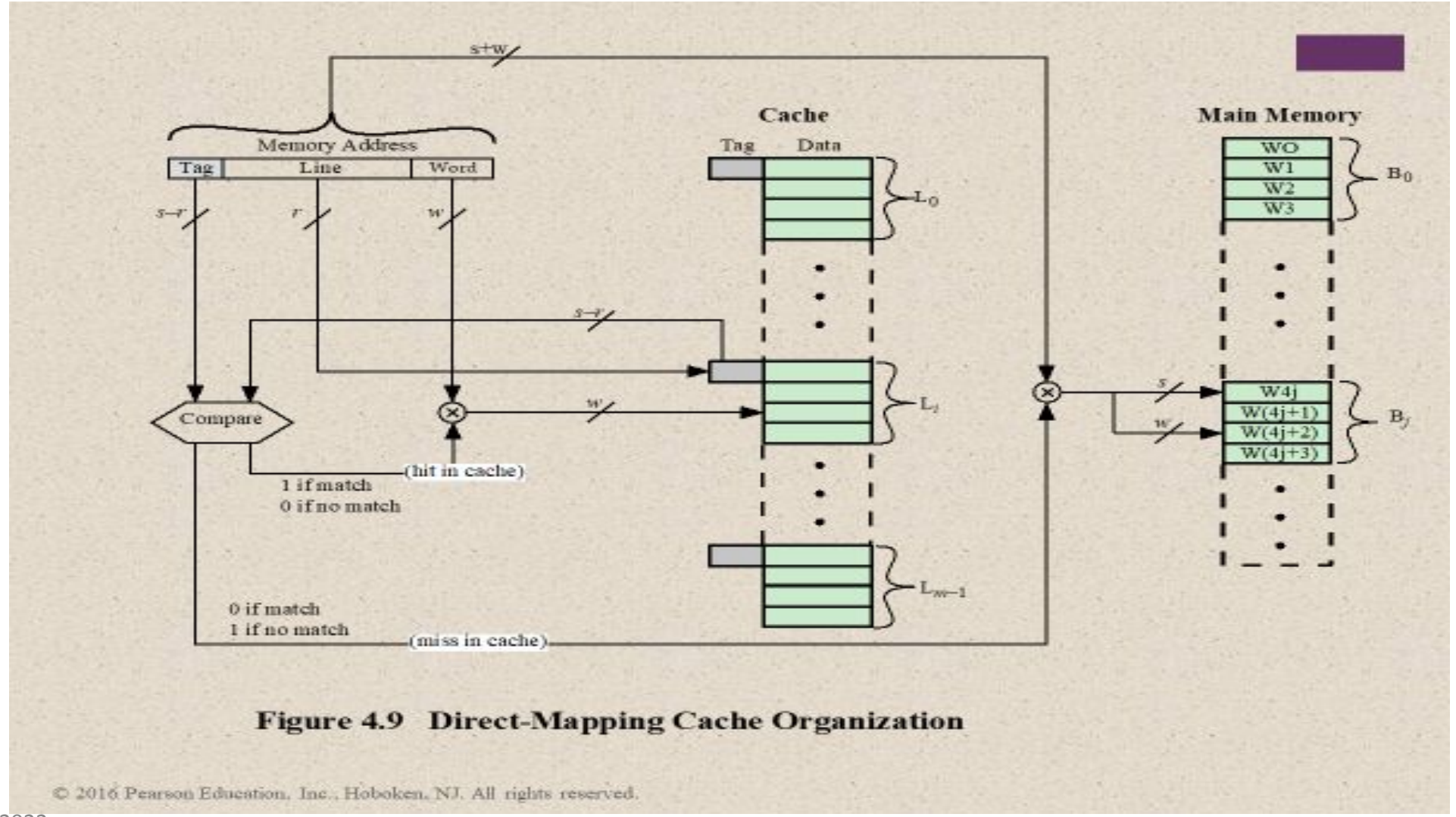


Figure 4.9 Direct-Mapping Cache Organization

Direct Memory Mapping Function

Main memory address can be viewed as consisting of three fields. The least significant w bits identify a unique word or byte within a block of main memory; in most contemporary machines, the address is at the byte level. The remaining s bits specify one of the 2^s blocks of main memory. The cache logic interprets these s bits as a tag of s - r bits (most significant portion) and a line field of r bits. This latter field identifies one of the m = 2^r lines of the cache. To summarize,

- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^w words or bytes
- Number of blocks in main memory = $2^{s+w}/2^w = 2^s$
- Number of lines in cache = m = 2^r
- Size of cache = 2^{r+w} words or bytes
- Size of tag = (s - r) bits

Direct Memory Mapping Function

Ex 1: MM Size: 4 GB
Cache Size: 1 MB
Block Size: 4 KB

1. P.A. bits' split?
2. Tag directory size?

Sol. MM Size = 4 GB = $2^2 \times 2^{30}$ B = $2^{(2+30)}$ B = 2^{32} B

∴ No. of P.A. bits = $\log_2 2^{32}$ = 32

Block Size = 4 KB = $2^2 \times 2^{10}$ B = 2^{12} B

∴ Block offset = $\log_2 2^{12}$ = 12

No. of Blocks in MM = $2^{32} / 2^{12}$ = 2^{20}

∴ Block number bits = $\log_2 2^{20}$ = 20

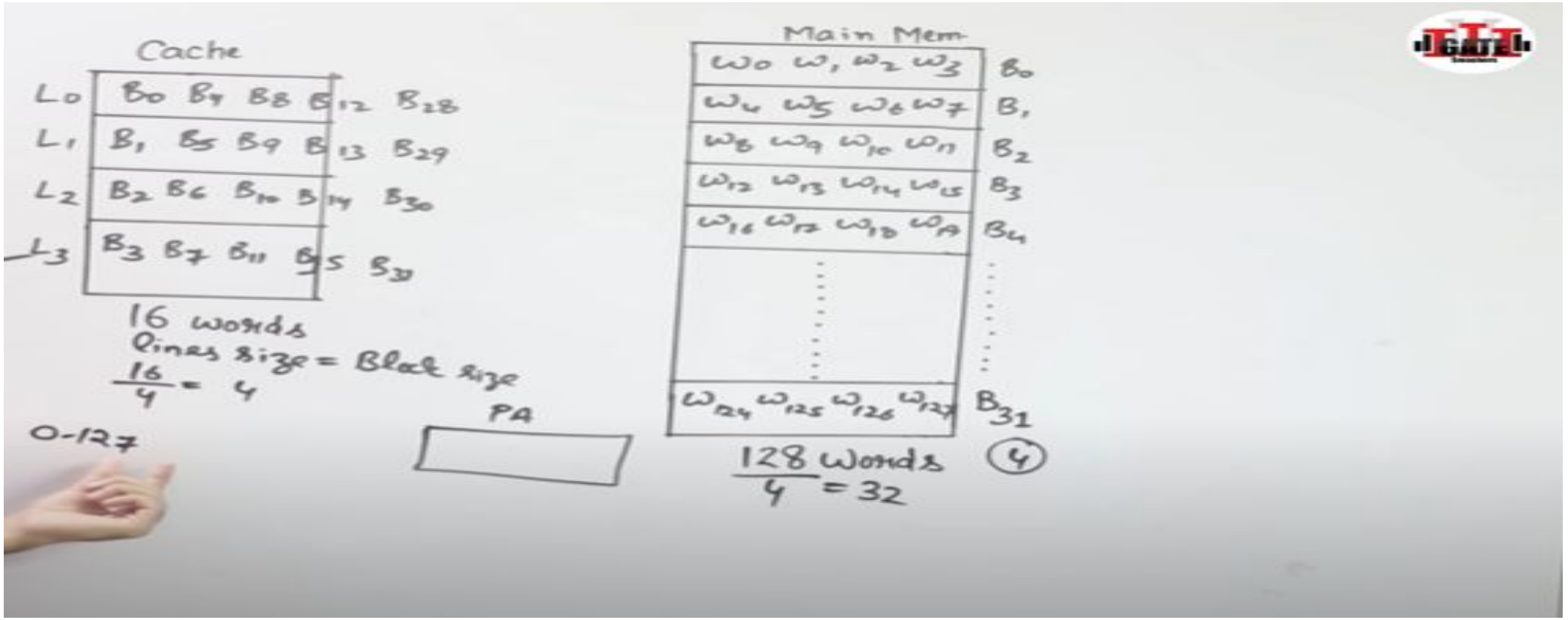
Cache Size = 1 MB = 1 x 2^{20} B = 2^{20} B

No. of Lines in Cache = $2^{20} / 2^{12}$ = 2^8

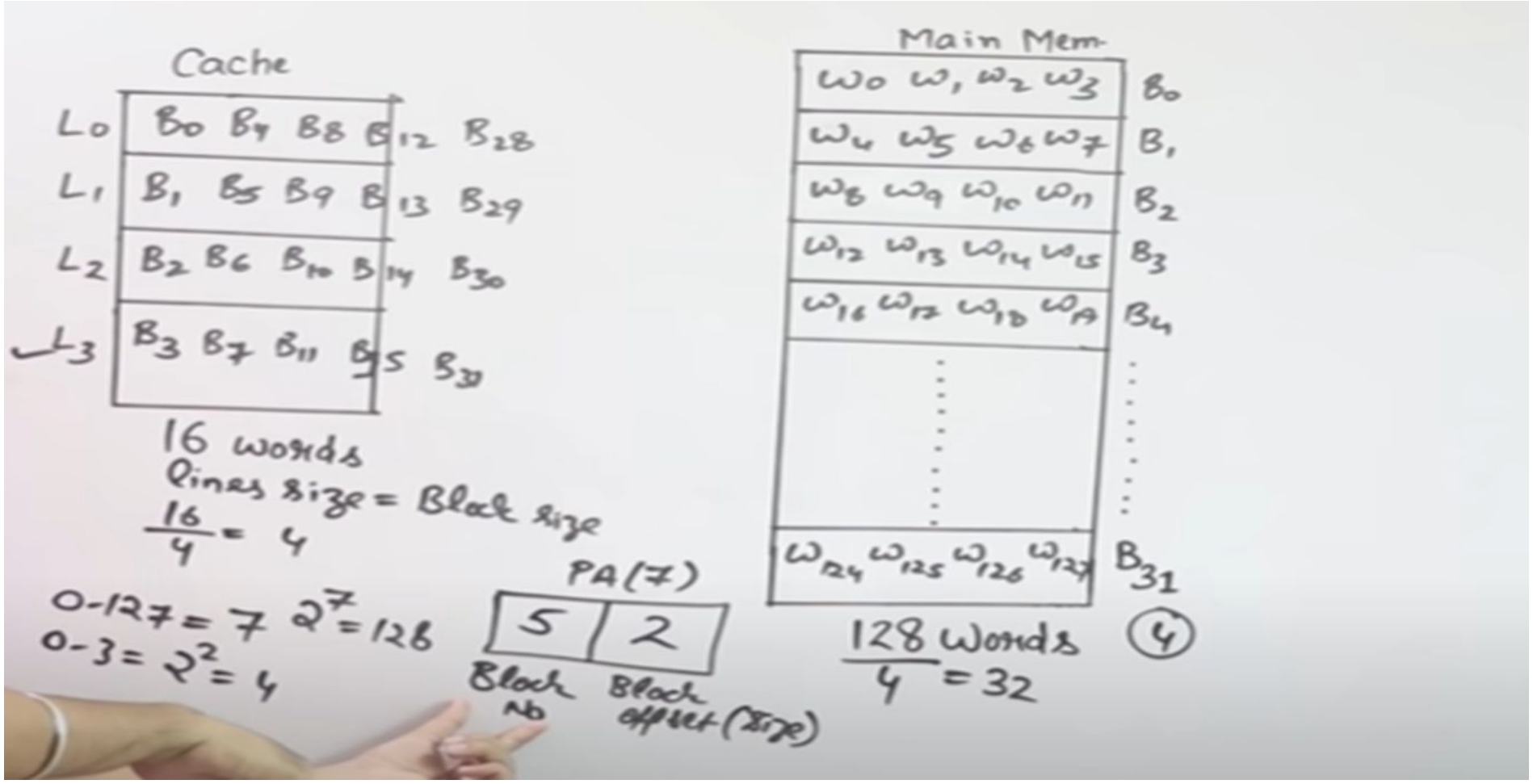
∴ Line number bits = $\log_2 2^8$ = 8

No. of Tag bits : P.A. bits - (Line no. bits + offset) = 32 - (8+12) = 12

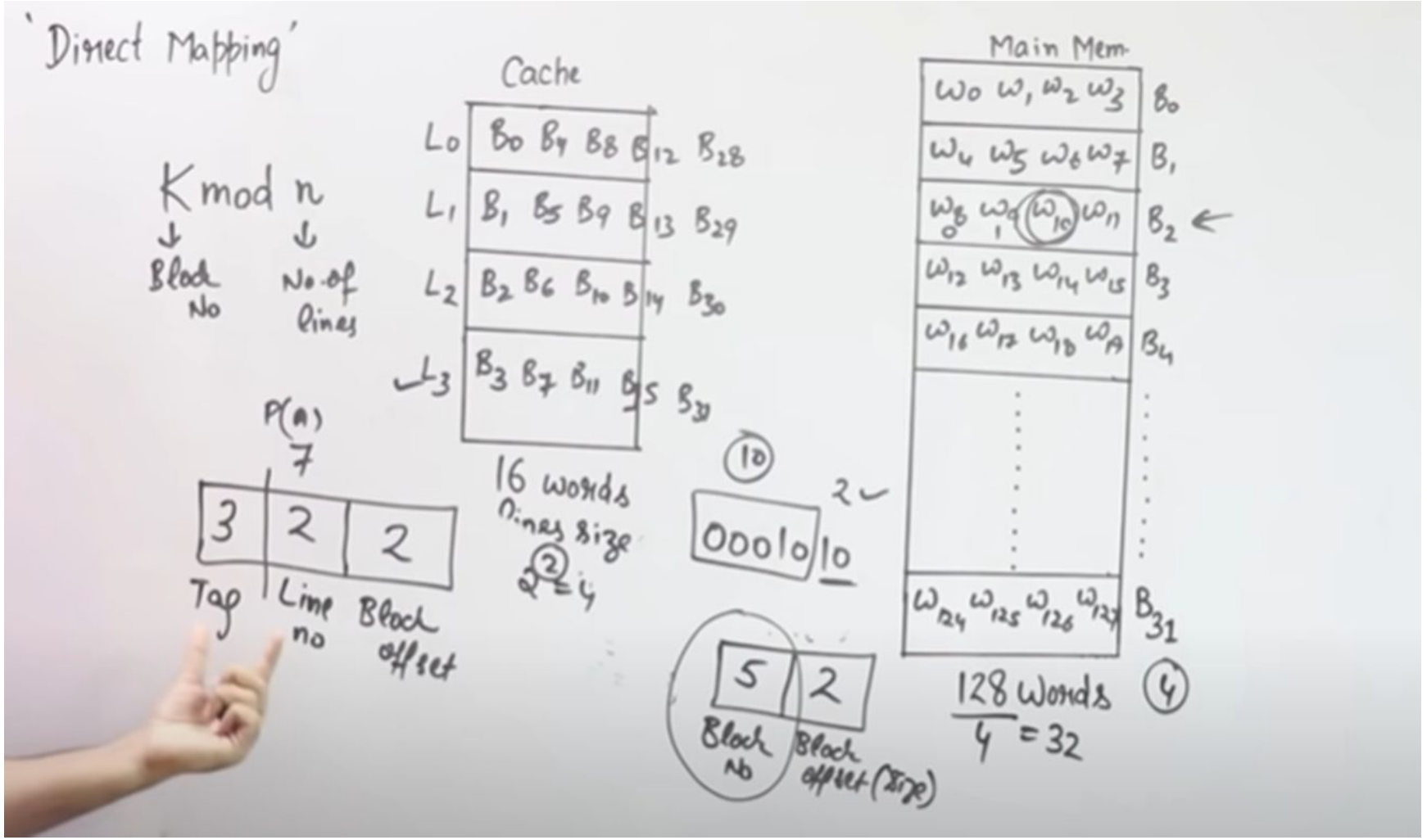
Direct Memory Mapping Function: Example



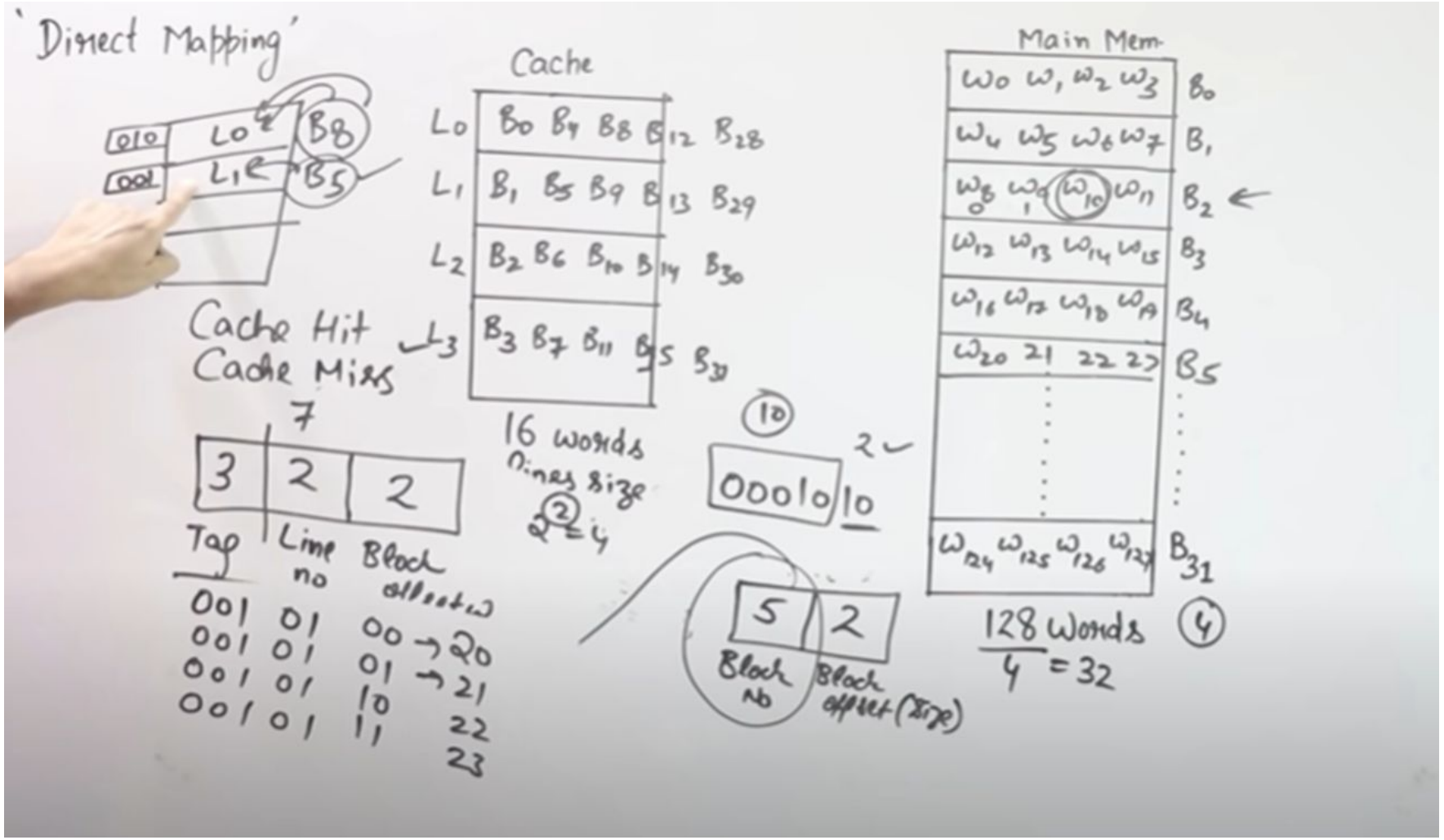
Direct Memory Mapping Function: Example



Direct Memory Mapping Function :Example



Direct Memory Mapping Function : Example



Direct Memory Mapping Function: Pros and Cons

- Simple
- Inexpensive
- Fixed location for given block
- If a program accesses 2 blocks that map to the same line repeatedly, cache misses are very high
- No page replacement algorithm is needed.
- Disadvantage: Multiple pages with the same tag no. once they can not be loaded.

Associative Memory Mapping Function

- Address length = (s + w) bits
- Number of addressable units = 2^{s+w} words or bytes
- Block size = line size = 2^{w} words or bytes
- Number of blocks in main memory = $2^{s+w}/2^{w} = 2^{s}$
- Number of lines in cache = undetermined
- Size of tag = s bits

Associative Memory Mapping Function

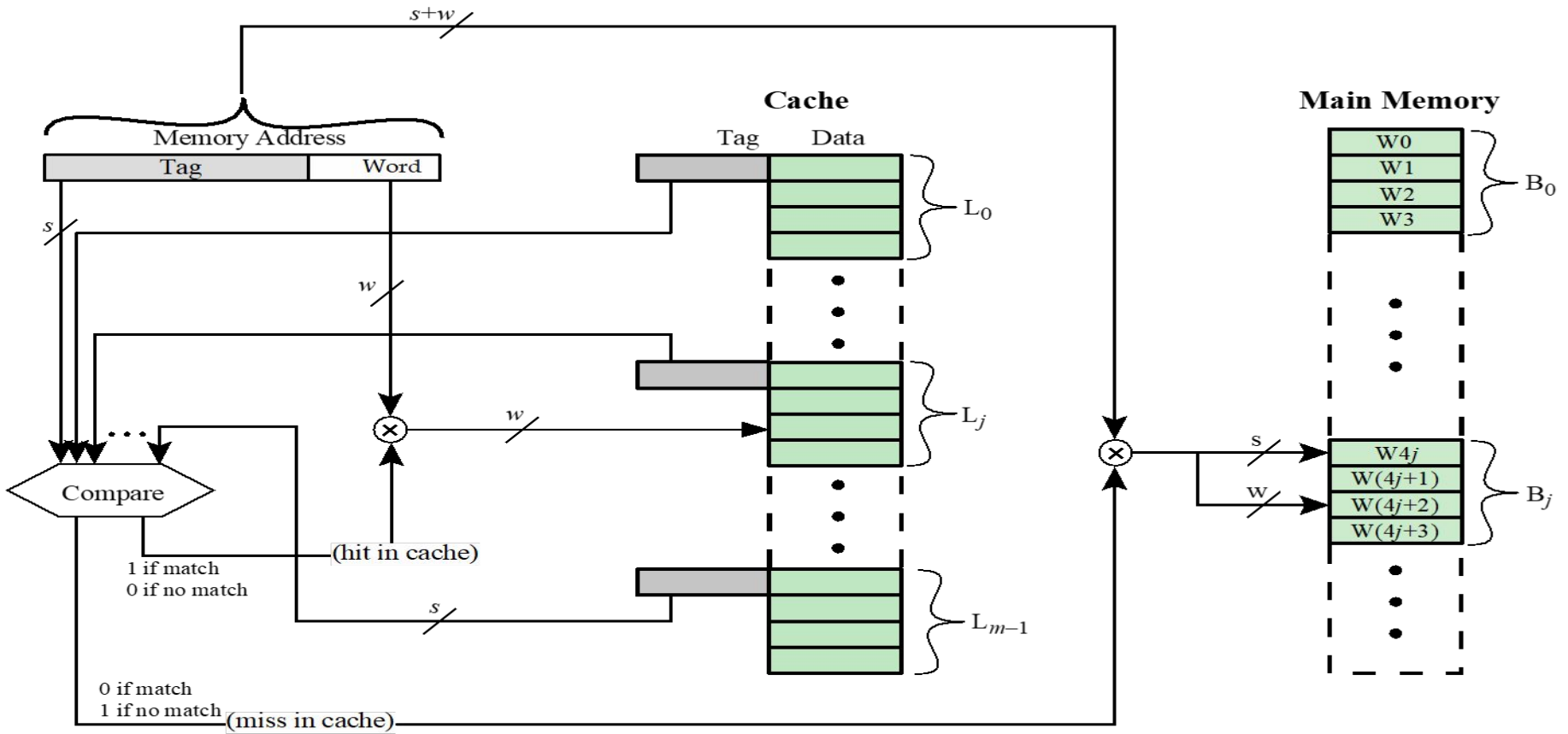
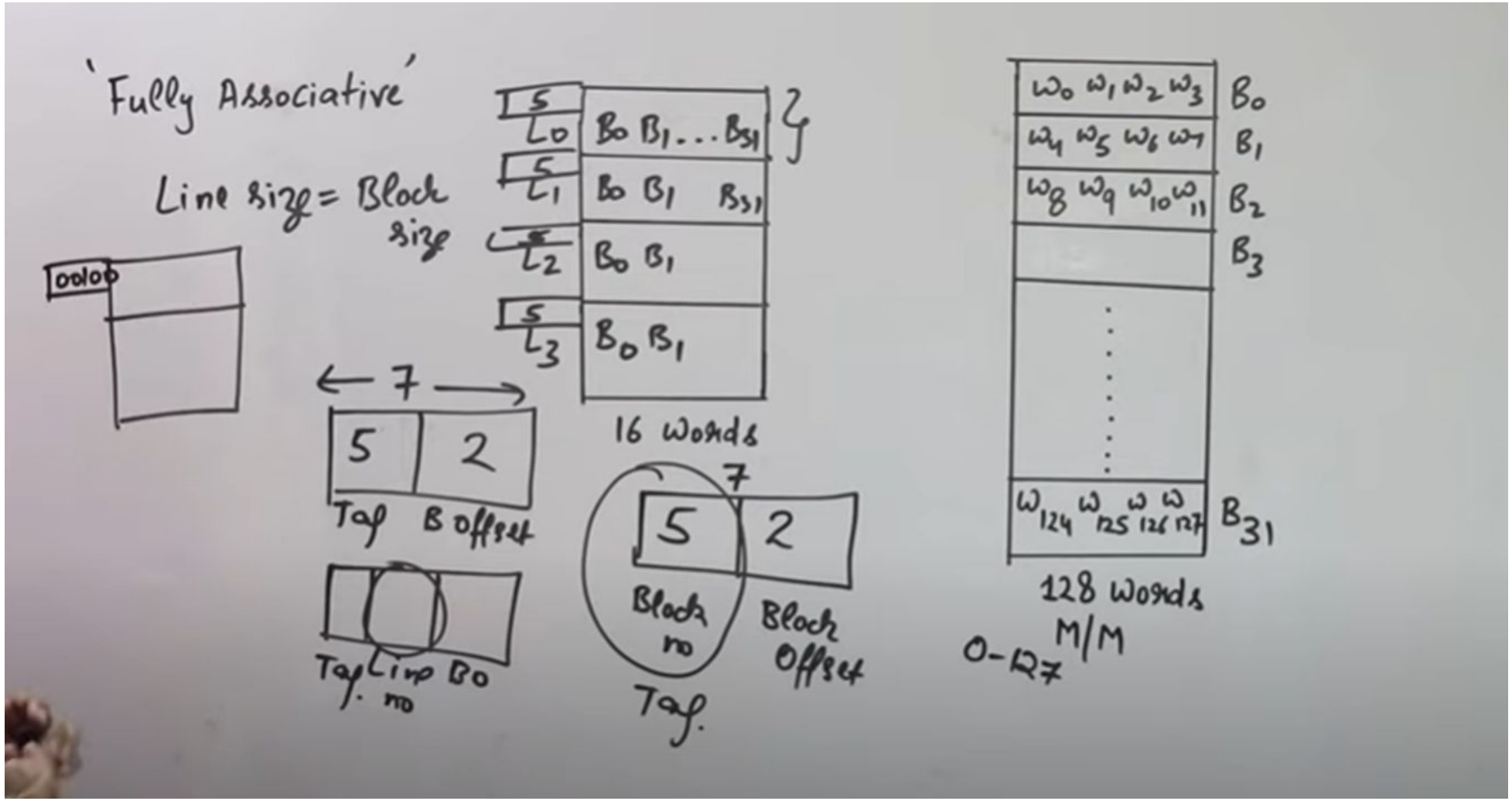


Figure 4.11 Fully Associative Cache Organization

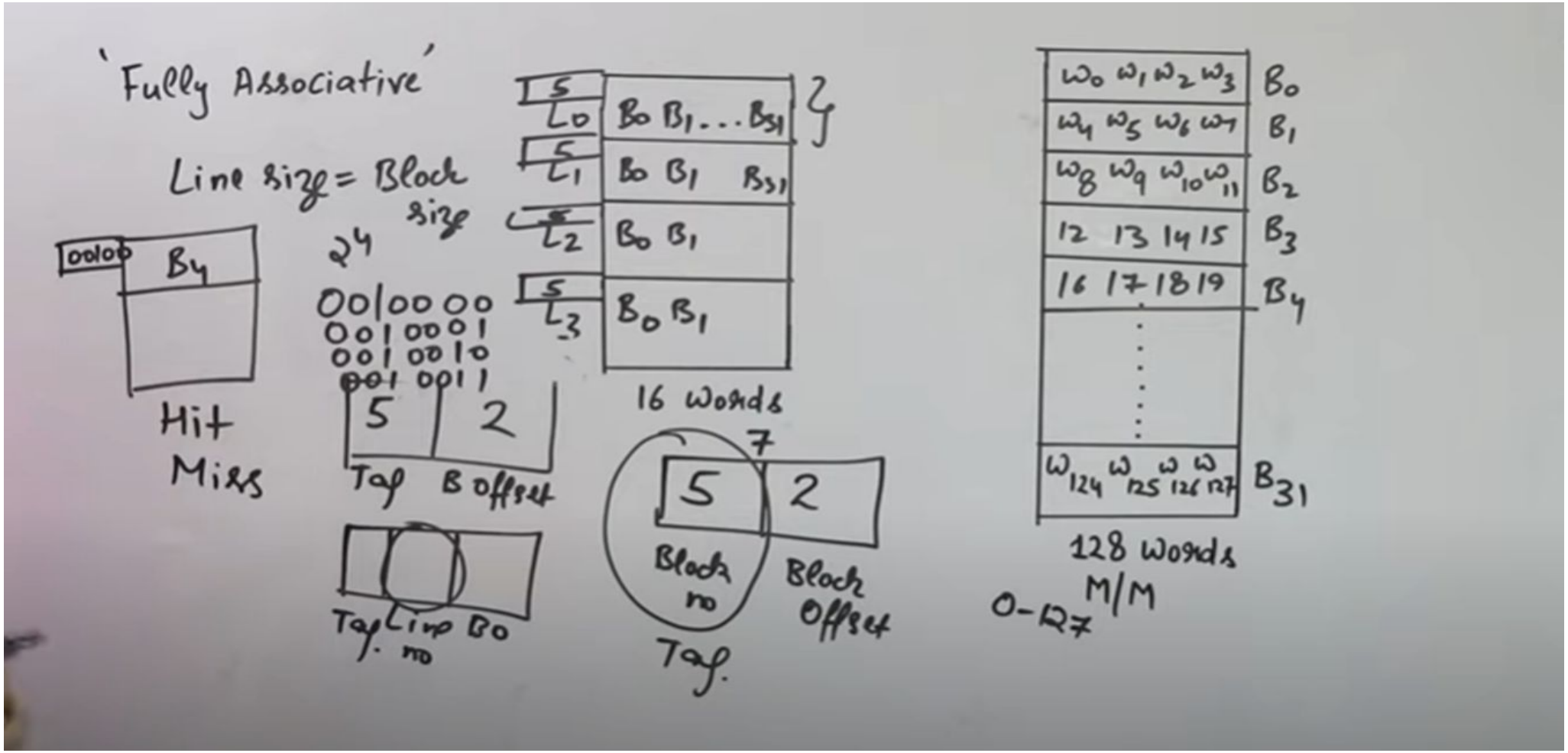
Associative Memory Mapping Function

- A main memory block can load into any line of cache
- Memory address is interpreted as tag and word
- Tag uniquely identifies block of memory
- Every line's tag is examined for a match
- Cache searching gets expensive

Associative Memory Mapping Function: Example



Associative Memory Mapping Function : Example



Associative Memory Mapping Function

- From main memory, any page (block) can be loaded into any line (frame) of cache memory.
- That flexibility we are enjoying.
- Memory address is interpreted as tag and word.
- Tag uniquely identifies block of memory.
- Every line's (Blocks) tag is examined for a match.
- Cache searching gets expensive.
- CAM (Content Addressable Memory) is very costly hardware. (12+7) = 19 bit CAM is very costly affair.

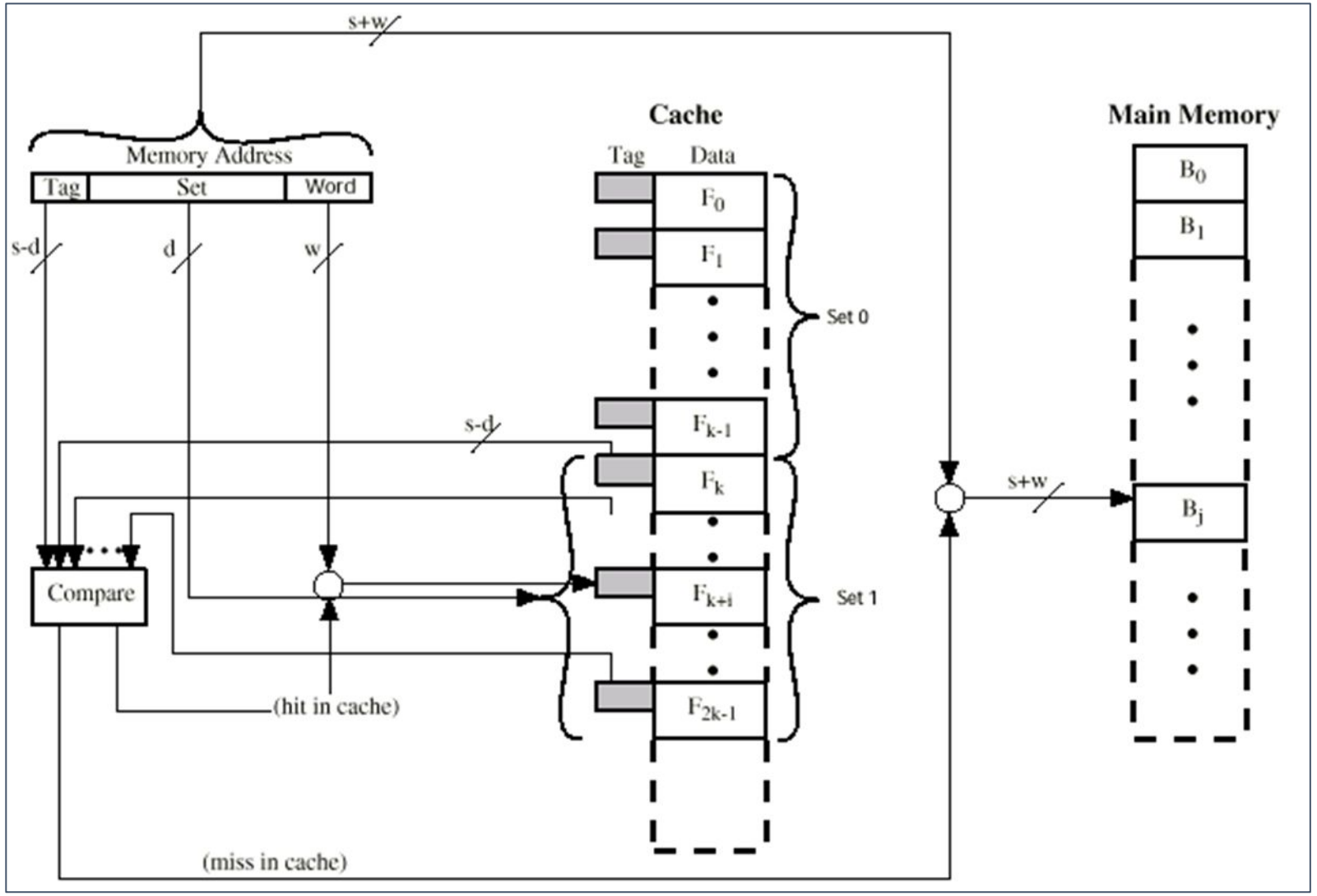
Set Associative Memory Mapping Function

- Cache is divided into a number of sets.
- Each set contains a number of lines.
- A given block maps to any line in a given set
  - e.g. Block B can be in any line of set I.
- 2 way associative mapping
  - A given block can be in one of 2 lines in only one set.

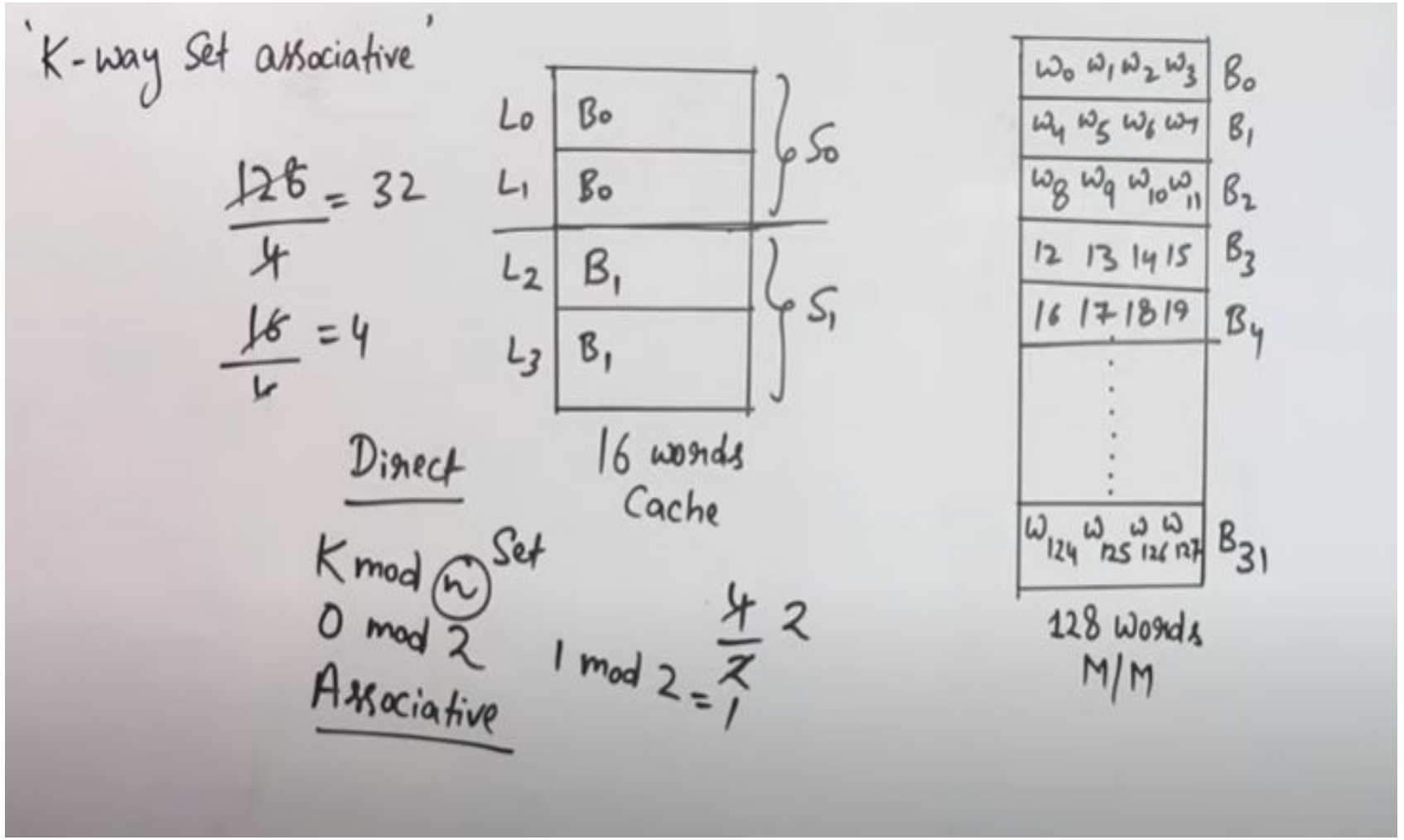
Set Associative Memory Mapping Function

- Compromise that exhibits the strengths of both the direct and associative approaches while reducing their disadvantages
- Cache consists of a number of sets
- Each set contains a number of lines
- A given block maps to any line in a given set
- e.g. 2 lines per set
  - 2 way associative mapping
  - A given block can be in one of 2 lines in only one set

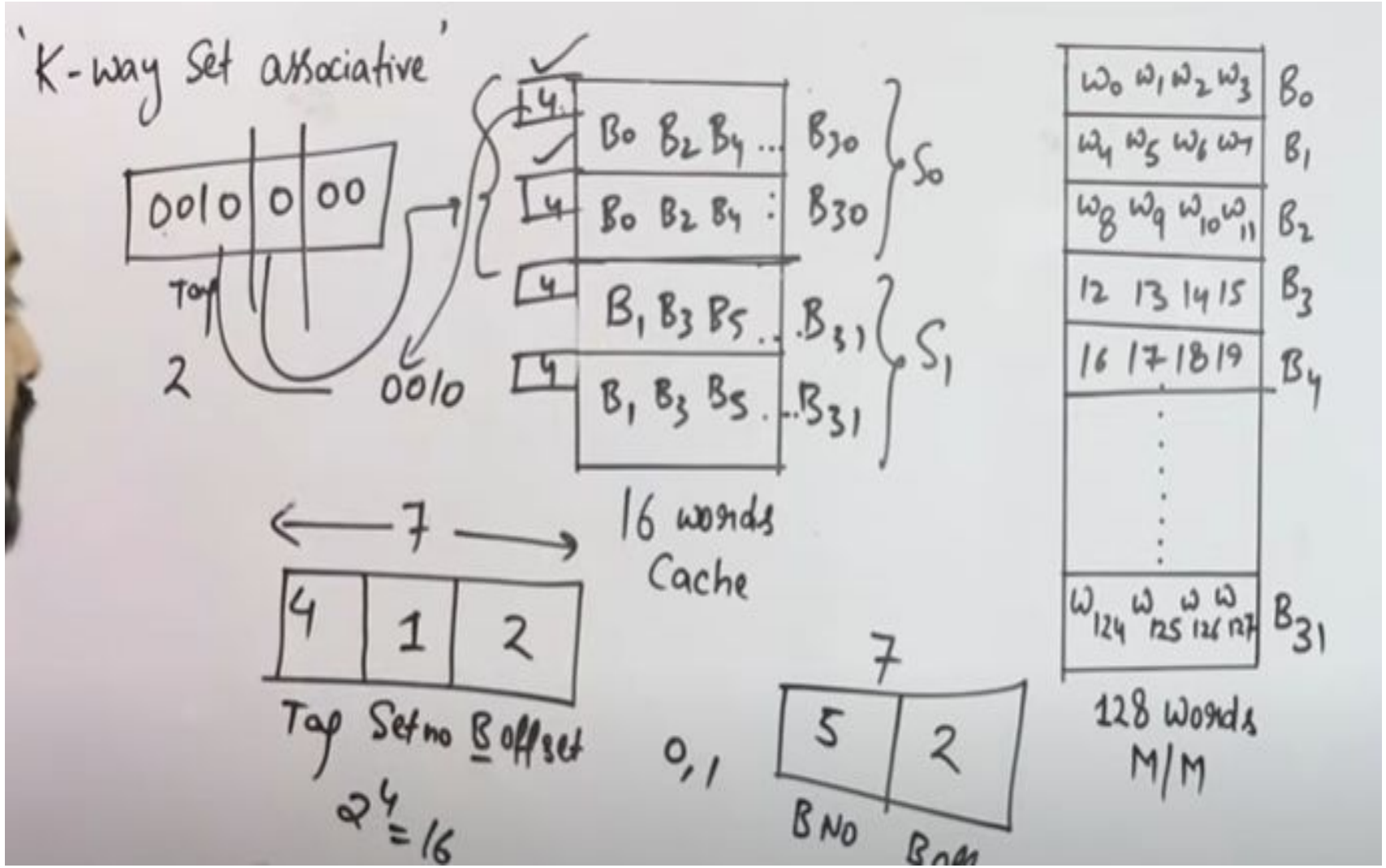
Set-Associative Mapping



Set Associative Memory Mapping Function



Set Associative Memory Mapping Function



Replacement Algorithms

- Once the cache has been filled, when a new block is brought into the cache, one of the existing blocks must be replaced.
- For direct mapping there is only one possible line for any particular block and no choice is possible.
- For the associative and set-associative techniques a replacement algorithm is needed.
- To achieve high speed, an algorithm must be implemented in hardware.

Most Common Replacement Algorithms

- Least recently used (LRU)
  1. Most effective
  2. Replace that block in the set that has been in the cache longest with no reference to it
  3. Because of its simplicity of implementation, LRU is the most popular replacement algorithm
- First-in-first-out (FIFO)
  1. Replace that block in the set that has been in the cache longest
  2. Easily implemented as a round-robin or circular buffer technique
- Least frequently used (LFU)
  1. Replace that block in the set that has experienced the fewest references
  2. Could be implemented by associating a counter with each line

Write Policy

- Must not overwrite a cache block unless main memory is up to date
- Multiple CPUs may have individual caches
- I/O may address main memory directly

Write Policy

When a block that is resident in the cache is to be replaced there are two cases to consider:

If the old block in the cache has not been altered then it may be overwritten with a new block without first writing out the old block

If at least one write operation has been performed on a word in that line of the cache then main memory must be updated by writing the line of cache out to the block of memory before bringing in the new block

There are two problems to contend with:

More than one device may have access to main memory

A more complex problem occurs when multiple processors are attached to the same bus and each processor has its own local cache - if a word is altered in one cache it could conceivably invalidate a word in other caches

Write Through and Write Back Policy

- **Write through**
  - Simplest technique
  - All write operations are made to main memory as well as to the cache
  - The main disadvantage of this technique is that it generates substantial memory traffic and may create a bottleneck
- **Write back**
  - Minimizes memory writes
  - Updates are made only in the cache
  - Portions of main memory are invalid and hence accesses by I/O modules can be allowed only through the cache
  - This makes for complex circuitry and a potential bottleneck

Line Size

- When a block of data is retrieved and placed in the cache not only the desired word but also some number of adjacent words are retrieved.
- As the block size increases the hit ratio will at first increase because of the principle of locality. As the block size increases more useful data are brought into the cache.
- The hit ratio will begin to decrease as the block becomes bigger and the probability of using the newly fetched information becomes less than the probability of reusing the information that has to be replaced

Two specific effects come into play:

Larger blocks reduce the number of blocks that fit into a cache

As a block becomes larger each additional word is farther from the requested word.

Number of cache, One level and Two level cache

- Cache memory is fast and expensive. Traditionally, it is categorized as "levels" that describe its closeness and accessibility to the microprocessor.
- There are three general cache levels:

- L1 cache, or primary cache, is extremely fast but relatively small, and is usually embedded in the processor chip as CPU cache.
- L2 cache, or secondary cache, is often more capacious than L1. L2 cache may be embedded on the CPU, or it can be on a separate chip or coprocessor and have a high-speed alternative system bus connecting the cache and CPU.
- Level 3 (L3) cache is specialized memory developed to improve the performance of L1 and L2.

Number of cache, One level and Two level cache

- As logic density has increased it has become possible to have a cache on the same chip as the processor
- The on-chip cache reduces the processor's external bus activity and speeds up execution time and increases overall system performance.
- When the requested instruction or data is found in the on-chip cache, the bus access is eliminated
  - ✔ On-chip cache accesses will complete appreciably faster than would even zero-wait state bus cycles.
  - ✔ During this period the bus is free to support other transfers

Performance characteristics of two level cache

- **Processor is much faster than the main memory**

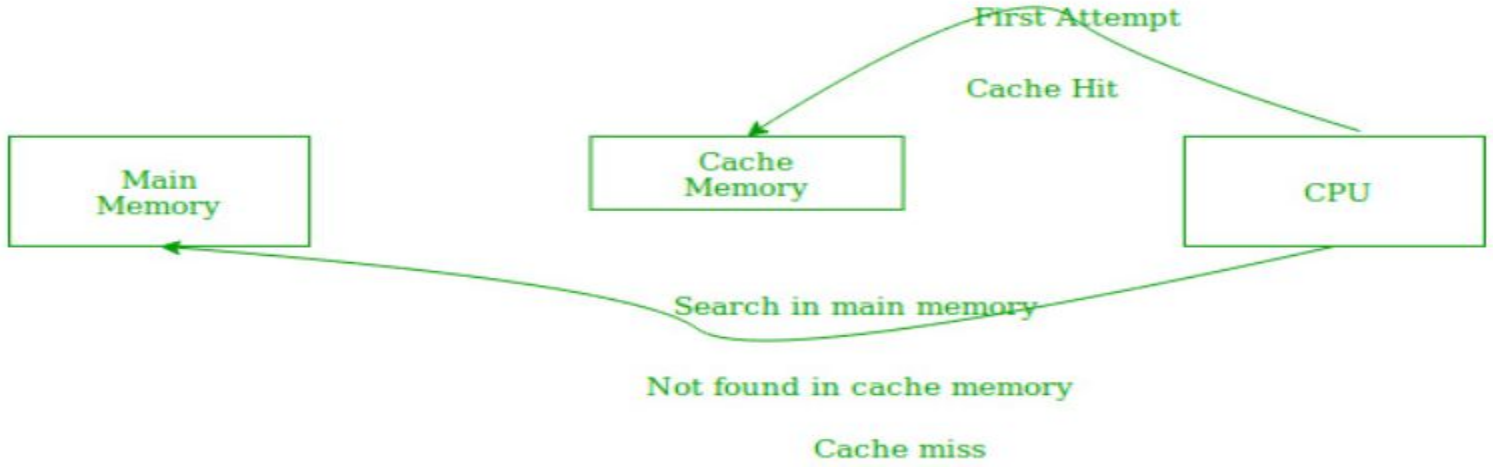
Processor has to spend much of its time waiting while instructions and data are being fetched from the main memory, which is major obstacle towards achieving good performance.

- Speed of the main memory cannot be increased beyond a certain point
- Cache memory is an architectural arrangement which makes the main memory appear faster to the processor than reality.
- Cache memory is based on the property of computer programs known as **"locality of reference".**

Locality of Reference

Locality of Reference refers to the tendency of the computer program to access instructions whose addresses are near one another. The property of locality of reference is mainly shown by:

1. Loops in program cause the CPU to repeatedly execute a set of instructions that constitute the loop.
2. References to data items also get localized, meaning the same data item is referenced again and again.



Locality of Reference

- Locality of reference: It is efficient method is to store the data or instruction in the cache memory so that if it is needed soon again in the near future it could be fetched in a much faster manner.
- Cache Operation:

There are two ways in which data or instruction are fetched from main memory then get stored in cache memory.

- Temporal Locality:

  Temporal locality means current data or instruction that is being fetched may be needed soon. So we should store that data or instruction in the cache memory so that we can avoid again searching in main memory for the same data. Recently executed instruction is likely to be executed again very soon (Time).

- MOV AL,BL

Locality of Reference

- Spatial Locality
  Spatial locality means instruction or data near to the current memory location that is being fetched, may be needed by the processor soon.
- Spatial locality of reference: 2000 3000 4000
  - Instructions with addresses close to a recently instruction are likely to be executed soon (Space or location)

Main memory and Secondary storage

- Primary Memory (Main Memory):

- Primary memory holds only those data and instructions on which the computer is currently working.
- The data and instruction required to be processed resides in the main memory. It is divided into two subcategories RAM and ROM.
- Characteristics of Main Memory

•These are semiconductor memories.
•It is known as the main memory.
•Usually volatile memory.
•Data is lost in case power is switched off.

Main memory

Memory Type	Category	Erasure	Write Mechanism	Volatility
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile
Programmable ROM (PROM)				
Erasable PROM (EPROM)	Read-mostly memory	UV light, chip-level	Electrically	
Electrically Erasable PROM (EEPROM)		Electrically, byte-level		
Flash memory		Electrically, block-level		

Secondary Memory

- This type of memory is also known as external memory or non-volatile. It is slower than the main memory. These are used for storing data/information permanently.
- Example, disk, CD-ROM, DVD, etc.

Characteristics of Secondary Memory

- These are magnetic and optical memories.
- It is known as the backup memory.
- It is a non-volatile memory.
- Data is permanently stored even if power is switched off.
- It is used for storage of data in a computer

Secondary Memory

- This type of memory is also known as external memory or non-volatile. It is slower than the main memory. These are used for storing data/information permanently.
- Example, disk, CD-ROM, DVD, etc.

Characteristics of Secondary Memory

- These are magnetic and optical memories.
- It is known as the backup memory.
- It is a non-volatile memory.
- Data is permanently stored even if power is switched off.
- It is used for storage of data in a computer

END