Unit IV :
Memory Organization

Characteristics of memory system, The memory hierarchy, Cache Memory- Cache memory principles, Replacement algorithms, Write policy, One level and two level cache, I/O modules- Module function and I/O module structure, Programmed I/O, Interrupt driven I/O.

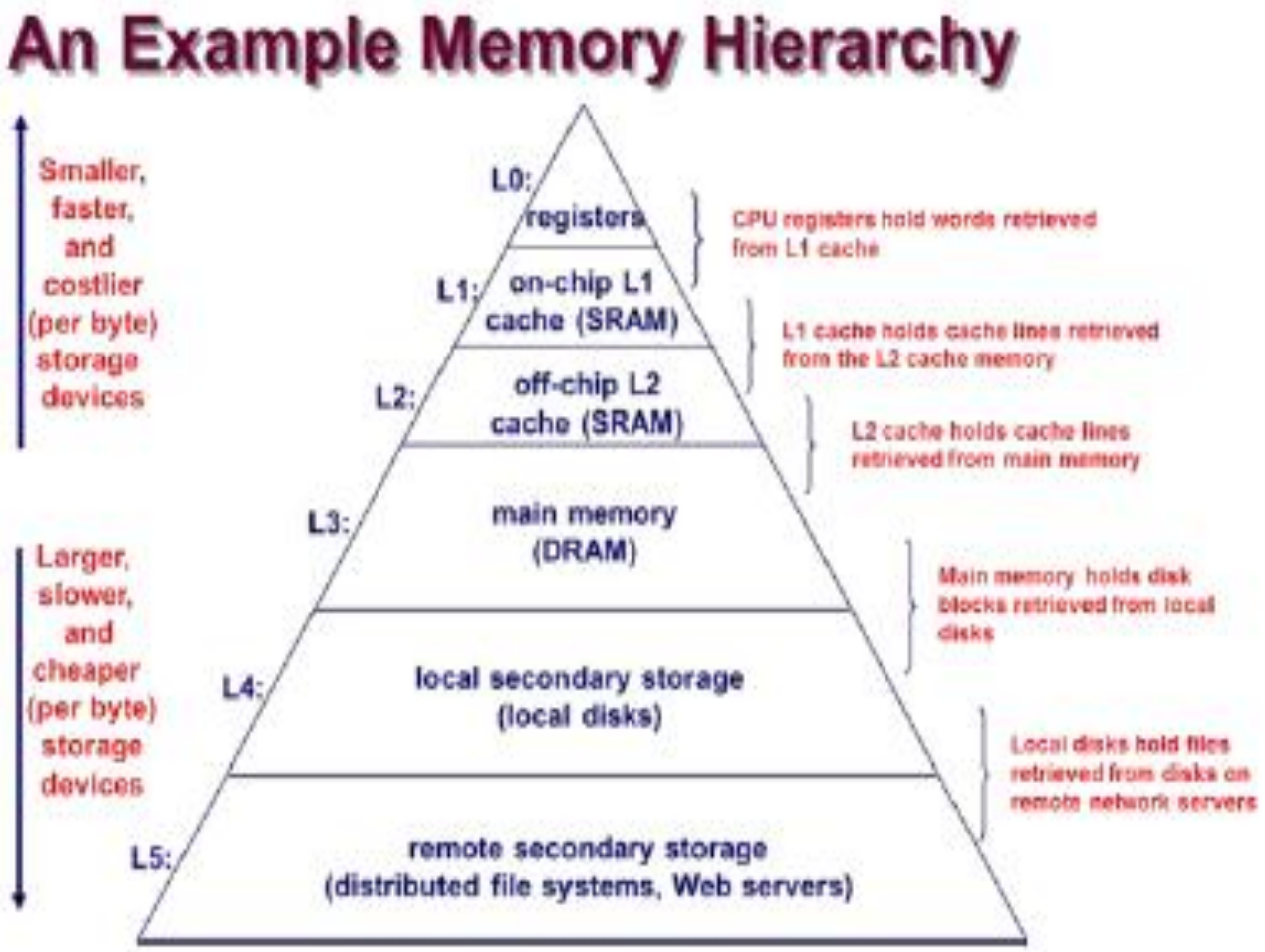
Microcomputer Memory

- Memory is an essential component of the microcomputer system.
- It stores binary instructions and datum for the microcomputer.
- The memory is the place where the computer holds current programs and data that are in use.
- None technology is optimal in satisfying the memory requirements for a computer system.
- Computer memory exhibits perhaps the widest range of type, technology, organization, performance and cost of any feature of a computer system.
- The memory unit that communicates directly with the CPU is called main memory.
- Devices that provide backup storage are called auxiliary memory or secondary memory

Memory Hierarchies

- Some fundamental and enduring properties of hardware and software:
- Fast storage technologies cost more per byte and have less capacity
- Gap between CPU and main memory speed is widening
- Well-written programs tend to exhibit good locality
- These fundamental properties complement each other beautifully
- They suggest an approach for organizing memory and storage systems known as a memory hierarchy

An Example Memory Hierarchy



Characteristics of memory systems

- The memory system can be characterized with their

Location, Capacity, Unit of transfer, Access method, Performance, Physical type, Physical characteristics, Organization.

Location

• Processor memory: The memory like registers is included within the processor and termed as processor memory.

• • Internal memory: It is often termed as main memory and resides within the CPU.

• • External memory: It consists of peripheral storage devices such as disk and magnetic tape that are accessible to processor via i/o controllers

Capacity

• Word size: Capacity is expressed in terms of words or bytes.

— The natural unit of organisation

• Number of words: Common word lengths are 8, 16, 32 bits etc.

— or Byte

Unit of Transfer

• Internal: For internal memory, the unit of transfer is equal

to the number of data lines into and out of the memory module.

• External: For external memory, they are transferred in block which is larger than a word.

• Addressable unit

— Smallest location which can be uniquely addressed

— Word internally

— Cluster on Magnetic disks

Access Method

- Sequential access: In this access, it must start with beginning and read through a specific linear sequence. This means access time of data unit depends on position of records (unit of data) and previous location. — e.g. tape
- Direct Access: Individual blocks of records have unique address based on location. Access is accomplished by jumping (direct access) to general vicinity plus a sequential search to reach the final location. — e.g. disk
- Random access: The time to access a given location is independent of the sequence of prior accesses and is constant. Thus any location can be selected out randomly and directly addressed and accessed. — e.g. RAM
- Associative access: This is random access type of memory that enables one to make a comparison of desired bit locations within a word for a specified match, and to do this for all words simultaneously.
- e.g. cache

Performance

• **Access time**: For random access memory, access time is the time it takes to perform a read or write operation i.e. time taken to address a memory plus to read / write from addressed memory location. Whereas for non-random access, it is the time needed to position read / write mechanism at desired location.

— Time between presenting the address and getting the valid data

• **Memory Cycle time:** It is the total time that is required to store next memory access operation from the previous memory access operation.

Memory cycle time = access time plus transient time (any additional time required before a second access can commence).

— Time may be required for the memory to “recover” before next access

— Cycle time is access + recovery

• **Transfer Rate**: This is the rate at which data can be transferred in and out of a memory unit.

— Rate at which data can be moved

— For random access, R = 1 / cycle time

— For non-random access, Tn = Ta + 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).

Physical Types

- Semiconductor
  - RAM
- Magnetic
  - Disk & Tape
- Optical
  - CD & DVD
- Others
  - Bubble
  - Hologram

Cache Memory

Principles

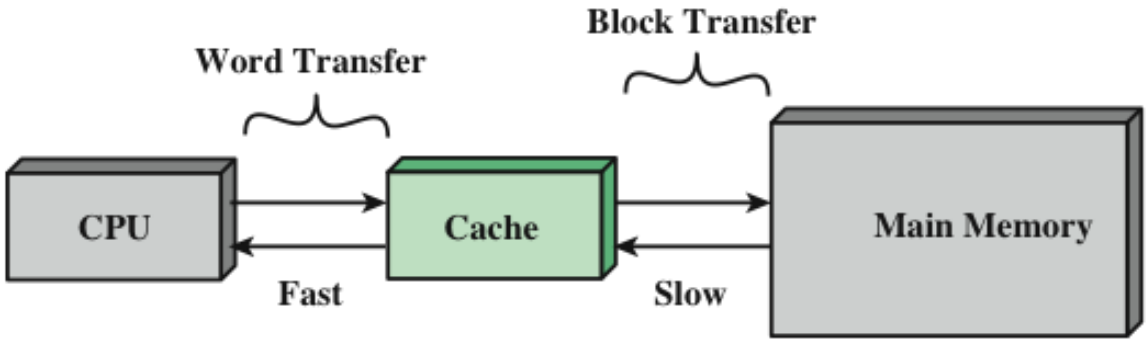
.Cache memory is intended to give memory speed, and at same time provide large memory size at the price of less expensive type of memories.

.there is a relatively large and slow main memory together with a small, faster cache memory.

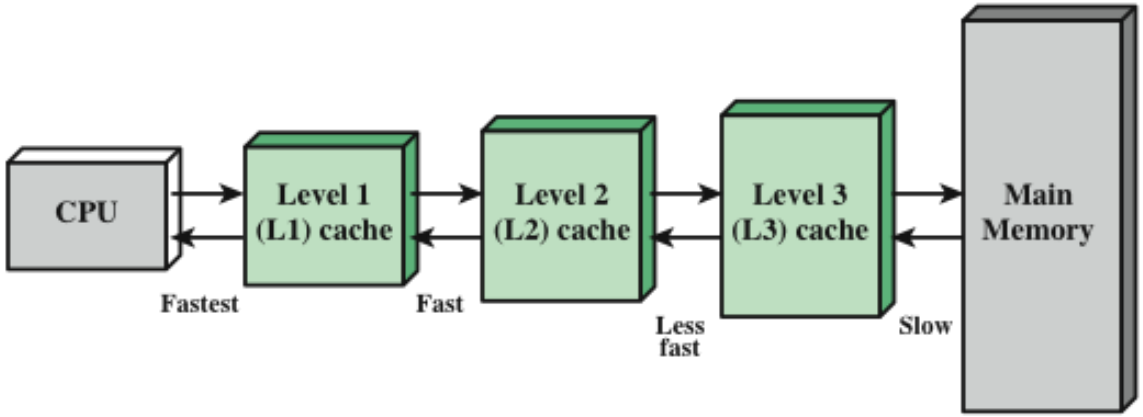
.Contain copy of portions of main memory.

.When processor attempts to read a word of memory, it first search in cache, if so, the word is delivered to the processor. If not a block of main memory is read into the cache and the word is delivered to the processor.

Cache and Main Memory



(a) Single cache



(b) Three-level cache organization

Figure 4.3 Cache and Main Memory

- Cache memory is designed to combine the memory access time of expensive, high-speed memory combined with the large memory size of less expensive, lower-speed memory.

- 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. When the processor attempts to read a word of memory, a check is made to determine if the word is in the cache. If so, the word is delivered to the processor. If not, a block of main memory, consisting of some fixed number of words, is read into the cache and then the word is delivered to the processor.
- Because of the phenomenon of locality of reference, when a block of data is fetched into the cache to satisfy a single memory reference, it is likely that there will be future references to that same memory location or to other words in the block.

- 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.

Principles

Main memory consist of up to 2^n addressable word, each word having a unique n bit address.

For mapping, memory is considered to consist of a number of fixed length of blocks of **K** words each. M=2^n/K blocks.

Cache consist of C lines of K words and C<<M.

If a word in a block of memory is read, that block is transferred to one of the cache line, an individual line cannot be uniquely and permanently dedicated to a particular block.

Each line include a tag that identifies which particular block is currently being stored.
Tag is usually a portion of the main memory address.

Cache/Main Memory Structure

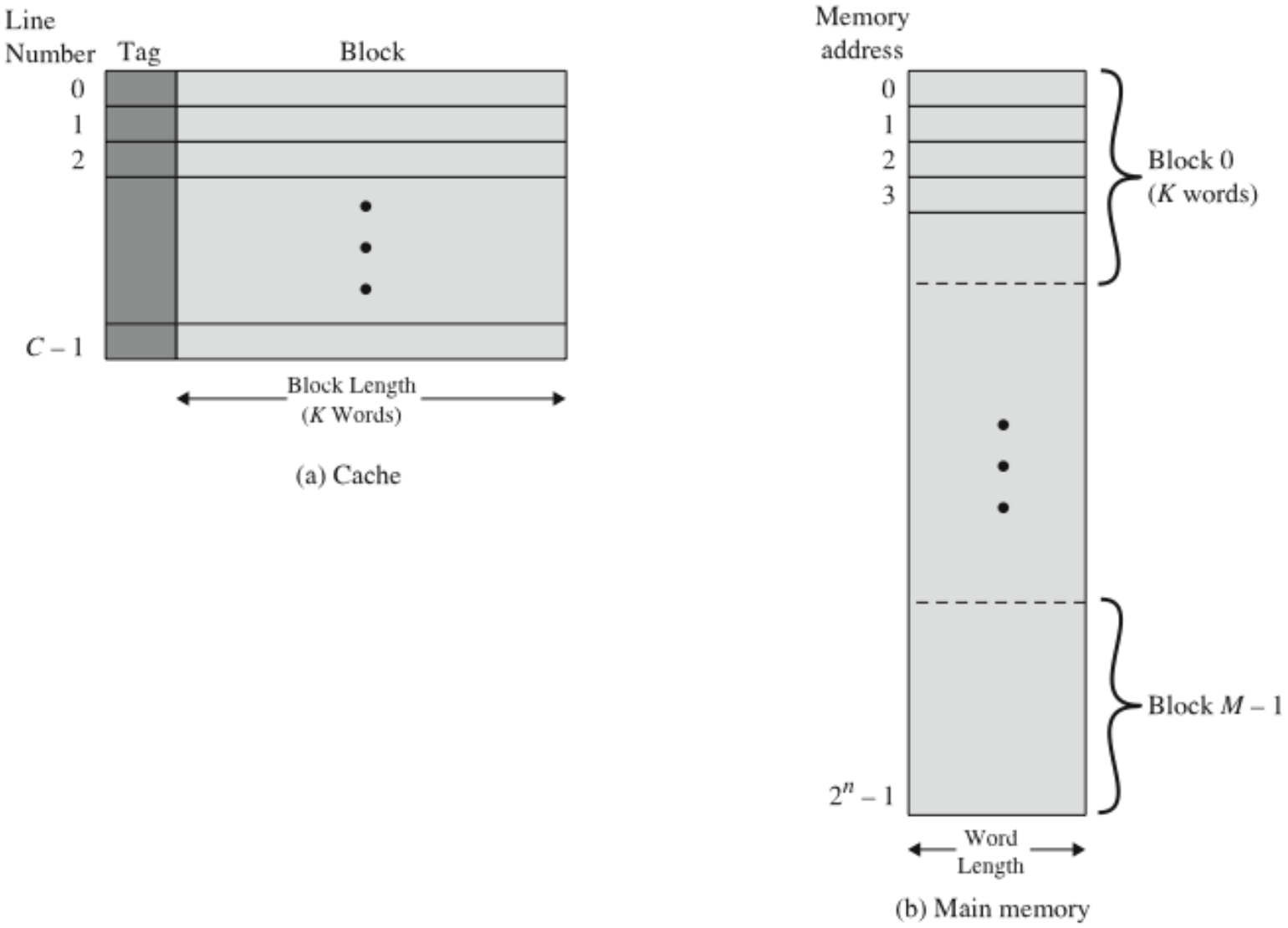


Figure 4.4 Cache/Main-Memory Structure

Cache Read Operation

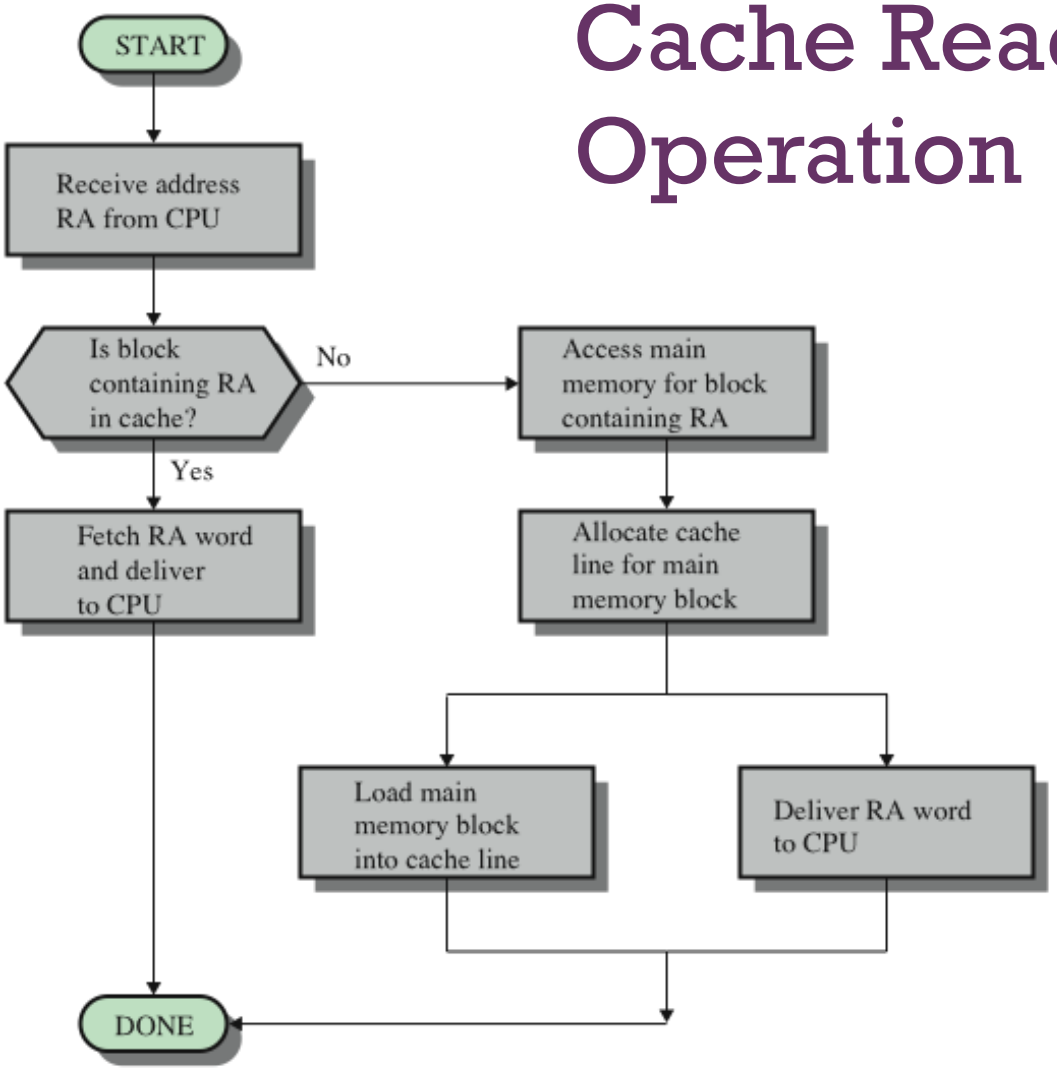


Figure 4.5 Cache Read Operation

The processor generates the read address (RA) of a word to be read. If the word is contained in the cache, it is delivered to the processor.

Otherwise, the block containing that word is loaded into the cache, and the word is delivered to the processor.

Typical Cache Organization

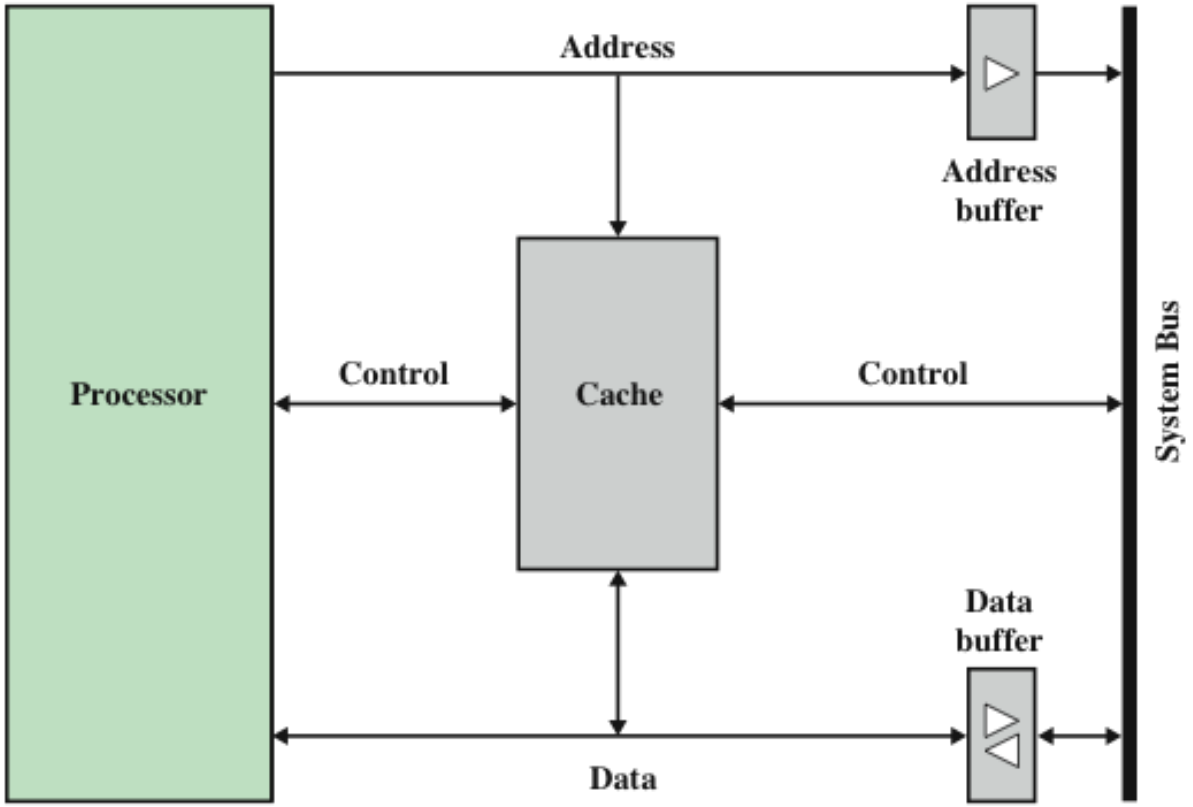


Figure 4.6 Typical Cache Organization

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

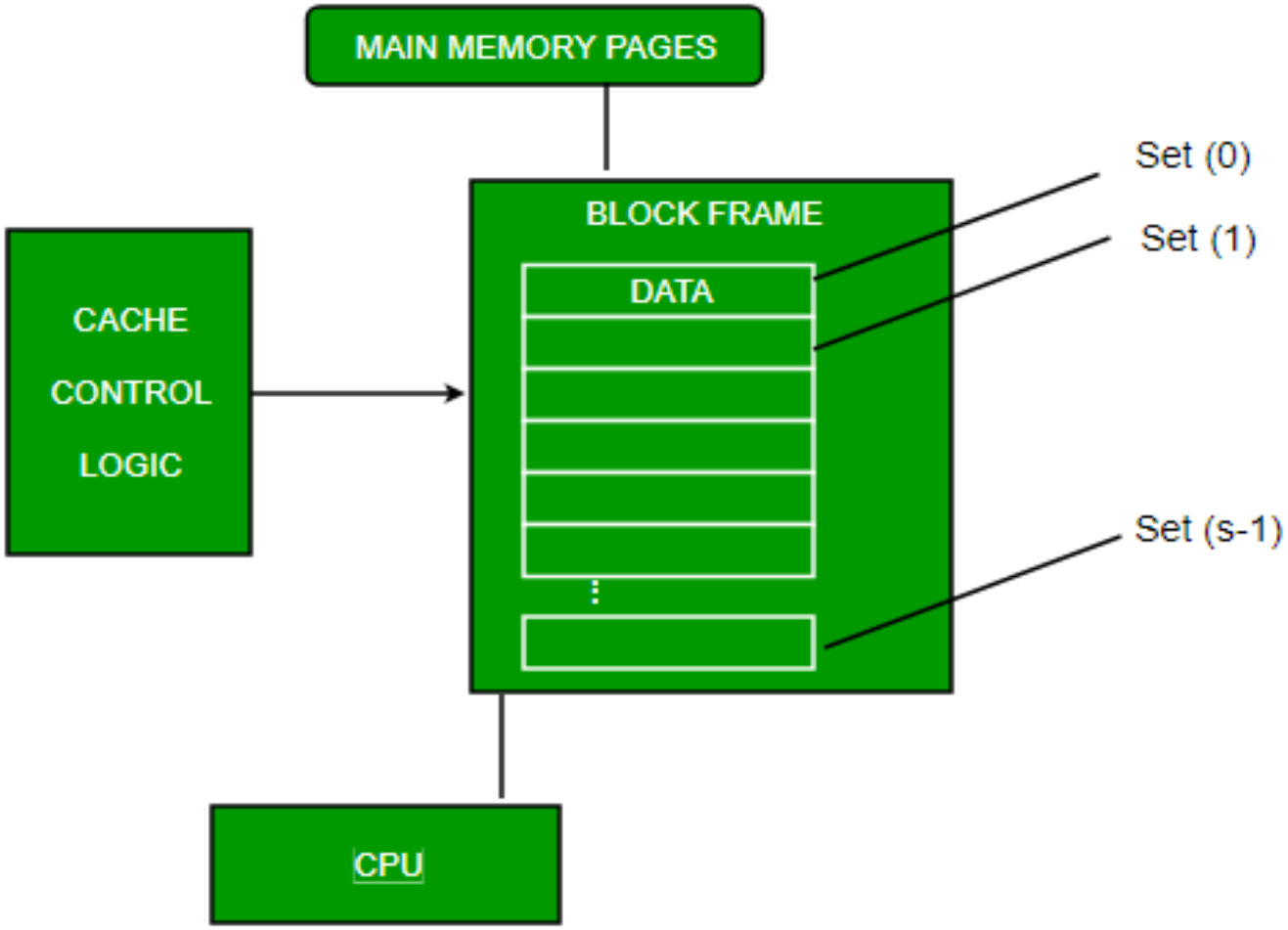
Associative

- 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

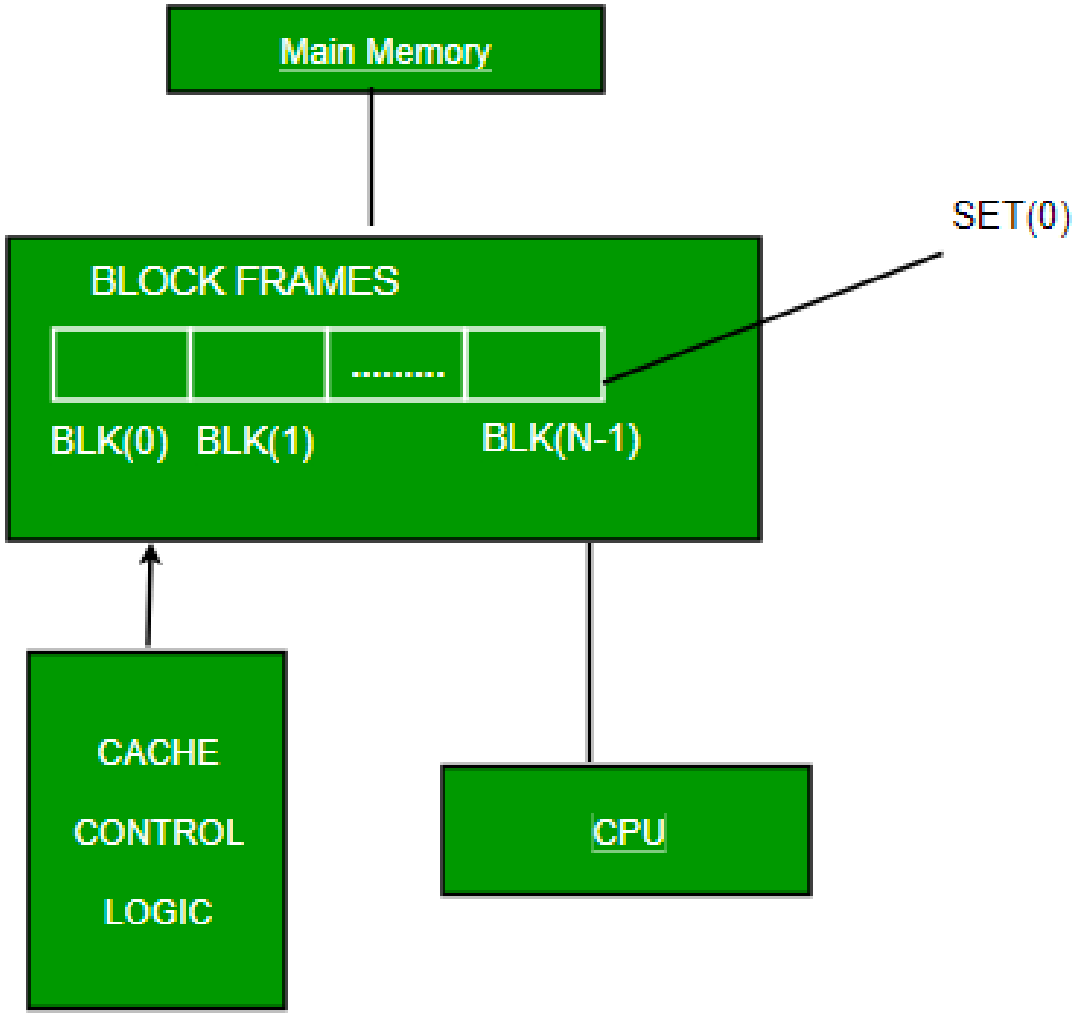
Set Associative

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

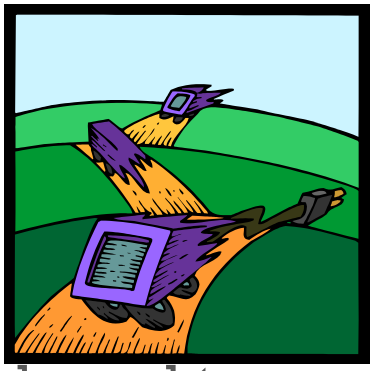
- **Direct Mapping –**
  The simplest technique, known as direct mapping, maps each block of main memory into only one possible cache line.
  In Direct mapping, assign each memory block to a specific line in the cache.
- If a line is previously taken up by a memory block when a new block needs to be loaded, the old block is trashed.
- An address space is split into two parts **index field** and a **tag field**.
- The cache is used to store the tag field whereas the rest is stored in the main memory.
- Direct mapping`s performance is directly proportional to the Hit ratio.


MAIN MEMORY PAGES
BLOCK FRAME
DATA
Set (0)
Set (1)
Set (s-1)
CACHE
CONTROL
LOGIC
CPU

- **Associative Mapping –**
  In this type of mapping, the associative memory is used to store content and addresses of the memory word. Any block can go into any line of the cache. This means that the word id bits are used to identify which word in the block is needed, but the tag becomes all of the remaining bits. This enables the placement of any word at any place in the cache memory. It is considered to be the fastest and the most flexible mapping form.


Main Memory
SET(0)
BLOCK FRAMES
.....
BLK(0)
BLK(1)
BLK(N-1)
CACHE
CONTROL
LOGIC
CPU

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

The four most common replacement algorithms are:

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

- A number of algorithms have been tried. We mention four of the most common. Probably the most
- effective is least recently used (LRU): Replace that block in the set that has been in
- the cache longest with no reference to it. For two-way set associative, this is easily
- implemented. Each line includes a USE bit. When a line is referenced, its USE bit
- is set to 1 and the USE bit of the other line in that set is set to 0. When a block is to
- be read into the set, the line whose USE bit is 0 is used. Because we are assuming
- that more recently used memory locations are more likely to be referenced, LRU
- should give the best hit ratio. LRU is also relatively easy to implement for a fully
- associative cache. The cache mechanism maintains a separate list of indexes to all
- the lines in the cache. When a line is referenced, it moves to the front of the list.
- For replacement, the line at the back of the list is used. Because of its simplicity of
- implementation, LRU is the most popular replacement algorithm.

- Another possibility is first-in-first-out (FIFO): Replace that block in the set
- that has been in the cache longest. FIFO is easily implemented as a round-robin
- or circular buffer technique. Still another possibility is least frequently used (LFU):
- Replace that block in the set that has experienced the fewest references. LFU could
- be implemented by associating a counter with each line. A technique not based on
- usage (i.e., not LRU, LFU, FIFO, or some variant) is to pick a line at random from
- among the candidate lines. Simulation studies have shown that random replacement
- provides only slightly inferior performance to an algorithm based on usage [SMIT82].

- **Types of Cache –**
- **Primary Cache –**
  A primary cache is always located on the processor chip. This cache is small and its access time is comparable to that of processor registers.
- **Secondary Cache –**
  Secondary cache is placed between the primary cache and the rest of the memory. It is referred to as the level 2 (L2) cache. Often, the Level 2 cache is also housed on the processor chip.

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

.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
- .-when update an occurs UPDATE bit associated with the line is set.
- .Then when a bock is replaced it is written back to main memory if and only if the UPDATE bit is set.
- .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

The simplest technique is called **write through.** Using this technique, all write operations are made to main memory as well as to the cache, ensuring that main memory is always valid. Any other processor–cache module can monitor traffic to main memory to maintain consistency within its own cache. The main disadvantage of this technique is that it generates substantial memory traffic and may create a bottleneck.

An alternative technique, known as **write back,** minimizes memory writes. With write back, updates are made only in the cache. When an update occurs, a **dirty bit,** or use bit, associated with the line is set. Then, when a block is replaced, it is written back to main memory if and only if the dirty bit is set. The problem with write back is that 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. Experience has shown that the percentage of memory references that are writes is on the order of 15% [SMIT82]. However, for HPC applications, this number may approach 33% (vector-vector multiplication) and can go as high as 50% (matrix transposition).

In a bus organization in which more than one device (typically a processor) has a cache and main memory is shared, a new problem is introduced. If data in one cache are altered, this invalidates not only the corresponding word in main memory, but also that same word in other caches (if any other cache happens to have that same word). Even if a write-through policy is used, the other caches may contain invalid data. A system that prevents this problem is said to maintain cache coherency. Possible approaches to cache coherency include the following:

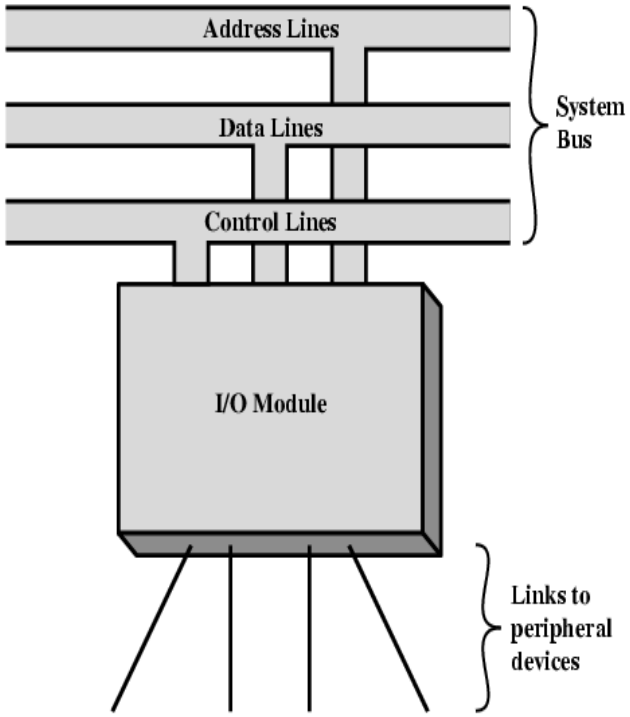
• **Bus watching with write through:** Each cache controller monitors the address lines to detect write operations to memory by other bus masters. If another master writes to a location in shared memory that also resides in the cache memory, the cache controller invalidates that cache entry. This strategy depends on the use of a write-through policy by all cache controllers.

• **Hardware transparency:** Additional hardware is used to ensure that all updates to main memory via cache are reflected in all caches. Thus, if one processor modifies a word in its cache, this update is written to main memory. In addition, any matching words in other caches are similarly updated.

• **Non-cacheable memory:** Only a portion of main memory is shared by more than one processor, and this is designated as non-cacheable. In such a system, all accesses to shared memory are cache misses, because the shared memory is never copied into the cache. The non-cacheable memory can be identified using chip-select logic or high-address bits.

I/O Module

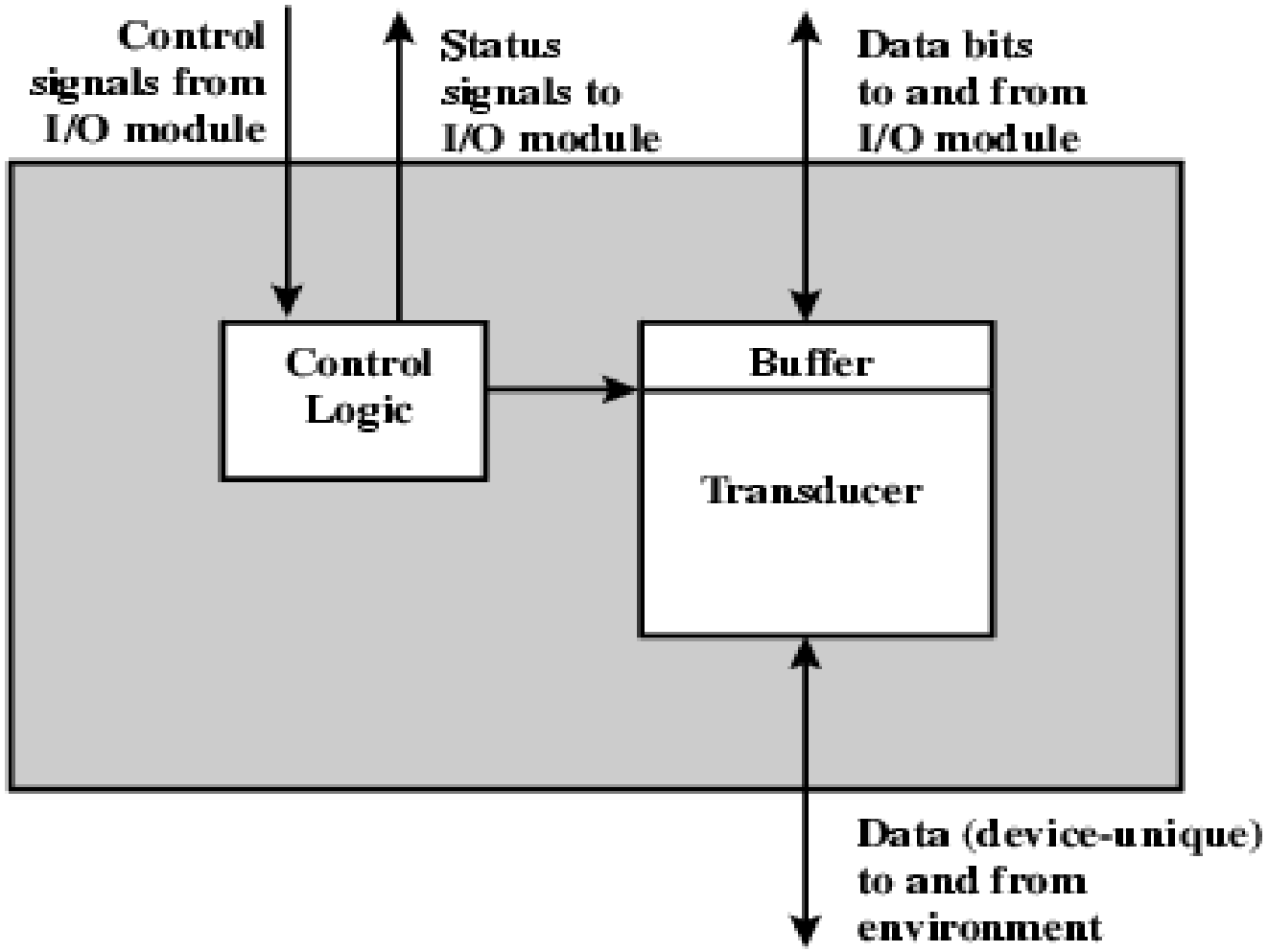
- Computer system consists of a set of I/O modules
- Each module interfaces one or more peripherals to system bus
- It contains a logic for performing a communication function between the peripheral and the bus.



External Devices

- Broadly classified into three categories
  - Human readable: suitable for communicating with the computer users, e.g. display, printer, keyboard
  - Machine readable: suitable for communicating with equipment, e.g. magnetic disk & tapes, sensors and actuators etc.
  - Communication: suitable for communicating with remote devices, e.g. Modem, Network Interface Card etc.

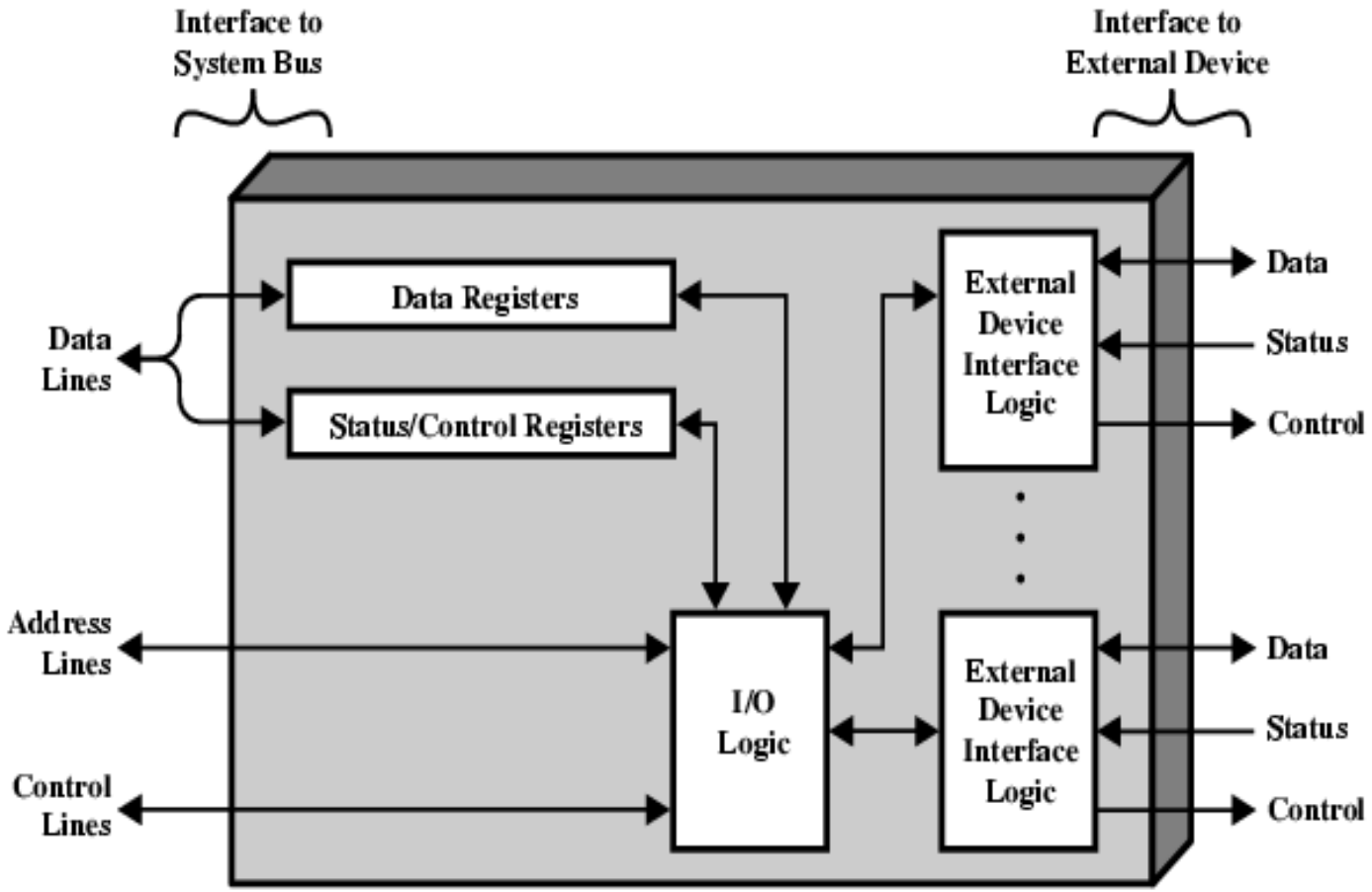
Block Diagram



I/O Modules

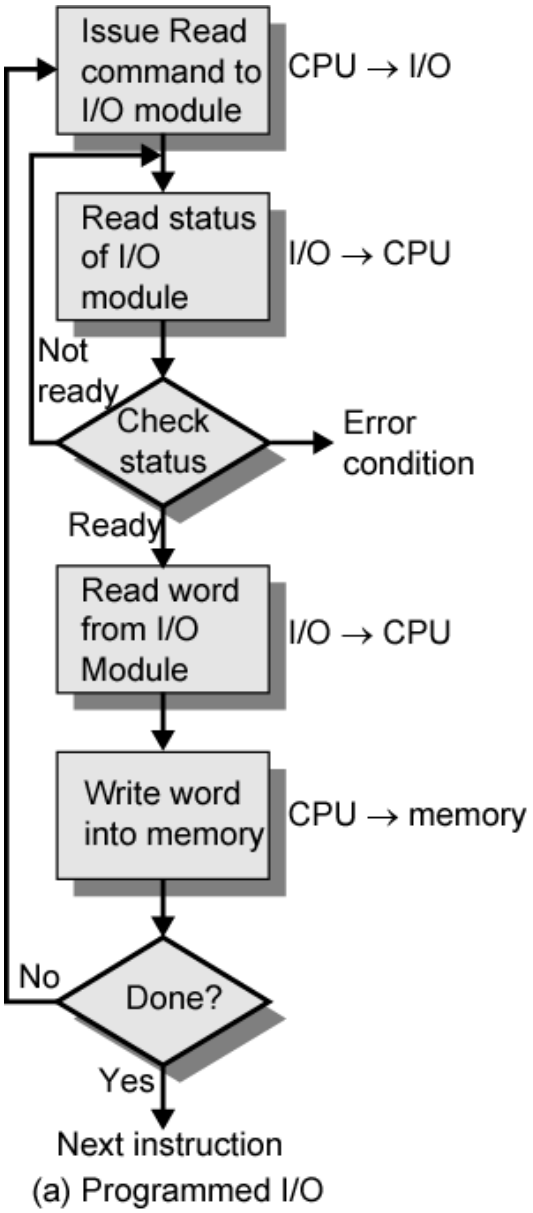
- I/O Module Function
  - Control and timing
  - Processor communication
  - Device communication
  - Data buffering
  - Error detection

I/O Module structure

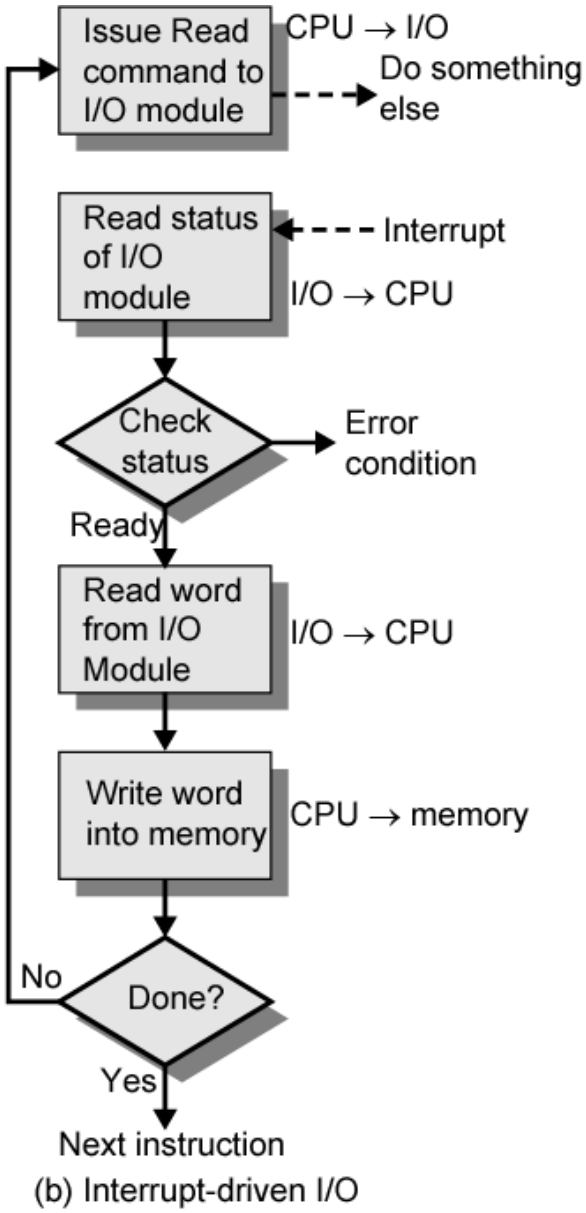


I/O Operations

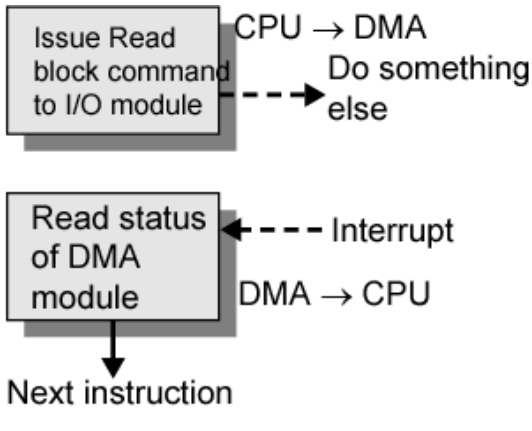
- Three techniques
  - Programmed I/O
  - Interrupt-driven I/O



(a) Programmed I/O



(b) Interrupt-driven I/O



(c) Direct memory access

Programmed I/O

- Programmed I/O instructions are the result of I/O instructions written in computer program. Each data item transfer is initiated by the instruction in the program.
- Usually the program controls data transfer to and from CPU and peripheral. Transferring data under programmed I/O requires constant monitoring of the peripherals by the CPU.

- I/O module performs requested action
- I/O module never alerts processor about the action taken
- It is processor's responsibility

I/O Commands

- To execute I/O instruction processor issues:
  - Address of particular I/O module & peripheral
  - An I/O command: Four types
    - Control: To activate a peripheral & tell what to do
    - Test: To test status of I/O module as well as peripheral
    - Read: Peripheral->I/O module buffer->Processor
    - Write: Processor->I/O module Buffer->Peripheral

I/O Instructions

- With programmed I/O, there is close correspondence between the I/O related instructions and I/O commands issues to an I/O module
- I/O command contains unique address of desired device
- Two modes of addressing I/O
  - Memory-mapped I/O
  - Isolated

Interrupt Driven I/O

- In the programmed I/O method the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer. This is time consuming process because it keeps the processor busy needlessly.
- This problem can be overcome by using **interrupt initiated I/O**. In this when the interface determines that the peripheral is ready for data transfer, it generates an interrupt. After receiving the interrupt signal, the CPU stops the task which it is processing and service the I/O transfer and then returns back to its previous processing task.
- Processor issues I/O command and then do some other useful work
- To request service, I/O module interrupts processor

How this work

I/O Module point of view

- Receives READ command from processor
- Proceeds to read peripheral
- Interrupts processor once data are available
- Wait for data transfer request from processor
- When requested, places data on data bus

CPU point of view

- Issues READ command to module
- Do some other useful work
- Check for interrupt at the end of each instruction cycle
- If interrupted:
  - Save context (registers)
  - Process interrupt
    - Fetch data & store

- Simple Interrupt Processing