UNIT 3: MEMORY SYSTEM

BASIC CONCEPTS

• Maximum size of memory that can be used in any computer is determined by addressing mode.

Address	Memory Locations
16 Bit	$2^{16} = 64$ K
32 Bit	$2^{32} = 4$G (Giga)
40 Bit	$2^{40} = 1$T (Tera)

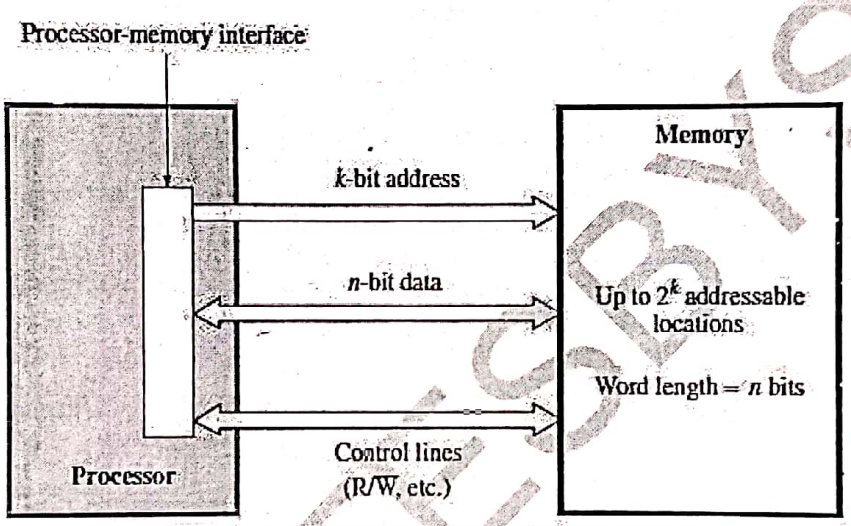


Figure 8.1 Connection of the memory to the processor.

• If MAR is k-bits long then
  → memory may contain upto 2^K addressable-locations
• If MDR is n-bits long, then
  → n-bits of data are transferred between the memory and processor.
• The data-transfer takes place over the processor-bus (Figure 8.1).
• The processor-bus has
  1) Address-Line
  2) Data-line &
  3) Control-Line (R/W', MFC – Memory Function Completed).
• The Control-Line is used for coordinating data-transfer.
• The processor reads the data from the memory by
  → loading the address of the required memory-location into MAR and
  → setting the R/W' line to 1.
• The memory responds by
  → placing the data from the addressed-location onto the data-lines and
  → confirms this action by asserting MFC signal.
• Upon receipt of MFC signal, the processor loads the data from the data-lines into MDR.
• The processor writes the data into the memory-location by
  → loading the address of this location into MAR &
  → setting the R/W' line to 0.
• **Memory Access Time:** It is the time that elapses between
  → initiation of an operation &
  → completion of that operation.
• **Memory Cycle Time:** It is the minimum time delay that required between the initiation of the two successive memory-operations.

RAM (Random Access Memory)

- In RAM, any location can be accessed for a Read/Write-operation in fixed amount of time,

Cache Memory

- It is a small, fast memory that is inserted between
  - → larger slower main-memory and
  - → processor.
- It holds the currently active segments of a program and their data.

Virtual Memory

- The address generated by the processor is referred to as a **virtual/logical address.**
- The virtual-address-space is mapped onto the physical-memory where data are actually stored.
- The mapping-function is implemented by MMU. (MMU = memory management unit).
- Only the active portion of the address-space is mapped into locations in the physical-memory.
- The remaining virtual-addresses are mapped onto the bulk storage devices such as magnetic disk.
- As the active portion of the virtual-address-space changes during program execution, the MMU
  - → changes the mapping-function &
  - → transfers the data between disk and memory.
- During every memory-cycle, MMU determines whether the addressed-page is in the memory.
  - If the page is in the memory.
    - Then, the proper word is accessed and execution proceeds.
      - Otherwise, a page containing desired word is transferred from disk to memory.

- Memory can be classified as follows:
  1) RAM which can be further classified as follows:
     i) Static RAM
     ii) Dynamic RAM (DRAM) which can be further classified as synchronous & asynchronous DRAM.
  2) ROM which can be further classified as follows:
     i) PROM
     II) EPROM
     iii) EEPROM &
     iv) Flash Memory which can be further classified as Flash Cards & Flash Drives.

SEMI CONDUCTOR RAM MEMORIES

INTERNAL ORGANIZATION OF MEMORY-CHIPS

- Memory-cells are organized in the form of array (Figure 8.2).
- Each cell is capable of storing 1-bit of information.
- Each row of cells forms a memory-word.
- All cells of a row are connected to a common line called as **Word-Line.**
- The cells in each column are connected to **Sense/Write** circuit by 2-bit-lines.
- The Sense/Write circuits are connected to data-input or output lines of the chip.
- During a write-operation, the sense/write circuit
  - → receive input information &
  - → store input info in the cells of the selected word.

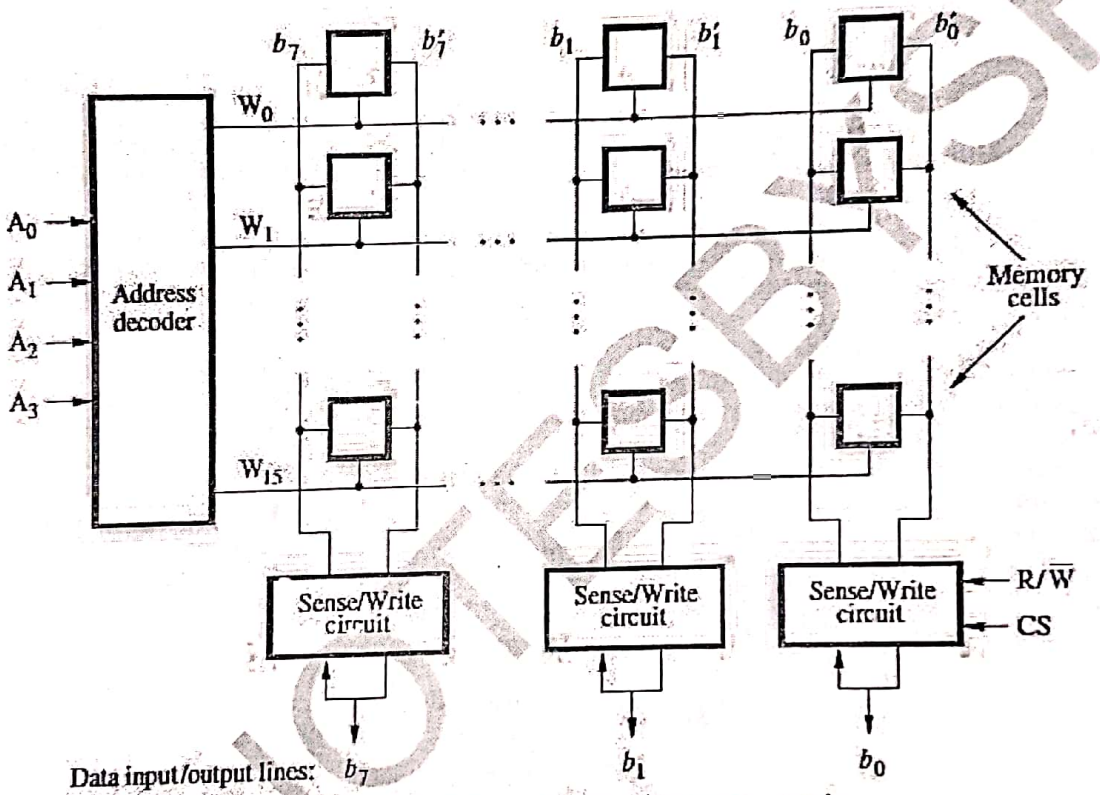


Figure 8.2 Organization of bit cells in a memory chip.

- The data-input and data-output of each Sense/Write circuit are connected to a single bidirectional data-line.
- Data-line can be connected to a data-bus of the computer.
- Following 2 control lines are also used:
  - **1) R/W'** → Specifies the required operation.
  - **2) CS'** → Chip Select input selects a given chip in the multi-chip memory-system.

Bit Organization	Requirement of external connection for address, data and control lines
128 (16x8)	14
(1024) 128x8(1k)	19

STATIC RAM (OR MEMORY)

• Memories consist of circuits capable of retaining their state as long as power is applied are known.

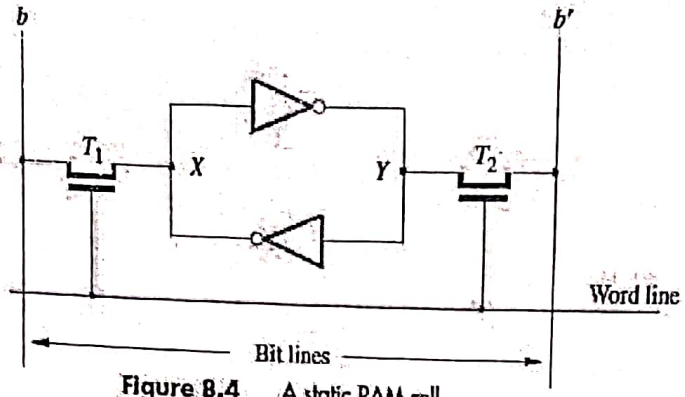


Figure 8.4 A static RAM cell.

• Two inverters are cross connected to form a latch (Figure 8.4).
• The latch is connected to 2-bit-lines by transistors T_1 and T_2.
• The transistors act as switches that can be opened/closed under the control of the word-line.
• When the word-line is at ground level, the transistors are turned off and the latch retain its state.

Read Operation

• To read the state of the cell, the word-line is activated to close switches T_1 and T_2.
• If the cell is in state 1, the signal on bit-line b is high and the signal on the bit-line b' is low.
• Thus, b and b' are complement of each other.
• Sense/Write circuit
  → monitors the state of b & b' and
  → sets the output accordingly.

Write Operation

• The state of the cell is set by
  → placing the appropriate value on bit-line b and its complement on b' and
  → then activating the word-line. This forces the cell into the corresponding state.
• The required signal on the bit-lines is generated by Sense/Write circuit.

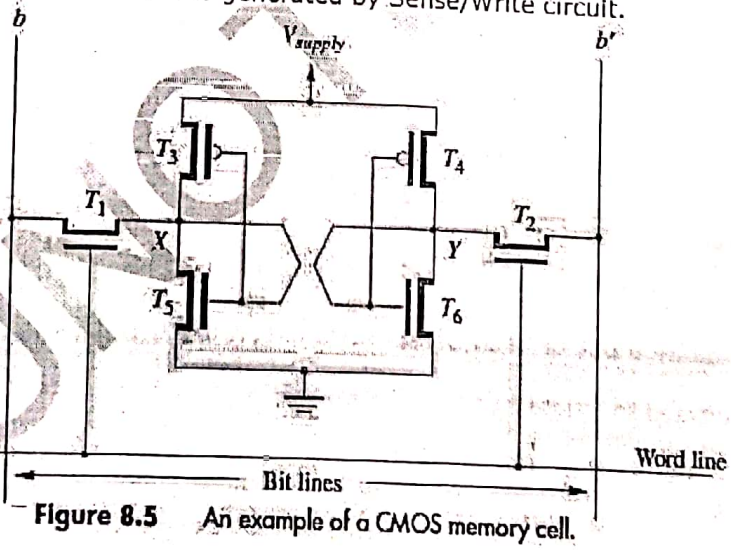


Figure 8.5 An example of a CMOS memory cell.

CMOS Cell

• Transistor pairs (T_3, T_5) and (T_4, T_6) form the inverters in the latch (Figure 8.5).
• In state 1, the voltage at point X is high by having T_5, T_6 ON and T_4, T_5 are OFF.
• Thus, T_1 and T_2 returned ON (Closed), bit-line b and b' will have high and low signals respectively.
• **Advantages:**
  1) It has low power consumption '.' the current flows in the cell only when the cell is active.
  2) Static RAM's can be accessed quickly. It access time is few nanoseconds.
• **Disadvantage:** SRAMs are said to be volatile memories '.' their contents are lost when power is interrupted.

ASYNCHRONOUS DRAM

- Less expensive RAMs can be implemented if simple cells are used.
- Such cells cannot retain their state indefinitely. Hence they are called **Dynamic RAM (DRAM).**
- The information stored in a dynamic memory-cell in the form of a charge on a capacitor.
- This charge can be maintained only for tens of milliseconds.
- The contents must be periodically refreshed by restoring this capacitor charge to its full value.

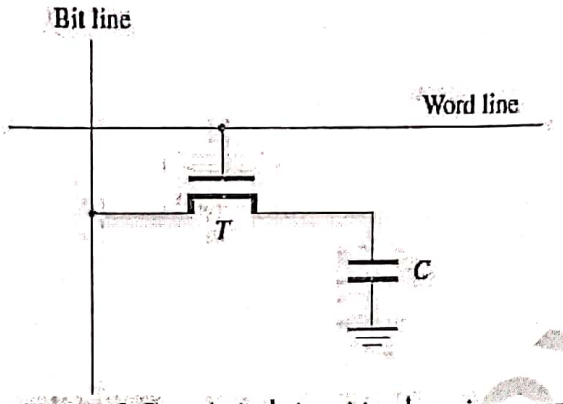


Figure 8.6 A single-transistor dynamic memory cell.

- In order to store information in the cell, the transistor T is turned 'ON' (Figure 8.6).
- The appropriate voltage is applied to the bit-line which charges the capacitor.
- After the transistor is turned off, the capacitor begins to discharge.
- Hence, info. stored in cell can be retrieved correctly before threshold value of capacitor drops down.
- During a read-operation,
  - → transistor is turned 'ON'
  - → a sense amplifier detects whether the charge on the capacitor is above the threshold value.
    - ➢ If (charge on capacitor) > (threshold value) → Bit-line will have logic value '1'.
    - ➢ If (charge on capacitor) < (threshold value) → Bit-line will set to logic value '0'.

ASYNCHRONOUS DRAM DESCRIPTION

- The 4 bit cells in each row are divided into 512 groups of 8 (Figure 5.7).
- 21 bit address is needed to access a byte in the memory. 21 bit is divided as follows:
    1) 12 address bits are needed to select a row.
        i.e. A_{8-0} → specifies row-address of a byte.
    2) 9 bits are needed to specify a group of 8 bits in the selected row.
        i.e. A_{20-9} → specifies column-address of a byte.

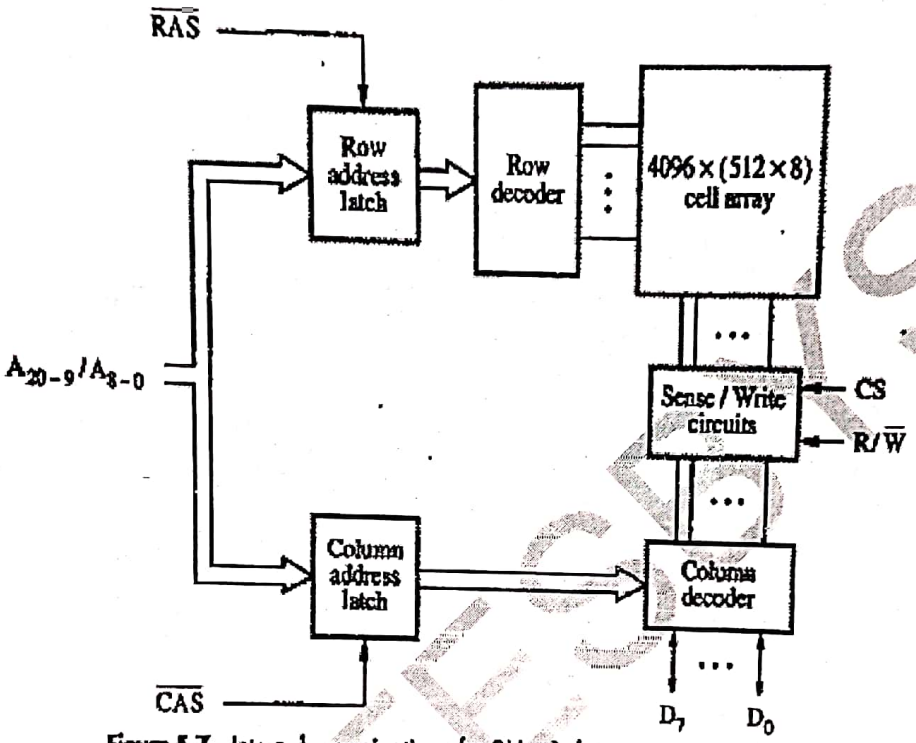


Figure 5.7 Internal organization of a 2M × 8 dynamic memory chip.

- During Read/Write-operation,
    → row-address is applied first.
    → row-address is loaded into row-latch in response to a signal pulse on **RAS'** input of chip.
    (RAS = Row-address Strobe CAS = Column-address Strobe)
- When a Read-operation is initiated, all cells on the selected row are read and refreshed.
- Shortly after the row-address is loaded, the column-address is
    → applied to the address pins &
    → loaded into **CAS'.**
- The information in the latch is decoded.
- The appropriate group of 8 Sense/Write circuits is selected.
    R/W'=1(read-operation) → Output values of selected circuits are transferred to data-lines D_0-D_7.
    R/W'=0(write-operation) → Information on D_0-D_7 are transferred to the selected circuits.
- RAS' & CAS' are active-low so that they cause latching of address when they change from high to low.
- To ensure that the contents of DRAMs are maintained, each row of cells is accessed periodically.
- A special memory-circuit provides the necessary control signals RAS' & CAS' that govern the timing.
- The processor must take into account the delay in the response of the memory.

Fast Page Mode

- Transferring the bytes in sequential order is achieved by applying the consecutive sequence of column-address under the control of successive CAS' signals.
- This scheme allows transferring a block of data at a faster rate.
- The block of transfer capability is called as *fast page mode.*

SYNCHRONOUS DRAM

- The operations are directly synchronized with clock signal (Figure 8.8).
- The address and data connections are buffered by means of registers.
- The output of each sense amplifier is connected to a latch.
- A Read-operation causes the contents of all cells in the selected row to be loaded in these latches.
- Data held in latches that correspond to selected columns are transferred into data-output register.
- Thus, data becoming available on the data-output pins.

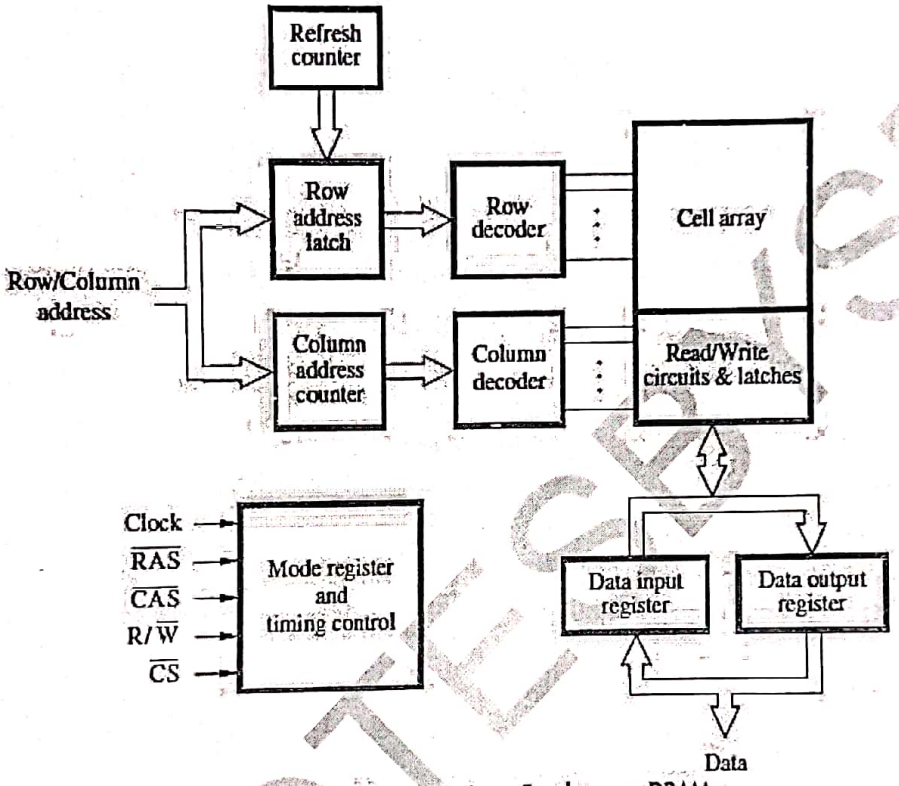


Figure 8.8 Synchronous DRAM.

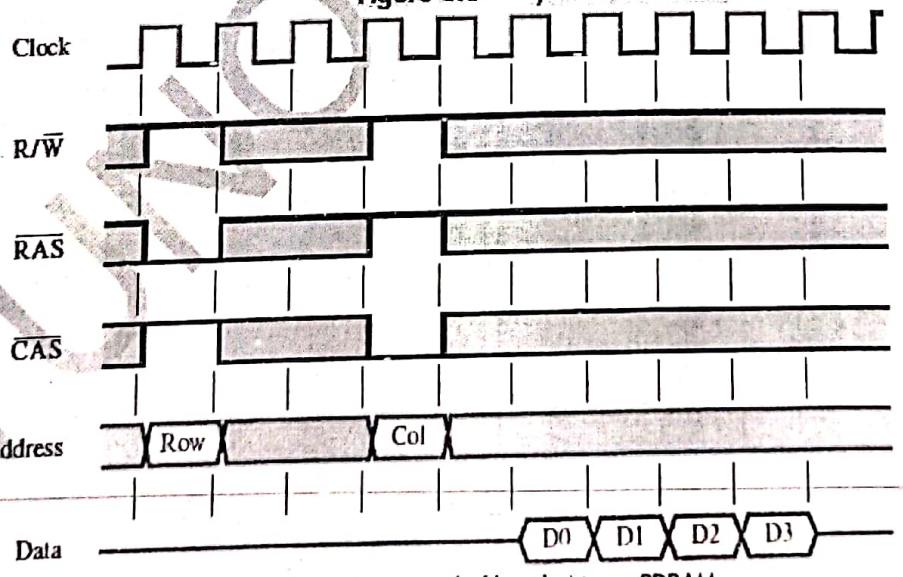


Figure 8.9 A burst read of length 4 in an SDRAM.

- First, the row-address is latched under control of RAS' signal (Figure 8.9).
- The memory typically takes 2 or 3 clock cycles to activate the selected row.
- Then, the column-address is latched under the control of CAS' signal.
- After a delay of one clock cycle, the first set of data bits is placed on the data-lines.
- SDRAM automatically increments column-address to access next 3 sets of bits in the selected row.

LATENCY & BANDWIDTH

• A good indication of performance is given by 2 parameters: 1) Latency 2) Bandwidth.

Latency

• It refers to the amount of time it takes to transfer a word of data to or from the memory.
• For a transfer of single word, the latency provides the complete indication of memory performance.
• For a block transfer, the latency denotes the time it takes to transfer the first word of data.

Bandwidth

• It is defined as the number of bits or bytes that can be transferred in one second.
• Bandwidth mainly depends on
    1) The speed of access to the stored data &
    2) The number of bits that can be accessed in parallel.

DOUBLE DATA RATE SDRAM (DDR-SDRAM)

• The standard SDRAM performs all actions on the rising edge of the clock signal.
• The DDR-SDRAM transfer data on both the edges (loading edge, trailing edge).
• The Bandwidth of DDR-SDRAM is doubled for long burst transfer.
• To make it possible to access the data at high rate, the cell array is organized into two banks.
• Each bank can be accessed separately.
• Consecutive words of a given block are stored in different banks.
• Such interleaving of words allows simultaneous access to two words.
• The two words are transferred on successive edge of the clock.

STRUCTURE OF LARGER MEMORIES

Dynamic Memory System

• The physical implementation is done in the form of memory-modules.
• If a large memory is built by placing DRAM chips directly on the Motherboard, then it will occupy large amount of space on the board.
• These packaging consideration have led to the development of larger memory units known as SIMM's & DIMM's.
    1) SIMM → Single Inline memory-module
    2) DIMM → Dual Inline memory-module
• SIMM/DIMM consists of many memory-chips on small board that plugs into a socket on motherboard.

MEMORY-SYSTEM CONSIDERATION

MEMORY CONTROLLER

- To reduce the number of pins, the dynamic memory-chips use multiplexed-address inputs.
- The address is divided into 2 parts:
  - **1) High Order Address Bit**
    - ➢ Select a row in cell array.
    - ➢ It is provided first and latched into memory-chips under the control of RAS' signal.
  - **2) Low Order Address Bit**
    - ➢ Selects a column.
    - ➢ They are provided on same address pins and latched using CAS' signals.
- The Multiplexing of address bit is usually done by **Memory Controller Circuit** (Figure 5.11).

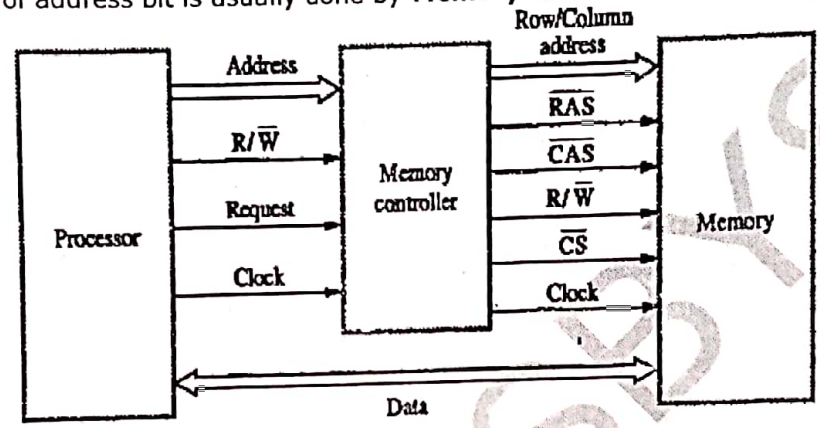


Figure 5.11 Use of a memory controller.

- The Controller accepts a complete address & R/W' signal from the processor.
- A Request signal indicates a memory access operation is needed.
- Then, the Controller
  - → forwards the row & column portions of the address to the memory.
  - → generates RAS' & CAS' signals &
  - → sends R/W' & CS' signals to the memory.

RAMBUS MEMORY

- The usage of wide bus is expensive.
- Rambus developed the implementation of narrow bus.
- Rambus technology is a fast signaling method used to transfer information between chips.
- The signals consist of much smaller voltage swings around a reference voltage V_{ref}.
- The reference voltage is about 2V.
- The two logical values are represented by 0.3V swings above and below V_{ref}.
- This type of signaling is generally is known as **Differential Signalling**.
- Rambus provides a complete specification for design of communication called as **Rambus Channel**.
- Rambus memory has a clock frequency of 400 MHz.
- The data are transmitted on both the edges of clock so that effective data-transfer rate is 800MHZ.
- Circuitry needed to interface to Rambus channel is included on chip. Such chips are called **RDRAM.** (RDRAM = Rambus DRAMs).
- Rambus channel has:
  1) 9 Data-lines (1^{st}-8^{th} line ->Transfer the data, 9^{th} line->Parity checking).
  2) Control-Line &
  3) Power line.
- A two channel rambus has 18 data-lines which has no separate Address-Lines.
- Communication between processor and RDRAM modules is carried out by means of packets transmitted on the data-lines.
- There are 3 types of packets:
  1) Request
  2) Acknowledge &
  3) Data.

READ ONLY MEMORY (ROM)

- Both SRAM and DRAM chips are volatile, i.e. They lose the stored information if power is turned off.
- Many application requires non-volatile memory which retains the stored information if power is turned off.
- For ex:

  OS software has to be loaded from disk to memory i.e. it requires non-volatile memory.
- Non-volatile memory is used in embedded system.
- Since the normal operation involves only reading of stored data, a memory of this type is called ROM.
  - **At Logic value '0'** → Transistor(T) is connected to the ground point (P).
    Transistor switch is closed & voltage on bit-line nearly drops to zero (Figure 8.11).
  - **At Logic value '1'** → Transistor switch is open.
    The bit-line remains at high voltage.

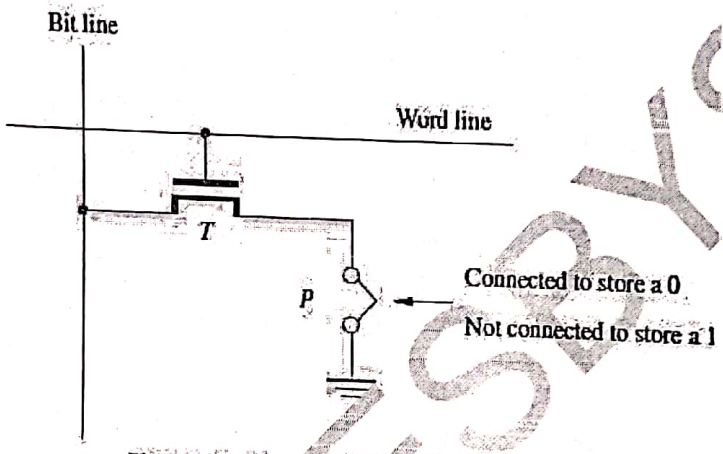


Figure 8.11 A ROM cell.

- To read the state of the cell, the word-line is activated.
- A Sense circuit at the end of the bit-line generates the proper output value.

TYPES OF ROM

- Different types of non-volatile memory are
  1) PROM
  2) EPROM
  3) EEPROM &
  4) Flash Memory (Flash Cards & Flash Drives)

PROM (PROGRAMMABLE ROM)

- PROM allows the data to be loaded by the user.
- Programmability is achieved by inserting a 'fuse' at point P in a ROM cell.
- Before PROM is programmed, the memory contains all 0's.
- User can insert 1's at required location by burning-out fuse using high current-pulse.
- This process is irreversible.
- **Advantages:**
  1) It provides flexibility.
  2) It is faster.
  3) It is less expensive because they can be programmed directly by the user.

EPROM (ERASABLE REPROGRAMMABLE ROM)

- EPROM allows
  - → stored data to be erased and
  - → new data to be loaded.
- In cell, a connection to ground is always made at 'P' and a special transistor is used.
- The transistor has the ability to function as
  - → a normal transistor or
  - → a disabled transistor that is always turned 'off'.
- Transistor can be programmed to behave as a permanently open switch, by injecting charge into it.
- Erasure requires dissipating the charges trapped in the transistor of memory-cells.
  This can be done by exposing the chip to ultra-violet light.
- **Advantages:**
  1) It provides flexibility during the development-phase of digital-system.
  2) It is capable of retaining the stored information for a long time.
- **Disadvantages:**
  1) The chip must be physically removed from the circuit for reprogramming.
  2) The entire contents need to be erased by UV light.

EEPROM (ELECTRICALLY ERASABLE ROM)

- **Advantages:**
  1) It can be both programmed and erased electrically.
  2) It allows the erasing of all cell contents selectively.
- **Disadvantage:** It requires different voltage for erasing, writing and reading the stored data.

FLASH MEMORY

- In EEPROM, it is possible to read & write the contents of a single cell.
- In Flash device, it is possible to read contents of a single cell & write entire contents of a block.
- Prior to writing, the previous contents of the block are erased.
  Eg. In MP3 player, the flash memory stores the data that represents sound.
- Single flash chips cannot provide sufficient storage capacity for embedded-system.
- **Advantages:**
  1) Flash drives have greater density which leads to higher capacity & low cost per bit.
  2) It requires single power supply voltage & consumes less power.
- There are 2 methods for implementing larger memory: 1) Flash Cards & 2) Flash Drives

1) Flash Cards

- ➢ One way of constructing larger module is to mount flash-chips on a small card.
- ➢ Such flash-card have standard interface.
- ➢ The card is simply plugged into a conveniently accessible slot.
- ➢ Memory-size of the card can be 8, 32 or 64MB.
- ➢ Eg: A minute of music can be stored in 1MB of memory. Hence 64MB flash cards can store an hour of music.

2) Flash Drives

- ➢ Larger flash memory can be developed by replacing the hard disk-drive.
- ➢ The flash drives are designed to fully emulate the hard disk.
- ➢ The flash drives are solid state electronic devices that have no movable parts.

Advantages:

1) They have shorter seek & access time which results in faster response.
2) They have low power consumption. ∴ they are attractive for battery driven application.
3) They are insensitive to vibration.

Disadvantages:

1) The capacity of flash drive (<1GB) is less than hard disk (>1GB).
2) It leads to higher cost per bit.
3) Flash memory will weaken after it has been written a number of times (typically at least 1 million times).

SPEED, SIZE COST

Characteristics	SRAM	DRAM	Magnetis Disk
Speed	Very Fast	Slower	Much slower than DRAM
Size	Large	Small	Small
Cost	Expensive	Less Expensive	Low price

Memory	Speed	Size	Cost
Registers	Very high	Lower	Very Lower
Primary cache	High	Lower	Low
Secondary cache	Low	Low	Low
Main memory	Lower than Seconadry cache	High	High
Secondary Memory	Very low	Very High	Very High

- The main-memory can be built with DRAM (Figure 8.14)
- Thus, SRAM's are used in smaller units where speed is of essence.
- The Cache-memory is of 2 types:
  - **1) Primary/Processor Cache** (Level1 or L1 cache)
    - ➢ It is always located on the processor-chip.
  - **2) Secondary Cache** (Level2 or L2 cache)
    - ➢ It is placed between the primary-cache and the rest of the memory.
- The memory is implemented using the dynamic components (SIMM, RIMM, DIMM).
- The access time for main-memory is about 10 times longer than the access time for L1 cache.

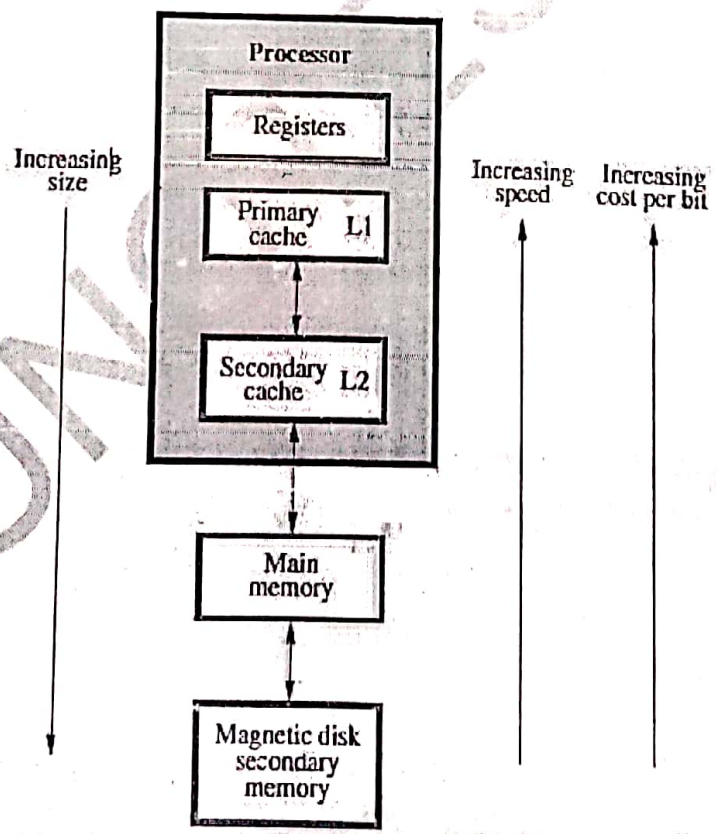


Figure 8.14 Memory hierarchy.

CACHE MEMORIES

- The effectiveness of cache mechanism is based on the property of **'Locality of Reference'.**

Locality of Reference

- Many instructions in the localized areas of program are executed repeatedly during some time period
- Remainder of the program is accessed relatively infrequently (Figure 8.15).
- There are 2 types:
  - **1) Temporal**
    - ➢ The recently executed instructions are likely to be executed again very soon.
  - **2) Spatial**
    - ➢ Instructions in close proximity to recently executed instruction are also likely to be executed soon.
- If active segment of program is placed in cache-memory, then total execution time can be reduced.
- **Block** refers to the set of contiguous address locations of some size.
- The cache-line is used to refer to the cache-block.

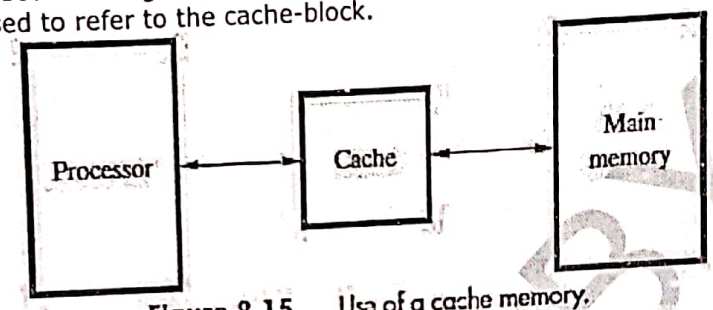


Figure 8.15 Use of a cache memory.

- The Cache-memory stores a reasonable number of blocks at a given time.
- This number of blocks is small compared to the total number of blocks available in main-memory.
- Correspondence b/w main-memory-block & cache-memory-block is specified by mapping-function.
- Cache control hardware decides which block should be removed to create space for the new block.
- The collection of rule for making this decision is called the **Replacement Algorithm.**
- The cache control-circuit determines whether the requested-word currently exists in the cache.
- The write-operation is done in 2 ways: 1) Write-through protocol & 2) Write-back protocol.
  - **Write-Through Protocol**
    - ➢ Here the cache-location and the main-memory-locations are updated simultaneously.
  - **Write-Back Protocol**
    - ➢ This technique is to
      - → update only the cache-location &
      - → mark the cache-location with associated flag bit called **Dirty/Modified Bit.**
    - ➢ The word in memory will be updated later, when the marked-block is removed from cache.

During Read-operation

- If the requested-word currently not exists in the cache, then **read-miss** will occur.
- To overcome the read miss, *Load-through/Early restart protocol* is used.
  - **Load–Through Protocol**
    - ➢ The block of words that contains the requested-word is copied from the memory into cache.
    - ➢ After entire block is loaded into cache, the requested-word is forwarded to processor.

During Write-operation

- If the requested-word not exists in the cache, then **write-miss** will occur.
  - 1) If **Write Through Protocol** is used, the information is written directly into main-memory.
  - 2) If **Write Back Protocol** is used,
    - → then block containing the addressed word is first brought into the cache &
    - → then the desired word in the cache is over-written with the new information.

MAPPING-FUNCTION

- Here we discuss about 3 different mapping-function:
  - 1) Direct Mapping
  - 2) Associative Mapping
  - 3) Set-Associative Mapping

DIRECT MAPPING

- The block-j of the main-memory maps onto block-j modulo-128 of the cache (Figure 8.16).
- When the memory-blocks 0, 128, & 256 are loaded into cache, the block is stored in cache-block 0.
  Similarly, memory-blocks 1, 129, 257 are stored in cache-block 1.
- The contention may arise when
  1) When the cache is full.
  2) When more than one memory-block is mapped onto a given cache-block position.
- The contention is resolved by
  allowing the new blocks to overwrite the currently resident-block.
- Memory-address determines placement of block in the cache.

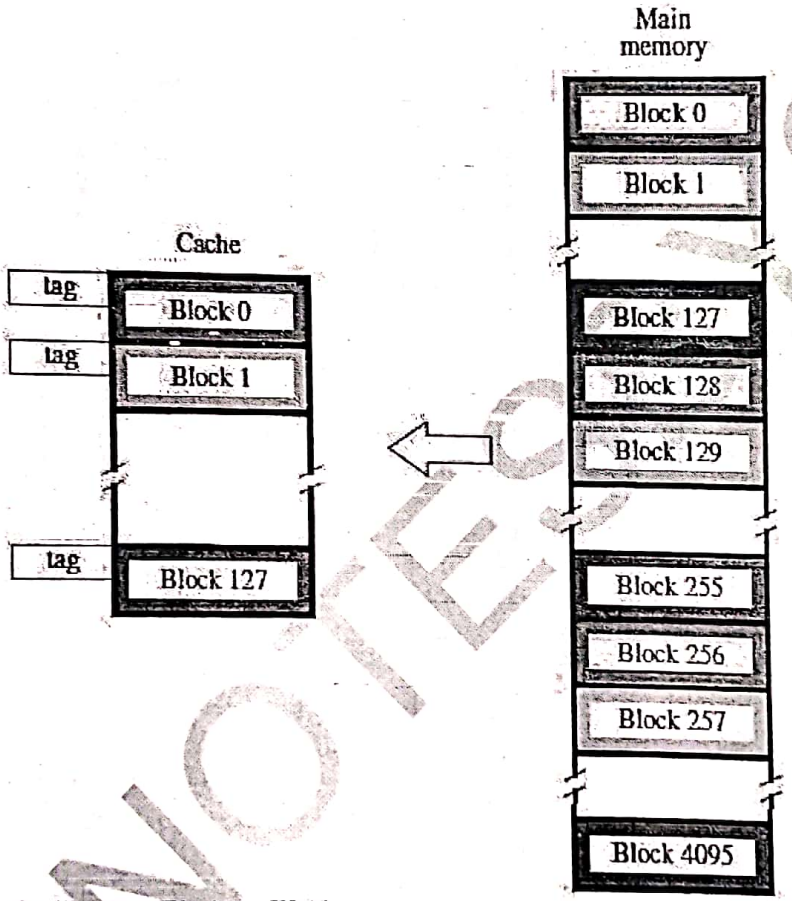


Figure 8.16 Direct-mapped cache.

- The memory-address is divided into 3 fields:
  1) Low Order 4 bit field
  ➢ Selects one of 16 words in a block.
  2) 7 bit cache-block field
  ➢ 7-bits determine the cache-position in which new block must be stored.
  3) 5 bit Tag field
  ➢ 5-bits memory-address of block is stored in 5 tag-bits associated with cache-location.
- As execution proceeds,
  5-bit tag field of memory-address is compared with tag-bits associated with cache-location.
  If they match, then the desired word is in that block of the cache.
  Otherwise, the block containing required word must be first read from the memory.
  And then the word must be loaded into the cache.

ASSOCIATIVE MAPPING

- The memory-block can be placed into any cache-block position. (Figure 8.17).
- 12 tag-bits will identify a memory-block when it is resolved in the cache.
- Tag-bits of an address received from processor are compared to the tag-bits of each block of cache.
- This comparison is done to see if the desired block is present.

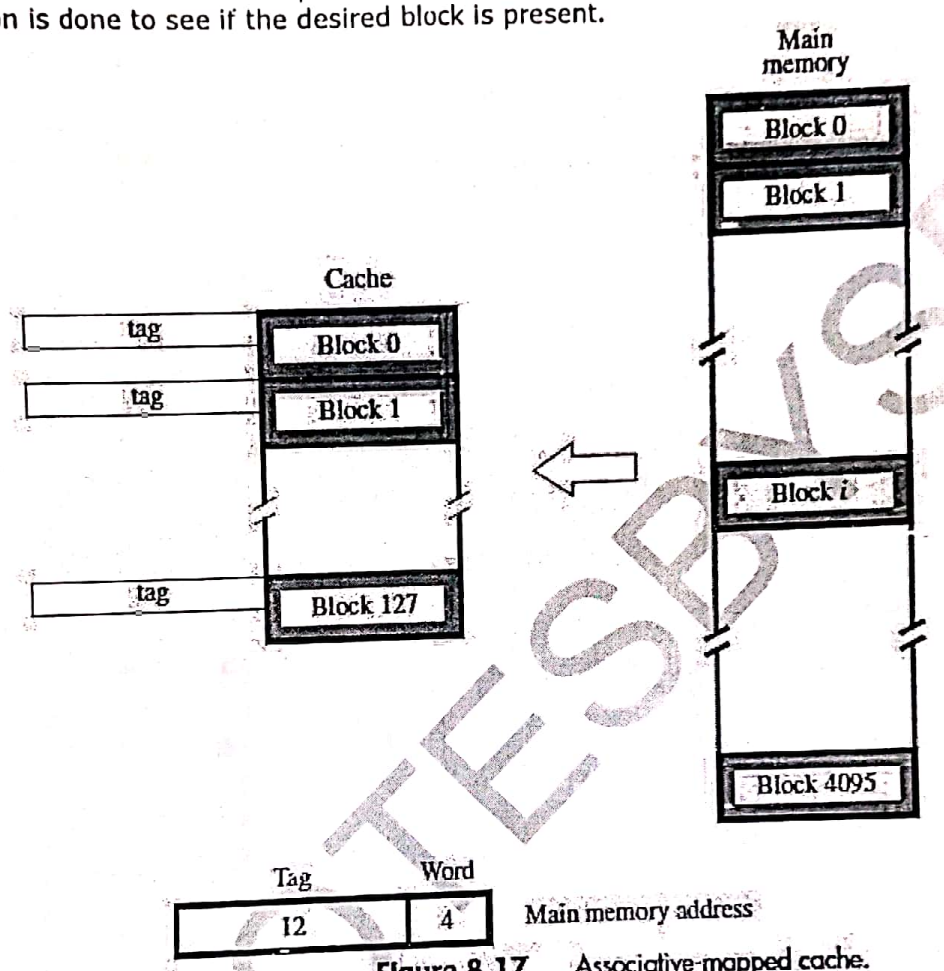


Figure 8.17 Associative-mapped cache.

- It gives complete freedom in choosing the cache-location.
- A new block that has to be brought into the cache has to replace an existing block if the cache is full.
- The memory has to determine whether a given block is in the cache.
- **Advantage:** It is more flexible than direct mapping technique.
- **Disadvantage:** Its cost is high.

SET-ASSOCIATIVE MAPPING

- It is the combination of direct and associative mapping. (Figure 8.18).
- The blocks of the cache are grouped into sets.
- The mapping allows a block of the main-memory to reside in any block of the specified set.
- The cache has 2 blocks per set, so the memory-blocks 0, 64, 128..... 4032 maps into cache set '0'.
- The cache can occupy either of the two block position within the set.

6 bit set field

➢ Determines which set of cache contains the desired block.

6 bit tag field

➢ The tag field of the address is compared to the tags of the two blocks of the set.
➢ This comparison is done to check if the desired block is present.

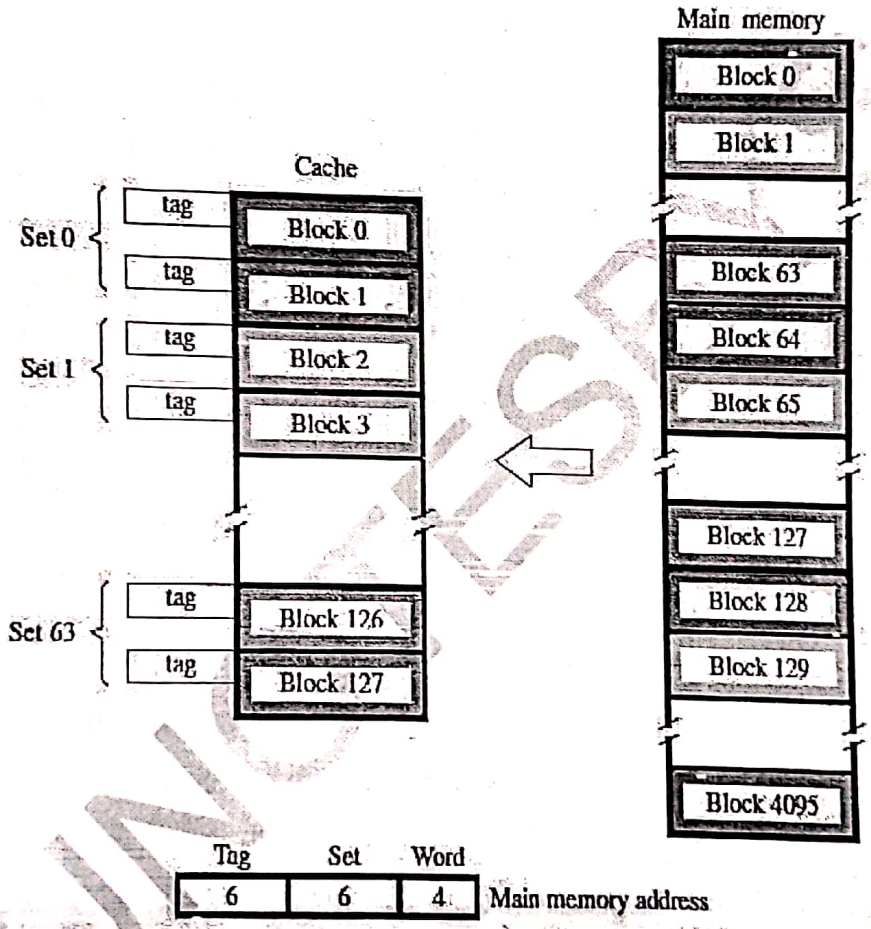


Figure 8.18 Set-associative-mapped cache with two blocks per set.

- The cache which contains 1 block per set is called **direct mapping.**
- A cache that has 'k' blocks per set is called as "**k-way set associative cache**".
- Each block contains a control-bit called a **valid-bit.**
- The Valid-bit indicates that whether the block contains valid-data.
- The dirty bit indicates that whether the block has been modified during its cache residency.

Valid-bit=0 → When power is initially applied to system.
Valid-bit=1 → When the block is loaded from main-memory at first time.

- If the main-memory-block is updated by a source & if the block in the source is already exists in the cache, then the valid-bit will be cleared to "0".
- If Processor & DMA uses the same copies of data then it is called as **Cache Coherence Problem.**
- **Advantages:**

1) Contention problem of direct mapping is solved by having few choices for block placement.
2) The hardware cost is decreased by reducing the size of associative search.

REPLACEMENT ALGORITHM

- In direct mapping method,
  the position of each block is pre-determined and there is no need of replacement strategy.
- In associative & set associative method,
  The block position is not pre-determined.
  If the cache is full and if new blocks are brought into the cache,
  then the cache-controller must decide which of the old blocks has to be replaced.
- When a block is to be overwritten, the block with longest time w/o being referenced is over-written.
- This block is called **Least recently Used (LRU) block** & the technique is called **LRU algorithm.**
- The cache-controller tracks the references to all blocks with the help of block-counter.
- **Advantage:** Performance of LRU is improved by randomness in deciding which block is to be over-written.

Eg:
Consider 4 blocks/set in set associative cache.

- 2 bit counter can be used for each block.
- When a **'hit'** occurs, then block counter=0; The counter with values originally lower than the referenced one are incremented by 1 & all others remain unchanged.
- When a **'miss'** occurs & if the set is full, the blocks with the counter value 3 is removed, the new block is put in its place & its counter is set to "0" and other block counters are incremented by 1.

PERFORMANCE CONSIDERATION

- Two key factors in the commercial success are 1) performance & 2) cost.
- In other words, the best possible performance at low cost.
- A common measure of success is called the **Pricel Performance ratio.**
- Performance depends on
  - → how fast the machine instructions are brought to the processor &
  - → how fast the machine instructions are executed.
- To achieve parallelism, *interleaving* is used.
- Parallelism means both the slow and fast units are accessed in the same manner.

INTERLEAVING

- The main-memory of a computer is structured as a collection of physically separate modules.
- Each module has its own
  - 1) ABR (address buffer register) &
  - 2) DBR (data buffer register).
- So, memory access operations may proceed in more than one module at the same time (Fig 5.25).
- Thus, the aggregate-rate of transmission of words to/from the main-memory can be increased.

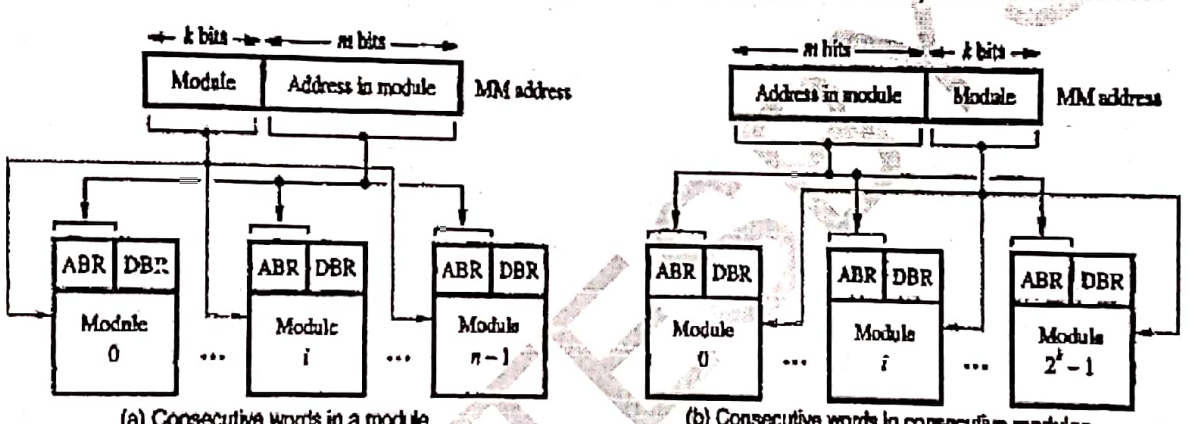


Figure 5.25 Addressing multiple-module memory systems.

- The low-order k-bits of the memory-address select a module.
  While the high-order m-bits name a location within the module.
  In this way, consecutive addresses are located in successive modules.
- Thus, any component of the system can keep several modules busy at any one time T.
- This results in both
  - → faster access to a block of data and
  - → higher average utilization of the memory-system as a whole.
- To implement the interleaved-structure, there must be 2k modules;
  Otherwise, there will be gaps of non-existent locations in the address-space.

Hit Rate & Miss Penalty

- The number of hits stated as a fraction of all attempted accesses is called the **Hit Rate.**
- The extra time needed to bring the desired information into the cache is called the **Miss Penalty.**
- High hit rates well over 0.9 are essential for high-performance computers.
- Performance is adversely affected by the actions that need to be taken when a miss occurs.
- A performance penalty is incurred because
  of the extra time needed to bring a block of data from a slower unit to a faster unit.
- During that period, the processor is stalled waiting for instructions or data.
- We refer to the total access time seen by the processor when a miss occurs as the miss penalty.
- Let h be the hit rate, M the miss penalty, and C the time to access information in the cache. Thus, the average access time experienced by the processor is

$$t_{avg} = hC + (1 - h)M$$

Problem 1:
Consider the dynamic memory cell. Assume that C = 30 femtofarads (10^{-15} F) and that leakage current through the transistor is about 0.25 picoamperes (10^{-12} A). The voltage across the capacitor when it is fully charged is 1.5 V. The cell must be refreshed before this voltage drops below 0.9 V. Estimate the minimum refresh rate.

Solution:
The minimum refresh rate is given by

$$\frac{50 \times 10^{-15} \times (4.5 - 3)}{9 \times 10^{-12}} = 8.33 \times 10^{-3} \text{ s}$$

Therefore, each row has to be refreshed every 8 ms.

Problem 2:
Consider a main-memory built with SDRAM chips. Data are transferred in bursts & the burst length is 8. Assume that 32 bits of data are transferred in parallel. If a 400-MHz clock is used, how much time does it take to transfer:

(a) 32 bytes of data
(b) 64 bytes of data

What is the latency in each case?

Solution:
(a) It takes 5 + 8 = 13 clock cycles.

$$\text{Total time} = \frac{13}{(133 \times 10^6)} = 0.098 \times 10^{-6} \text{ s} = 98 \text{ ns}$$

$$\text{Latency} = \frac{5}{(133 \times 10^6)} = 0.038 \times 10^{-6} \text{ s} = 38 \text{ ns}$$

(b) It takes twice as long to transfer 64 bytes, because two independent 32-byte transfers have to be made. The latency is the same, i.e. 38 ns.

Problem 3:
Give a critique of the following statement: "Using a faster processor chip results in a corresponding increase in performance of a computer even if the main-memory speed remains the same."

Solution:
A faster processor chip will result in increased performance, but the amount of increase will not be directly proportional to the increase in processor speed, because the cache miss penalty will remain the same if the main-memory speed is not improved.

Problem 4:
A block-set-associative cache consists of a total of 64 blocks, divided into 4-block sets. The main-memory contains 4096 blocks, each consisting of 32 words. Assuming a 32-bit byte-addressable address-space,

(a) how many bits are there in main-memory address
(b) how many bits are there in each of the Tag, Set, and Word fields?

Solution:
(a) 4096 blocks of 128 words each require 12+7 = 19 bits for the main-memory address.
(b) TAG field is 8 bits. SET field is 4 bits. WORD field is 7 bits.

Problem 5:
The cache block size in many computers is in the range of 32 to 128 bytes. What would be the main advantages and disadvantages of making the size of cache blocks larger or smaller?

Solution:
Larger size
- Fewer misses if most of the data in the block are actually used
- Wasteful if much of the data are not used before the cache block is ejected from the cache

Smaller size
- More misses

UNIT 4: ARITHMETIC

NUMBERS, ARITHMETIC OPERATIONS AND CHARACTERS

NUMBER REPRESENTATION

- Numbers can be represented in 3 formats:
  1) Sign and magnitude
  2) 1's complement
  3) 2's complement
- In all three formats, MSB=0 for +ve numbers & MSB=1 for -ve numbers.
- In **sign-and-magnitude system,**
  negative value is obtained by changing the MSB from 0 to 1 of the corresponding positive value.
  For ex, +5 is represented by 0101 &
  -5 is represented by 1101.
- In **1's complement system,**
  negative values are obtained by complementing each bit of the corresponding positive number.
  For ex, -5 is obtained by complementing each bit in 0101 to yield 1010.

(In other words, the operation of forming the 1's complement of a given number is equivalent to subtracting that number from 2^n-1).

- In **2's complement system,**
  forming the 2's complement of a number is done by subtracting that number from 2^n.
  For ex, -5 is obtained by complementing each bit in 0101 & then adding 1 to yield 1011.

(In other words, the 2's complement of a number is obtained by adding 1 to the 1's complement of that number).

- 2's complement system yields the most efficient way to carry out addition/subtraction operations.

B	Values represented		
$b_3b_2b_1b_0$	Sign and magnitude	1's complement	2's complement
0 1 1 1	+7	+7	+7
0 1 1 0	+6	+6	+6
0 1 0 1	+5	+5	+5
0 1 0 0	+4	+4	+4
0 0 1 1	+3	+3	+3
0 0 1 0	+2	+2	+2
0 0 0 1	+1	+1	+1
0 0 0 0	+0	+0	+0
1 0 0 0	−0	−7	−8
1 0 0 1	−1	−6	−7
1 0 1 0	−2	−5	−6
1 0 1 1	−3	−4	−5
1 1 0 0	−4	−3	−4
1 1 0 1	−5	−2	−3
1 1 1 0	−6	−1	−2
1 1 1 1	−7	−0	−1

Figure 1.3 Binary, signed-integer representations.

ADDITION OF POSITIVE NUMBERS

- Consider adding two 1-bit numbers.
- The sum of 1 & 1 requires the 2-bit vector 10 to represent the value 2. We say that sum is 0 and the carry-out is 1.

```
  0      1      0      1
+ 0    + 0    + 1    + 1
---    ---    ---    ---
  0      1      1     10
                      ↑
                   Carry-out
```

Figure 2.2 Addition of 1-bit numbers.

ADDITION & SUBTRACTION OF SIGNED NUMBERS

• Following are the two rules for addition and subtraction of n-bit signed numbers using the 2's complement representation system (Figure 1.6).

Rule 1:

- **To Add** two numbers, add their n-bits and ignore the carry-out signal from the MSB position.
- Result will be algebraically correct, if it lies in the range -2^{n-1} to $+2^{n-1}-1$.

Rule 2:

- **To Subtract** two numbers X and Y (that is to perform X-Y), take the 2's complement of Y and then add it to X as in rule 1.
- Result will be algebraically correct, if it lies in the range (2^{n-1}) to $+(2^{n-1}-1)$.

• When the result of an arithmetic operation is outside the representable-range, an arithmetic overflow is said to occur.

• To represent a signed in 2's complement form using a larger number of bits, repeat the sign bit as many times as needed to the left. This operation is called **sign extension.**

• In 1's complement representation, the result obtained after an addition operation is not always correct. The carry-out(c_n) cannot be ignored. If c_n=0, the result obtained is correct. If c_n=1, then a 1 must be added to the result to make it correct.

OVERFLOW IN INTEGER ARITHMETIC

• When result of an arithmetic operation is outside the representable-range, an **arithmetic overflow** is said to occur.

• For example: If we add two numbers +7 and +4, then the output sum S is 1011(←0111+0100), which is the code for -5, an incorrect result.

• An overflow occurs in following 2 cases

1) Overflow can occur only when adding two numbers that have the same sign.
2) The carry-out signal from the sign-bit position is not a sufficient indicator of overflow when adding signed numbers.

(a)	0010	(+2)	(b)	0100	(+4)
	+ 0011	(+3)		+ 1010	(−6)
	0101	(+5)		1110	(−2)
(c)	1011	(−5)	(d)	0111	(+7)
	+ 1110	(−2)		+ 1101	(−3)
	1001	(−7)		0100	(+4)
(e)	1101	(−3)		1101	
	− 1001	(−7)	⇒	+ 0111	
				0100	(+4)
(f)	0010	(+2)		0010	
	− 0100	(+4)	⇒	+ 1100	
				1110	(−2)
(g)	0110	(+6)		0110	
	− 0011	(+3)	⇒	+ 1101	
				0011	(+3)
(h)	1001	(−7)		1001	
	− 1011	(−5)	⇒	+ 0101	
				1110	(−2)
(i)	1001	(−7)		1001	
	− 0001	(+1)	⇒	+ 1111	
				1000	(−8)
(j)	0010	(+2)		0010	
	− 1101	(−3)	⇒	+ 0011	
				0101	(+5)

Figure 1.6 2's-complement Add and Subtract operations.

ADDITION & SUBTRACTION OF SIGNED NUMBERS

n-BIT RIPPLE CARRY ADDER

- A cascaded connection of n full-adder blocks can be used to add 2-bit numbers.
- Since carries must propagate (or ripple) through cascade, the configuration is called an n-bit ripple carry adder (Figure 9.1).

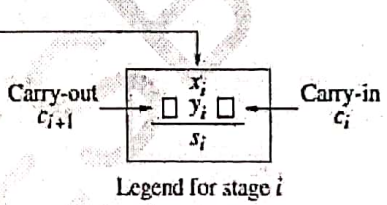
x_i	y_i	Carry-in c_i	Sum s_i	Carry-out c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$s_i = \overline{x}_i\overline{y}_i c_i + \overline{x}_i y_i \overline{c}_i + x_i\overline{y}_i\overline{c}_i + x_i y_i c_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = y_i c_i + x_i c_i + x_i y_i$$

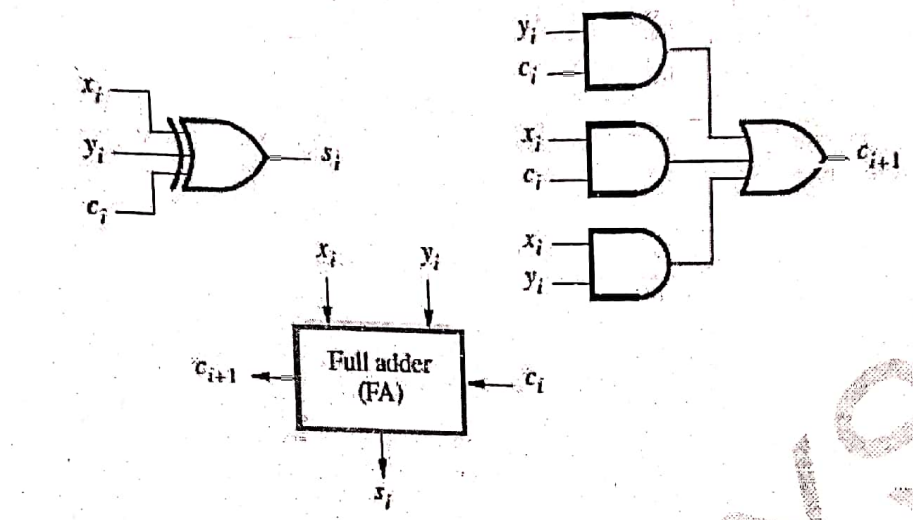
Example:

$$\begin{array}{r} X \\ +Y \\ \hline Z \end{array} = \begin{array}{r} 7 \\ +6 \\ \hline 13 \end{array} = \begin{array}{r} 0\ 1\ \boxed{1}\ 1 \\ +\ 0\ 0\ 1\ 1\ \boxed{1\ 0}\ 0\ 0 \\ \hline 1\ 1\ \boxed{0}\ 1 \end{array}$$

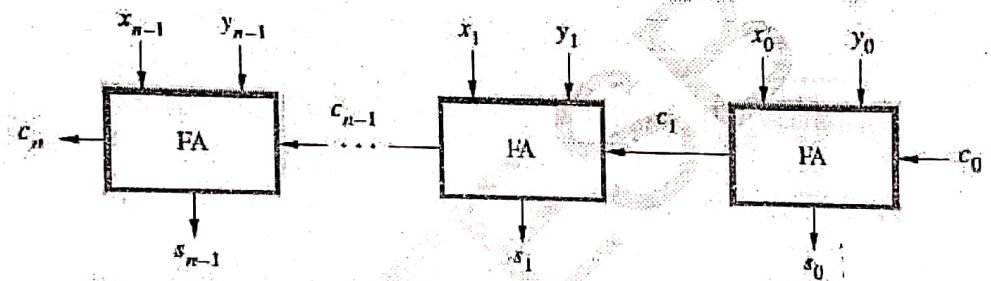


Legend for stage i

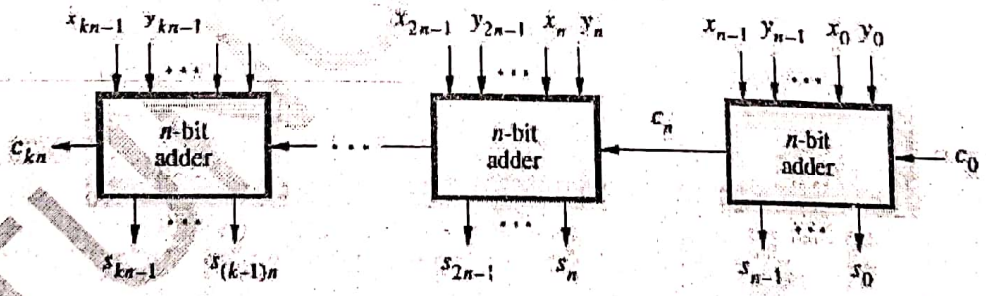
Figure 9.1 Logic specification for a stage of binary addition.



(a) Logic for a single stage



(b) An n-bit ripple-carry adder



(c) Cascade of k n-bit adders

Figure 9.2 Logic for addition of binary numbers.

ADDITION/SUBTRACTION LOGIC UNIT

- The n-bit adder can be used to add 2's complement numbers X and Y (Figure 9.3).
- **Overflow** can only occur when the signs of the 2 operands are the same.
- In order to perform the subtraction operation X-Y on 2's complement numbers X and Y; we form the 2's complement of Y and add it to X.
- Addition or subtraction operation is done based on value applied to the Add/Sub input control-line.
- Control-line=0 for addition, applying the Y vector unchanged to one of the adder inputs.
  Control-line=1 for subtraction, the Y vector is 2's complemented.

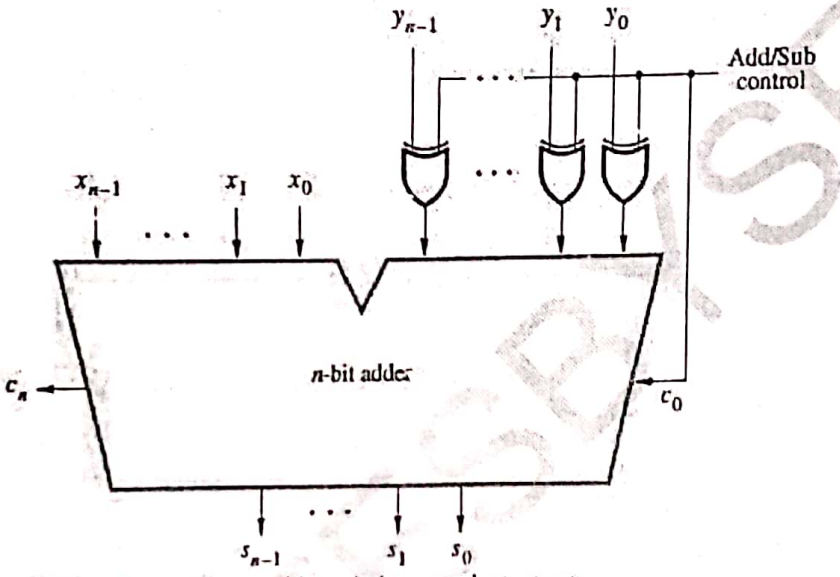


Figure 9.3 Binary addition/subtraction logic circuit.

DESIGN OF FAST ADDERS

- **Drawback of ripple carry adder:** If the adder is used to implement the addition/subtraction, all sum bits are available in 2n gate delays.
- Two approaches can be used to reduce delay in adders:
  1) Use the fastest possible electronic-technology in implementing the ripple-carry design.
  2) Use an augmented logic-gate network structure.

CARRY-LOOKAHEAD ADDITIONS

- The logic expression for s_i(sum) and c_{i+1}(carry-out) of stage i are

  $s_i = x_i + y_i + c_i$ ------(1) $c_{i+1} = x_i y_i + x_i c_i + y_i c_i$ ------(2)

- Factoring (2) into

  $c_{i+1} = x_i y_i + (x_i + y_i) c_i$

  we can write

  $c_{i+1} = G_i + P_i C_i$ where $G_i = x_i y_i$ and $P_i = x_i + y_i$

- The expressions G_i and P_i are called generate and propagate functions (Figure 9.4).
- If $G_i = 1$, then $c_{i+1} = 1$, independent of the input carry c_i. This occurs when both x_i and y_i are 1. Propagate function means that an input-carry will produce an output-carry when either $x_i = 1$ or $y_i = 1$.
- All G_i and P_i functions can be formed independently and in parallel in one logic-gate delay.
- Expanding c_i terms of i-1 subscripted variables and substituting into the c_{i+1} expression, we obtain

  $c_{i+1} = G_i + P_i G_{i-1} + P_i P_{i-1} G_{i-2} \ldots\ldots + P_1 G_0 + P_i P_{i-1} \ldots P_0 c_0$

- Conclusion: Delay through the adder is 3 gate delays for all carry-bits & 4 gate delays for all sum-bits.
- Consider the design of a 4-bit adder. The carries can be implemented as

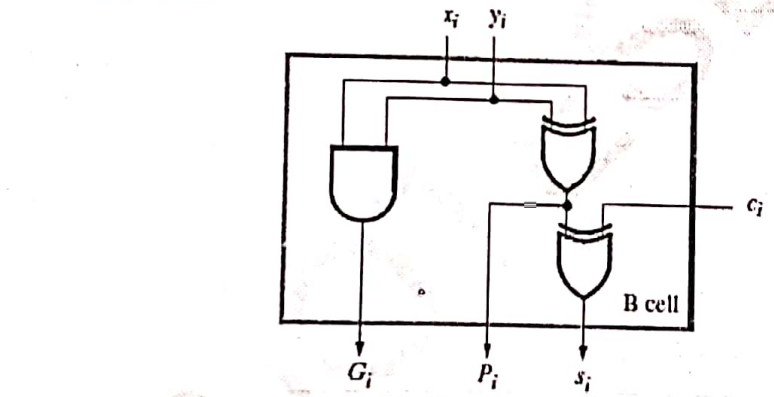
  $c_1 = G_0 + P_0 c_0$

  $c_2 = G_1 + P_1 G_0 + P_1 P_0 c_0$

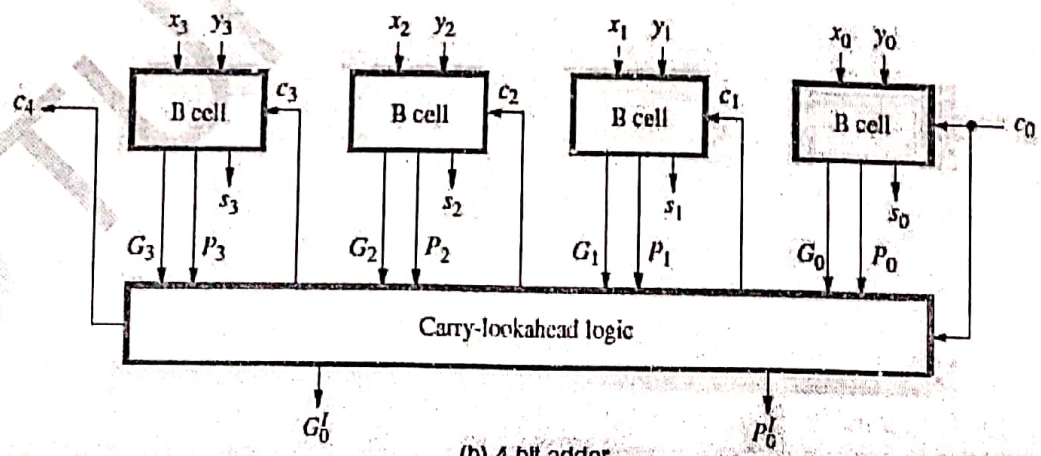
  $c_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 c_0$

  $c_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 c_0$

- The carries are implemented in the block labeled carry-lookahead logic. An adder implemented in this form is called a **Carry-Lookahead Adder.**
- Limitation: If we try to extend the carry-lookahead adder for longer operands, we run into a problem of gate fan-in constraints.



(a) Bit-stage cell



(b) 4-bit adder

Figure 9.4 A 4-bit carry-lookahead adder

HIGHER-LEVEL GENERATE & PROPAGATE FUNCTIONS

- 16-bit adder can be built from four 4-bit adder blocks (Figure 9.5).
- These blocks provide new output functions defined as G_k and P_k,
  where k=0 for the first 4-bit block,
  k=1 for the second 4-bit block and so on.
- In the first block,

$$P_0=P_3P_2P_1P_0$$

&

$$G_0=G_3+P_3G_2+P_3P_2G_1+P_3P_2P_1G_0$$

- The first-level G_i and P_i functions determine whether bit stage i generates or propagates a carry, and the second level G_k and P_k functions determine whether block k generates or propagates a carry.
- Carry c_{16} is formed by one of the carry-lookahead circuits as

$$c_{16}=G_3+P_3G_2+P_3P_2G_1+P_3P_2P_1G_0+P_3P_2P_1P_0c_0$$

- Conclusion: All carries are available 5 gate delays after X, Y and c_0 are applied as inputs.

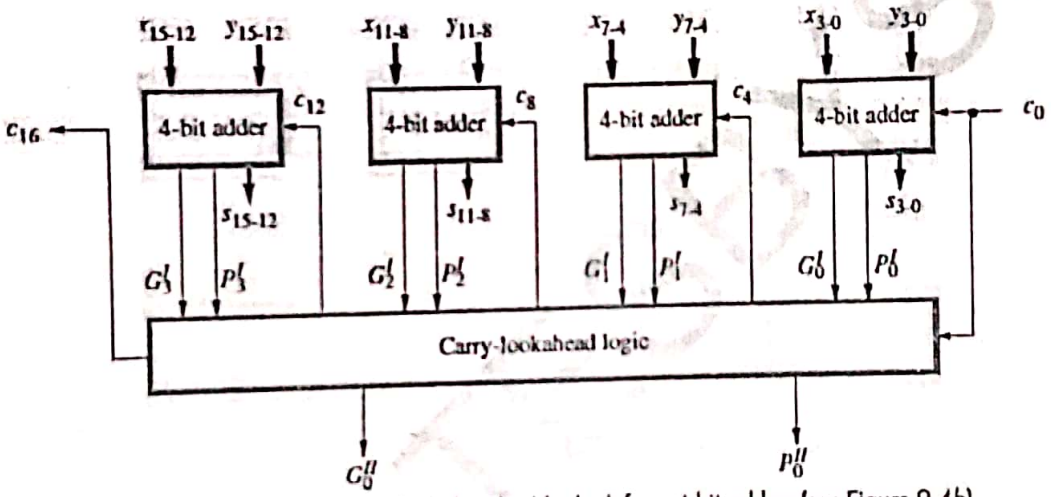
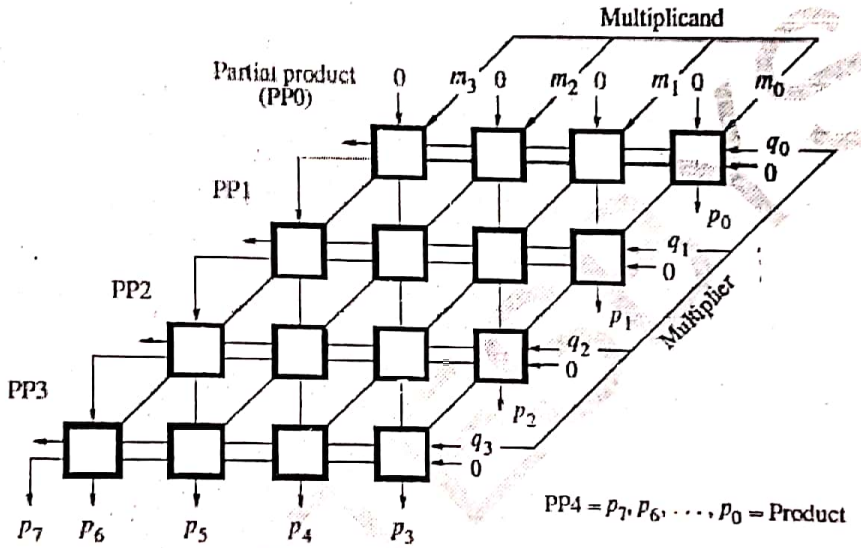


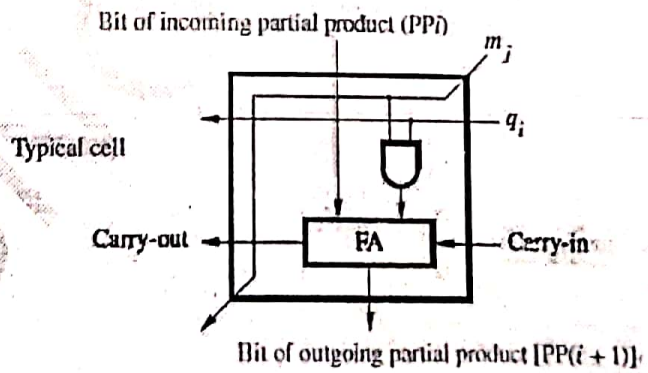
Figure 9.5 A 16-bit carry-lookahead adder built from 4-bit adders (see Figure 9.4b).

MULTIPLICATION OF POSITIVE NUMBERS

```
                1 1 0 1      (13) Multiplicand M
              × 1 0 1 1      (11) Multiplier Q
              ---------
                1 1 0 1
              1 1 0 1
            0 0 0 0
          1 1 0 1
          -----------
          1 0 0 0 1 1 1 1    (143) Product P
```

(a) Manual multiplication algorithm





(b) Array implementation

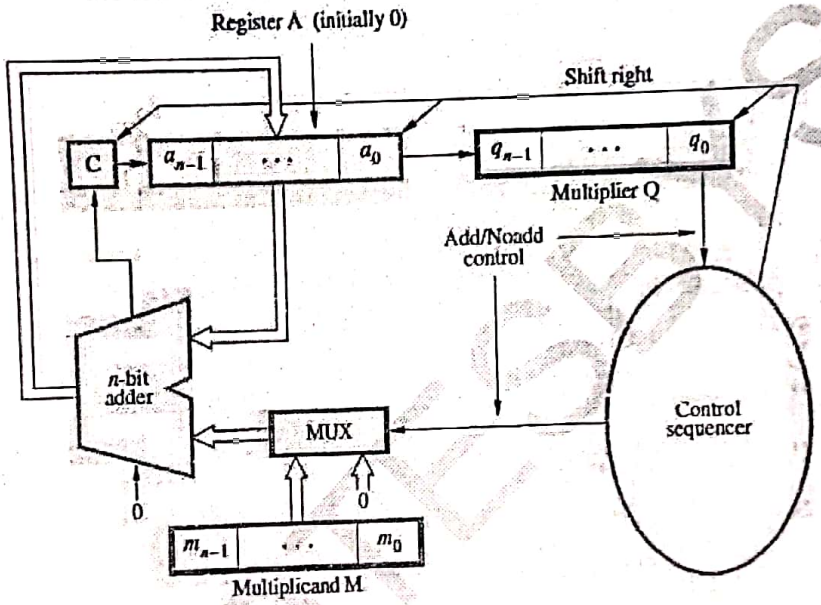
Figure 9.6 Array multiplication of unsigned binary operands.

ARRAY MULTIPLICATION

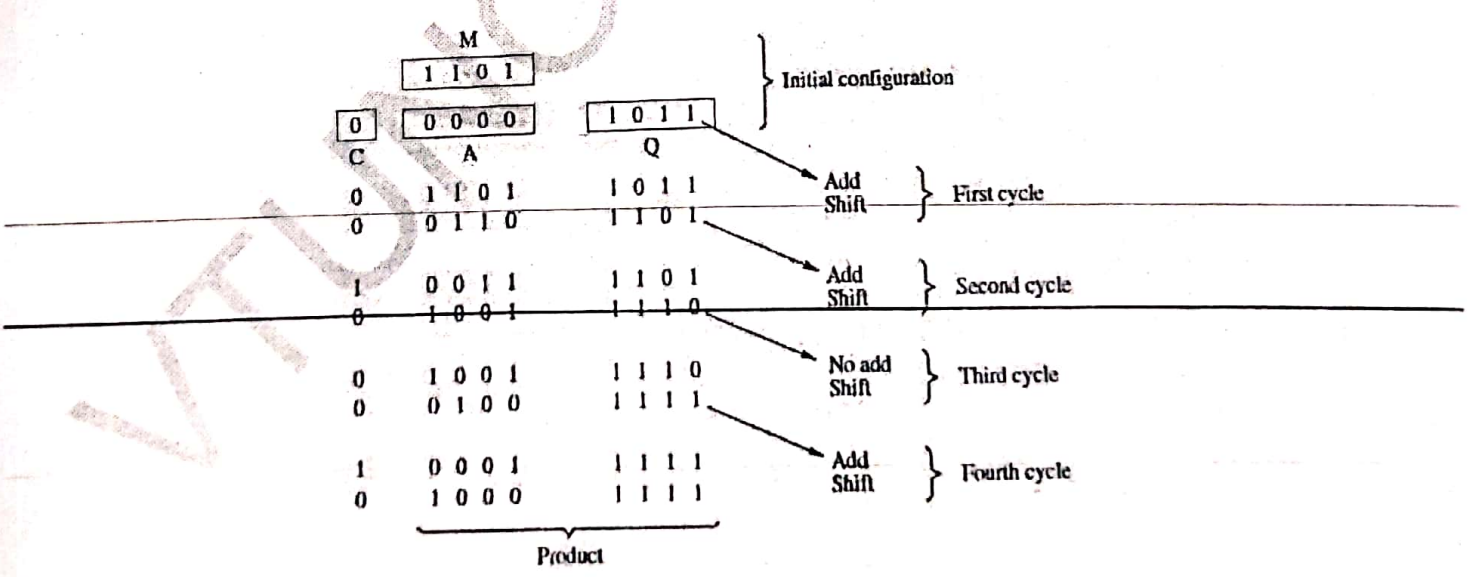
- The main component in each cell is a full adder(FA)..
- The AND gate in each cell determines whether a multiplicand bit m_j, is added to the incoming partial-product bit, based on the value of the multiplier bit q_i (Figure 9.6).

SEQUENTIAL CIRCUIT BINARY MULTIPLIER

- Registers A and Q combined hold PP_i(partial product)
  while the multiplier bit q_i generates the signal Add/Noadd.
- The carry-out from the adder is stored in flip-flop C (Figure 9.7).
- Procedure for multiplication:
  1) Multiplier is loaded into register Q,
     Multiplicand is loaded into register M and
     C & A are cleared to 0.
  2) If q_0=1, add M to A and store sum in A. Then C, A and Q are shifted right one bit-position.
     If q_0=0, no addition performed and C, A & Q are shifted right one bit-position.
  3) After n cycles, the high-order half of the product is held in register A and
     the low-order half is held in register Q.



(a) Register configuration



(b) Multiplication example

Figure 9.7 Sequential circuit binary multiplier.

SIGNED OPERAND MULTIPLICATION
BOOTH ALGORITHM

- This algorithm
  - → generates a 2n-bit product
  - → treats both positive & negative 2's-complement n-bit operands uniformly(Figure 9.9-9.12).
- Attractive feature: This algorithm achieves some efficiency in the number of addition required when the multiplier has a few large blocks of 1s.
- This algorithm suggests that we can reduce the number of operations required for multiplication by representing multiplier as a difference between 2 numbers.

  For e.g. multiplier(Q) 14(001110) can be represented as
  010000 (16)
  -000010 (2)
  001110 (14)
- Therefore, product P=M*Q can be computed by adding 2^4 times the M to the 2's complement of 2^1 times the M.

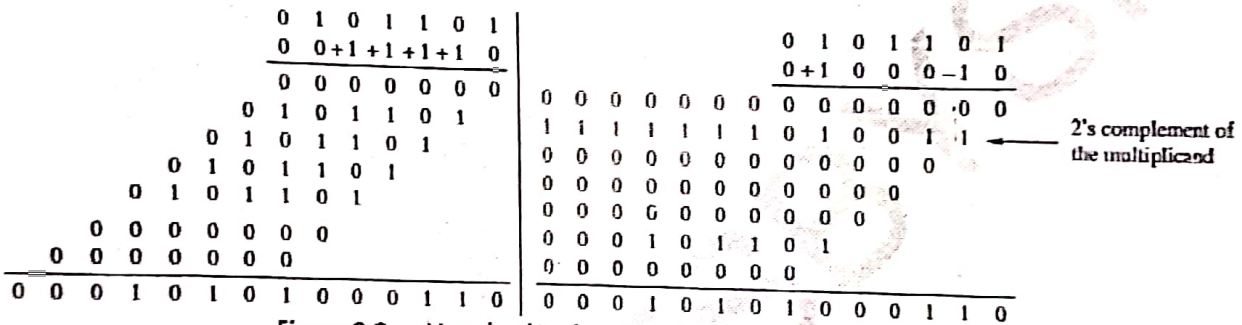


Figure 9.9 Normal and Booth multiplication schemes.

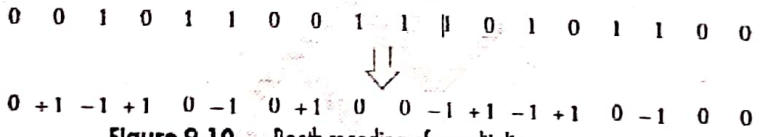


Figure 9.10 Booth recoding of a multiplier.

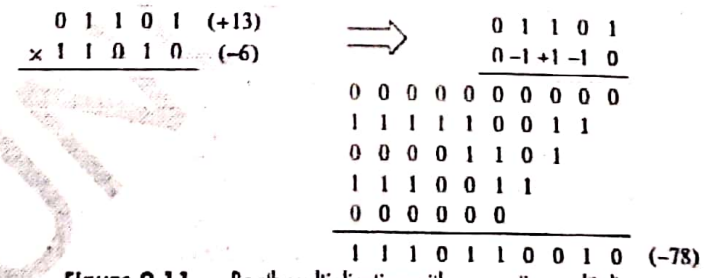


Figure 9.11 Booth multiplication with a negative multiplier.

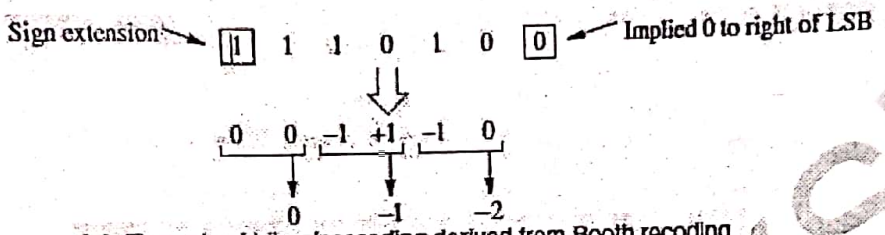
Multiplier		Version of multiplicand selected by bit i
Bit i	Bit $i-1$	
0	0	$0 \times M$
0	1	$+1 \times M$
1	0	$-1 \times M$
1	1	$0 \times M$

Figure 9.12 Booth multiplier recoding table.

FAST MULTIPLICATION

BIT-PAIR RECODING OF MULTIPLIERS

- This method
  - → derived from the booth algorithm
  - → reduces the number of summands by a factor of 2
- Group the Booth-recoded multiplier bits in pairs. (Figure 9.14 & 9.15).
- The pair (+1 -1) is equivalent to the pair (0 +1).



(a) Example of bit-pair recoding derived from Booth recoding

Multiplier bit-pair		Multiplier bit on the right	Multiplicand selected at position i
$i+1$	i	$i-1$	
0	0	0	$0\times M$
0	0	1	$+1\times M$
0	1	0	$+1\times M$
0	1	1	$+2\times M$
1	0	0	$-2\times M$
1	0	1	$-1\times M$
1	1	0	$-1\times M$
1	1	1	$0\times M$

(b) Table of multiplicand selection decisions

Figure 9.14 Multiplier bit-pair recoding.

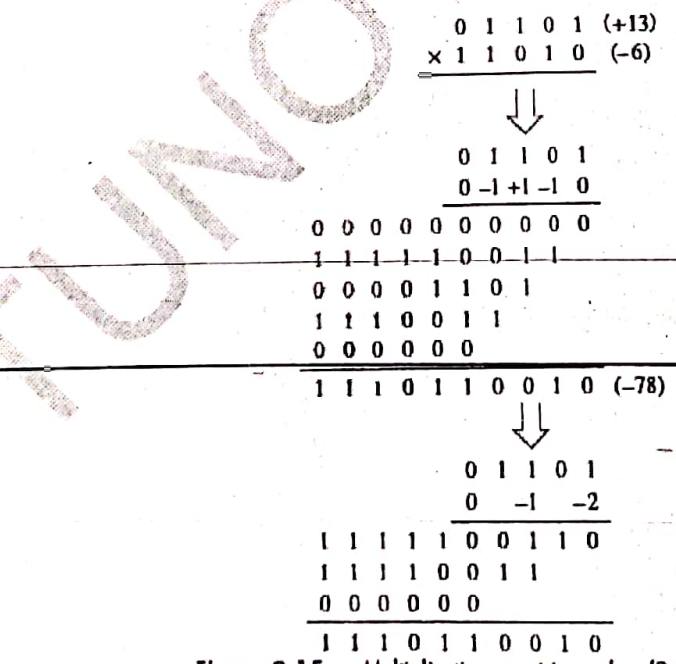


Figure 9.15 Multiplication requiring only $n/2$ summands.

INTEGER DIVISION

- An n-bit positive-divisor is loaded into register M.
  An n-bit positive-dividend is loaded into register Q at the start of the operation.
  Register A is set to 0 (Figure 9.21).
- After division operation, the n-bit quotient is in register Q, and
  the remainder is in register A.

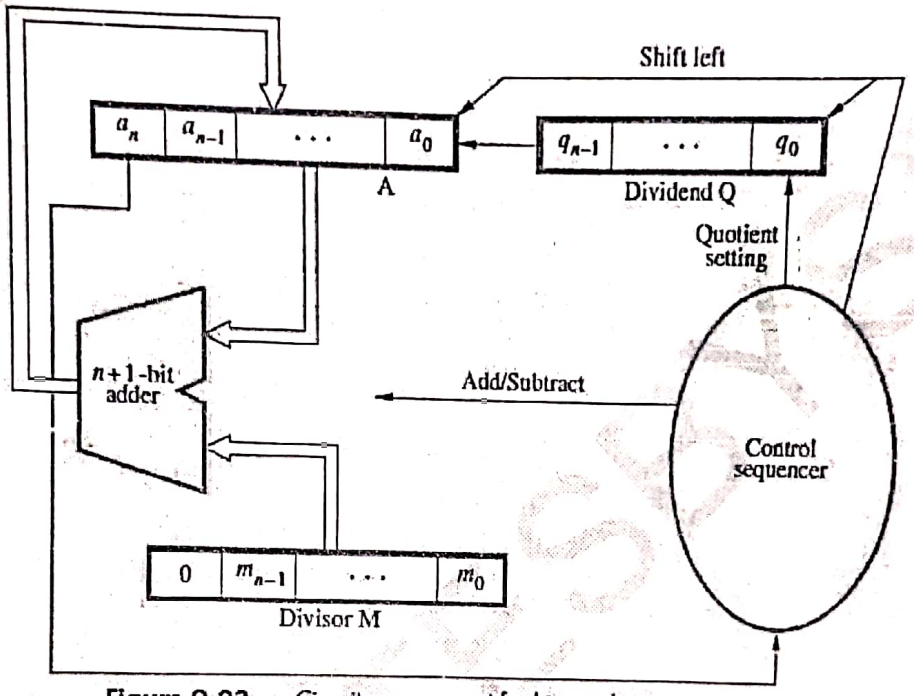


Figure 9.23 Circuit arrangement for binary division.

```
      21                    10101
13 ) 274          1101 ) 100010010
     26                    1101
     --                   -----
     14                    10000
     13                    1101
     --                   -----
      1                     1110
                            1101
                           -----
                               1
```

Figure 9.22 Longhand division examples.

NON-RESTORING DIVISION

- Procedure:

Step 1: Do the following n times

i) If the sign of A is 0, shift A and Q left one bit position and subtract M from A; otherwise, shift A and Q left and add M to A (Figure 9.23).

ii) Now, if the sign of A is 0, set q_0 to 1; otherwise set q_0 to 0.

Step 2: If the sign of A is 1, add M to A (restore).

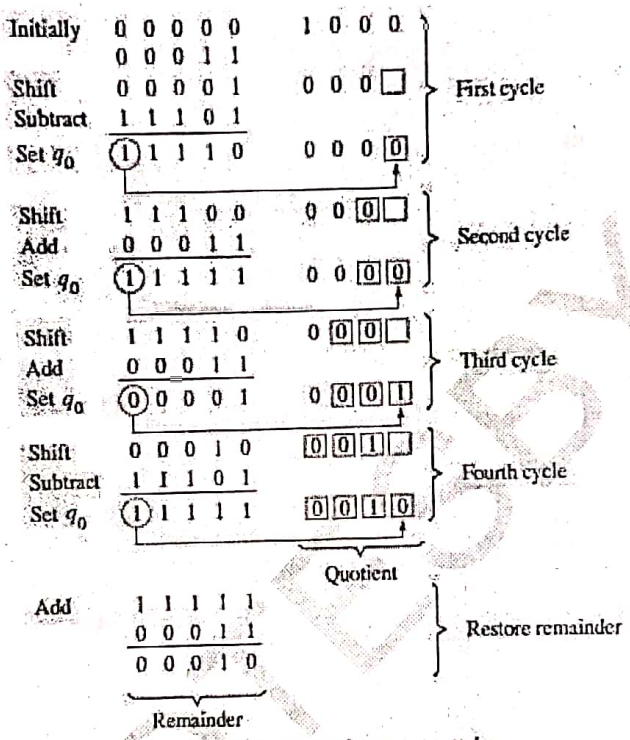


Figure 9.25 A non-restoring division example.

RESTORING DIVISION

- Procedure: Do the following n times
  1) Shift A and Q left one binary position (Figure 9.22).
  2) Subtract M from A, and place the answer back in A
  3) If the sign of A is 1, set q_0 to 0 and add M back to A(restore A).
     If the sign of A is 0, set q_0 to 1 and no restoring done.

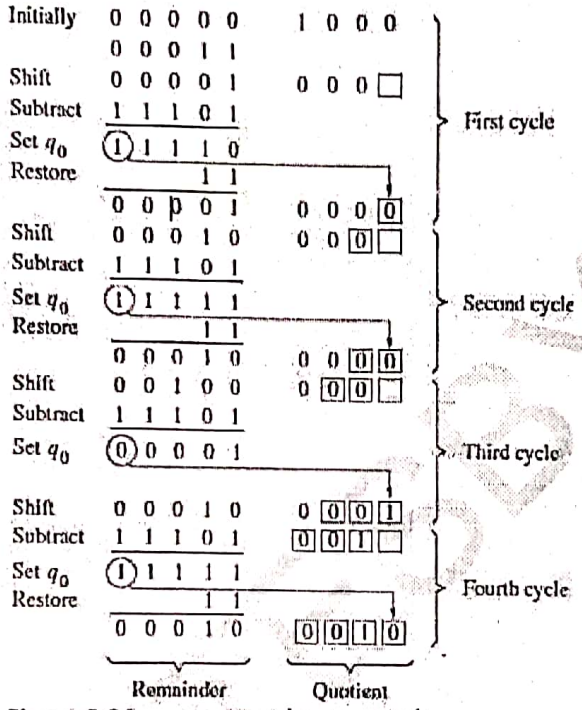


Figure 9.24 A restoring division example.

Problem 1:

Represent the decimal values 5, -2, 14, -10, 26, -19, 51 and -43 as signed 7-bit numbers in the following binary formats:

(a) sign-and-magnitude
(b) 1's-complement
(c) 2's-complement

Solution:

The three binary representations are given as:

Decimal values	Sign-and-magnitude representation	1's-complement representation	2's-complement representation
5	0000101	0000101	0000101
−2	1000010	1111101	1111110
14	0001110	0001110	0001110
−10	1001010	1110101	1110110
26	0011010	0011010	0011010
−19	1010011	1101100	1101101
51	0110011	0110011	0110011
−43	1101011	1010100	1010101

Problem 2:

(a) Convert the following pairs of decimal numbers to 5-bit 2's-complement numbers, then add them. State whether or not overflow occurs in each case.

a) 5 and 10 b) 7 and 13
c) −14 and 11 d) −5 and 7
e) −3 and −8

(b) Repeat Problem 1.7 for the subtract operation, where the second number of each pair is to be subtracted from the first number. State whether or not overflow occurs in each case.

Solution:

(a)

(a)	(b)	(c)
00101 + 01010 = 01111	00111 + 01101 = 10100	10010 + 01011 = 11101
no overflow	overflow	no overflow

(d)	(e)	(f)
11011 + 00111 = 00010	11101 + 11000 = 10101	10110 + 10011 = 01001
no overflow	no overflow	overflow

(b) To subtract the second number, form its 2's-complement and add it to the first number.

(a)	(b)	(c)
00101 + 10110 = 11011	00111 + 10011 = 11010	10010 + 10101 = 00111
no overflow	no overflow	overflow

(d)	(e)	(f)
11011 + 11001 = 10100	11101 + 01000 = 00101	10110 + 01101 = 00011
no overflow	no overflow	no overflow

Problem 3:

Perform following operations on the 6-bit signed numbers using 2's complement representation system. Also indicate whether overflow has occurred.

010110	101011	111111
+001001	+100101	+000111
011001	110111	010101
+010000	+111001	+101011
010110	111110	100001
−011111	−100101	−011101
111111	000111	011010
−000111	−111000	−100010

Solution:

010110	(+22)	101011	(−21)	111111	(−1)
+ 001001	+ (+9)	+ 100101	+ (−27)	+ 000111	+ (+7)
011111	(+31)	010000	(−48)	000110	(+6)
		overflow			
011001	(+25)	110111	(−9)	010101	(+21)
+ 010000	+ (+16)	+ 111001	+ (−7)	+ 101011	+ (−21)
101001	(+41)	110000	(−16)	000000	(0)
overflow					

010110	(+22)	010110
− 011111	− (+31)	+ 100001
	(−9)	110111
111110	(−2)	111110
− 100101	− (−27)	+ 011011
	(+25)	011001
100001	(−31)	100001
− 011101	− (+29)	+ 100011
	(−60)	000100
		overflow
111111	(−1)	111111
− 000111	− (+7)	+ 111001
	(−8)	111000
000111	(+7)	000111
− 111000	− (−8)	+ 001000
	(+15)	001111
011010	(+26)	011010
− 100010	− (−30)	+ 011110
	(+56)	111000
		overflow

Problem 4:

Perform signed multiplication of following 2's complement numbers using Booth's algorithm.

(a) A=010111 and B=110110 (b) A=110011 and B=101100
(c) A=110101 and B=011011 (d) A=001111 and B=001111
(e) A=10100 and B=10101 (f) A=01110 and B=11000

Solution:

010111 × 110110 = +23 × -10 = -230

```
                    0 1 0 1 1 1
                  × 0 -1 +1 0 -1 0
                                  0
                  1 1 1 1 1 1 0 1 0 0 1
                                0
sign extension    0 0 0 0 1 0 1 1 1
                  1 1 1 1 0 0 0 1 1 0 1 0
```

110011 × 101100 = -13 × -20 = 260

```
                    1 1 0 0 1 1
                  × -1 +1 0 -1 0 0
                                0
                              0
                  0 0 0 0 0 0 1 1 0 1
                            0
sign extension    1 1 1 1 1 0 0 1 1
                  0 0 0 1 0 0 0 0 0 1 0 0
```

110101 × 011011 = -11 × 27 = -297

```
                    1 1 0 1 0 1
                  × +1 0 -1 +1 0 -1
                                0
                  0 0 0 0 0 0 0 0 1 0 1 1
                          0
sign extension    1 1 1 1 1 1 1 0 1 0 1
                  0 0 0 0 0 0 1 0 1 1
                        0
                  1 1 1 1 0 1 1 0 1 0 1 1 1
```

001111 × 001111 = 15 × 15 = 225

```
                    0 0 1 1 1 1
                  × 0 +1 0 0 0 -1
                  1 1 1 1 1 1 1 1 0 0 0 1
                  0 0 0 0 1 1 1 1
                  0 0 0 0 1 1 1 0 0 0 0 1
```

10100(-12) × 10101(-11) →

```
                    1 0 1 0 0
                    -1 1 -1 1 -1   (recoded multiplier)
                  0 0 0 0 0 1 1 0 0
                  1 1 1 1 0 1 0 0
                  0 0 0 1 1 0 0
                  1 1 0 1 0 0
                  0 1 1 0 0
                  0 1 0 0 0 0 1 0 0   (+132)
```

01110(+14) × 11000(-8) →

```
                    0 1 1 1 0
                    0 -1 0 0 0   (recoded multiplier)
                  0 0 0 0 0 0 0 0 0 0
                  0 0 0 0 0 0 0 0 0
                  0 0 0 0 0 0 0 0
                  1 1 1 0 0 1 0
                  0 0 0 0 0 0
                  1 1 1 0 0 1 0 0 0 0   (-112)
```

Problem 5:

Perform signed multiplication of following 2's complement numbers using bit-pair recoding method.

(a) A=010111 and B=110110 (b) A=110011 and B=101100

(c) A=110101 and B=011011 (d) A=001111 and B=001111

Solution:

010111
× 110110

```
                    0 1 0 1 1 1
                      -1 +2 -2
1 1 1 1 1 1 | 1 0 1 0 0 1 0
0 0 0 | 0 1 0 1 1 1 0
1 1 1 1 0 2 1 1 0 1 0 1
-------------------------
1 1 1 1 0 0 0 1 1 0 1 0
```

110011
× 101100

```
                    1 1 0 0 1 1
                      -1 -1  0
                             0
0 0 0 | 0 0 0 1 1 0 1
0 | 0 0 0 1 1 1 0 1 1
-------------------------
0 0 0 1 0 0 0 0 0 1 0 0
```

110101
× 011011

```
                    1 1 0 1 0 1
                      +2 -1 -1
0 0 0 0 0 0 | 0 0 0 1 0 1 1
0 0 0 | 0 0 0 1 0 1 1
1 1 1 0 1 0 1 1   1
-------------------------
1 1 1 0 1 1 0 1 0 1 1 1
```

001111
× 001111

```
                    0 0 1 1 1 1
                      +1      -1
1 1 1 1 1 1 | 1 1 0 0 0 1
0 0 0 0 1 1 1 1
-------------------------
0 0 0 0 1 1 1 0 0 0 0 1
```

Problem 6:
Given A=10101 and B=00100, perform A/B using restoring division algorithm.
Solution:

Initially	000000(A) 000100(M)	10101(Q)
Shift	000001	0101□
Subtract	111100	
Set q0	(1)11101	
Restore	100	
	000001	0101[0]
Shift	000010	1010
Subtract	111100	
Set q0	(1)11110	
Restore	100	
	000010	1010[0]
Shift	000101	0100
Subtract	111100	
Set q0	(0)00001	
No restore	000000	
	000001	1010[0]
		01001
Shift	000001	
Subtract	111100	
Set q0	(1)11110	
Restore	100	
	000010	1001[0]
Shift	000101	0010
Subtract	111100	
Set q0	(0)00001	
No restore	000000	
	000001	0010[1]
	remainder	quotient

Problem 7:
Given A=10101 and B=00101, perform A/B using non-restoring division algorithm.
Solution:

	[000000] A	[10101] Q	Initial configuration
	[000101] M		
shift	000001	0 1 0 1 □	1st cycle
subtract	111011		
	111100	0 1 0 1 [0]	
shift	111000	1 0 1 [0] □	2nd cycle
add	000101		
	111101	1 0 1 [0] [0]	
shift	111011	0 1 [0] [0] □	3rd cycle
add	000101		
	000000	0 1 [0] [0] [1]	
shift	000000	1 [0] [0] [1] □	4th cycle
subtract	111011		
	111011	1 [0] [0] [1] [0]	
shift	110111	[0] [0] [1] [0] □	5th cycle
add	000101		
	111100	[0] [0] [1] [0] [0]	
add	000101	quotient	
	000001		
	remainder		

Problem 8:

Represent 1259.12510 in single precision and double precision formats

Solution:

Step 1: Convert decimal number to binary format

$1259_{(10)} = 10011101011_{(2)}$

Fractional Part

$0.125_{(10)} = 0.001$

Binary number = 10011101011+0.001

= 10011101011.001

Step 2: Normalize the number

$10011101011.001 = 1.0011101011001 \times 2^{10}$

Step 3: Single precision format:

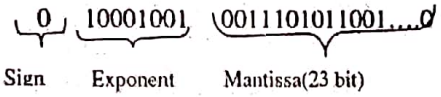
For a given number S=0, E=10 and M=0011101011001

Bias for single precision format is = 127

$E' = E+127 = 10+127 = 137_{(10)}$

$= 10001001_{(2)}$

Number in single precision format is given as



Step 4: Double precision format:

For a given number S=0, E=10 and M=0011101011001

Bias for double precision format is = 1023

$E' = E+1023 = 10+1023 = 1033_{(10)}$

$= 10000001001_{(2)}$

Number in double precision format is given as