||Jai Sri Gurudev ||
Sri Adichunchanagiri Shikshana Trust®

SJB INSTITUTE OF TECHNOLOGY

Accredited by NBA & NAAC with 'A' Grade
No. 67, BGS Health & Education City, Dr. Vishnuvardhan Road
Kengeri, Bangalore – 560 060

SJBIT

Department of Electronics & Communication Engineering

Computer Organization & ARM Microcontrollers [21EC52]

MODULE – 2

Memory System and Basic Processing Unit

Notes (as per VTU Syllabus)

V SEMESTER – B. E

Academic Year: 2023 – 2024 (ODD)

Course Coordinator :	**Dr. Supreeth H S G**
Designation :	**Associate Professor**

Syllabus

[As per Choice Based Credit System (CBCS) scheme]
SEMESTER – V

Subject Code: 21EC52 **IA Marks: 50**
Number of Lecture Hours/Week: 3 **Exam Marks: 50**
Total Number of Lecture Hours: 40 **Exam Hours 03**
Credits – 04

Course objectives: This course will enable students to

1. Explain the basic organization of a computer system.
2. Demonstrate functioning of different sub systems, such as processor, Input/output, and memory.
3. Describe the architectural features and instructions of 32-bit microcontroller ARM Cortex M3.
4. Apply the knowledge gained for Programming ARM Cortex M3 for different applications.
5. Understand the basic hardware components and their selection method based on the characteristics and attributes of an embedded system

Module – 02

Memory System: Basic Concepts, Semiconductor RAM Memories, Read Only Memories, Speed, Size, and Cost, Cache Memories – Mapping Functions, Replacement Algorithms, Performance Considerations.

Text book 1: Chapter 5 – 5.1 to 5.4, 5.5 (5.5.1, 5.5.2), 5.6

Basic Processing Unit: Some Fundamental Concepts, Execution of a Complete Instruction, Multiple Bus Organization, Hard-wired Control, Micro programmed Control. Basic concepts of pipelining,

Text book 1: Chapter7, Chapter 8 – 8.1

Text Books:

1. Carl Hamacher, Zvonko Vranesic, Safwat Zaky, Computer Organization, 5th Edition, Tata McGraw Hill, 2002. (Listed topics only from Chapters 1, 2, 4, 5, 8).

2. Andrew N Sloss, Dominic System and Chris Wright, “ARM System Developers Guide”, Elsevier, Morgan Kaufman publisher, 1st Edition, 2008.

INDEX SHEET

MODULE – 02
Memory System & Basic Processing Unit

Section 1: Basic Concepts:

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

- Maximum size of memory that can be used in any computer is determined by addressing mode.
- If MAR is k-bits long then

  → memory may contain upto 2^K addressable-locations

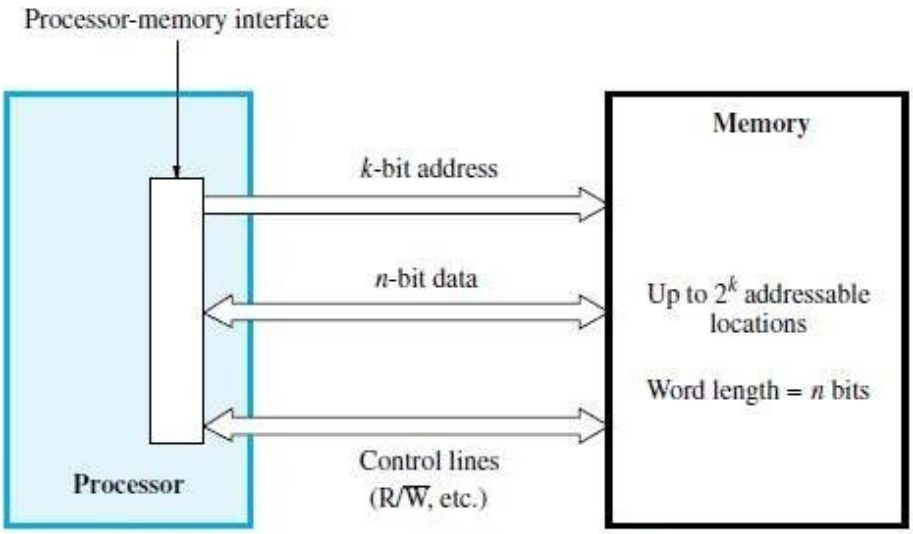


Figure 8.1 Connection of the memory to the processor.

- 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

Section 02: 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 magneticdisk.
- 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.

- 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

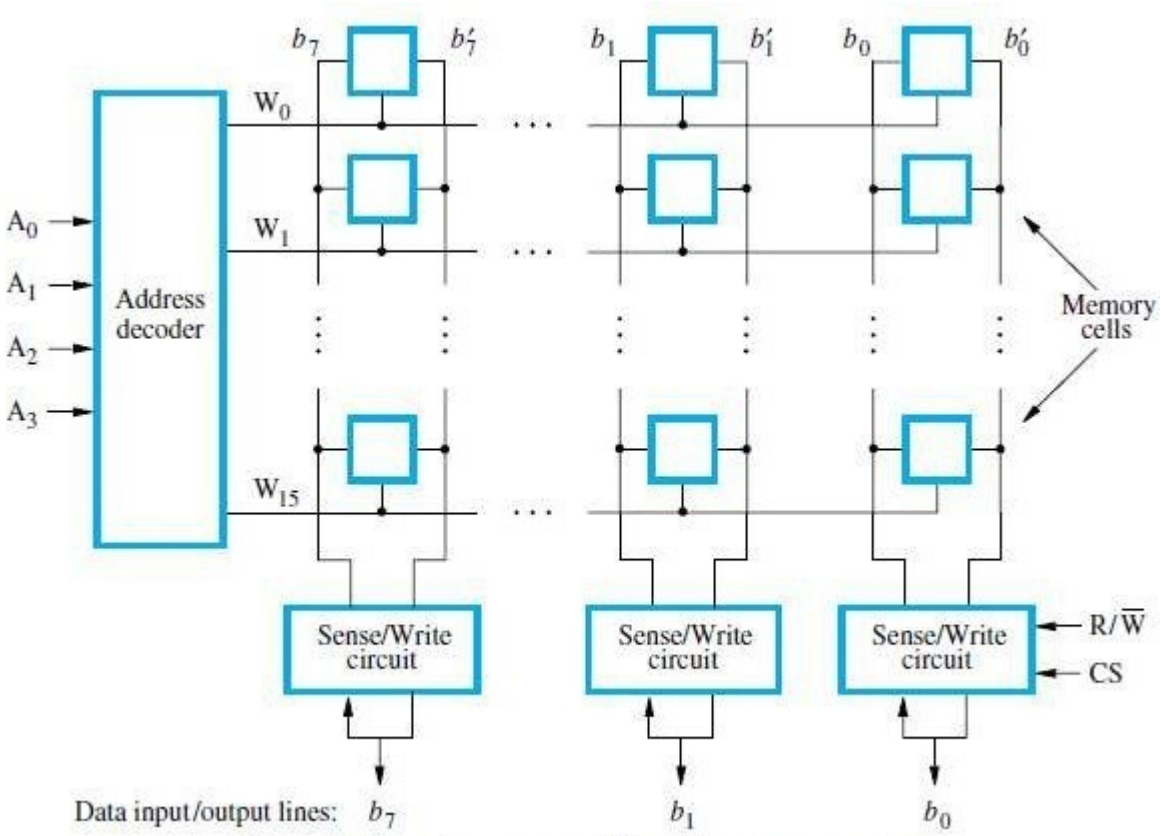


Figure 8.2 Organization of bit cells in a memory chip.

STATIC RAM (OR MEMORY)

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

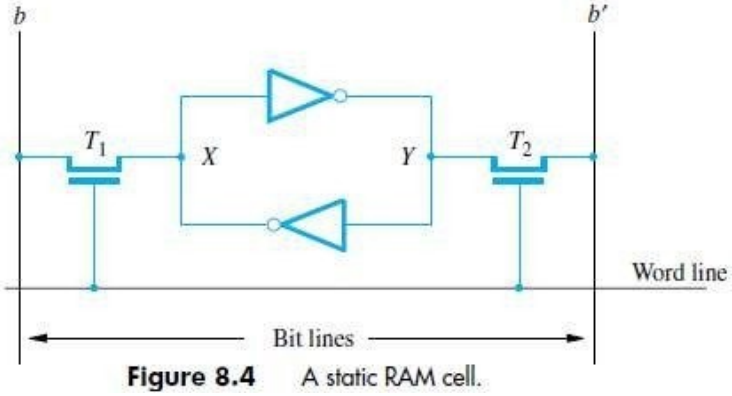


Figure 8.4 A static RAM cell.

- Two inverters are cross connected to form a latch (Figure 8.4).
- The latch is connected to 2-bit-lines by transistors T1 and T2.
- 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 T1 and T2.
- 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.

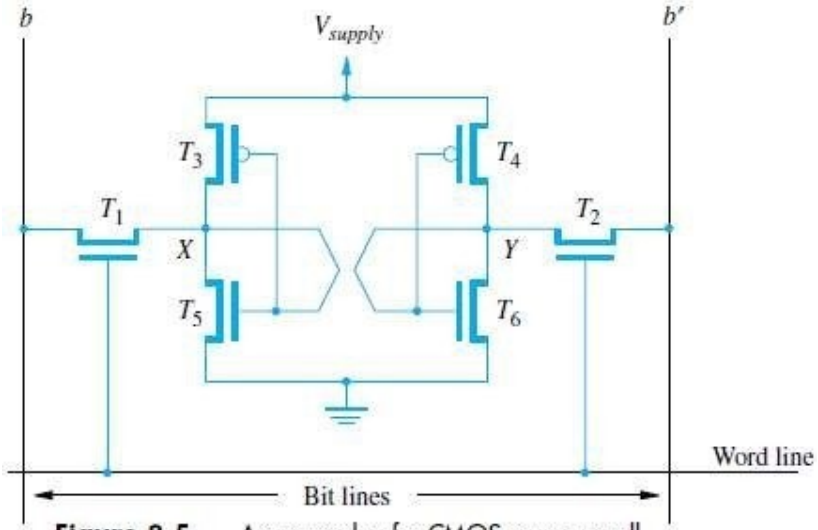


Figure 8.5 An example of a CMOS memory cell.

CMOS Cell

- Transistor pairs (T3, T5) and (T4, T6) form the inverters in the latch (Figure 8.5).
- In state 1, the voltage at point X is high by having T5, T6 ON and T4, T5 are OFF.
- Thus, T1 and T2 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.

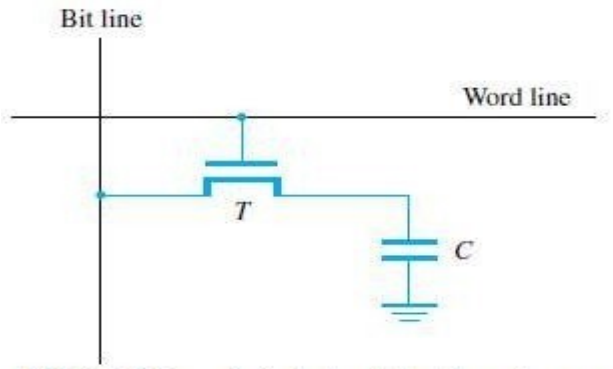


Figure 8.6 A single-transistor dynamic memory cell.

- 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. A8-0 → specifies row-address of a byte.
  2) 9 bits are needed to specify a group of 8 bits in the selected row.
     i.e. A20-9 → specifies column-address of a byte.

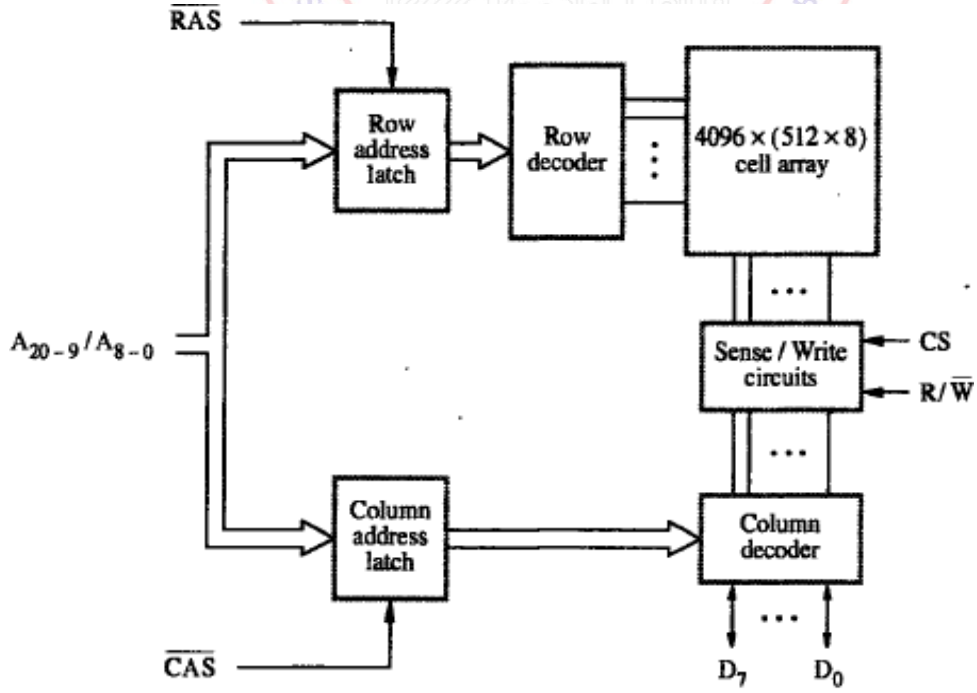


Figure 5.7 Internal organization of a 2M × 8 dynamic memory chip.

- 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 D0-D7.
  R/W'=0(write-operation) □ Information on D0-D7 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

- 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.

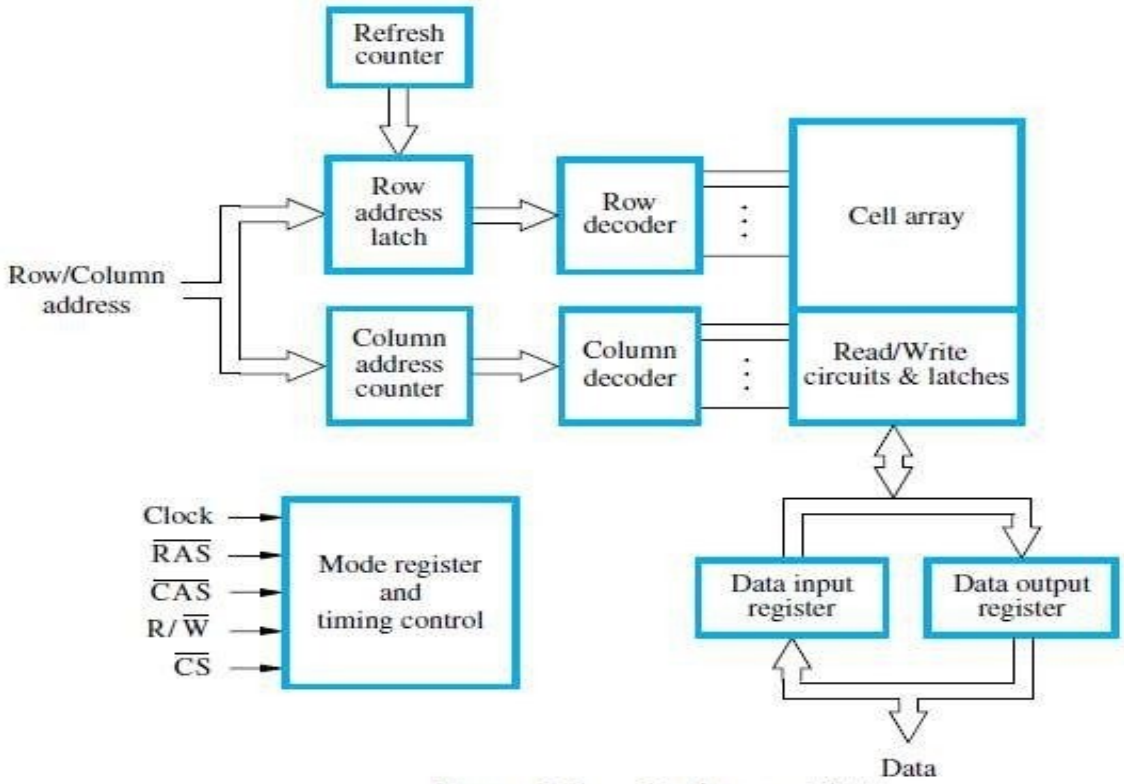


Figure 8.8 Synchronous DRAM.

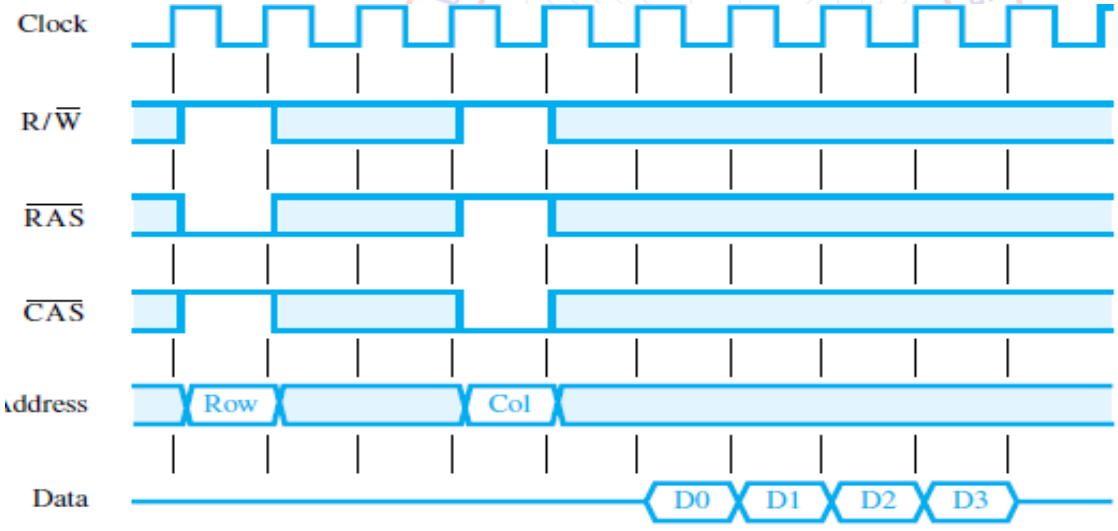


Figure 8.9 A burst read of length 4 in an SDRAM.

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).

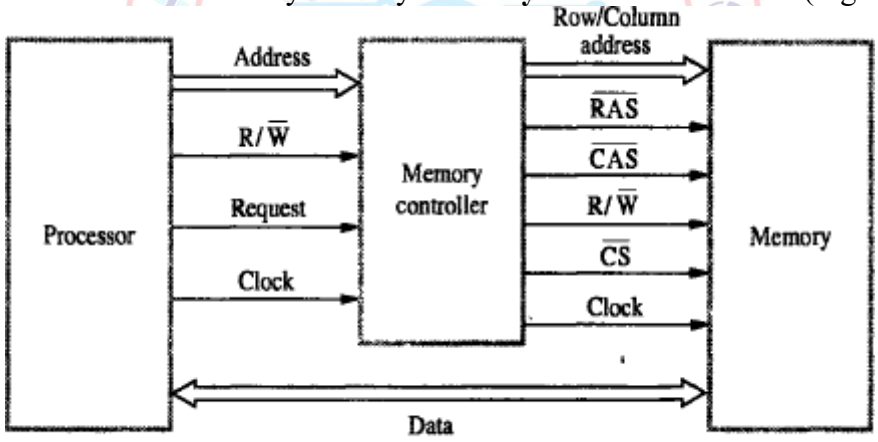


Figure 5.11 Use of a memory controller.

- 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 Vref.
- The reference voltage is about 2V.

- The two logical values are represented by 0.3V swings above and below Vref.
- 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.

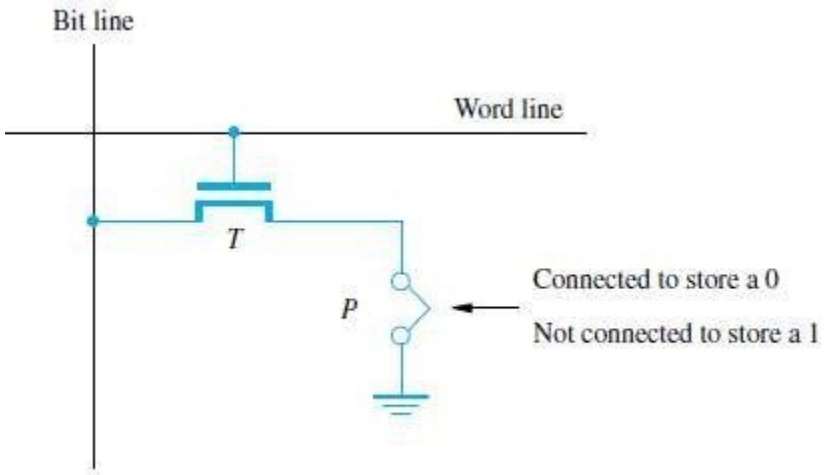


Figure 8.11 A ROM cell.

- 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 drivenapplication.
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.

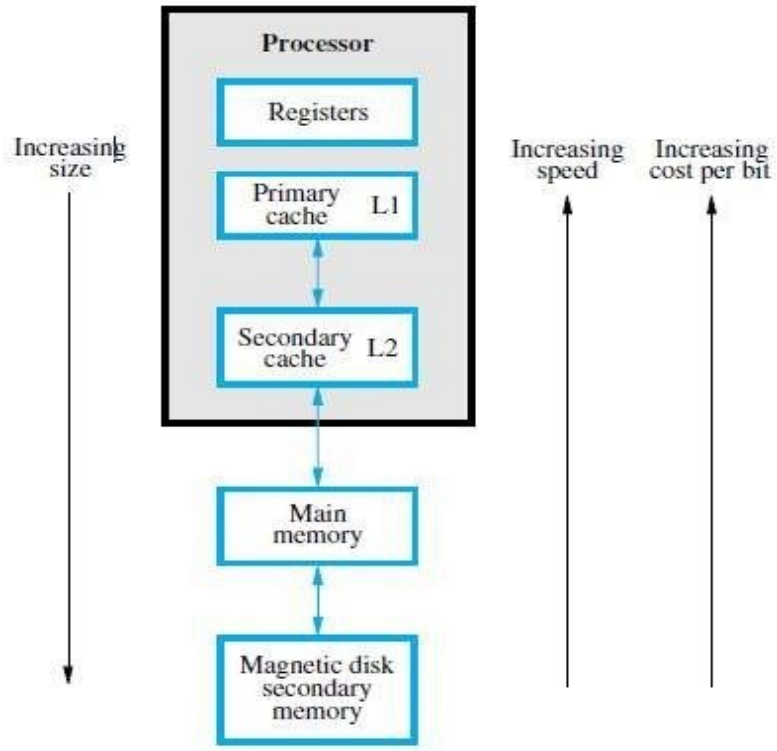


Figure 8.14 Memory hierarchy.

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.

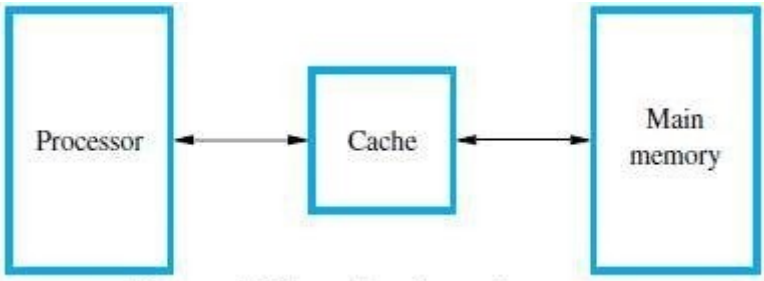


Figure 8.15 Use of a cache memory.

- 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.

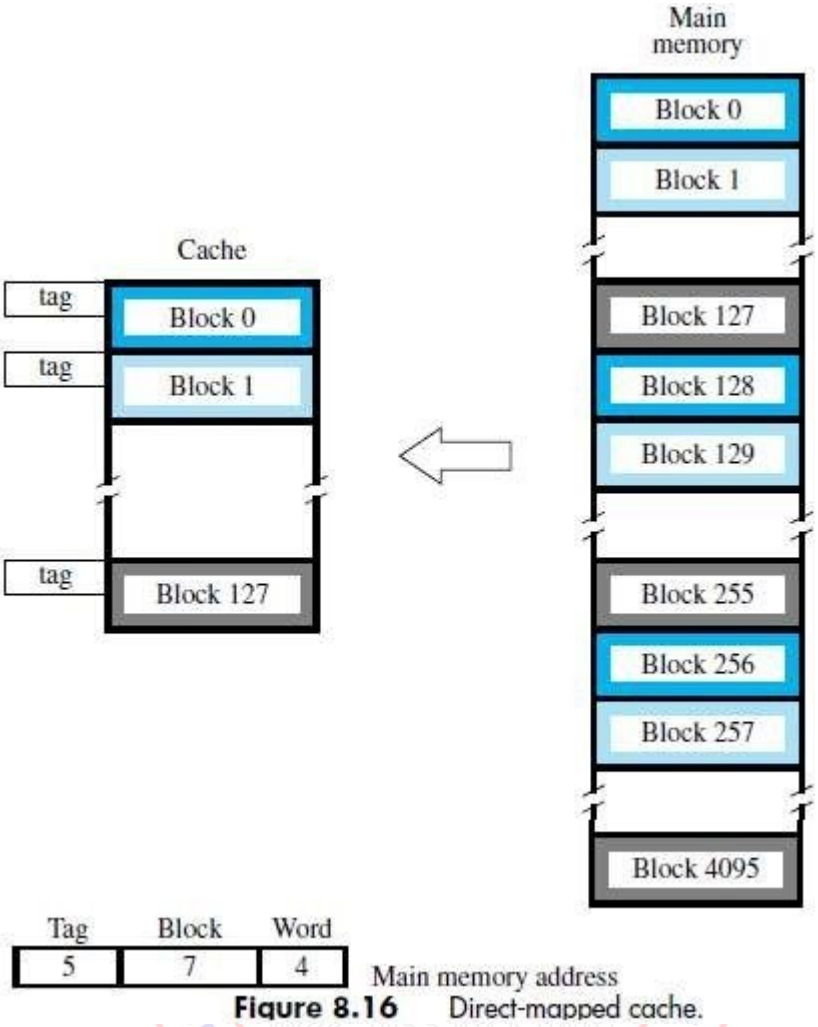


Figure 8.16 Direct-mapped cache.

- 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.

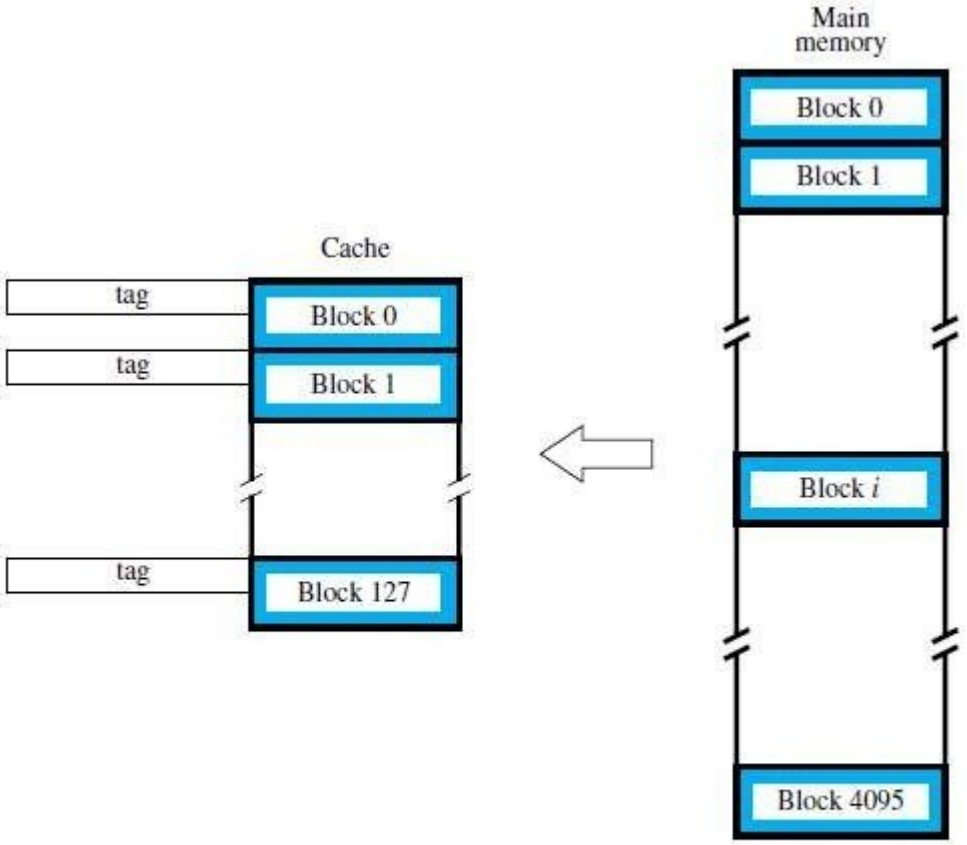


Figure 8.17 Associative-mapped cache.

- 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.

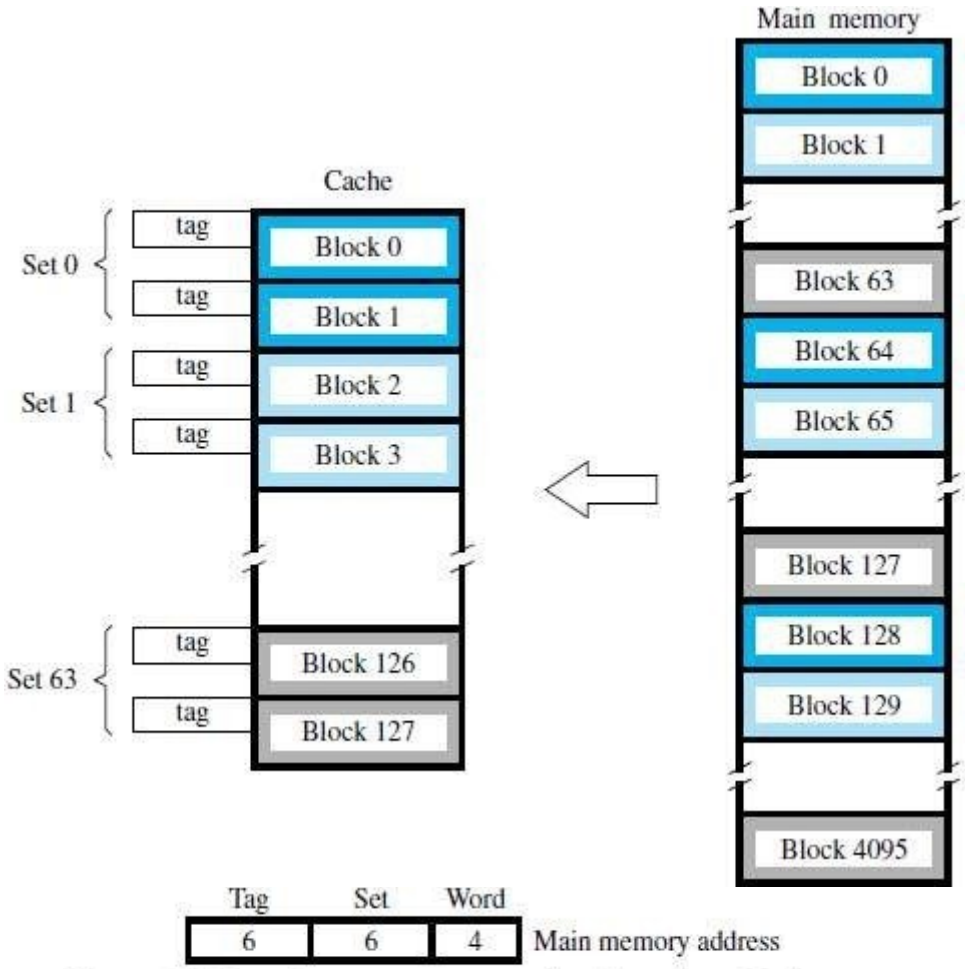


Figure 8.18 Set-associative-mapped cache with two blocks per set.

- 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.

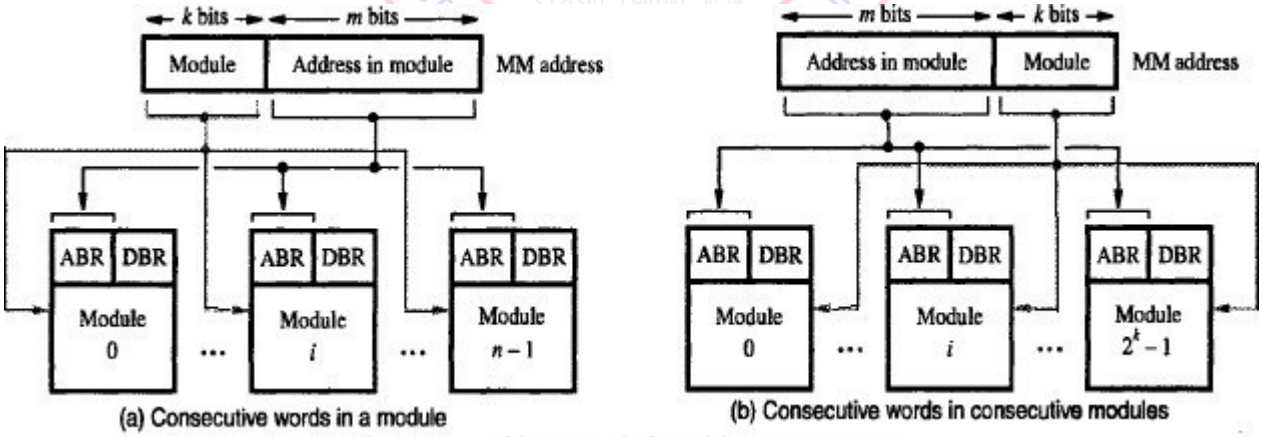


Figure 5.25 Addressing multiple-module memory systems.

- 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

$$tavg = hC + (1 - h)M$$

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 thecache

Smaller size

- More misses

Problem 6:

Consider a computer system in which the available pages in the physical memory are divided among several application programs. The operating system monitors the page transfer activity and dynamically adjusts the number of pages allocated to various programs. Suggest a suitable strategy that the operating system can use to minimize the overall rate of page transfers.

Solution:

The operating system may increase the main-memory pages allocated to a program that has a large number of page faults, using space previously allocated to a program with a few page faults

Problem 7:

In a computer with a virtual-memory system, the execution of an instruction may be interrupted by a page fault. What state information has to be saved so that this instruction can be resumed later? Note that bringing a new page into the main-memory involves a DMA transfer, which requires execution of other instructions. Is it simpler to abandon the interrupted instruction and completely re-execute it later? Can this be done?

Solution:

Continuing the execution of an instruction interrupted by a page fault requires saving the entire state of the processor, which includes saving all registers that may have been affected by the instruction as well as the control information that indicates how far the execution has progressed. The alternative of re-executing the instruction from the beginning requires a capability to reverse any changes that may have been caused by the partial execution of the instruction.

Problem 8:

When a program generates a reference to a page that does not reside in the physical main-memory, execution of the program is suspended until the requested page is loaded into the main-memory from a disk. What difficulties might arise when an instruction in one page has an operand in a different page? What capabilities must the processor have to handle this situation?

Solution:

The problem is that a page fault may occur during intermediate steps in the execution of a single instruction. The page containing the referenced location must be transferred from the disk into the main-memory before execution can proceed.

Since the time needed for the page transfer (a disk operation) is very long, as compared to instruction execution time, a context-switch will usually be made.

(A context-switch consists of preserving the state of the currently executing program, and ”switching” the processor to the execution of another program that is resident in the main- memory.) The page transfer, via DMA, takes place while this other program executes. When the page transfer is complete, the original program can be resumed.

Therefore, one of two features are needed in a system where the execution of an individual instruction may be suspended by a page fault. The first possibility is to save the state of instruction execution. This involves saving more information (temporary programmer- transparent registers, etc.) than needed when a program is interrupted between instructions. The second possibility is to ”unwind” the effects of the portion of the instruction completed when the page fault occurred, and then execute the instruction from the beginning when the program is resumed.

Problem 9:

Magnetic disks are used as the secondary storage for program and data files in most virtual-memory systems. Which disk parameter(s) should influence the choice of page size?

Solution:

The sector size should influence the choice of page size, because the sector is the smallest directly addressable block of data on the disk that is read or written as a unit. Therefore, pages should be some small integral number of sectors in size.

Problem 10:

A disk unit has 24 recording surfaces. It has a total of 14,000 cylinders. There is an average of 400 sectors per track. Each sector contains 512 bytes of data.

(a) What is the maximum number of bytes that can be stored in this unit?
(b) What is the data-transfer rate in bytes per second at a rotational speed of 7200 rpm?
(c) Using a 32-bit word, suggest a suitable scheme for specifying the disk address.

Solution:

(a) The maximum number of bytes that can be stored on this disk is 24 X 14000 X 400 X 512 = 68.8 X 10^9 bytes.

(b) The data-transfer rate is (400 X 512 X 7200)/60 = 24.58 X 10^6 bytes/s.

(c) Need 9 bits to identify a sector, 14 bits for a track, and 5 bits for a surface.

Thus, a possible scheme is to use address bits A8-0 for sector, A22-9 for track, and A27-23 for surface identification. Bits A31-28 are not used.

BASIC PROCESSING UNIT:

SOME FUNDAMENTAL CONCEPTS

- To execute an instruction, processor has to perform following 3 steps:
  1) Fetch contents of memory-location pointed to by PC. Content of this location is an instruction to be executed. The instructions are loaded into IR, Symbolically, this operation is written as:
     IR□ [[PC]]
  2) Increment PC by 4.
     PC□ [PC] +4
  3) Carry out the actions specified by instruction (in the IR).
- The first 2 steps are referred to as **Fetch Phase**.
  Step 3 is referred to as **Execution Phase**.

- The operation specified by an instruction can be carried out by performing one or more of the following actions:
  1) Read the contents of a given memory-location and load them into a register.
  2) Read data from one or more registers.
  3) Perform an arithmetic or logic operation and place the result into a register.
  4) Store data from a register into a given memory-location.
- The hardware-components needed to perform these actions are shown in Figure 5.1.

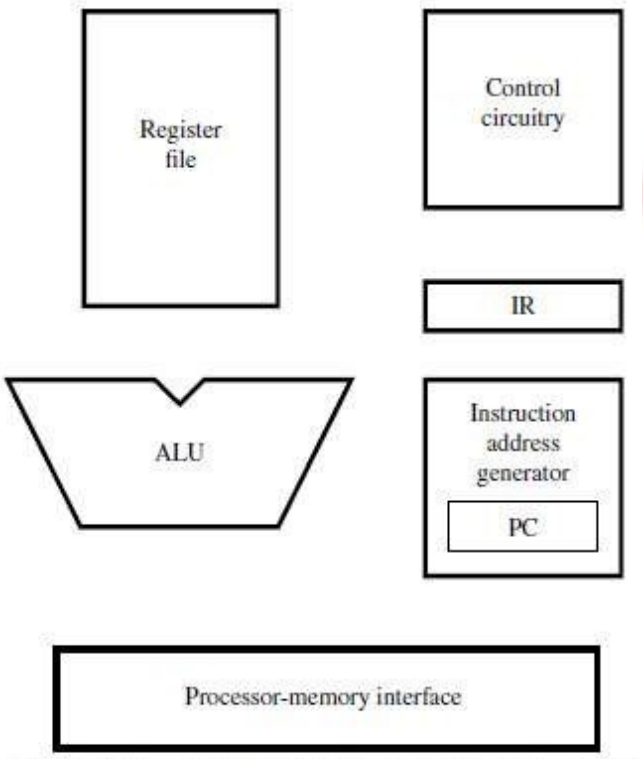


Figure 5.1 Main hardware components of a processor.

SINGLE BUS ORGANIZATION

- ALU and all the registers are interconnected via a **Single Common Bus** (Figure 7.1).
- Data & address lines of the external memory-bus is connected to the internal processor-bus via MDR & MAR respectively. (MDR□ Memory Data Register, MAR □ Memory Address Register).
- **MDR** has 2 inputs and 2 outputs. Data may be loaded
  → into MDR either from memory-bus (external) or
  → from processor-bus (internal).

- **MAR**‟s input is connected to internal-bus; MAR‟s output is connected to external-bus.
- **Instruction Decoder & Control Unit** is responsible for
  → issuing the control-signals to all the units inside the processor.
  → implementing the actions specified by the instruction (loaded in the IR).
- Register R0 through R(n-1) are the **Processor Registers**.
  The programmer can access these registers for general-purpose use.
- Only processor can access 3 registers **Y**, **Z** & **Temp** for temporary storage during program-execution. The programmer cannot access these 3 registers.
- In **ALU**, 1) „A‟ input gets the operand from the output of the multiplexer(MUX).
  2) „B‟ input gets the operand directly from the processor-bus.
- There are 2 options provided for „A‟ input of the ALU.
- MUX is used to select one of the 2 inputs.
- **MUX** selects either
  → output of Y or
  → constant-value 4(which is used to increment PC content).

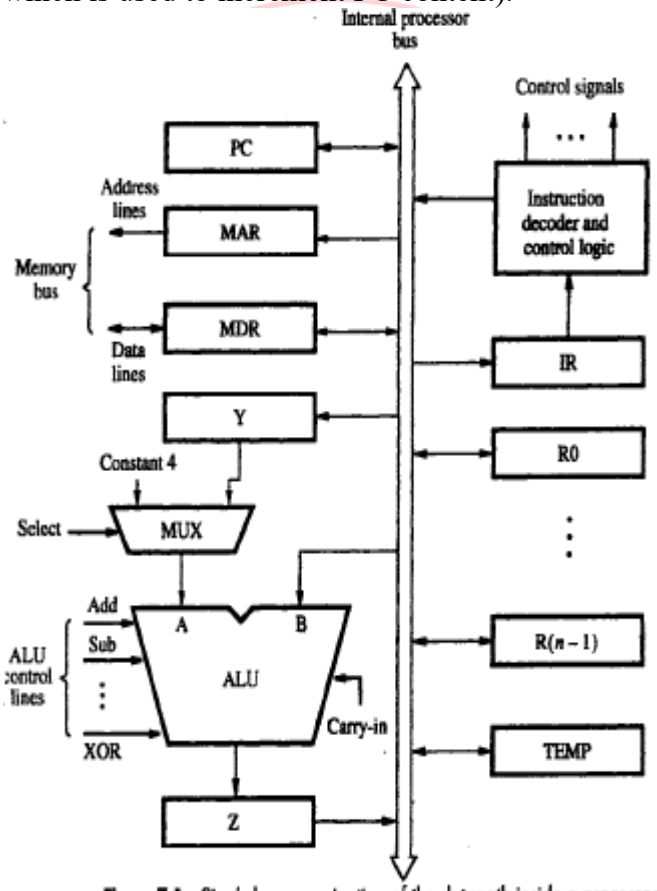


Figure 7.1 Single-bus organization of the datapath inside a processor.

- An instruction is executed by performing one or more of the following operations:
  1) Transfer a word of data from one register to another or to the ALU.
  2) Perform arithmetic or a logic operation and store the result in a register.
  3) Fetch the contents of a given memory-location and load them into a register.
  4) Store a word of data from a register into a given memory-location.
- **Disadvantage:** Only one data-word can be transferred over the bus in a clock cycle.

Solution: Provide multiple internal-paths. Multiple paths allow several data-transfers to take place in parallel.

REGISTER TRANSFERS

- Instruction execution involves a sequence of steps in which data are transferred from one register to another.
- For each register, two control-signals are used: Riin & Riout. These are called **Gating Signals.**
- Riin=1 □ data on bus is loaded into Ri. Riout=1
  □ content of Ri is placed onbus.
  Riout=0, □ bus can be used for transferring data from other registers.
- For example, *Move R1, R2;* This transfers the contents of register R1 to register R2. This can be accomplished as follows:
  1) Enable the output of registers R1 by setting R1out to 1 (Figure 7.2).
  This places the contents of R1 on processor-bus.
  2) Enable the input of register R2 by setting R2out to 1.
  This loads data from processor-bus into register R4.
- All operations and data transfers within the processor take place within time-periods defined by the

processor-clock.

- The control-signals that govern a particular transfer are asserted at the start of the clock cycle.

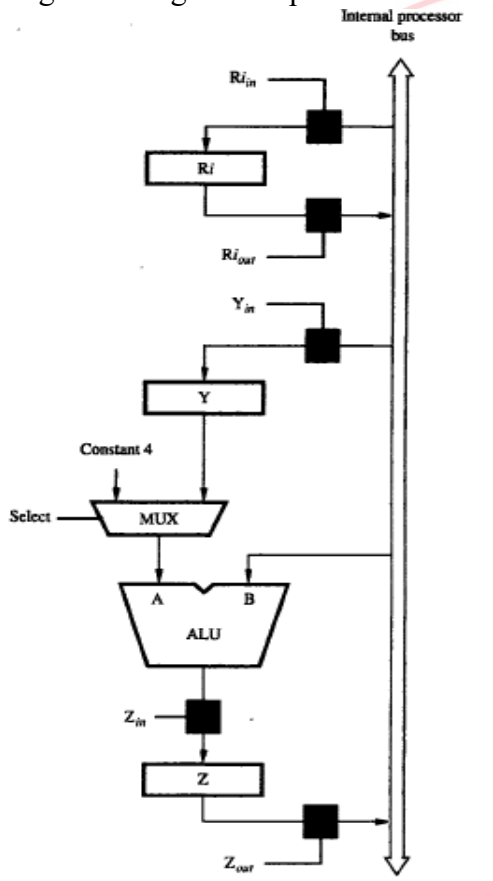


Figure 7.2 Input and output gating for the registers in Figure 7.1.

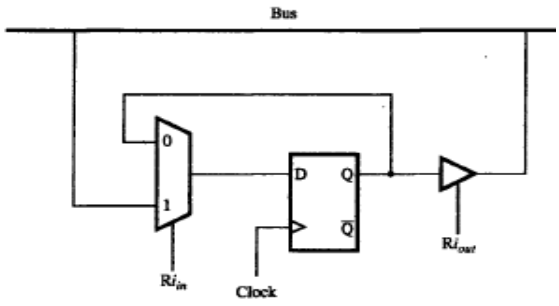


Figure 7.3 Input and output gating for one register bit.

Input & Output Gating for one Register Bit

- A 2-input multiplexer is used to select the data applied to the input of an edge-triggered D flip-flop.
- Riin=1 □ mux selects data on bus. This data will be loaded into flip-flop at rising-edge of clock.
  Riin=0 □ mux feeds back the value currently stored in flip-flop (Figure7.3).
- Q output of flip-flop is connected to bus via a tri-state gate.
  Riout=0 □ gate's output is in the high-impedancestate.
  Riout=1 □ the gate drives the bus to 0 or 1, depending on the value of Q.

PERFORMING AN ARITHMETIC OR LOGIC OPERATION

- The ALU performs arithmetic operations on the 2 operands applied to its A and B inputs.
- One of the operands is output of MUX;
  And, the other operand is obtained directly from processor-bus.
- The result (produced by the ALU) is stored temporarily in register Z.
- The sequence of operations for [R3]□[R1]+[R2] is as follows:
  1) R1out, Yin
  2) R2out, SelectY, Add, Zin
  3) Zout, R3in
- Instruction execution proceeds as follows:
  Step 1 --> Contents from register R1 are loaded into register Y.
  Step2 --> Contents from Y and from register R2 are applied to the A and B inputs of ALU;
  Addition is performed &
  Result is stored in the Z register.
  Step 3 --> The contents of Z register is stored in the R3 register.
- The signals are activated for the duration of the clock cycle corresponding to that step. All other signals are inactive.

CONTROL-SIGNALS OF MDR

- The MDR register has 4 control-signals (Figure 7.4):
  1) MDRin & MDRout control the connection to the internal processor data bus &
  2) MDRinE & MDRoutE control the connection to the memory Data bus.
- MAR register has 2 control-signals.
  1) MARin controls the connection to the internal processor address bus &
  2) MARout controls the connection to the memory address bus.

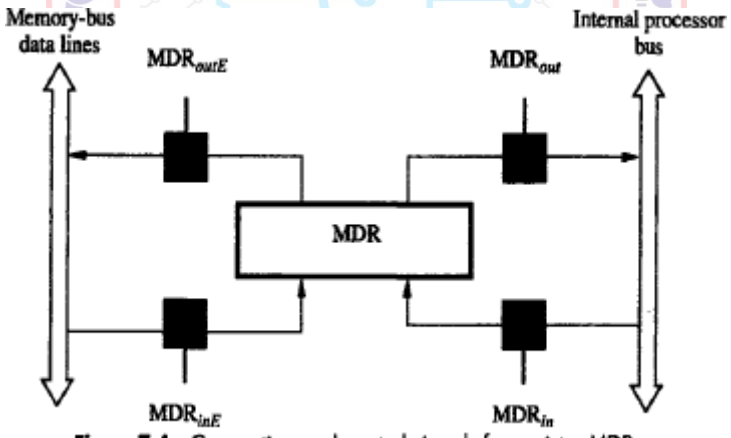


Figure 7.4 Connection and control signals for register MDR.

FETCHING A WORD FROM MEMORY

- To fetch instruction/data from memory, processor transfers required address to MAR. At the same time, processor issues Read signal on control-lines of memory-bus.
- When requested-data are received from memory, they are stored in MDR. From MDR, they are transferred to other registers.
- The response time of each memory access varies (based on cache miss, memory-mapped I/O). To accommodate this, MFC is used. (MFC □ Memory Function Completed).
- MFC is a signal sent from addressed-device to the processor. MFC informs the processor that the

requested operation has been completed by addressed-device.

- Consider the instruction Move (R1),R2. The sequence of steps is (Figure 7.5):

1) R1out, MARin, Read ;desired address is loaded into MAR & Read command is issued.
2) MDRinE, WMFC ;load MDR from memory-bus & Wait for MFC response from memory.
3) MDRout, R2in ;load R2 from MDR.

where WMFC=control-signal that causes processor's control. circuitry to wait for arrival of MFC signal.

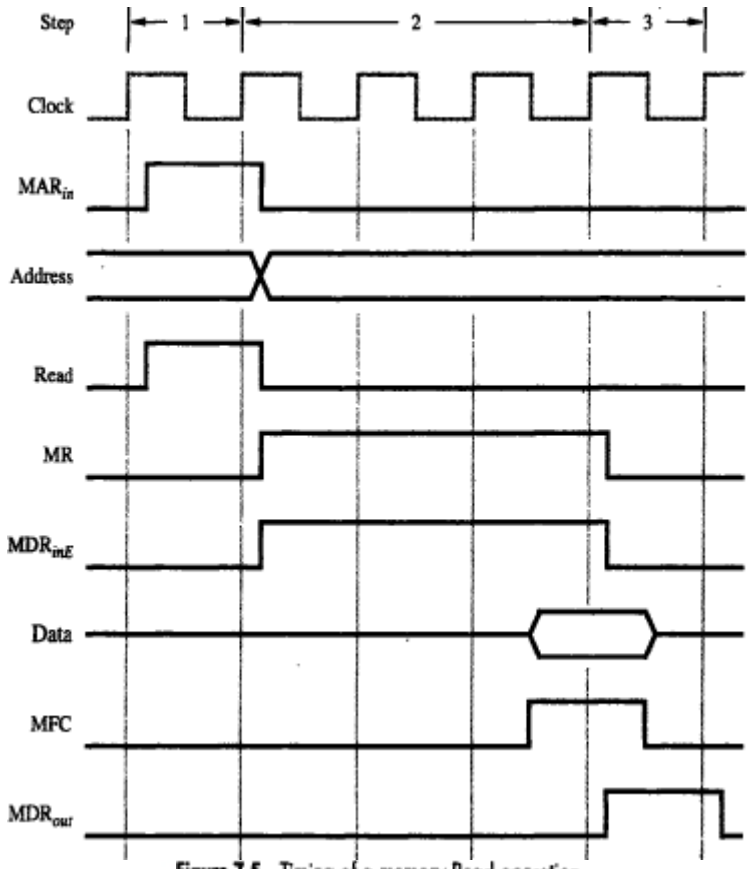


Figure 7.5 Timing of a memory Read operation.

Storing a Word in Memory

- Consider the instruction *Move R2,(R1).* This requires the following sequence:

1) R1out, MARin ;desired address is loaded into MAR.
2) R2out, MDRin, Write ;data to be written are loaded into MDR & Write command is issued.
3) MDRoutE, WMFC ;load data into memory-location pointed by R1 from MDR.

EXECUTION OF A COMPLETE INSTRUCTION

- Consider the instruction *Add (R3),R1* which adds the contents of a memory-location pointed by R3 to register R1. Executing this instruction requires the following actions:

1) Fetch the instruction.
2) Fetch the first operand.
3) Perform the addition &
4) Load the result into R1.

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, SelectY, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

Figure 7.6 Control sequence for execution of the instruction Add (R3),R1

- Instruction execution proceeds as follows:

Step1--> The instruction-fetch operation is initiated by
→ loading contents of PC into MAR &
→ sending a Read request to memory.
The Select signal is set to Select4, which causes the Mux to select constant 4. This value is added to operand at input B (PC‟s content), and the result is stored in Z.

Step2--> Updated value in Z is moved to PC. This completes the PC increment operation and PC will now point to next instruction.

Step3--> Fetched instruction is moved into MDR and then to IR.
The step 1 through 3 constitutes the **Fetch Phase**.
At the beginning of step 4, the instruction decoder interprets the contents of the IR. This enables the control circuitry to activate the control-signals for steps 4 through 7.
The step 4 through 7 constitutes the **Execution Phase**.

Step4--> Contents of R3 are loaded into MAR & a memory read signal is issued.

Step5--> Contents of R1 are transferred to Y to prepare for addition.

Step6--> When Read operation is completed, memory-operand is available in MDR, and the addition is performed.

Step7--> Sum is stored in Z, then transferred to R1.The End signal causes a new instruction fetch cycle to begin by returning to step1.

EXECUTION OF A COMPLETE INSTRUCTION

- Consider the instruction *Add (R3),R1* which adds the contents of a memory-location pointed by R3 to register R1. Executing this instruction requires the following actions:

5) Fetch the instruction.
6) Fetch the first operand.
7) Perform the addition &
8) Load the result into R1.

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, SelectY, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

Figure 7.6 Control sequence for execution of the instruction Add (R3),R1

- Instruction execution proceeds as follows:

Step1--> The instruction-fetch operation is initiated by
→ loading contents of PC into MAR &
→ sending a Read request to memory.
The Select signal is set to Select4, which causes the Mux to select constant 4. This value is added to operand at input B (PC‟s content), and the result is stored in Z.

Step2--> Updated value in Z is moved to PC. This completes the PC increment operation and PC will now point to next instruction.

Step3--> Fetched instruction is moved into MDR and then to IR.
The step 1 through 3 constitutes the **Fetch Phase**.
At the beginning of step 4, the instruction decoder interprets the contents of the IR. This enables the control circuitry to activate the control-signals for steps 4 through 7.
The step 4 through 7 constitutes the **Execution Phase**.

Step4--> Contents of R3 are loaded into MAR & a memory read signal is issued.

Step5--> Contents of R1 are transferred to Y to prepare for addition.

Step6--> When Read operation is completed, memory-operand is available in MDR, and the addition is performed.

Step7--> Sum is stored in Z, then transferred to R1.The End signal causes a new instruction fetch cycle to begin by returning to step1.

MULTIPLE BUS ORGANIZATION

- **Disadvantage of Single-bus organization:** Only one data-word can be transferred over the bus in a clock cycle. This increases the steps required to complete the execution of the instruction

Solution: To reduce the number of steps, most processors provide multiple internal-paths. Multiple paths enable several transfers to take place in parallel.

- As shown in fig 7.8, three buses can be used to connect registers and the ALU of the processor.
- All general-purpose registers are grouped into a single block called the **Register File**.
- Register-file has 3 ports:
  1) Two output-ports allow the contents of 2 different registers to be simultaneously placed on buses A & B.
  2) Third input-port allows data on bus C to be loaded into a third register during the same clock-cycle.
- Buses A and B are used to transfer source-operands to A & B inputs of ALU.
- The result is transferred to destination over bus C.

Step	Action
1	PC_{out}, R=B, MAR_{in}, Read, IncPC
2	WMFC
3	MDR_{outB}, R=B, IR_{in}
4	$R4_{outA}$, $R5_{outB}$, SelectA, Add, $R6_{in}$, End

Figure 7.9 Control sequence for the instruction Add R4,R5,R6

- **Incrementer Unit** is used to increment PC by 4.
- Instruction execution proceeds as follows:

Step 1--> Contents of PC are
→ passed through ALU using R=B control-signal &
→ loaded into MAR to start memory Read operation. At the same time, PC is incremented by 4.
Step2--> Processor waits for MFC signal from memory.
Step3--> Processor loads requested-data into MDR, and then transfers them to IR.
Step4--> The instruction is decoded and add operation takes place in a single step.

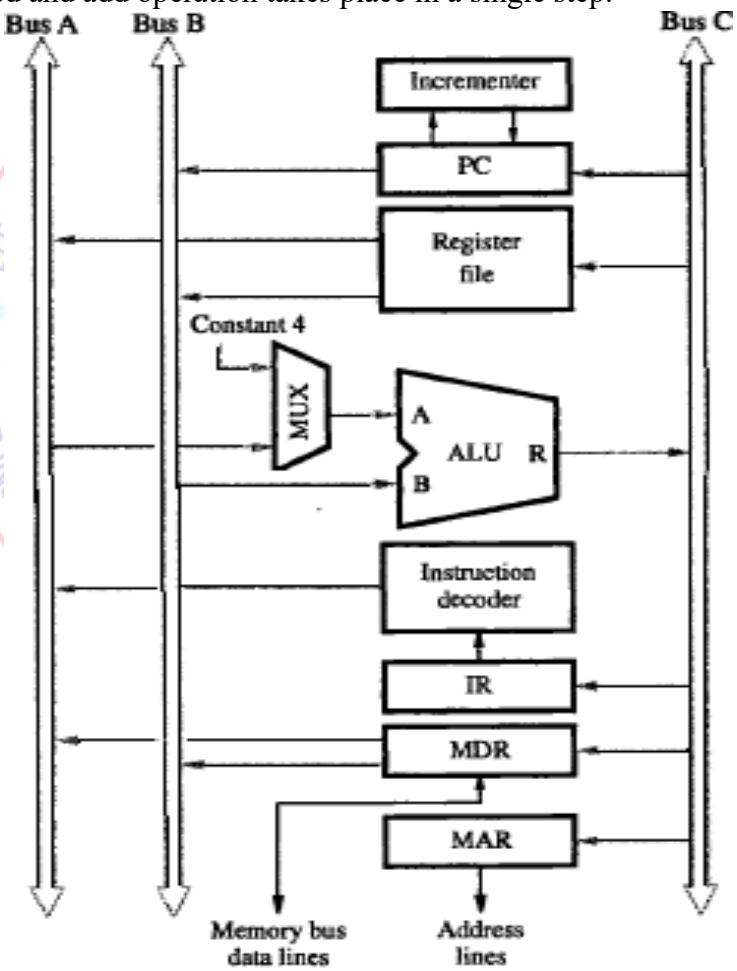


Figure 7.8 Three-bus organization of the datapath.

COMPLETE PROCESSOR

- This has separate processing-units to deal with integer data and floating-point data.

Integer Unit → To process integer data. (Figure 7.14).

Floating Unit → To process floating –point data.

- **Data-Cache** is inserted between these processing-units & main-memory. The integer and floating unit gets data from data cache.
- **Instruction-Unit** fetches instructions
  → from an instruction-cache or
  → from main-memory when desired instructions are not already in cache.
- Processor is connected to system-bus & hence to the rest of the computer by means of a **Bus Interface.**

- Using separate caches for instructions & data is common practice in many processors today.
- A processor may include several units of each type to increase the potential for concurrent operations.
- The 80486 processor has 8-kbytes single cache for both instruction and data.
  Whereas the Pentium processor has two separate 8 kbytes caches for instruction and data.

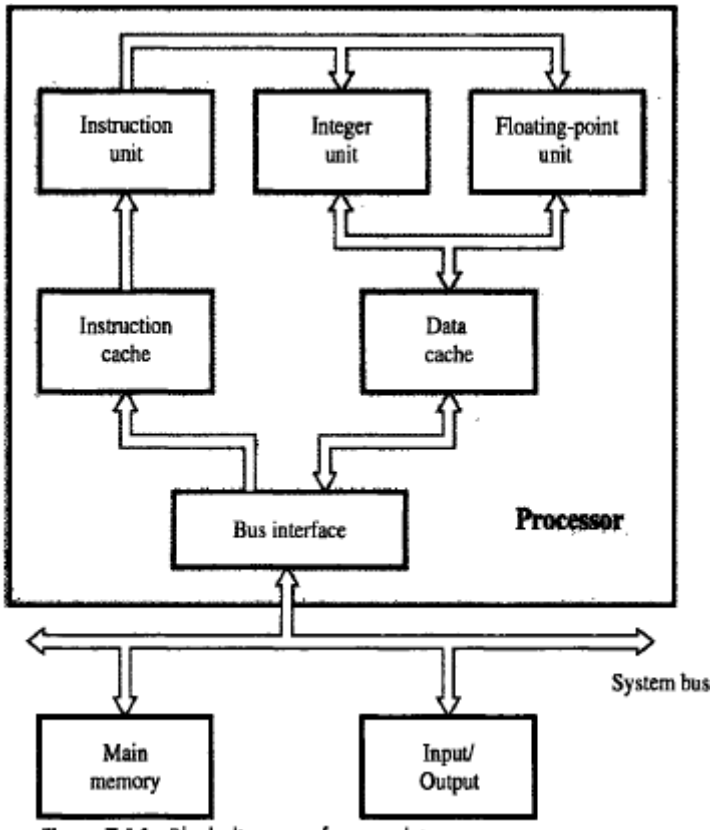


Figure 7.14 Block diagram of a complete processor.

Note:

To execute instructions, the processor must have some means of generating the control-signals. There are two approaches for this purpose:

1) Hardwired control and 2) Microprogrammed control.

HARDWIRED CONTROL

- Hardwired control is a method of control unit design (Figure 7.11).
- The control-signals are generated by using logic circuits such as gates, flip-flops, decoders etc.
- **Decoder/Encoder Block** is a combinational-circuit that generates required control-outputs depending on state of all its inputs.
- **Instruction Decoder**
  - It decodes the instruction loaded in the IR.

- If IR is an 8 bit register, then instruction decoder generates 2^8(256 lines); one for each instruction.
- It consists of a separate output-lines INS1 through INSm for each machine instruction.
- According to code in the IR, one of the output-lines INS1 through INSm is set to 1, and all other lines are set to 0.

- **Step-Decoder** provides a separate signal line for each step in the control sequence.
- **Encoder**
  - It gets the input from instruction decoder, step decoder, external inputs and condition codes.
  - It uses all these inputs to generate individual control-signals: Yin, PCout, Add, End and so on.
  - For example (Figure 7.12), *Zin=T1+T6.ADD+T4.BR*

This signal is asserted during time-slot T1 for all instructions. during T6 for an Add instruction. during T4 for unconditional branch instruction

- When **RUN**=1, counter is incremented by 1 at the end of every clock cycle.
  When RUN=0, counter stops counting.
- After execution of each instruction, **end** signal is generated. End signal resets step counter.
- Sequence of operations carried out by this machine is determined by wiring of logic circuits, hence the name "**hardwired**".
- **Advantage**: Can operate at high speed.
- **Disadvantages:**
  1) Since no. of instructions/control-lines is often in hundreds, the complexity of control unit is very high.
  2) It is costly and difficult to design.
  3) The control unit is inflexible because it is difficult to change the design.

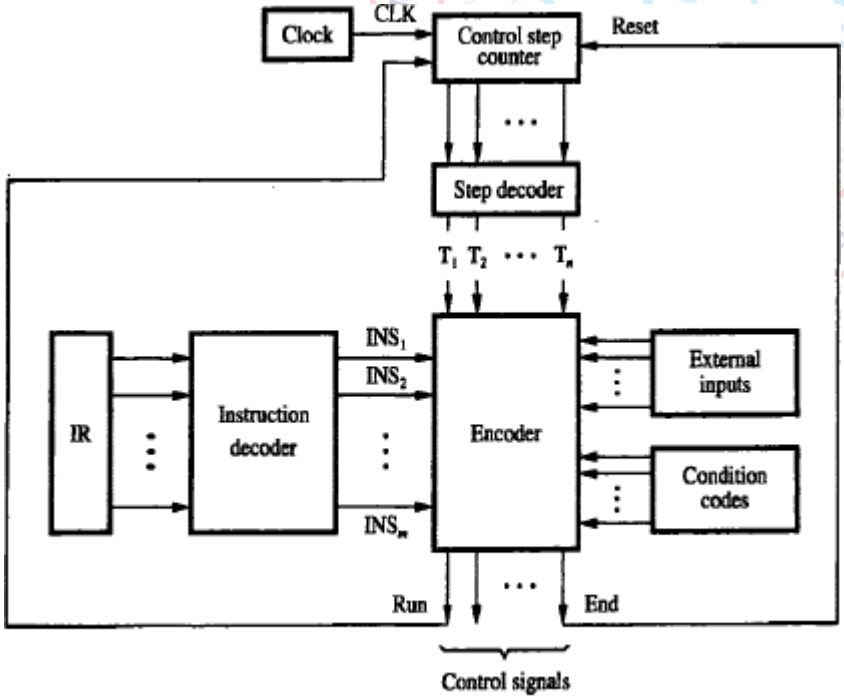


Figure 7.11 Separation of the decoding and encoding functions.

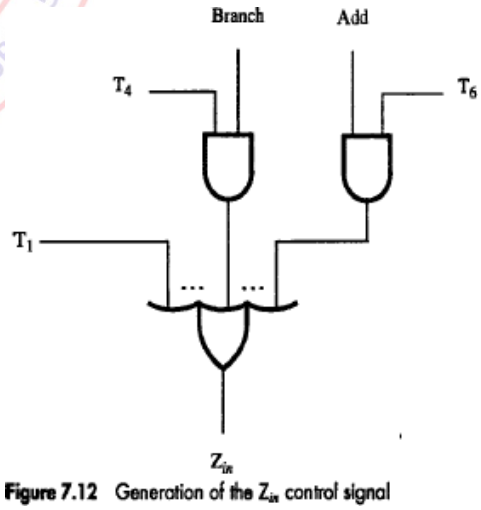
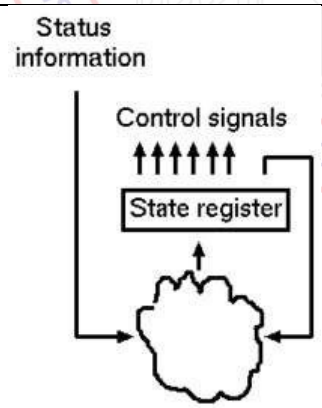
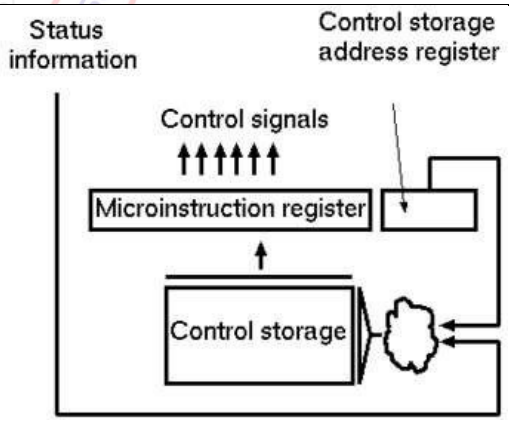


Figure 7.12 Generation of the Z_{in} control signal

HARDWIRED CONTROL VS MICROPROGRAMMED CONTROL

Attribute	Hardwired Control	Microprogrammed Control
Definition	Hardwired control is a control mechanism to generate control-signals by using gates, flip-flops, decoders, and other digital circuits.	Micro programmed control is a control mechanism to generate control-signals by using a memory called control store (CS), which contains the control-signals.
Speed	Fast	Slow
Control functions	Implemented in hardware.	Implemented in software.
Flexibility	Not flexible to accommodate new system specifications or new instructions.	More flexible, to accommodate new system specification or new instructions redesign is required.
Ability to handle large or complex instruction sets	Difficult.	Easier.
Ability to support operating systems & diagnostic features	Very difficult.	Easy.
Design process	Complicated.	Orderly and systematic.
Applications	Mostly RISC microprocessors.	Mainframes, some microprocessors.
Instructionset size	Usually under 100 instructions.	Usually over 100 instructions.
ROM size	-	2K to 10K by 20-400 bit microinstructions.
Chip area efficiency	Uses least area.	Uses more area.
Diagram		

MICROPROGRAMMED CONTROL

- Microprogramming is a method of control unit design (Figure 7.16).
- Control-signals are generated by a program similar to machine language programs.
- **Control Word(CW)** is a word whose individual bits represent various control-signals (like Add, PCin).
- Each of the control-steps in control sequence of an instruction defines a unique combination of 1s & 0s in CW.
- Individual control-words in microroutine are referred to as **microinstructions** (Figure 7.15).
- A sequence of CWs corresponding to control-sequence of a machine instruction constitutes the **microroutine.**
- The microroutines for all instructions in the instruction-set of a computer are stored in a special memory called the **Control Store (CS)**.
- Control-unit generates control-signals for any instruction by sequentially reading CWs of corresponding microroutine from CS.
- **µPC** is used to read CWs sequentially from CS. (µPC Microprogram Counter).
- Every time new instruction is loaded into IR, o/p of **Starting Address Generator** is loaded into µPC.
- Then, µPC is automatically incremented by clock;

causing successive microinstructions to be read from CS.

Hence, control-signals are delivered to various parts of processor in correct sequence.

Advantages

- It simplifies the design of control unit. Thus it is both, cheaper and less error prone implement.
- Control functions are implemented in software rather than hardware.
- The design process is orderly and systematic.
- More flexible, can be changed to accommodate new system specifications or to correct the design errors quickly and cheaply.
- Complex function such as floating point arithmetic can be realized efficiently.

Disadvantages

- A microprogrammed control unit is somewhat slower than the hardwired control unit, because time is required to access the microinstructions from CM.
- The flexibility is achieved at some extra hardware cost due to the control memory and its access circuitry.

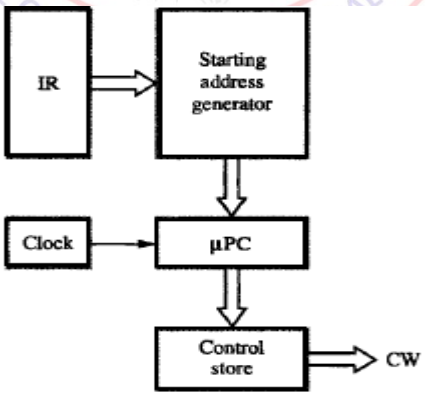


Figure 7.16 Basic organization of a microprogrammed control unit.

Micro-instruction	..	PC_{in}	PC_{out}	MAR_{in}	Read	MDR_{out}	IR_{in}	Y_{in}	Select	Add	Z_{in}	Z_{out}	$R1_{out}$	$R1_{in}$	$R3_{out}$	WMFC	End	..
1		0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	
2		1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	
3		0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	

Figure 7.15 An example of microinstructions for Figure 7.6.

ORGANIZATION OF MICROPROGRAMMED CONTROL UNIT TO SUPPORT CONDITIONALBRANCHING

- **Drawback of previous Microprogram control:**
  - ➢ It cannot handle the situation when the control unit is required to check the status of the condition codes or external inputs to choose between alternative courses of action.

Solution:

- ➢ Use conditional branch microinstruction.

- In case of conditional branching, microinstructions specify which of the external inputs, condition-codes should be checked as a condition for branching to take place.
- **Starting and Branch Address Generator Block** loads a new address into µPC when a microinstruction instructs it to do so (Figure 7.18).
- To allow implementation of a conditional branch, inputs to this block consist of
  - → external inputs and condition-codes &
  - → contents of IR.
- µPC is incremented every time a new microinstruction is fetched from microprogram memory except in following situations:
  1) When a new instruction is loaded into IR, µPC is loaded with starting-address of microroutine for that instruction.
  2) When a Branch microinstruction is encountered and branch condition is satisfied, µPC is loaded with branch-address.
  3) When an End microinstruction is encountered, µPC is loaded with address of first CW in microroutine for instruction fetch cycle.

Address	Microinstruction
0	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
1	Z_{out}, PC_{in}, Y_{in}, WMFC
2	MDR_{out}, IR_{in}
3	Branch to starting address of appropriate microroutine
25	If N=0, then branch to microinstruction 0
26	Offset-field-of-IR_{out}, SelectY, Add, Z_{in}
27	Z_{out}, PC_{in}, End

Figure 7.17 Microroutine for the instruction Branch < 0.

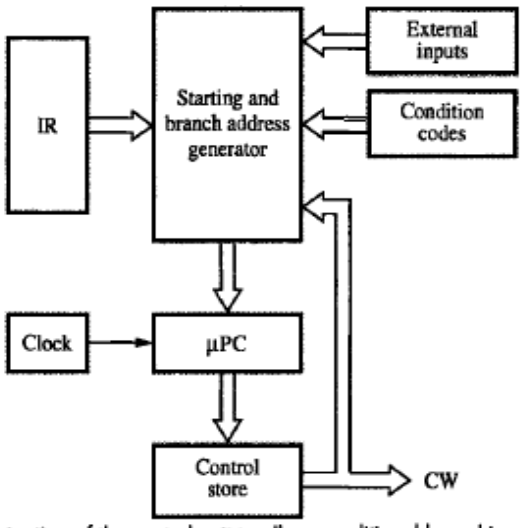


Figure 7.18 Organization of the control unit to allow conditional branching in the microprogram.

MICROINSTRUCTIONS

- A simple way to structure microinstructions is to assign one bit position to each control-signal required in the CPU.
- There are 42 signals and hence each microinstruction will have 42 bits.
- **Drawbacks of microprogrammed control:**
  1) Assigning individual bits to each control-signal results in long microinstructions because the number of required signals is usually large.
  2) Available bit-space is poorly used because only a few bits are set to 1 in any given microinstruction.
- **Solution:** Signals can be grouped because
  1) Most signals are not needed simultaneously.
  2) Many signals are mutually exclusive. E.g. only 1 function of ALU can be activated at a time.
     For ex: Gating signals: IN and OUT signals (Figure 7.19).
     Control-signals: Read, Write.
     ALU signals: Add, Sub, Mul, Div, Mod.
- Grouping control-signals into fields requires a little more hardware because decoding-circuits must be used to decode bit patterns of each field into individual control-signals.
- **Advantage:** This method results in a smaller control-store (only 20 bits are needed to store the patterns for the 42 signals).

Microinstruction

F1	F2	F3	F4	F5

F1 (4 bits)	F2 (3 bits)	F3 (3 bits)	F4 (4 bits)	F5 (2 bits)
0000: No transfer	000: No transfer	000: No transfer	0000: Add	00: No action
0001: PC_{out}	001: PC_{in}	001: MAR_{in}	0001: Sub	01: Read
0010: MDR_{out}	010: IR_{in}	010: MDR_{in}	⋮	10: Write
0011: Z_{out}	011: Z_{in}	011: $TEMP_{in}$	1111: XOR	
0100: $R0_{out}$	100: $R0_{in}$	100: Y_{in}	(16 ALU functions)	
0101: $R1_{out}$	101: $R1_{in}$			
0110: $R2_{out}$	110: $R2_{in}$			
0111: $R3_{out}$	111: $R3_{in}$			
1010: $TEMP_{out}$				
1011: $Offset_{out}$				

F6	F7	F8	...

F6 (1 bit)	F7 (1 bit)	F8 (1 bit)
0: SelectY	0: No action	0: Continue
1: Select4	1: WMFC	1: End

Figure 7.19 An example of a partial format for field-encoded microinstructions.

TECHNIQUES OF GROUPING OF CONTROL-SIGNALS

- The grouping of control-signal can be done either by using
  1) Vertical organization &
  2) Horizontal organisation.

Vertical Organization	Horizontal Organization
Highly encoded schemes that use compact codes to specify only a small number of control functions in each microinstruction are referred to as a vertical organization.	The minimally encoded scheme in which many resources can be controlled with a single microinstuction is called a horizontal organization.
Slower operating-speeds.	Useful when higher operating-speed is desired.
Short formats.	Long formats.
Limited ability to express parallel microoperations.	Ability to express a high degree of parallelism.
Considerable encoding of the control information.	Little encoding of the control information.

MICROPROGRAM SEQUENCING

- The task of microprogram sequencing is done by microprogram sequencer.
- Two important factors must be considered while designing the microprogram sequencer:
  1) The size of the microinstruction &
  2) The address generation time.
- The size of the microinstruction should be minimum so that the size of control memory required to store microinstructions is also less.
- This reduces the cost of control memory.
- With less address generation time, microinstruction can be executed in less time resulting better throughout.
- During execution of a microprogram the address of the next microinstruction to be executed has 3 sources:
  1) Determined by instruction register.
  2) Next sequential address &
  3) Branch.
- Microinstructions can be shared using microinstruction branching.
- **Disadvantage of microprogrammed branching**:
  1) Having a separate microroutine for each machine instruction results in a large total number of microinstructions and a large control-store.
  2) Execution time is longer because it takes more time to carry out the required branches.
- Consider the instruction *Add src,Rdst* ;which adds the source-operand to the contents of Rdst and places the sum in Rdst.
- Let source-operand can be specified in following addressing modes (Figure 7.20):
  a) Indexed
  b) Autoincrement
  c) Autodecrement
  d) Register indirect &
  e) Register direct
- Each box in the chart corresponds to a microinstruction that controls the transfers and operations indicated within the box.
- The microinstruction is located at the address indicated by the octal number (001,002).

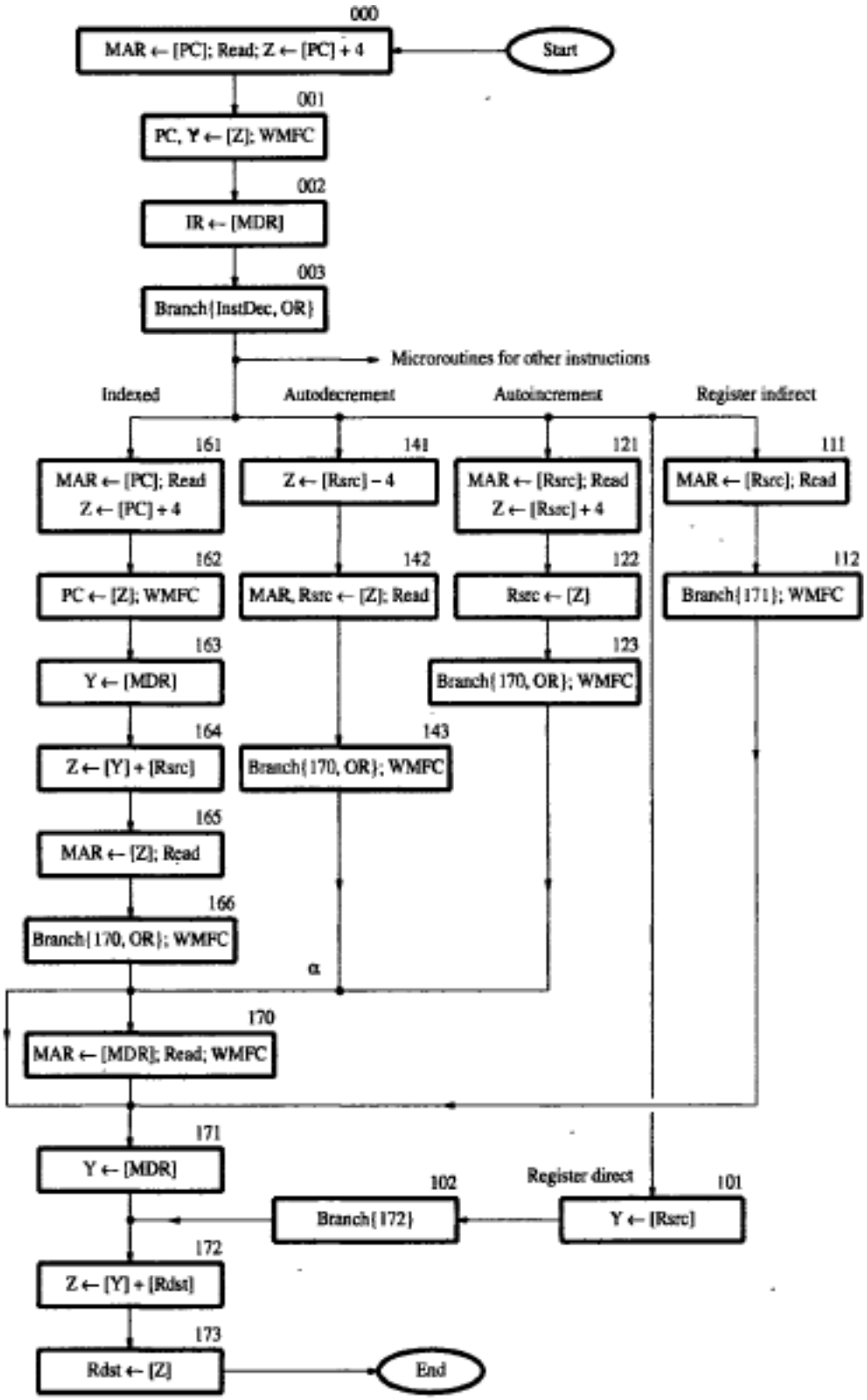


Figure 7.20 Flowchart of a microprogram for the Add src,Rdst instruction.

BRANCH ADDRESS MODIFICATION USING BIT-ORING

- The branch address is determined by ORing particular bit or bits with the current address of microinstruction.
- **Eg:** If the current address is 170 and branch address is 171 then the branch address can be generated by ORing 01(bit 1), with the current address.
- Consider the point labeled α in the figure. At this point, it is necessary to choose between direct and indirect addressing modes.
- If indirect-mode is specified in the instruction, then the microinstruction in location 170 is performed to fetch the operand from the memory.

  If direct-mode is specified, this fetch must be bypassed by branching immediately to location 171.
- The most efficient way to bypass microinstruction 170 is to have bit-ORing of
  → current address 170 &
  → branch address 171.

WIDE BRANCH ADDRESSING

- The instruction-decoder (InstDec) generates the starting-address of the microroutine that implements the instruction that has just been loaded into the IR.
- Here, register IR contains the Add instruction, for which the instruction decoder generates the microinstruction address 101. (However, this address cannot be loaded as is into the μPC).
- The source-operand can be specified in any of several addressing-modes. The bit-ORing technique can be used to modify the starting-address generated by the instruction-decoder to reach the appropriate path.

Use of WMFC

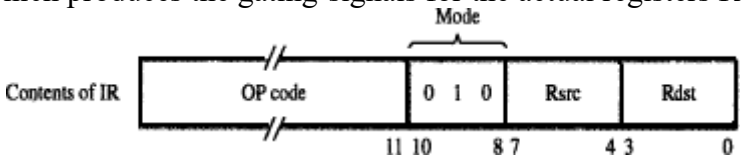
- WMFC signal is issued at location 112 which causes a branch to the microinstruction in location 171.
- WMFC signal means that the microinstruction may take several clock cycles to complete. If the branch is allowed to happen in the first clock cycle, the microinstruction at location 171 would be fetched and executed prematurely. To avoid this problem, WMFC signal must inhibit any change in the contents of the μPC during the waiting-period.

Detailed Examination of Add (Rsrc)+,Rdst

- Consider *Add (Rsrc)+,Rdst;* which adds Rsrc content to Rdst content, then stores the sum in Rdst and finally increments Rsrc by 4 (i.e. auto-increment mode).
- In bit 10 and 9, bit-patterns 11, 10, 01 and 00 denote indexed, auto-decrement, auto-increment and register modes respectively. For each of these modes, bit 8 is used to specify the indirect version.
- The processor has 16 registers that can be used for addressing purposes; each specified using a 4- bit-code (Figure 7.21).
- There are 2 stages of decoding:

  1) The microinstruction field must be decoded to determine that an Rsrc or Rdst register is involved.

  2) The decoded output is then used to gate the contents of the Rsrc or Rdst fields in the IR into a second decoder, which produces the gating-signals for the actual registers R0 to R15.



MICROINSTRUCTIONS WITH NEXT-ADDRESS FIELDS

Address (octal)	Microinstruction
000	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
001	Z_{out}, PC_{in}, Y_{in}, WMFC
002	MDR_{out}, IR_{in}
003	μBranch {μPC ← 101 (from Instruction decoder); $\mu PC_{5,4} \leftarrow [IR_{10,9}]$; $\mu PC_3 \leftarrow [\overline{IR_{10}}] \cdot [\overline{IR_9}] \cdot [IR_8]$}
121	$Rsrc_{out}$, MAR_{in}, Read, Select4, Add, Z_{in}
122	Z_{out}, $Rsrc_{in}$
123	μBranch {μPC ← 170; $\mu PC_0 \leftarrow [\overline{IR_8}]$}, WMFC
170	MDR_{out}, MAR_{in}, Read, WMFC
171	MDR_{out}, Y_{in}
172	$Rdst_{out}$, SelectY, Add, Z_{in}
173	Z_{out}, $Rdst_{in}$, End

Figure 7.21 Microinstruction for Add (Rsrc)+,Rdst.

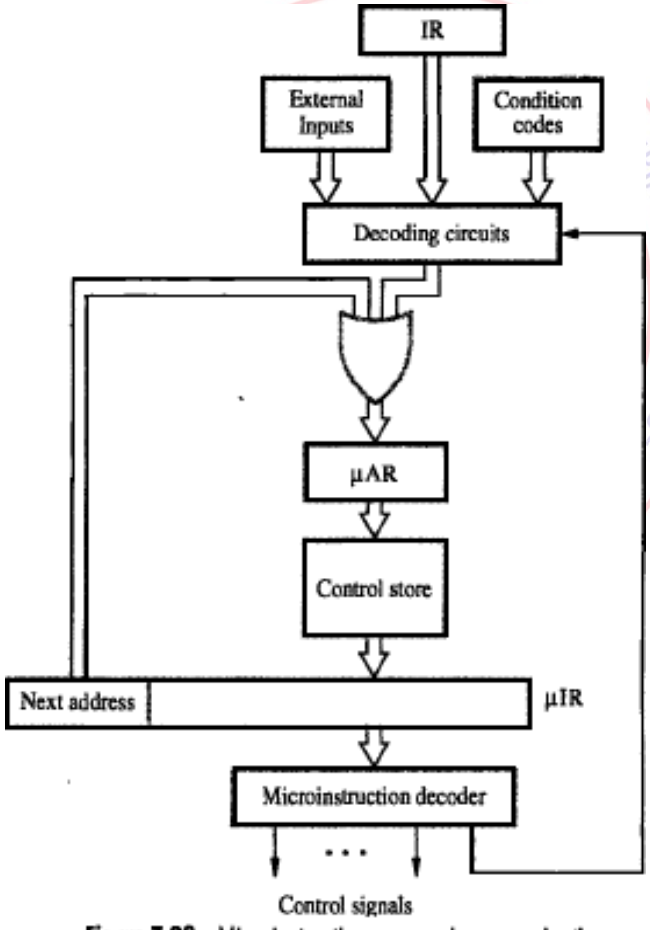


Figure 7.22 Microinstruction-sequencing organization.

- **Drawback of previous organization:**
  - The microprogram requires several branch microinstructions which perform no useful operation. Thus, they detract from the operating-speed of the computer.

Solution:

- Include an address-field as a part of every microinstruction to indicate the location of the next microinstruction to be fetched. (Thus, every microinstruction becomes a branch microinstruction).

- The flexibility of this approach comes at the expense of additional bits for the address-field(Fig 7.22).
- **Advantage:** Separate branch microinstructions are virtually eliminated. (Figure 7.23-24).
- **Disadvantage:** Additional bits for the address field (around 1/6).
- There is no need for a counter to keep track of sequential address. Hence, µPC is replaced with µAR.
- The next-address bits are fed through the OR gate to the µAR, so that the address can be modified on the basis of the data in the IR, external inputs and condition-codes.
- The decoding circuits generate the starting-address of a given microroutine on the basis of the opcode in the IR. (µAR Microinstruction Address Register).

Microinstruction

F0	F1	F2	F3

F0 (8 bits)	F1 (3 bits)	F2 (3 bits)	F3 (3 bits)
Address of next microinstruction	000: No transfer 001: PC_{out} 010: MDR_{out} 011: Z_{out} 100: $Rsrc_{out}$ 101: $Rdst_{out}$ 110: $TEMP_{out}$	000: No transfer 001: PC_{in} 010: IR_{in} 011: Z_{in} 100: $Rsrc_{in}$ 101: $Rdst_{in}$	000: No transfer 001: MAR_{in} 010: MDR_{in} 011: $TEMP_{in}$ 100: Y_{in}

F4	F5	F6	F7

F4 (4 bits)	F5 (2 bits)	F6 (1 bit)	F7 (1 bit)
0000: Add 0001: Sub ⋮ 1111: XOR	00: No action 01: Read 10: Write	0: SelectY 1: Select4	0: No action 1: WMFC

F8	F9	F10

F8 (1 bit)	F9 (1 bit)	F10 (1 bit)
0: NextAdrs 1: InstDec	0: No action 1: OR_{mode}	0: No action 1: OR_{indsrc}

Figure 7.23 Format for microinstructions in the example of Section 7.5.3.

Octal address	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
000	00000001	001	011	001	0000	01	1	0	0	0	0
001	00000010	011	001	100	0000	00	0	1	0	0	0
002	00000011	010	010	000	0000	00	0	0	0	0	0
003	00000000	000	000	000	0000	00	0	0	1	1	0
121	01010010	100	011	001	0000	01	1	0	0	0	0
122	01111000	011	100	000	0000	00	0	1	0	0	1
170	01111001	010	000	001	0000	01	0	1	0	0	0
171	01111010	010	000	100	0000	00	0	0	0	0	0
172	01111011	101	011	000	0000	00	0	0	0	0	0
173	00000000	011	101	000	0000	00	0	0	0	0	0

Figure 7.24 Implementation of the microroutine of Figure 7.21 using a next-microinstruction address field. (See Figure 7.23 for encoded signals.)

PREFETCHING MICROINSTRUCTIONS

- **Disadvantage of Microprogrammed Control:** Slower operating-speed because of the time it takes to fetch microinstructions from the control-store.

  Solution: Faster operation is achieved if the next microinstruction is pre-fetched while the current one is being executed.

Emulation

- The main function of microprogrammed control is to provide a means for simple, flexible and relatively inexpensive execution of machine instruction.
- Its flexibility in using a machine's resources allows diverse classes of instructions to be implemented.
- Suppose we add to the instruction-repository of a given computer M1, an entirely new set of instructions that is in fact the instruction-set of a different computer M2.
- Programs written in the machine language of M2 can be then be run on computer M1 i.e. M1 emulates M2.
- Emulation allows us to replace obsolete equipment with more up-to-date machines.
- If the replacement computer fully emulates the original one, then no software changes have to be made to run existing programs.
- Emulation is easiest when the machines involved have similar architectures.

Problem 1:

Why is the Wait-for-memory-function-completed step needed for reading from or writing to the main memory?

Solution:

The WMFC step is needed to synchronize the operation of the processor and the main memory.

Problem 2:

For the single bus organization, write the complete control sequence for the instruction: Move (R1), R1

Solution:

PCout, MARin, Read, Select4, Add, Zin

2) Zout, PCin, Yin, WMFC
3) MDRout, IRin
4) R1out, MARin, Read
5) MDRinE, WMFC
6) MDRout, R2in, End

Problem 3:

1)

Write the sequence of control steps required for the single bus organization in each of the following instructions:

a) Add the immediate number NUM to register R1.
b) Add the contents of memory-location NUM to register R1.
c) Add the contents of the memory-location whose address is at memory-location NUM to register R1.

Assume that each instruction consists of two words. The first word specifies the operation andN the addressing mode, and the second word contains the number NUM

Solution:

(*a*) 1. PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2. Z_{out}, PC_{in}, Y_{in}, WMFC
3. MDR_{out}, IR_{in}
4. PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
5. Z_{out}, PC_{in}, Y_{in}
6. $R1_{out}$, Y_{in}, WMFC
7. MDR_{out}, SelectY, Add, Z_{in}
8. Z_{out}, $R1_{in}$, End

(*b*) 1-4. Same as in (*a*)
5. Z_{out}, PC_{in}, WMFC
6. MDR_{out}, MAR_{in}, Read
7. $R1_{out}$, Y_{in}, WMFC
8. MDR_{out}, Add, Z_{in}
9. Z_{out}, $R1_{in}$, End

(*c*) 1-5. Same as in (*b*)
6. MDR_{out}, MAR_{in}, Read, WMFC
7-10. Same as 6-9 in (*b*)

Problem 4:

Show the control steps for the Branch on Negative instruction for a processor with three-bus organization of the data path

Solution:

1 PC_{out}, R=B, MAR_{in}, Read, IncPC

2 WMFC

3 MDR_{outB}, R=B, IR_{in}

4. PC_{out}, Offset field of IR_{out}, Add, If N = 1 then PC_{in}, End

ELECTRONICS & COMMUNICATION ENGINEERING
ECE
SJB INSTITUTE OF TECHNOLOGY, BENGALURU - 560060

ELECTRONICS & COMMUNICATION ENGINEERING
ECE
SJB INSTITUTE OF TECHNOLOGY, BENGALURU - 560060