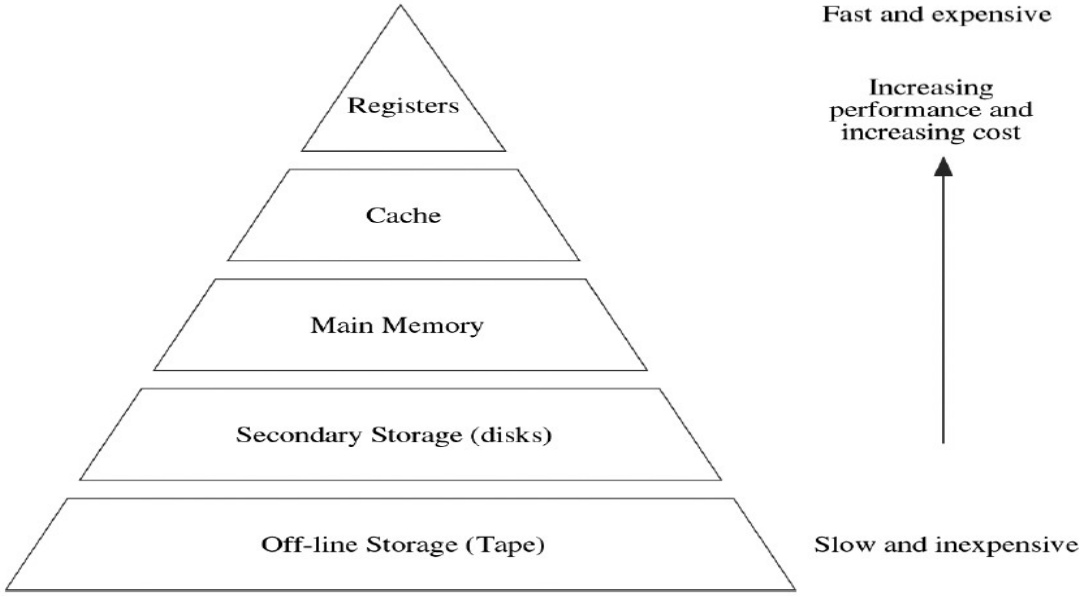
Unit-4

The Memory Hierarchy

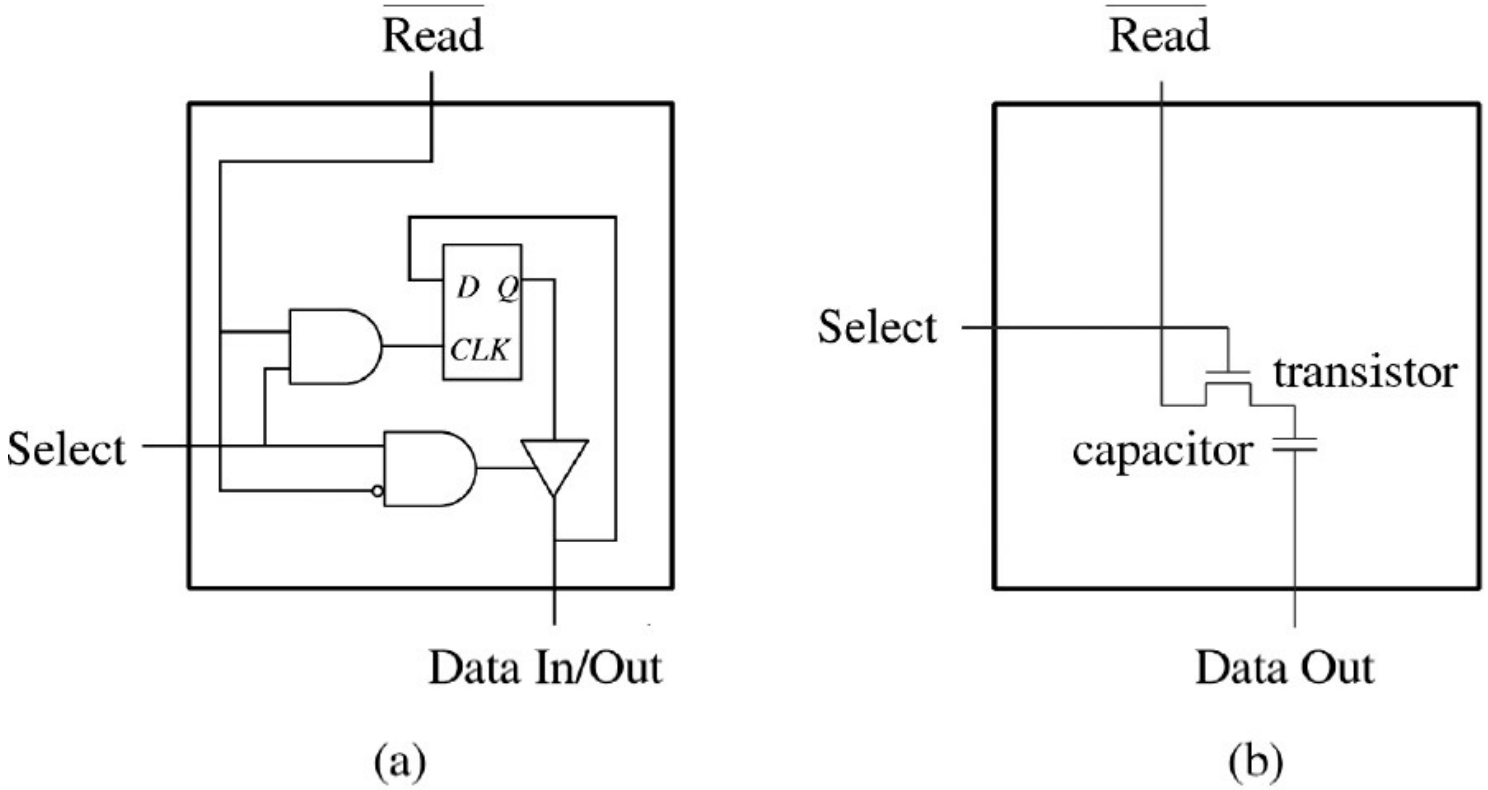


Memory type	Access time	Cost/MB	Typical amount used	Typical cost
Registers	0.5 ns	High	2 KB	–
Cache	5–20 ns	$80	2 MB	$160
Main memory	40–80ns	$0.40	512 MB	$205
Disk memory	5 ms	$0.005	40 GB	$200

The Memory Hierarchy

- At the top of the hierarchy are registers that are matched in speed to the CPU.
- small number of registers in a processor, on the order of a few hundred or less.
- At the bottom of the hierarchy are secondary and off-line storage memories such as hard magnetic disks and magnetic tapes in which the cost per stored bit is small in terms of money and electrical power, but the access time is very long when compared with registers.
- Between the registers and secondary storage are a number of other forms of memory that bridge the gap between the two.
- As we move up through the hierarchy, greater performance is realized, at a greater cost.
- Access times vary by approximately factors of 10 except for disks, which have access times 100,000 times slower than main memory.

Functional Behavior of a RAM Cell



Static RAM cell (a) and dynamic RAM cell (b).

Functional Behavior of a RAM Cell

- **random access memory** (RAM) the term “random” means that any memory location can be accessed in the same amount of time, regardless of its position in the memory.

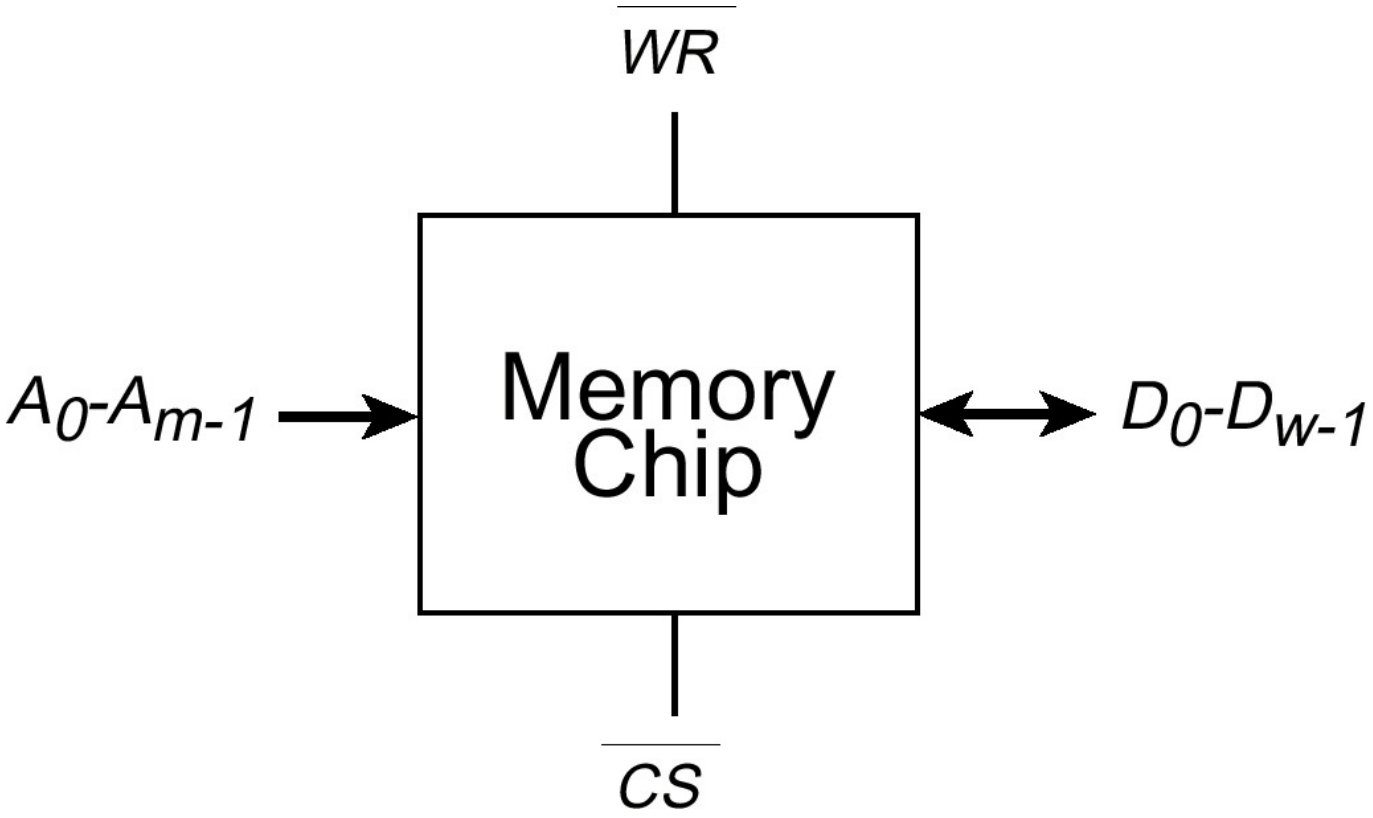
- The memory element as a D flip-flop, with additional controls to allow the cell to be selected, read, and written.
- There is a (bidirectional) data line for data input and output.

Functional Behavior of a RAM Cell

- RAM chips that are based upon flip-flops, as in Figure, are referred to as **static** RAM (SRAM), chips, because the contents of each location persist as long as power is applied to the chips.
- **Dynamic RAM** chips, referred to as **DRAM**s, employ a **capacitor**, which stores a minute amount of electric charge, in which the charge level represents a 1 or a 0.
- Capacitors are much smaller than flip-flops, and so a capacitor based DRAM can hold much more information in the same area than an SRAM. Since the charges on the capacitors dissipate with time, the charge in the capacitor storage cells in DRAMs must be restored, or **refreshed** frequently.

- With the drastic reduction in DRAM prices and the increased uptimes of PCs operating as automated teller machines (ATMs) and network file servers (NFSs), error detection circuitry is now commonplace in PCs.

Simplified RAM Chip Pinout

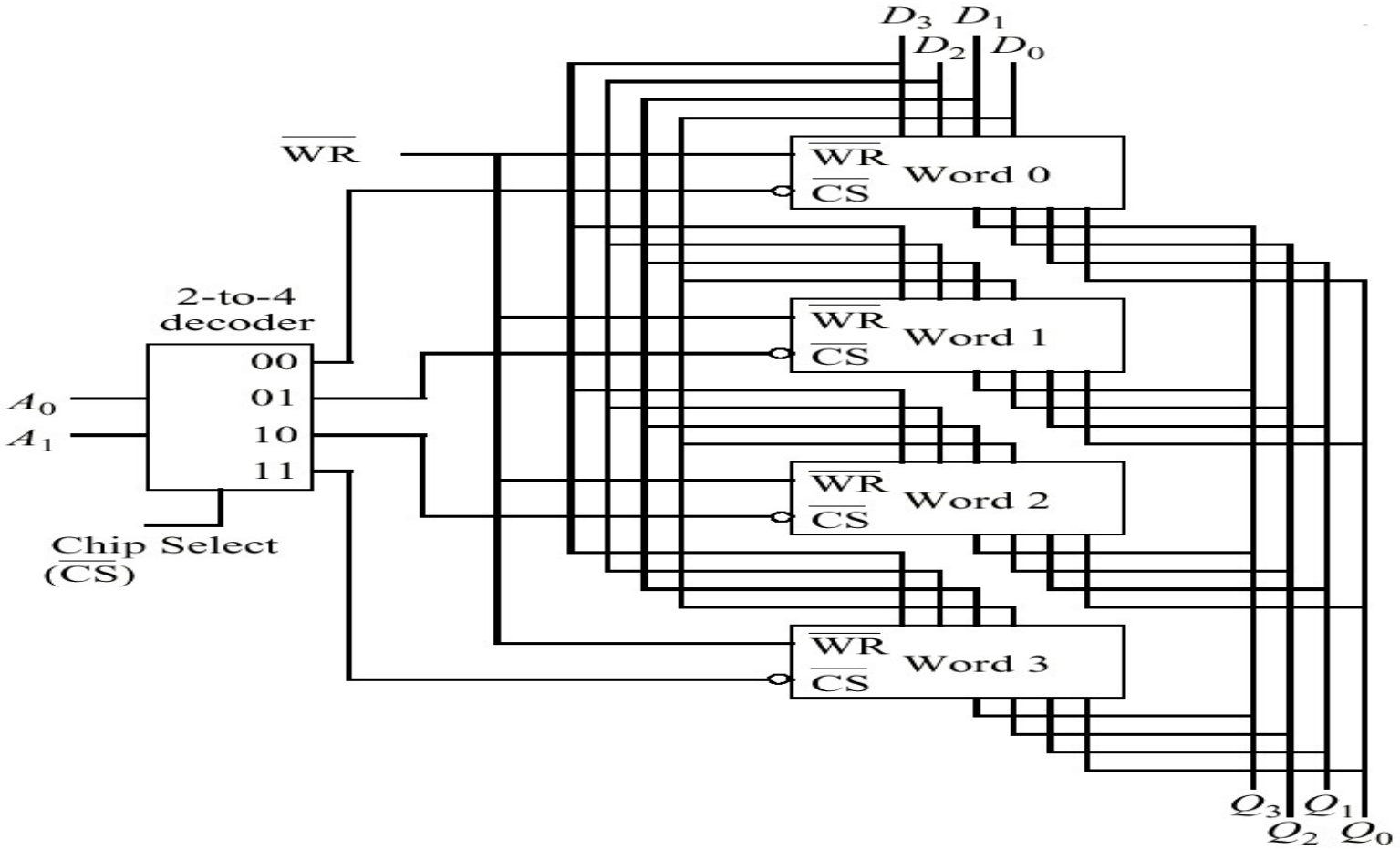


Simplified RAM Chip Pinout

- An *m*-bit address, having lines numbered from 0 to *m*-1 is applied to pins *A*0-*Am*-1.
- While asserting CS (Chip Select), and either WR (for writing data to the chip) or WR (for reading data from the chip).
- The overbars on CS and WR indicate that the chip is selected when CS=0 and that a write operation will occur when WR=0.
- When reading data from the chip, after a time period *tAA* (the time delay from when the address lines are made valid to the time the data is available at the output), the *w*-bit data word appears on the data lines *D*0- *Dw*-1.
- When writing data to a chip, the data lines must also be held valid for a time period *tAA*.
- The data lines are bidirectional.

- The address lines *A*0- *Am*-1 in the RAM chip shown in contain an address, which is decoded from an *m*-bit address into one of 2^*m* locations within the chip, each of which has a *w*-bit word associated with it. The chip thus contains 2^*m* ´ *w* bits.

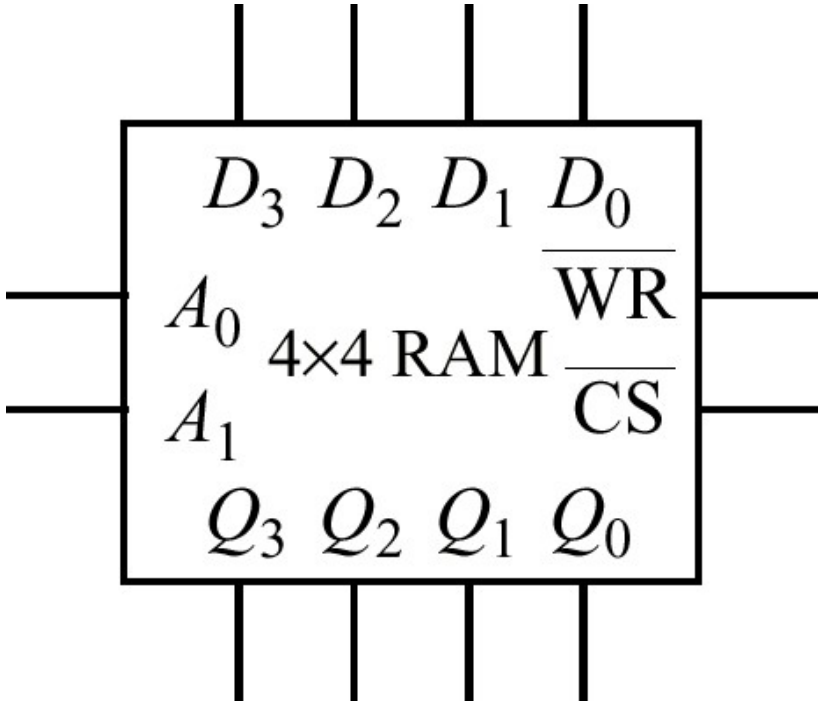
A Four-Word Memory with Four Bits per Word in a 2D Organization



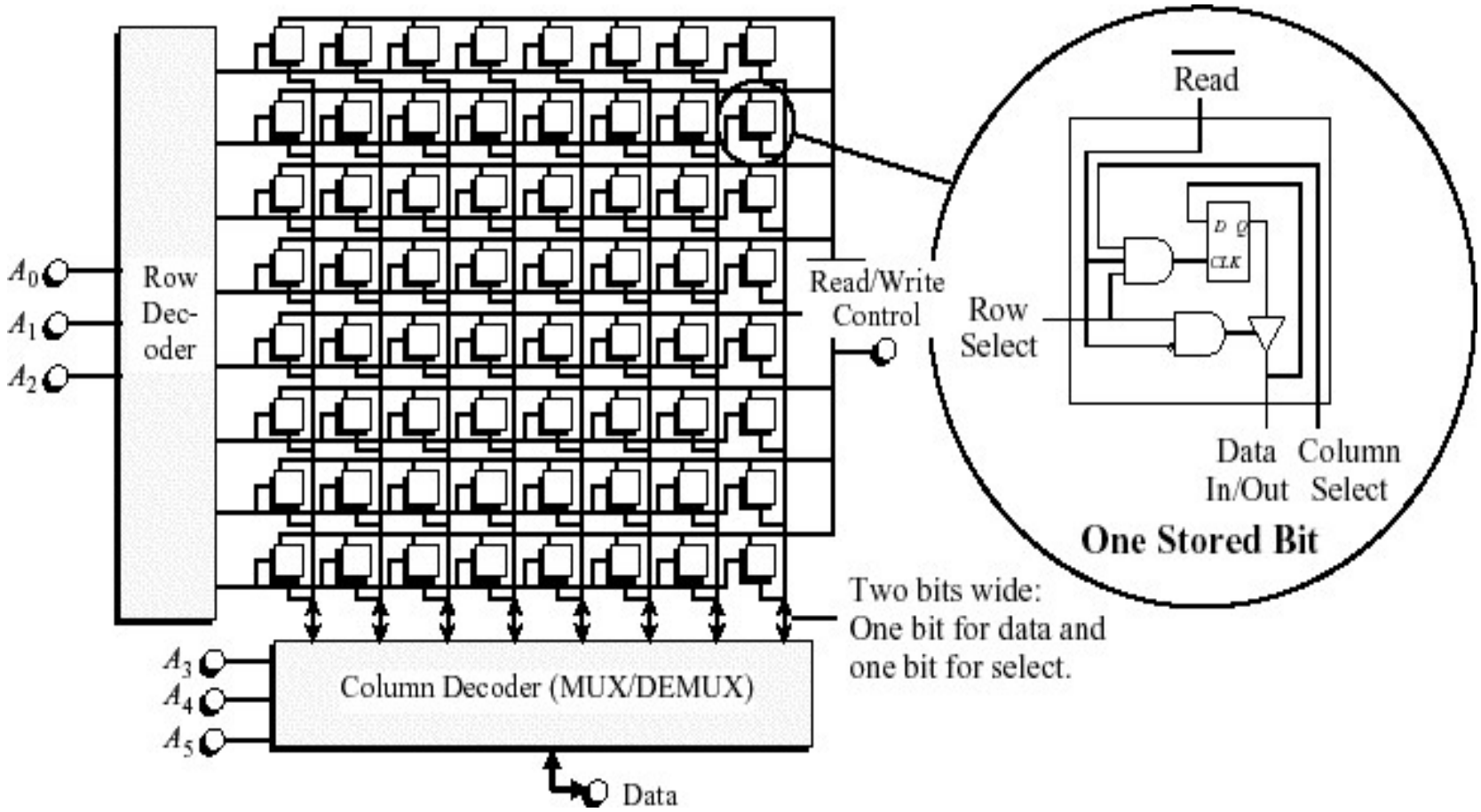
A Four-Word Memory with Four Bits per Word in a 2D Organization

- consider RAM that stores four four-bit words. (4 x4)
- A RAM can be thought of as a collection of registers.
- We can use four-bit registers to store the words.
- Using an addressing mechanism , one of the words to be selected for reading or for writing.
- Two address lines *A*0 and *A*1 select a word for reading or writing via the 2-to-4 decoder.
- The 2-to-4 decoder ensures that at most one register is enabled at a time, and the disabled registers are electrically disconnected through the use of tri-state buffers.
- The Chip Select line in the decoder is not necessary, but will be used later in constructing larger RAMs
- In the smallest RAM chips it is practical to use a single decoder to select one out of 2^*m* words, each of which is *w* bits wide.
- However, this organization is not economical in ordinary RAM chips. Consider that a 64M ´1 chip has 26 address lines (64M = 2^26

A Simplified Representation of the Four-Word by Four-Bit RAM



2-1/2D Organization of a 64-Word by One-Bit RAM



2-1/2D Organization of a 64-Word by One-Bit RAM

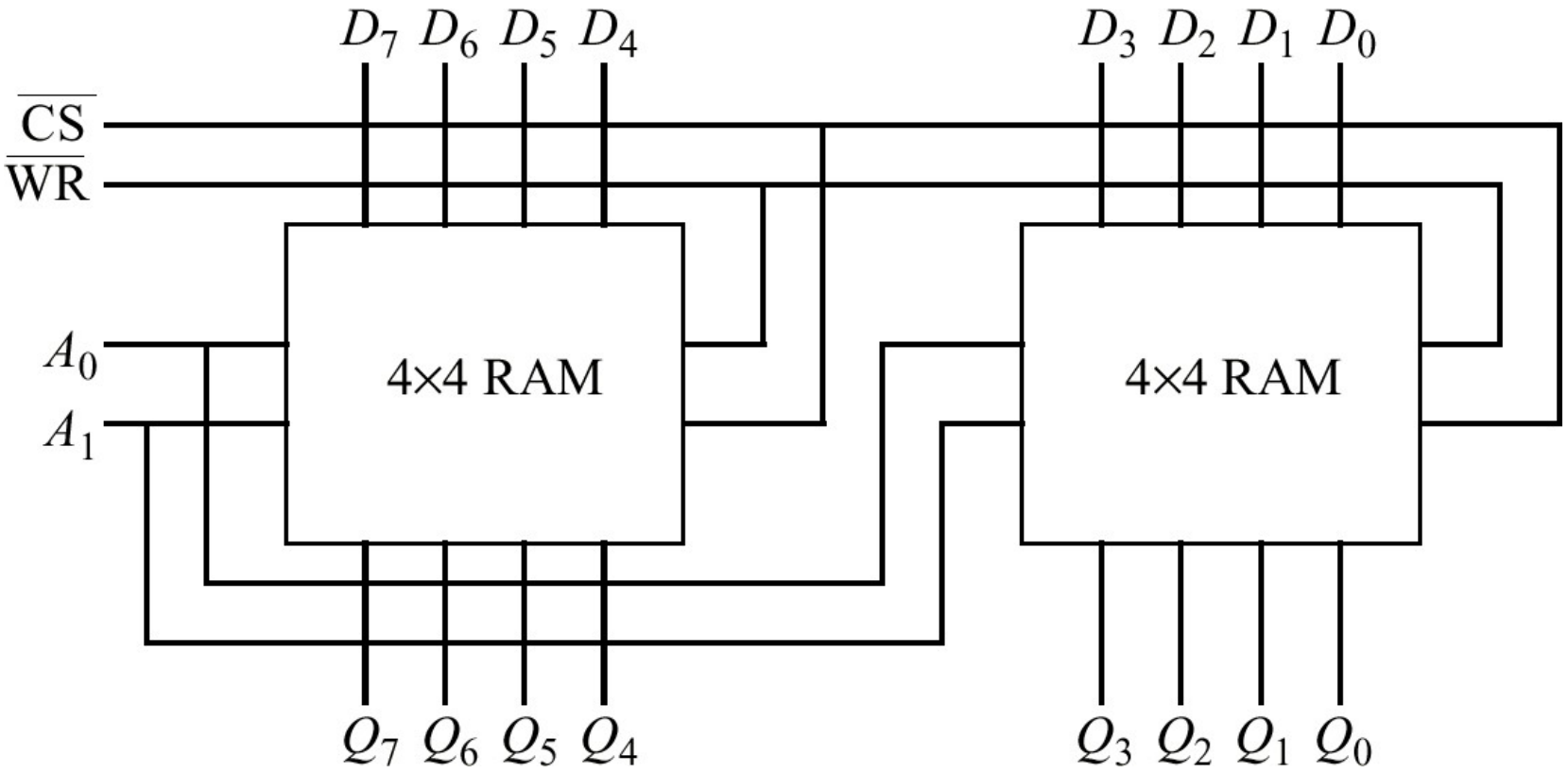
- The 2-1/2D organization is by far the most prevalent organization for RAM ICs. Figure shows a 2^6-word ´1-bit RAM with a 2 1/2D organization.
- The six address lines are evenly split between a row decoder and a column decoder (the column decoder is actually a MUX/DEMUX combination). A single bidirectional data line is used for input and output.
- During a read operation, an entire row is selected and fed into the column MUX, which selects a single bit for output.
- During a write operation, the single bit to be written is distributed by the DEMUX to the target column, while the row decoder selects the proper row to be written.

2-1/2D Organization of a 64-Word by One-Bit RAM

- In practice, to reduce pin count, there are generally only *m*/2 address pins on the chip, and the row and column addresses are time-multiplexed on these *m*/2 address lines.
- First, the *m*/2-bit row address is applied along with a row address strobe, RAS, signal.
- The row address is latched and decoded by the chip. Then the *m*/2-bit column address is applied, along with a column address strobe, CAS. There may be additional pins to control the chip refresh and other memory functions.

- Although DRAMs are very economical, SRAMs offer greater speed. The refresh cycles, error detection circuitry, and the low operating powers of DRAMs create a speed difference that is roughly 1/4 of SRAM speed, but SRAMs also incur a significant cost.

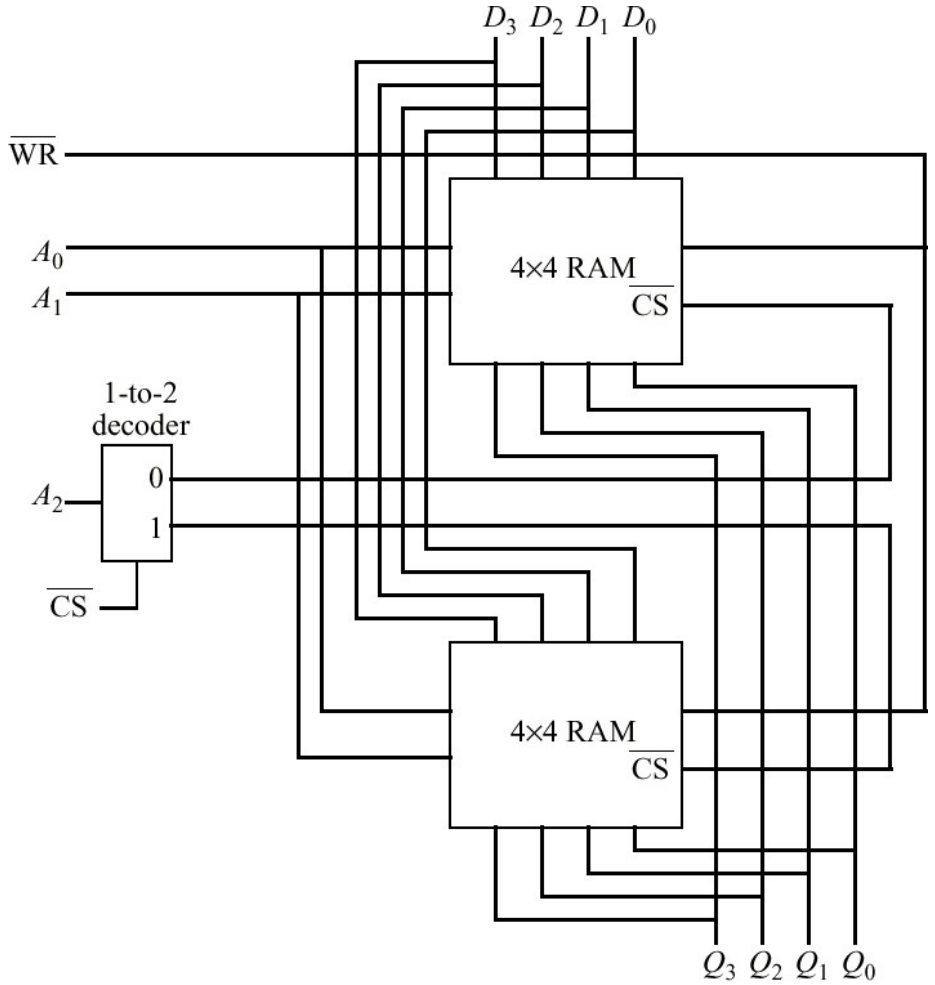
Two Four-Word by Four-Bit RAMs are Used in Creating a Four-Word by Eight-Bit RAM



Two Four-Word by Four-Bit RAMs are Used in Creating a Four-Word by Eight-Bit RAM

- We can construct larger RAM modules from smaller RAM modules.
- Both the word size and the number of words per module can be increased.
- As a simple example, consider using the 4 word x 4-bit RAM chip.
- Consider first the problem of increasing the word width from four bits to eight. We can accomplish this by simply using two chips, tying their CS (chip select) lines together so they are oth selected together, and juxtaposing their data lines, as shown in Figure.

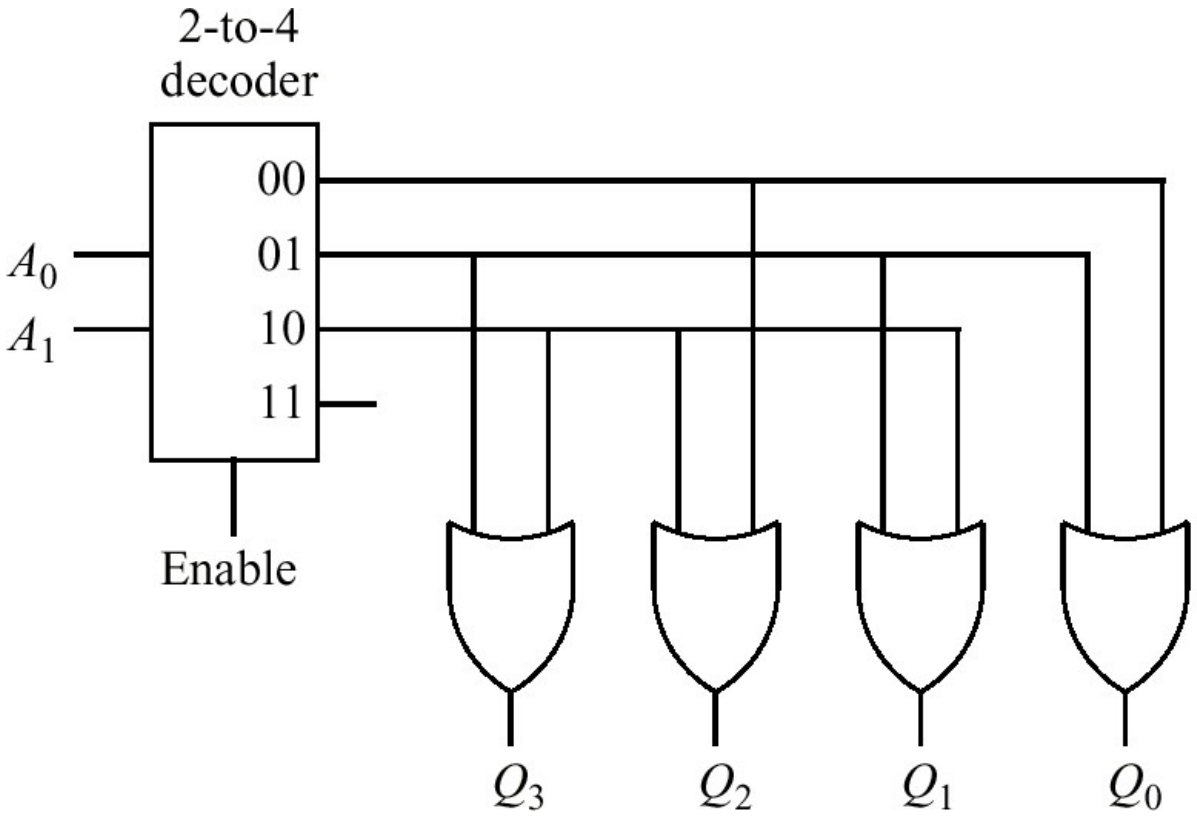
Two Four-Word by Four-Bit RAMs Make up an Eight-Word by Four-Bit RAM



Two Four-Word by Four-Bit RAMs Make up an Eight-Word by Four-Bit RAM

- Consider now the problem of increasing the number of words from four to eight.
- The eight words are distributed over the two four-word RAMs. Address line *A*2 is needed because there are now eight words to be addressed.
- A decoder for *A*2 enables either the upper or lower memory module by using the CS lines, and then the remaining address lines (*A*0 and *A*1) are decoded within the enabled module.
- A combination of these two approaches can be used to scale both the word size and number of words to arbitrary sizes.

A ROM Stores Four Four-Bit Words



Location	Stored word
00	0101
01	1011
10	1110
11	0000

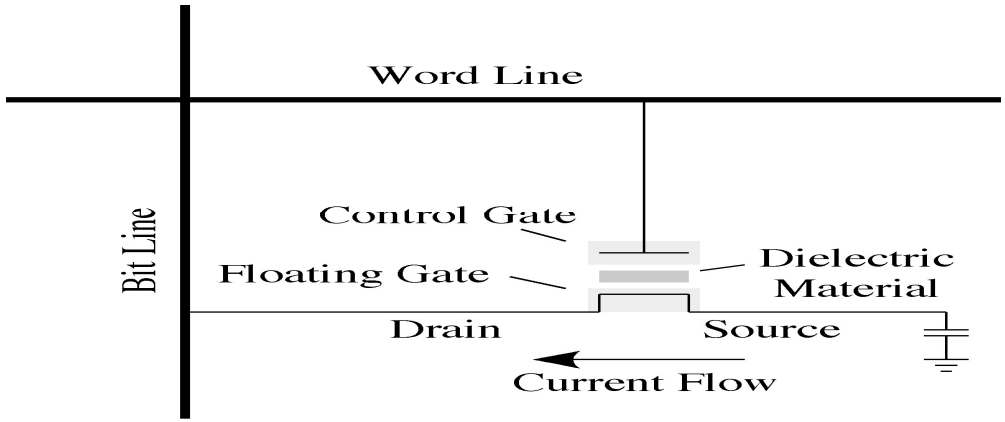
A ROM Stores Four Four-Bit Words

- When a computer program is loaded into the memory, it remains in the memory until it is overwritten or until the power is turned off.
- For some applications, the program never changes, and so it is hardwired into a **Read-only Memory** (ROM).
- ROMs are used to store programs in videogames, calculators, microwave ovens, and automobile fuel injection controllers, among many other applications.
- The ROM is a simple device. All that is needed is a decoder, some output lines, and a few logic gates.
- There is no need for flip-flops or capacitors.
- Figure shows a four-word ROM that stores four four-bit words (0101, 1011, 1110, and 0000). Each address input (00, 01, 10, or 11) corresponds to a different stored word.
- For high-volume applications, ROMs are factory-programmed.

A ROM Stores Four Four-Bit Words

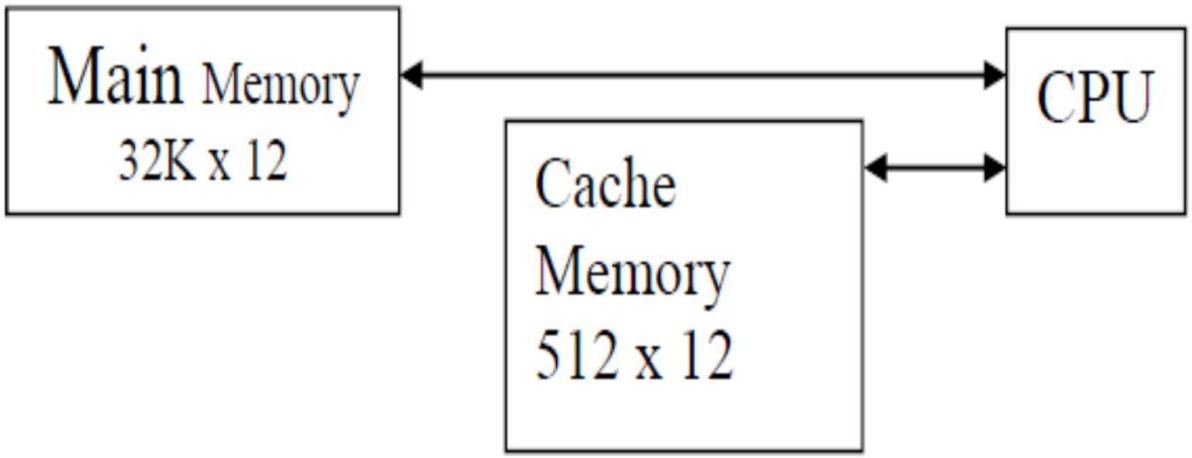
- As an alternative, for low-volume or prototyping applications, programmable ROMs (PROMs) are often used, which allow their contents to be written by a user with a relatively inexpensive device called a **PROM burner**.
- Unfortunately for the early videogame industry, these PROM burners are also capable of reading the contents of a PROM, which can then be duplicated onto another PROM, or worse still, the contents can be deciphered through reverse engineering and then modified and written to a new, contraband game cartridge

Cell Structure for Flash Memory



- Current flows from source to drain when a sufficient negative charge is placed on the dielectric material, preventing current flow through the word line. This is the logical 0 state. When the dielectric material is not charged, current flows between the bit and word lines, which is the logical 1 state.

Cache Memory



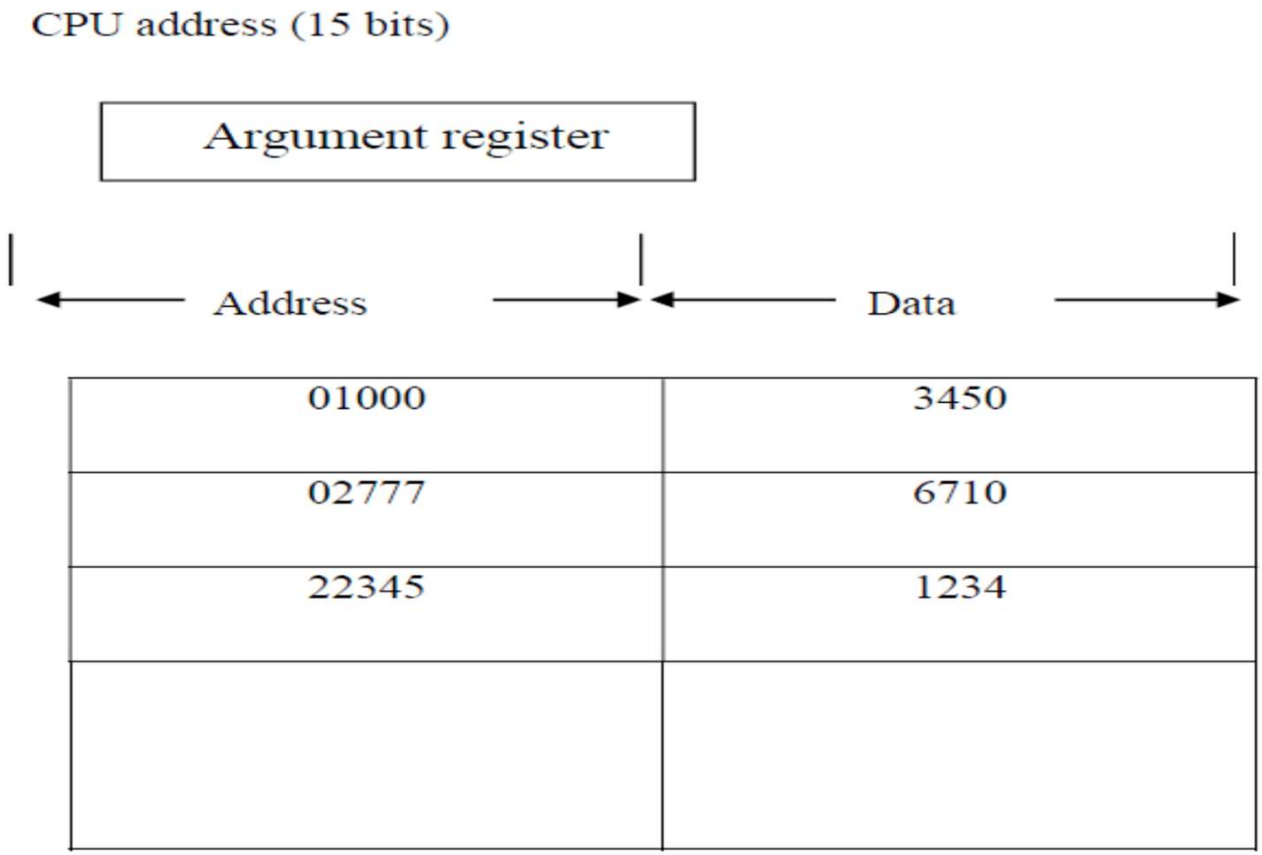
Cache Memory

- If the active portions of the program and the data are placed in a fast small memory, the average memory access time can be reduced; thus reducing the total execution time of the program.
- Such a fast small memory is referred to as a cache memory. It is placed between the CPU and the main memory.
- The cache memory access time is less than the access time of the memory by a factor of 5 to 10.
- The cache is the fastest component in the memory hierarchy and approach the speed of the CPU components.
- The fundamentals idea of cache organization is that by keeping the most frequently accessed instruction and data in the fast cache memory. The average memory access time will approach the access time of the cache

Cache Mapping Methods

- Associative Mapping
- Direct Mapping
- Set associative Mapping

Cache Memory : Associative Mapping



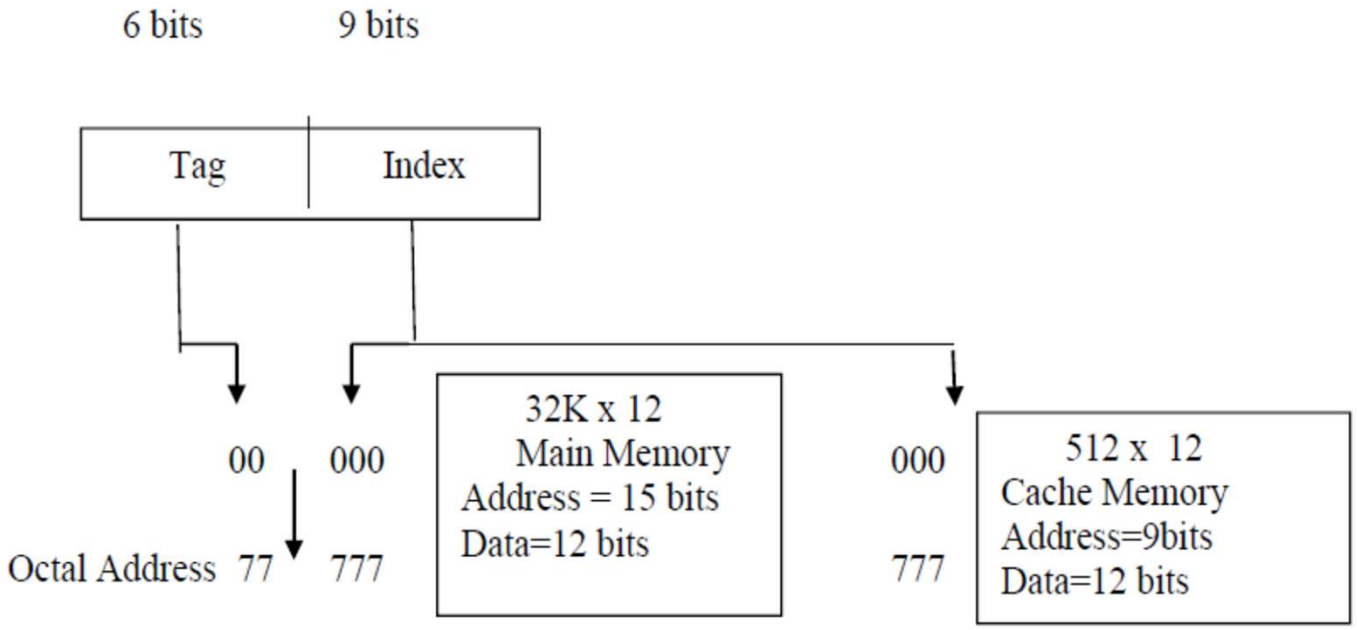
Cache Memory : Associative Mapping

- The fastest and most flexible cache organization uses an associative memory.
- The associative memory stores both the address and the content (data) of the memory word.
- The address value of 15bits is shown as five-digit octal number and its corresponding 12-bit word is shown as a four digit octal number.
- A CPU address of 15 bits is placed in the argument register and the associative memory is searched on a matching address.
- If the address is found the corresponding 12-bit data is head and sent to the CPU.
- If no match occurs, the main memory is accessed for the word.
- If the cache is full an Address - Data pair must be displaced to make room a pair that is needed and presently in the cache.

Cache Memory : Associative Mapping

- The decision as to what pair is replaced is defined from the replacement algorithm
- A simple procedure is to replace the cells of the cache in round- robin order whenever a new word is requested from main memory. This constitutes a FIFO (first- in- first out) replacement policy.

Cache Memory : Direct Mapping



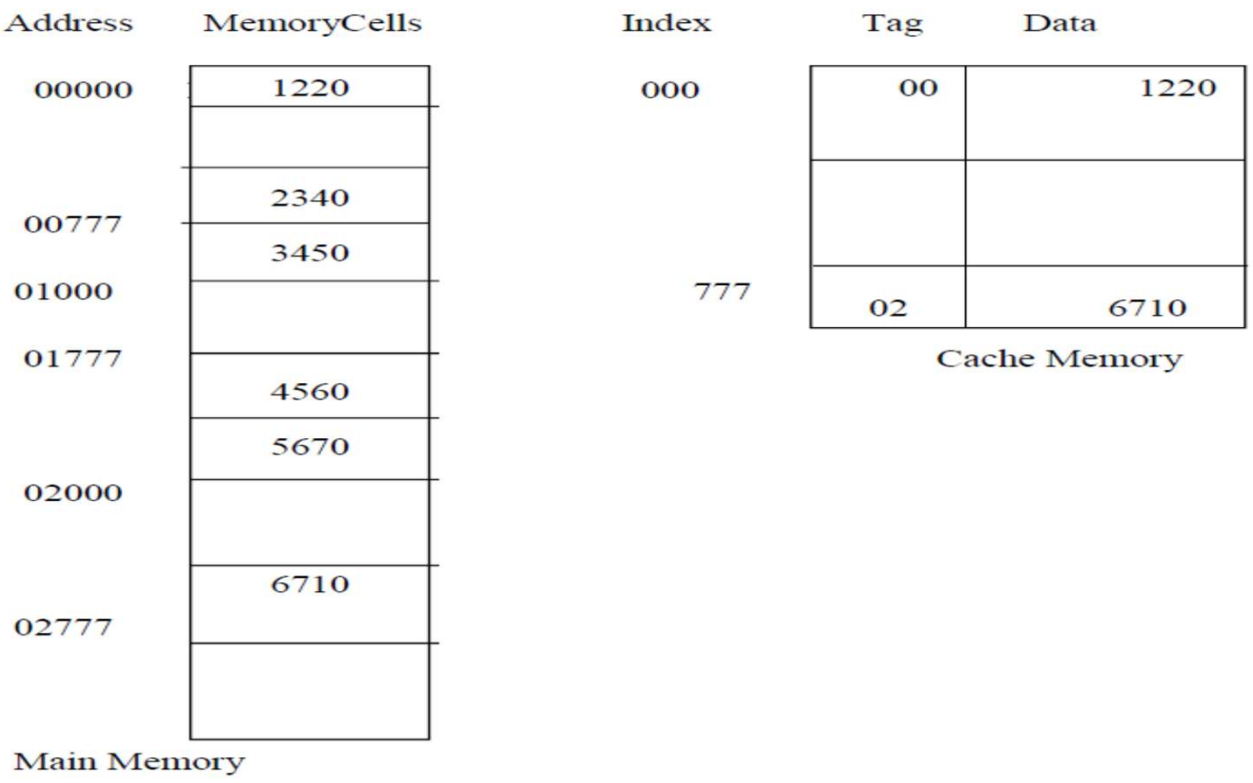
Cache Memory : Direct Mapping

- The CPU address of 15 bits is divided into two fields the 9- least significant bits constitute the index field and the remaining 6 bits from the tag field.
- The main memory needs an address that includes both the tag and the index bits the no of bits in the index field is equal to the no of address bits required to access the cache memory.
- In the general case there are 2K words in the cache memory and 2n words in main memory the n –bit memory address is divided into two field s k- bits for the index field and(n-k) bits for the tag field

Cache Memory : Direct Mapping

- Each word in the cache consists of data word and it associated tag when a new word is first brought in to the cache the tag bits are stored along side the data bits.
- when the cpu generates the memory request the index field is used for the address to access the cache , the tag field of the cpu address is compared with the tag in the word read from the cache.
- If the two tags match there is a hit and the desired data word is in cache if there is no match there is amiss and the required word is read from the main memory it is then stored in the cache together with the new tag.

Cache Memory : Direct Mapping



Cache Memory : Direct Mapping

- Consider the opn—of the direct mapping organization the word at address zero is presently stored in the cache (index= 000,tag=00, data= 1220).
- suppose the cpu wants to access the word at address 02000 the index address is 000 , so it is used to access the cache the two tags are then compared .the cache tag is 00 and the address tag is 02 which do not produce a match
- The main memory is accessed and the data word 5670 is transferred to the cpu the cache word at index address 000 is then replaced with a tag of 02 and data of 5670.
- The disadvantages of direct mapping is that the hit ratio can drop considerably if two or more words whose address have the same index but diff tags are accused repeatedly

Cache Memory : Set Associative Mapping

Index	Tag 1	Data 1	Tag 2	Data 2
000	01	3450	02	5670
777	02	6710	00	2340

Cache Memory : Set Associative Mapping

- Set associative mapping is an improvement over the direct mapping.
- Each word of cache can store two or more words of memory under the same index address.
- Each data word is stored together with its tag and the no of tag- data items in one word of cache is said to form a set.
- Each index address refers to two data words and their associated tags each tag req 6 bits and each data word has 12 bits. So the word length is 2 x (6+12)= 36 bits.
- An index address of 9 bits can accommodate 512 words .Thus the size of cache memory is 512*36
- It can accommodate 1024 words of main memory

Cache Memory : Set Associative Mapping

- The words stored at address 01000 and 02000 of main memory are stored in cache memory at index address 000.
- Similarly the words at address 02777 and 00777 are stored in cache at index address 777.
- when the cpu generates a memory request, the index value of the address is used to access the cache.
- The tag field of the cpu address is then compared with both tags in the cache to check if a match occurs.
- The hit ratio will improves as the set size increases because more words with the same index but diff tags can reside in the cache

Replacement algorithms

- Random replacement.
  - with the random replacement policy the control chooses one, tags data item for the replacement at random
- First- in - First – out (FIFO)
  - The FIFO procedure selects fn replacement the item that has been in the set the longest
- Least recently used (LRU)
  - The LRU algorithm selects for replacement the item that has been least recently used by the CPU.

Performance

- Hit rate = $\frac{\text{no. of hit accesses}}{\text{Total no. of memory accesses}}$

$= \frac{\text{No. of hits}}{\text{No. hits+ No. of Misses}}$

Performance

- Average memory Access time = Hit time of cache + Miss Rate * Miss Penalty

  = Hit time of Cache + Miss Rate * Main memory access time

  = Hit time of Cache + (1-Hit rate)* Main memory access time

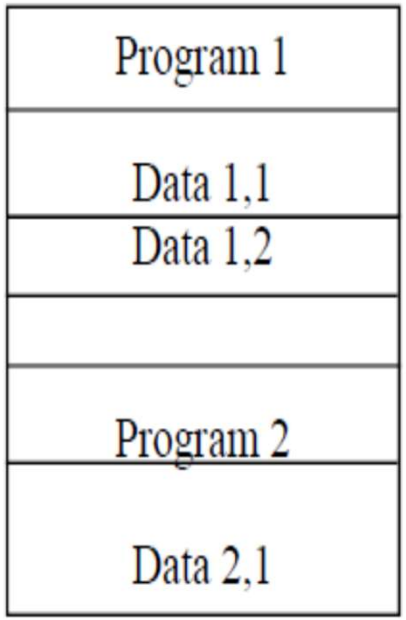
Virtual Memory

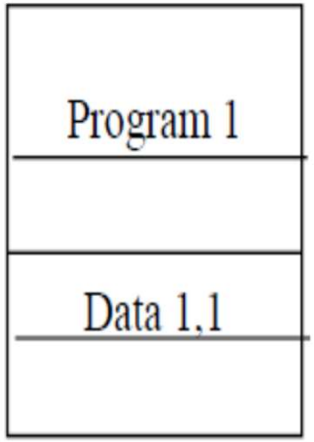
- Virtual memory is a concept used in some large computer systems that permit the user to construct programs as though a large memory space were available equal to the totality of the auxiliary memory.
- Each address that is referenced by the CPU goes through an address mapping from the so-called virtual address to a physical address in main memory.
- A virtual memory system provides a mechanism for translating program-generated addresses into correct main memory locations.

Virtual Memory

- An address used by a program will be called a virtual address; and a set of such addresses the address space.
- An address in main memory is called the location or physical address. The set of such locations is called memory space.

Virtual Memory





Address space

$N = 1024, K = 2^{20}$

Memory space $M = 32K = 2^{15}$

Virtual Memory

- In a multiprogramming computer system, program and data are transferred to and from auxiliary memory and the main memory based on the demands imposed by the CPU.
- Suppose that program 1 is currently being executed in the CPU. Program 1 and a portion of the associated data are moved from the auxiliary memory into main memory.
- Portions of program and the data need not be contiguous location in memory.

Virtual Memory

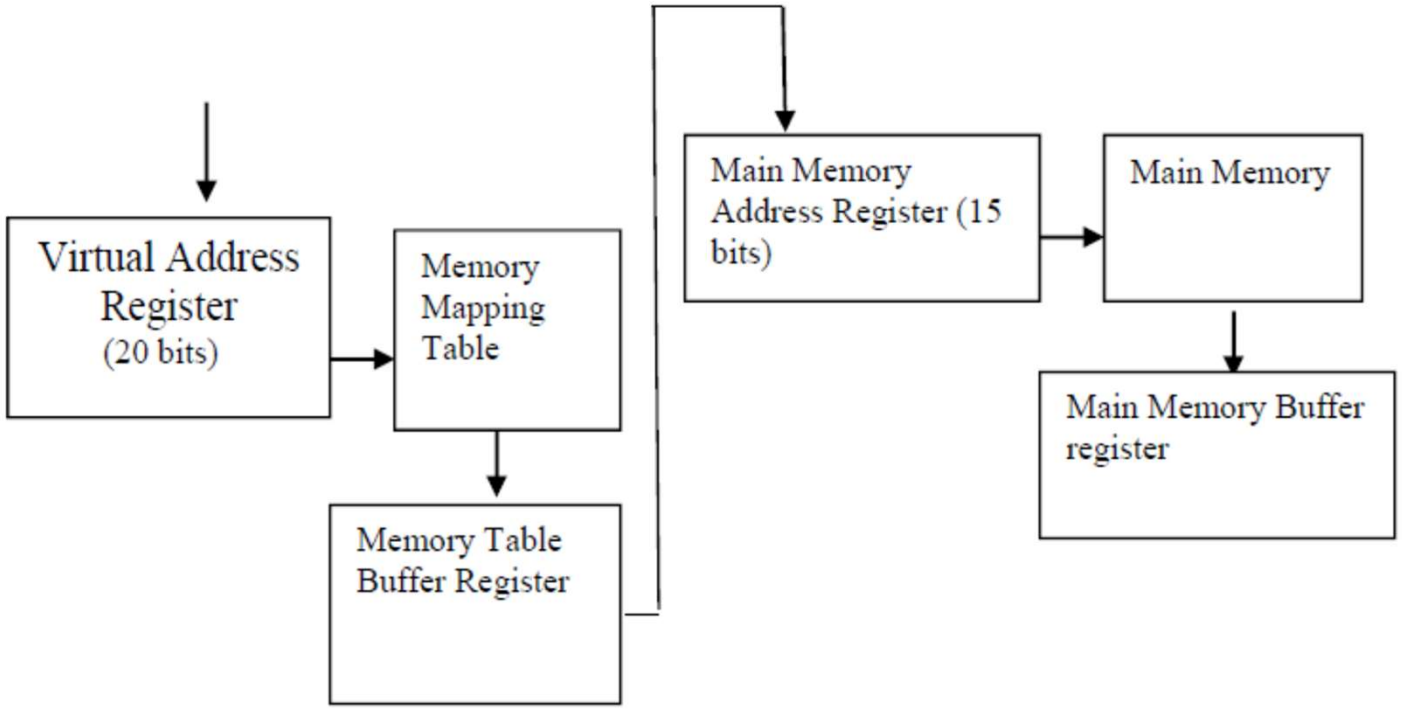


Fig: - Memory Table For Mapping a Virtual Address

Virtual Memory

- In our example the address field of an instruction code will consists of 20bits but physical memory addresses must be specified with only 15bits.
- The CPU will reference instruction at this address must be taken from physical memory because access for individual words will be prohibitively long.
- A table is needed to map a virtual address of 20 bits to physical address of 15bits.
- In the first case an additional memory unit is required as well as one extra memory access time. In the second case the table takes spaces from main memory and two accesses to memory are required with program running at half speed. The third alternative is to use an associative memory

Virtual Memory

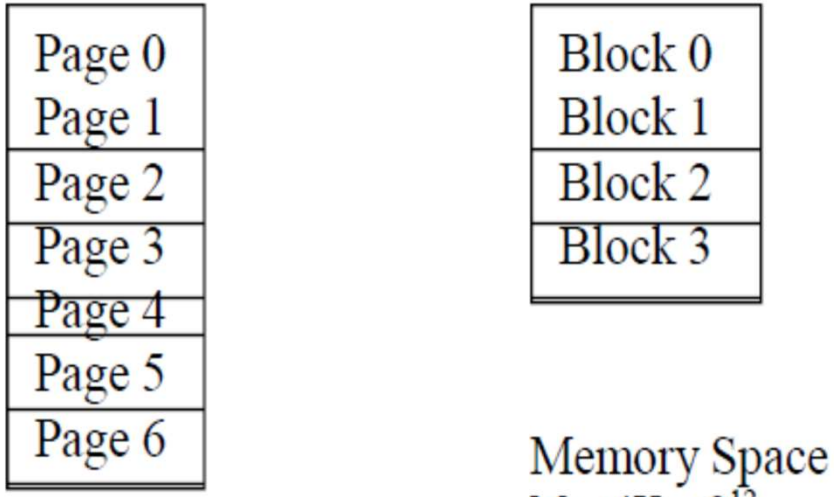
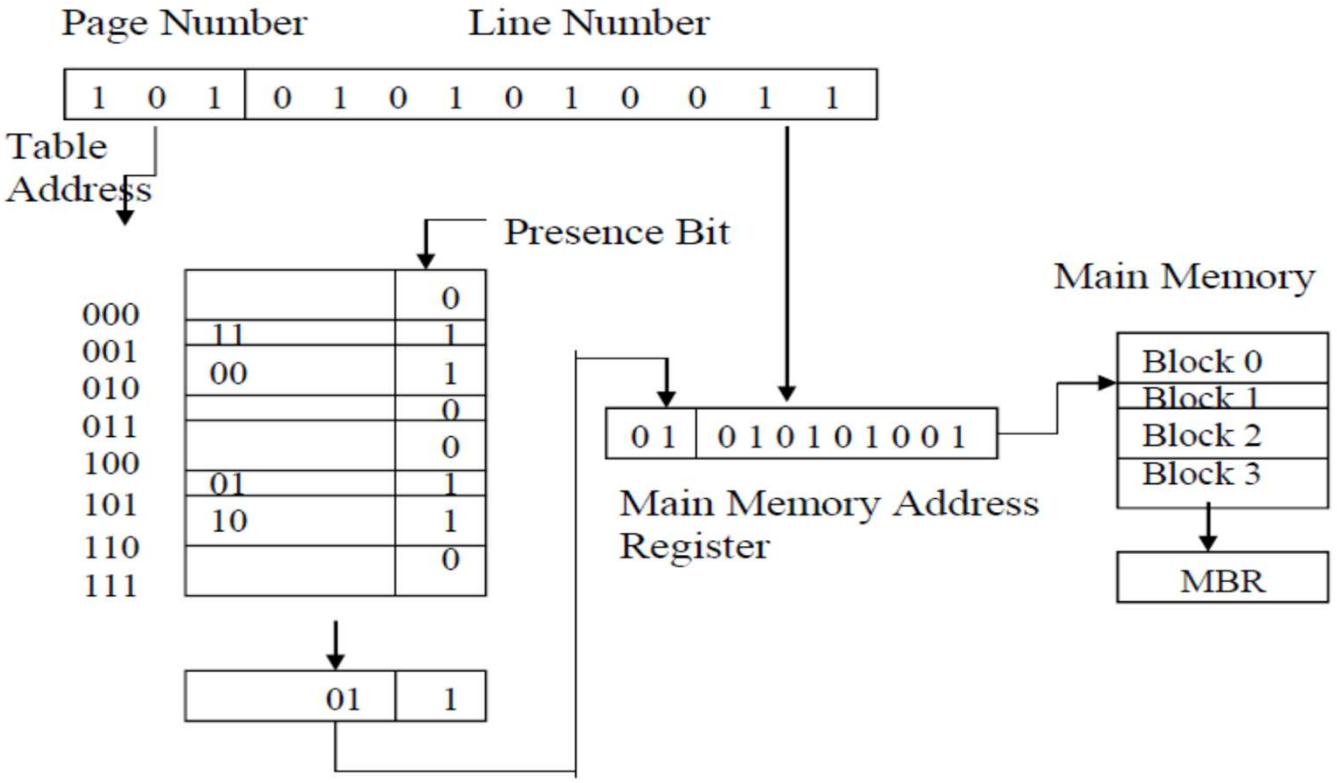


Fig: - Address space and memory space split into groups of 1K words

Virtual Memory

- Consider a computer with an address space of 8K and a memory space of 4K.
- If we split each into groups of 1K words we obtain eight pages and four blocks.
- At any given time, up to four pages of address space may reside in main memory in any one of the four blocks.

Virtual Memory



Memory Page Table

Virtual Memory

- The memory page table consists of eight words one for each page.
- The address in the page table denotes a page number and the content of the word gives the block number where that page is stored in the main memory.
- The pages 1,2,5 and 6 are available in the main memory in blocks 3,0,1 and 2 respectively.
- A presence bit in each location indicates whether the page has been transferred from auxiliary memory into main memory.
- A 0 in the presence bit indicates that this page is not available in main memory.
- The CPU references a word in memory with a virtual address of 13 bits.
- The three high-order bits of virtual address specify a page number and also an address for memory page table.

Virtual Memory

- The content of the memory page table at the page number address is read out into the memory table buffer register.
- If the presence bit is 1, the block number thus read is transferred to the two high order bits of the main memory address register.
- The line number from the virtual address register is transferred into the 10-low order bits of the memory address register.
- A read signal to main memory transfers the content of the word to the main buffer register ready to be used by the CPU.
- If the presence bit in the word read from the page table is 0 it signifies that the content of the word referenced by the virtual address does not reside in the main memory.
- A call to the operating system is then generated to fetch the required page from the auxiliary memory and place it into main memory before resuming computation.

Modes of Data Transfer

- Programmed IO
- Interrupt Driven IO
- DMA Transfer

Programmed IO

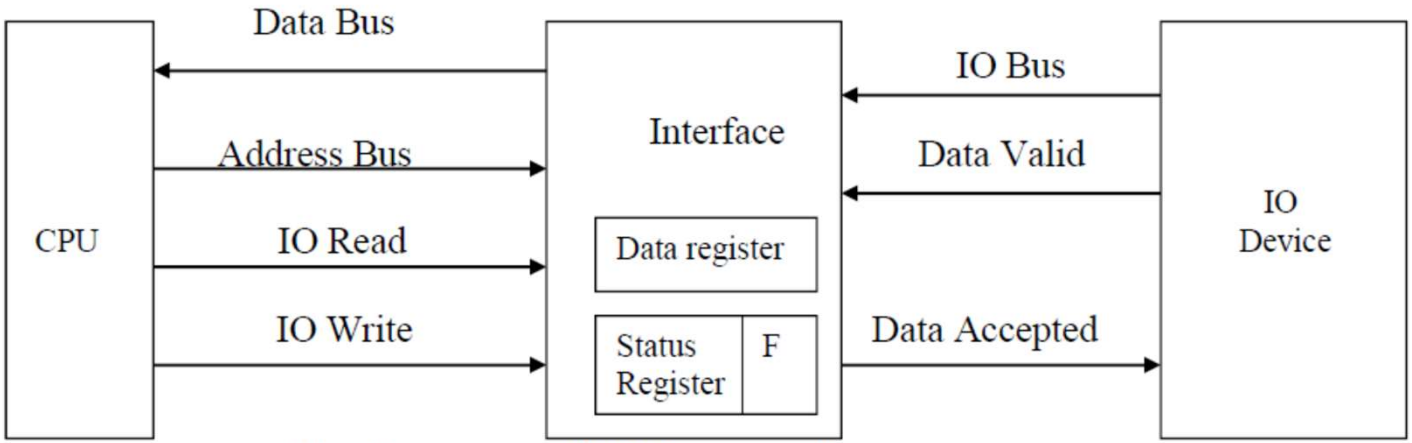


Fig: Programmed IO data transfer

Programmed IO

- Each data item transfer is initiated by an instruction in the program. Usually the transfer is to and from the CPU register and the peripheral.
- Transferring data under program control requires the constant monitoring of the peripheral by the CPU.
- Once the data transfer is initiated the CPU is required to monitor the interface to see that when the transfer can again be made.
- In the programmed IO method the CPU stays in the program loop until the IO unit indicates that it is ready for data transfer .
- This is a time consuming process. Since it keeps the processor needlessly.
- In the programmed IO method the IO device does not have direct access to memory.
- A transfer from an IO device to memory requires the execution of several instructions by the CPU , including a input instruction to transfer the data from the device to the CPU, and store instruction to transfer the data from CPU to memory.
- Other instructions are needed to verify that the data are available from the device and to count the number of words transferred .

Programmed IO

- When the byte of data is available the device places it in the IO bus and enables its data valid line.
- The interface accepts the byte in to its data register and enables the data accepted.
- The interface sets a bit in the status register i.e F (or) flag bit. The device can now disable the data valid line but will not transfer until the data accepted line is disabled by the interface.
- A program is written to check the flag in the status register to determine if a byte has been placed in the data register by the IO device. This is done by reading the status register in to the CPU register and checking the value of the flag bit .
- If the flag is equal to 1 the CPU reads the data from the data register .The flag bit will be reset to 0 by either the CPU or the interface depending on how the interface circuits are designed.

Programmed IO

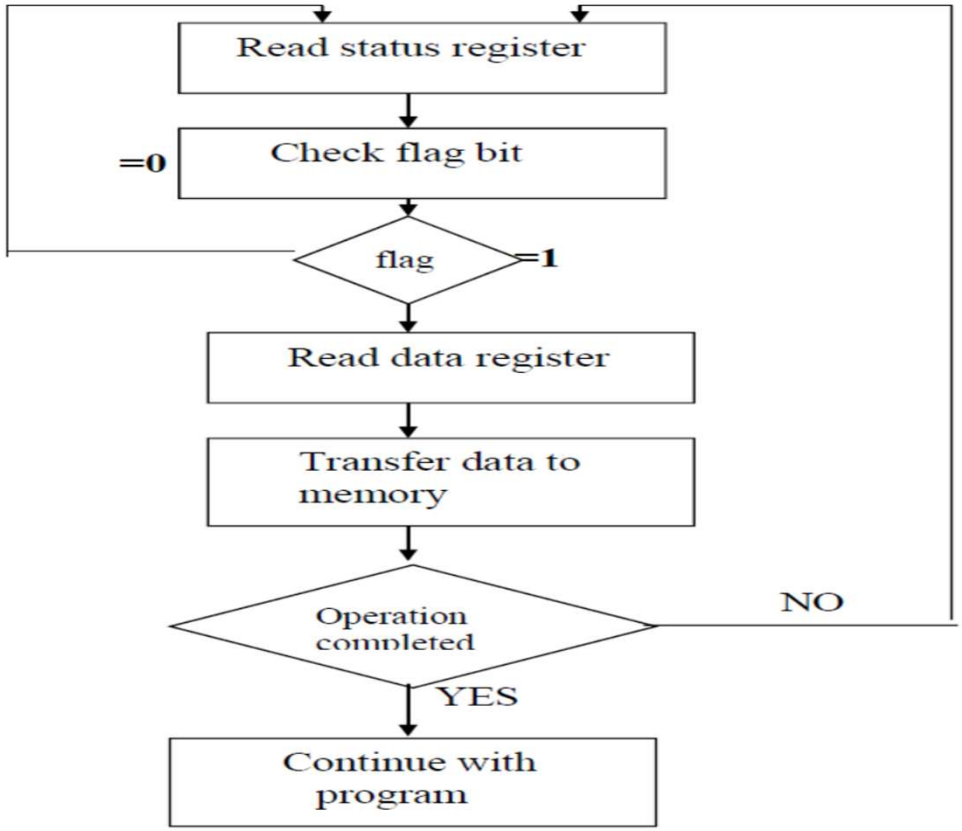


Fig: Programmed IO method of data transfer

Programmed IO

- The transfer of each byte requires three instructions:
  - Read the status register.
  - Check the status of the flag bit and branch to step1. if not set or or to step 3 if set .
  - Read the data register. The programmed IO method is particularly useful in small low speed computer

Interrupt driven data transfer

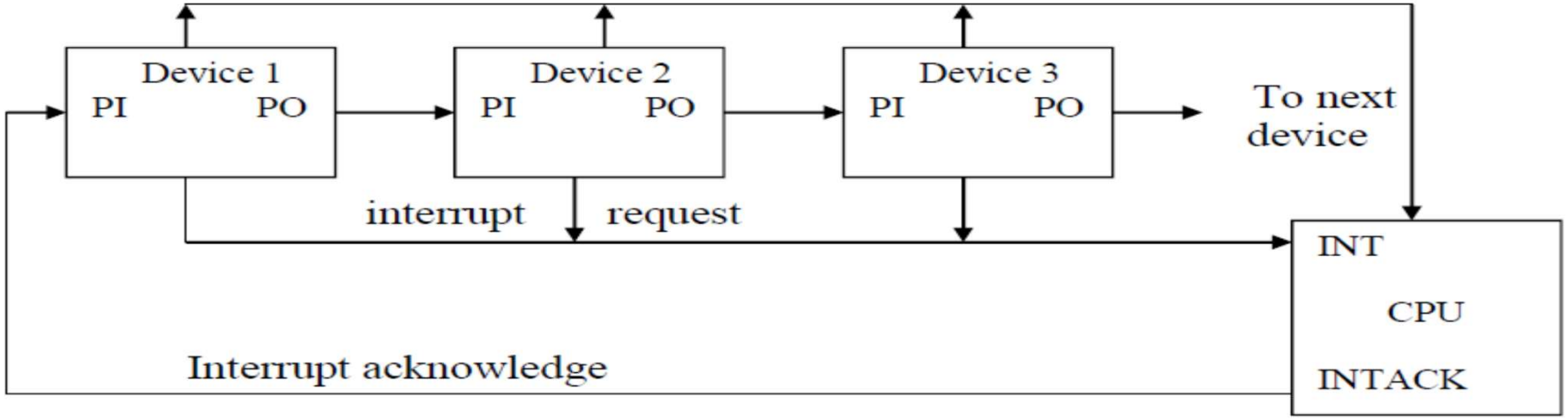


Fig: Daisy chain priority interrupt

Interrupt driven data transfer

- The daisy chaining method of establishing the priority consists of connection of all the devices that request an interrupt.
- The device with the highest priority will be positioned in the first position followed by the low priority devices up to lowest priority device which is positioned last in the chain.
- The interrupt request line is common to all the devices and forms an wired logic connection.
- If any device has its interrupt signal in the low level state , the interrupt goes to the low level state and enables the interrupt input in the CPU.
- When no interrupts are pending the interrupt line stays in the high level state and no interrupts are recognized by the CPU. This is equivalent to the –ve logic OR operation.

Interrupt driven data transfer

- The CPU responds to an interrupt request by enabling the interrupt acknowledgement line.
- This signal is received by the device 1 at its PI (priority In) input .
- The acknowledgement line passes to the next device through PO (priority Out) output only if device 1 is not requesting an interrupt.
- If device 1 has pending interrupt it blocks the acknowledgement signal from the next device by placing the 0 in the PO output.
- It then proceeds to insert its own interrupt vector address (VAD) in to the data bus for the CPU to use during the interrupt cycle

Interrupt driven data transfer

- A device with a 0 in its PI input generates a 0 in its PO output to inform the next lower priority device that the acknowledge signal has been blocked.
- A device that is requesting an interrupt and has 1 in its PI input will intercept acknowledge signal by placing the 0 in its PO output.
- If the device does not have any pending interrupts it transmits the acknowledgement signal to the next device by placing the 1 in the PO output.
- Thus the device with PI=1 and PO=0 is the one with the highest priority that is requesting an interrupt and this device places an vectored address on the data bus.
- The daisy chain arrangement gives the highest priority to the device that receives the acknowledgement signal from the CPU.

DMA mode of data transfer

- The transfer of data between the fast storage device such as magnetic disk and memory is of ten limited by the speed of the CPU.
- Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of the transfer.
- This transfer technique is called the direct access memory (DMA) During DMA transfer the CPU is idle and has no control of memory buses.
- A DMA controller takes control over the buses to manage the transfer directly between the IO device and the memory

DMA mode of data transfer

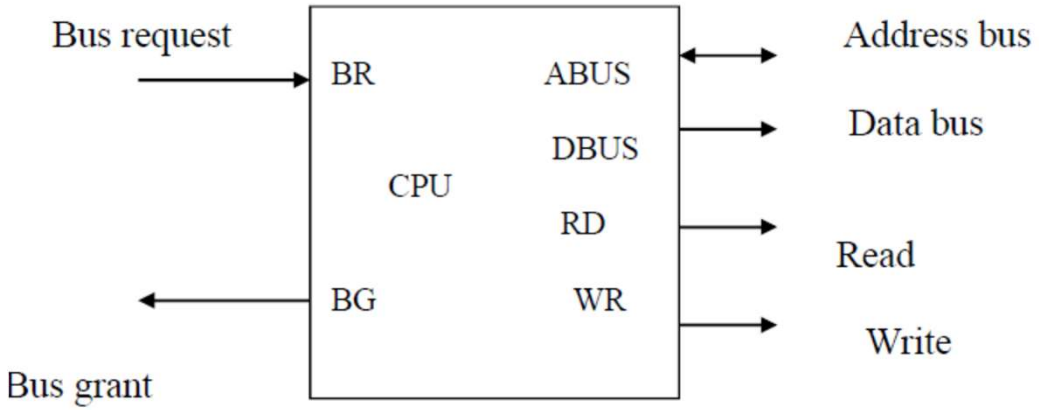


FIG: CPU bus signals for DMA transfer.

DMA mode of data transfer

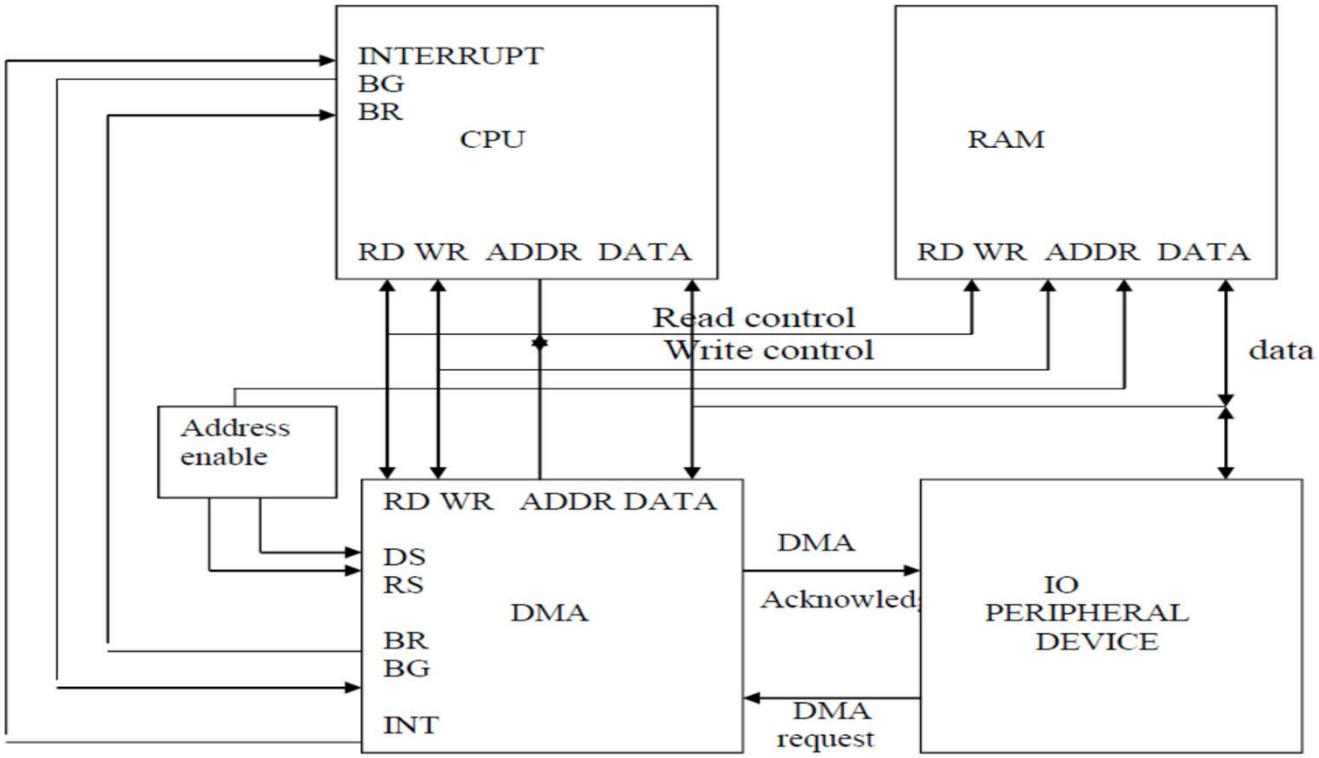
- The bus request (BR) is used by the DMAC (DMA controller) to request the CPU to relinquish control of the buses.
- When this input is active the CPU terminates the execution of the current instruction and places the address bus , data bus and R/W lines in to high impedance state.
- The high impedance state behaves like an open circuit which means that the output is disconnected and does not have logic significance.
- The CPU activates the bus grant (BG) output to inform the external DMA that the buses are in the high impedance state.
- The DMA that originated the bus request can now take control of the buses to conduct memory transfer with out processor intervention.

DMA mode of data transfer

- DMA Transfer can be :

- Burst mode➔ A block sequence consisting of number of memory words is transferred in continuous burst while a DMA controller is a master of the memory buses. This mode of transfer is needed for fast devices such as magnetic disks , where the data transfer can not be stopped or slowed down until an entire block is transferred.
- Cycle stealing➔ Allows the DMA controller to transfer one data word at a time , after which it must return control of the buses to the CPU. The CPU merely delays its operation for one memory cycle to allow the direct memory IO transfer to steal one memory cycle.

DMA mode of data transfer



DMA Transfer

DMA mode of data transfer

- When the peripheral device sends a DMA request line the DMAC activates the BR line informing the CPU to relinquish the buses.
- The CPU responds with its BG line informing the DMA that its buses are disabled.
- The DMA then puts the current value of the address register in to the address bus initiates the RD or WR signal and sends a DMA ACK to the peripheral device.
- The RD or WR lines in the DMAC are bi-directional. The direction of transfer depends on the status of the BG signal.
- If BG=0 the RD and WR are input lines allowing the CPU to communicate with the internal DMA registers.

DMA mode of data transfer

- If BG=1 the RD and WR are the output lines from the DMC to the RAM to specify the read or write operation for the data.
- When the peripheral device receives the DMA ACK it puts the word in the data bus (write) or receives the word from the data bus (read).
- Thus the DMA controls the R/W operation and supplies the addresses for the memory.
- The peripheral unit can then be communicate with the memory thru the data bus for direct transfer between the two units while the CPU is momentarily disabled.

DMA mode of data transfer

- For each word that is transferred the DMA increments its address register and decrements its word count register.
- If the word count register reaches to zero the DMA stops any further transfers and removes its bus request .
- It also informs the CPU of the termination by means of the interrupt.
- When the CPU responds the interrupt it reads the contents of the word count register .
- The zero value of the register indicates that all words were transferred successfully

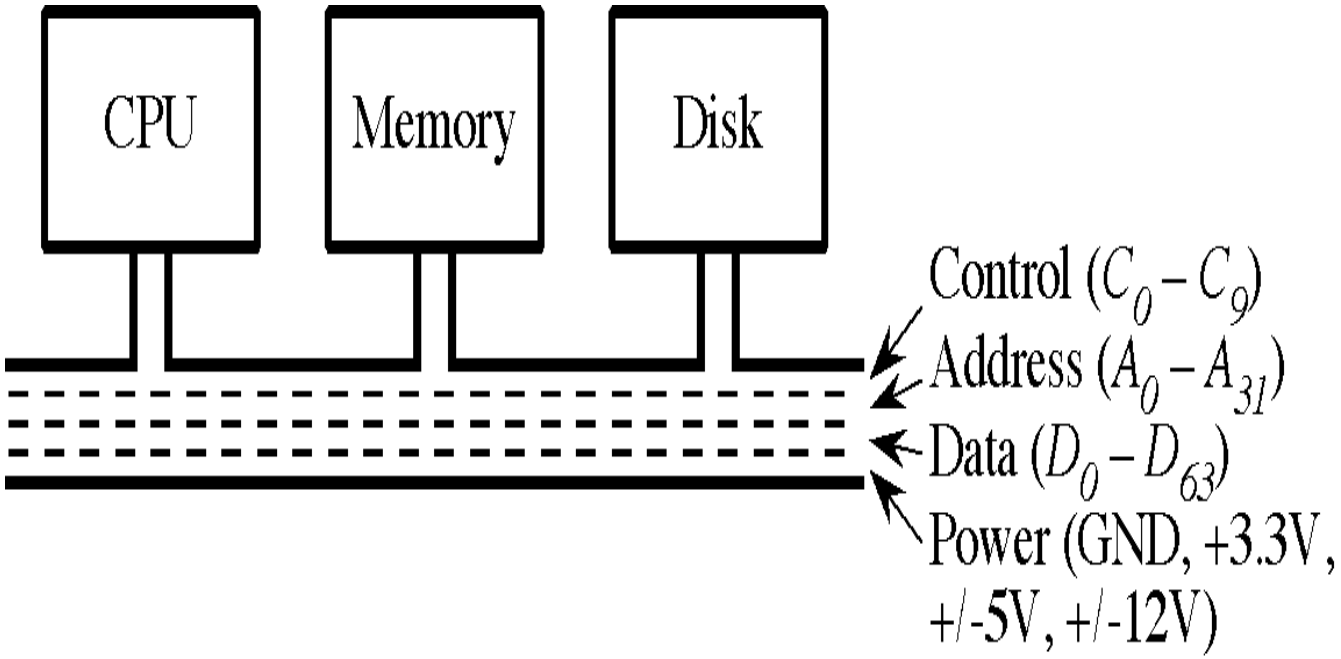
Simple Bus Architecture

- A **bus is a common pathway that connects a number of devices. An example of a** bus can be found on the **motherboard (the main circuit board that contains the** central processing unit) of a personal computer

BUS STRUCTURE, PROTOCOL, AND CONTROL

- A bus consists of the physical parts, like connectors and wires, and a bus protocol.
- The wires can be partitioned into separate groups for control, address, data, and power.
- A single bus may have a few different power lines, has lines for ground (GND) at 0 V, and positive and negative voltages at +5 V, and –15 V, respectively.
- The devices share a common set of wires, and only one device may send data at any one time.
- All devices simultaneously listen, but normally only one device receives. Only one device can be a bus master, and the remaining devices are then considered to be slaves. The master controls the bus, and can be either a sender or a receiver

BUS STRUCTURE, PROTOCOL, AND CONTROL



BUS STRUCTURE, PROTOCOL, AND CONTROL

- An advantage of using a bus is to eliminate the need for connecting every device with every other device, which avoids the wiring complexity that would quickly dominate the cost of such a system.
- Disadvantages of using a bus include the slowdown introduced by the master/slave configuration, the time involved in implementing a protocol (see below), and the lack of scalability to large sizes due to fan-out and timing constraints

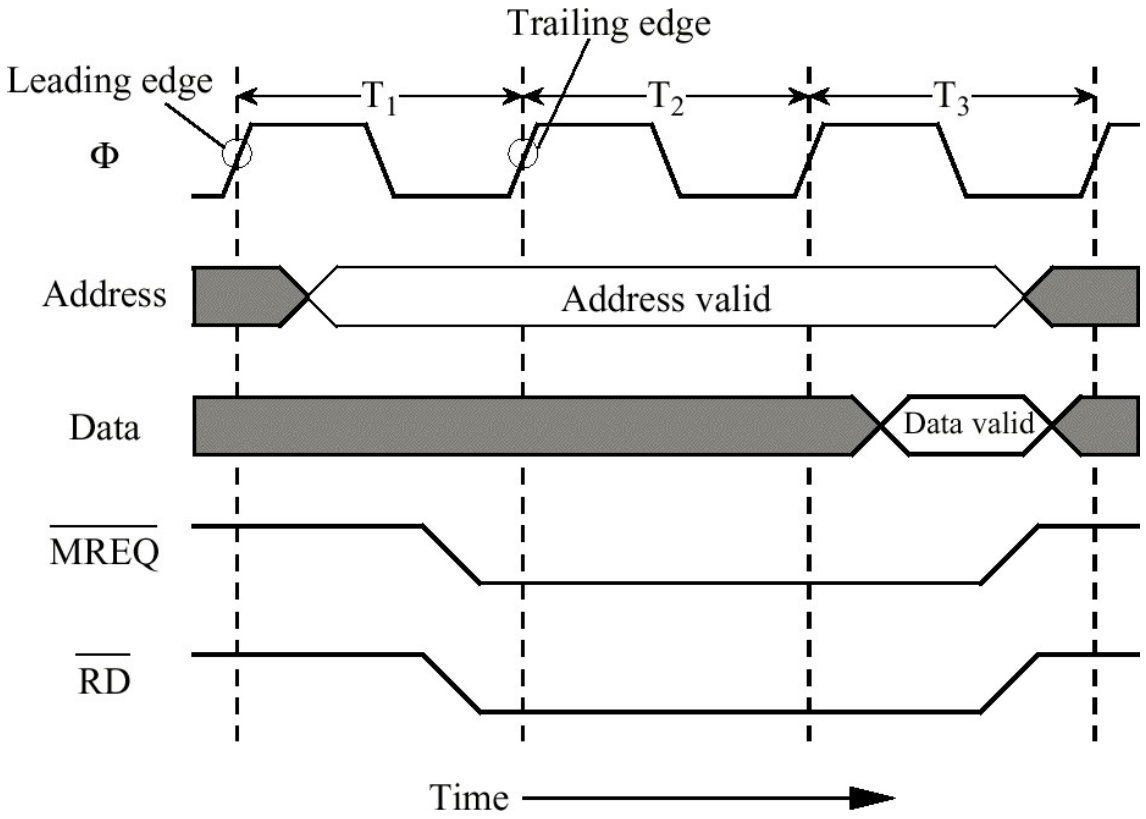
- A bus can be classified as one of two types: **synchronous or asynchronous**

The Synchronous Bus

- a synchronous read of a word of memory by a CPU.
- At some point early in time interval *T1, while the* clock is high, the CPU places the address of the location it wants to read onto the address lines of the bus.
- At some later time during *T1, after the voltages on the* address lines have become stable, or “settled,” the *MREQ* and RD lines are asserted by the CPU.
- *MREQ* informs the memory that it is selected for the transfer (as opposed to another device, like a disk).
- The RD line informs the selected device to perform a read operation.
- The overbars on *MREQ* and RD indicate that a 0 must be placed on these lines in order to assert them.

The Synchronous Bus

Timing diagram for a synchronous memory read



The Synchronous Bus

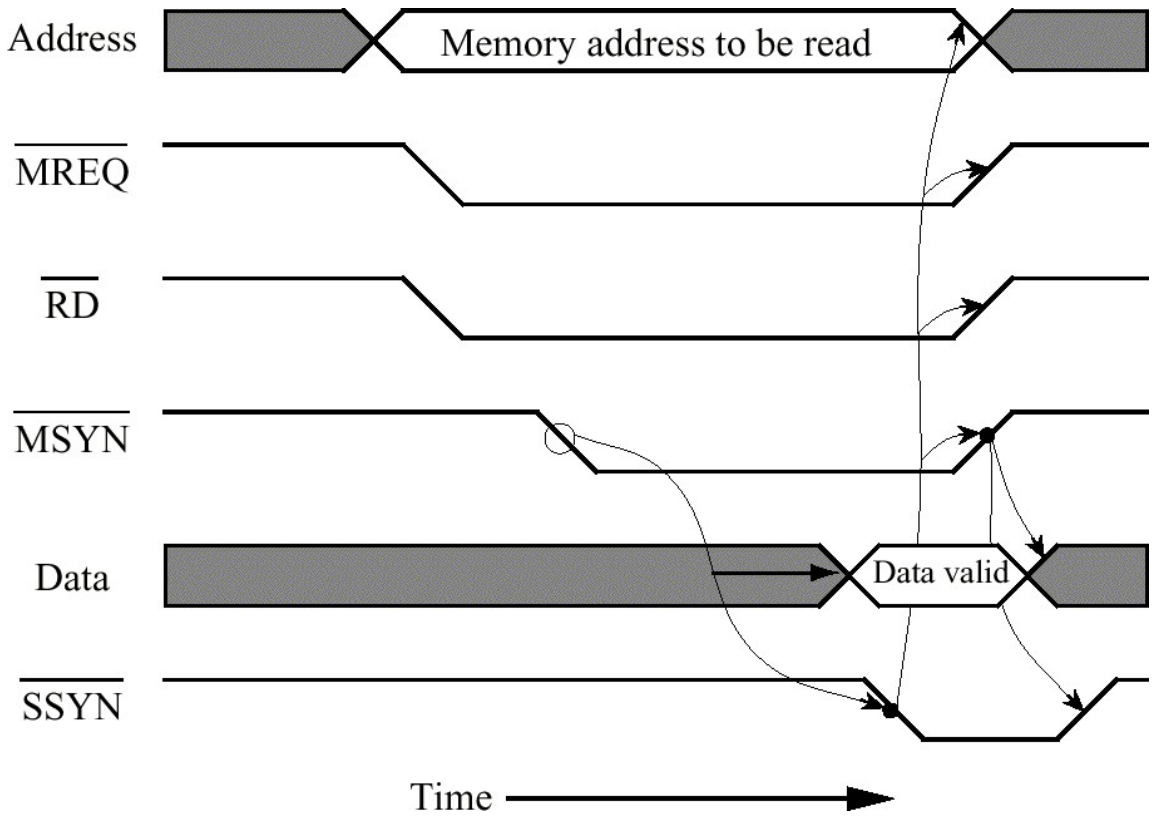
As an example of how communication takes place over a synchronous bus, consider the timing diagram shown in Figure 8-4 which is for a synchronous read of a word of memory by a CPU. At some point early in time interval T_1, while the clock is high, the CPU places the address of the location it wants to read onto the address lines of the bus. At some later time during T_1, after the voltages on the address lines have become stable, or "settled," the $\overline{MREQ}$ and $\overline{RD}$ lines are asserted by the CPU. $\overline{MREQ}$ informs the memory that it is selected for the transfer (as opposed to another device, like a disk). The $\overline{RD}$ line informs the selected device to perform a read operation. The overbars on $\overline{MREQ}$ and $\overline{RD}$ indicate that a 0 must be placed on these lines in order to assert them.

The Asynchronous Bus

If we replace the memory on a synchronous bus with a faster memory, then the memory access time will not improve because the bus clock is unchanged. If we increase the speed of the bus clock to match the faster speed of the memory, then slower devices that use the bus clock may not work properly.

An asynchronous bus solves this problem, but is more complex, because there is no bus clock. A master on an asynchronous bus puts everything that it needs on the bus (address, data, control), and then asserts $\overline{MSYN}$ (master synchronization). The slave then performs its job as quickly as it can, and then asserts $\overline{SSYN}$ (slave synchronization) when it is finished. The master then de-asserts $\overline{MSYN}$, which signals the slave to de-assert $\overline{SSYN}$.

The Asynchronous Bus



The Asynchronous Bus

In order for a CPU to read a

word from memory, it places an address on the bus, followed by asserting $\overline{MREQ}$ and $\overline{RD}$. After these lines settle, the CPU asserts $\overline{MSYN}$. This event triggers the memory to perform a read operation, which results in $\overline{SSYN}$ eventually being asserted by the memory. This is indicated by the **cause-and-effect** arrow between $\overline{MSYN}$ and $\overline{SSYN}$ shown in Figure 8-5. This method of synchronization is referred to as a "full handshake." In this particular implementation of a full handshake, asserting $\overline{MSYN}$ initiates the transfer, followed by the slave asserting $\overline{SSYN}$, followed by the CPU de-asserting $\overline{MSYN}$, followed by the memory de-asserting $\overline{SSYN}$. Notice the absence of a bus clock signal.

Asynchronous busses can be more difficult to debug than synchronous busses when there is a problem, and interfaces for asynchronous busses can be more difficult to make. For these reasons, synchronous busses are very common, particularly in personal computers.

BUS ARBITRATION—MASTERS AND SLAVES

- Suppose now that more than one device wants to be a bus master at the same time. How is a decision made as to who will be the bus master? This is the **bus arbitration problem, and there are two basic schemes: centralized and decentralized** (distributed). Figure 8-6 illustrates four organizati schemes. In Figure 8-6a, a centralized arbitration scheme is used.
- Devices 0 through *n are all attached to the same bus (not shown), and they also share a **bus*** **request line that goes into an arbiter. When a device wants to be a bus master, it** asserts the bus request line. When the arbiter sees the bus request, it determines if a **bus grant can be issued (it may be the case that the current bus master will not** allow itself to be interrupted).

BUS ARBITRATION—MASTERS AND SLAVES

- If a bus grant can be issued, then the arbiter asserts the bus grant line. The bus grant line is **daisy chained from one device to the** next. The first device that sees the asserted bus grant and also wants to be the bus master takes control of the bus and does not propagate the bus grant to higher for these two numbered devices.
- If a device does not want the bus, then it simply passes the bus grant to the next device. In this way, devices that are electrically closer to the arbiter have higher priorities than devices that are farther away.

BUS ARBITRATION—MASTERS AND SLAVES

- Sometimes an absolute priority ordering is not appropriate, and a number of bus request/bus grant lines are used as shown in Figure (b). Lower numbered bus request lines have higher priorities than higher numbered bus request lines. In order to raise the priority of a device that is far from the arbiter, it can be assigned to a lower numbered bus request line. Priorities are assigned within a group on the same bus request level by electrical proximity to the arbiter.
- Taking this to an extreme, each device can have its own bus request/bus grant line as shown in Figure (c). This fully centralized approach is the most powerful from a logical standpoint, but from a practical standpoint, it is the least scalable of all of the approaches. A significant cost is the need for additional lines (a precious commodity) on the bus

BUS ARBITRATION—MASTERS AND SLAVES

- In a fourth approach, a decentralized bus arbitration scheme is used as illustrated in Figure 8-6(d). Notice the lack of a central arbiter. A device that wants to become a bus master first asserts the bus request line, and then it checks if the bus is busy. If the busy line is not asserted, then the device sends a 0 to the next higher numbered device on the daisy chain, asserts the busy line, and de-asserts the bus request line. If the bus is busy, or if a device does not want the bus, then it simply propagates the bus grant to the next device.
- Arbitration needs to be a fast operation, and for that reason, a centralized scheme will only work well for a small number of devices (up to about eight). For a large number of devices, a decentralized scheme is more appropriate.

BUS ARBITRATION—MASTERS AND SLAVES

•(a) Simple centralized bus arbitration;

• (b) centralized arbitration with priority levels;

• (c) fully centralized bus arbitration;

• (d) decentralized bus arbitration.

•(Source: adapted from [Tanenbaum, 1999].)