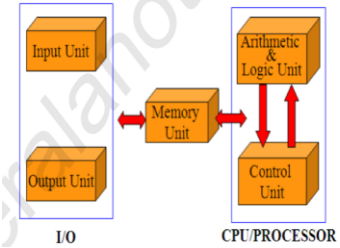
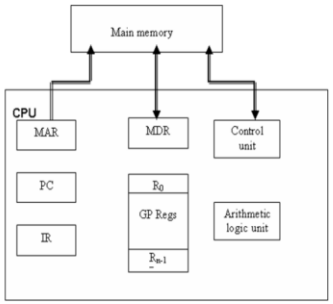
MOD1 Explain the functional units of computer



*Input Unit: Data/ instructions are fed to a computer through input unit Examples: keyboard, mouse. Scanner, joystick*Arithmetic & Logic Unit: ALU consist of necessary logic circuits like adder, comparator etc., to perform arithmetic and logic operations such as addition, multiplication, comparison of two numbers etc.*Control Unit: Control unit co-ordinates activities of all units by issuing control signals. Control signals issued by control unit govern the data transfers and then appropriate operations take place. Control unit interprets or decides the operation/action to be performed. ALU &Control Unit together called Central Processing Unit (CPU). CPU is the brain of a computer system.• Memory Unit: Memory unit stores the program instructions (Code), data and results of computations etcExamples: ROM,RAM Output Unit: Computer after computation returns the computed results, error messages, etc. via output unit. Examples: Monitor, printer, speaker

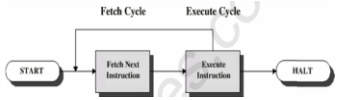
2. Describe the operational concepts of a computer?.A set of instructions called a program reside in the main memory of computer. Data is also stored in the memory *The CPU fetches those instructions sequentially one-by-one from the main memory, decodes them and performs the specified operation on associated data operands in ALU.*Processed data and results will be stored in the memory and displayed on an output unit.*

3. What is a register? Describe the functions of various registers in CPU with a neat diagram./ Explain the internal architecture of a processor.? Registers are high-speed storage area within the CPU. All data must be stored in a register before it can be processed.In addition to ALU CPU also contains CPU also contains Instruction Register (IR), the Program Counter(PC), the general-purpose registers, the Memory Address Register(MAR) and Memory Data Register(MDR).All activities pertaining to processing and data movement inside the computer machine are governed by control unit.

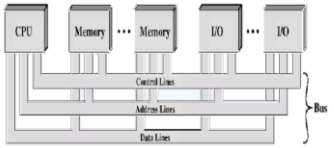


Functions*Instruction Register(IR): contains the instruction that is being executed. Its output is available to the control circuits, that generate the timing signals for control of the actual processing circuits needed to execute the instruction.*Program Counter(PC): It contains the memory address of the next instruction to be fetched and executed*Memory Address Register (MAR): holds the address of the memory location to or from which data is to be transferred.*Memory Data Register(MDR): contains the data to be written into or read-out of the addressed memory location. *General-purpose Registers: are used for holding data, intermediate results of operations. They are also known as scratch-pad registers.

4. Illustrate the basic operational concepts in transferring data between main memory and processor.?• Programs reside in the memory & usually get these through the input unit.*Execution of the program starts when the PC is set to point at the first instruction of the program,*Contents of PC are transferred to MAR and a Read Control Signal is sent to the memory.*The addressed word is read out of the memory and loaded into the MDR.*Now contents of MDR are transferred to the IR & now the instruction is ready to be decoded and executed..*If the instruction involves an operation by the ALU, it is necessary to obtain the required operands.*An operand in the memory is fetched by sending its address to MAR & Initiating a read cycle.• When the operand has been read from the memory to the MDR, it is transferred from MDR to the ALU.*After one or two such repeated cycles, the ALU can perform the desired operation.*If the result of this operation is to be stored in the memory, the result is sent to MDR.*Address of location where the result is stored is sent to MAR & a write cycle is initiated.*The contents of PC are incremented so that PC points to the next instruction that is to be executed.



5. List different types of buses used to interconnect various functional units in a computer.? Data Bus: It is used for transmission of data. The number of data lines corresponds to the number of bits in a word.*Address Bus: It carries the address of the main memory location from where the data can be accessed.*Control Bus: It is used to indicate the direction of data transfer and to coordinate the timing of events during the transfer.



6. Explain zero, one, two and three address instruction with an example for each.?Three-address instruction format*This instruction format consists of three addresses along with an operation field. The three addresses include the address of the first operand, address of the second operand, addressto store the result.*Format: Operation code Sourcel,source2, destination *Example: Add A,B,

[C]<- [A]+[B]

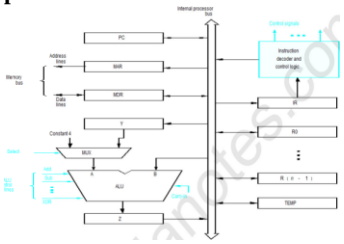
Two-address instruction format This instruction format consists of two addresses along with an operation field. The two addresses include the address of the first operand, address of the second operand; the result is stored in one of the operand address. es/co Example: Add A, B ,B<-[A]+[B] One -address instruction format*This instruction format consists of one address along with an operation field. The address is that of the first operand. The second operand and the result are stored in a CPU register called accumulator. A machine has only one accumulator, it need not be explicitly mentioned in the instruction ,Example: Add A,Zero-address instruction format*A stack is included in the CPU for performing arithmetic and logic instructions with no addresses. The operands are pushed onto the stack from memory and ALU operations are implicitly performed on the top elements of the stack. Example: Add,Top of stack = top of stack + second top of stack

7. What is an addressing mode? List different addressing modes used in an instruction execution.? The address generated by the CPU in-order to access the operand in the memory is termed as an effective address. The methods used to provide an access path to operands in memory and CPU registers is addressing mode.Various addressing modes are *Immediate*Register* Direct(Absolute)*Indirect*Index (Displacement)*Base with index*Base with index and

offset*Relative*Auto increment*Auto decrement

8. Describe auto increment addressing mode with the help of an example.? After accessing the operand the content of this register is automatically incremented to point the next item in the list. Increment R erates.com Add (RI)+, R2

10. With the help of a diagram, describe the data-path inside the processor?



Processor fetches one instruction at a time and perform the operation specified.• Instructions are fetched from successive memory locations until a branch or a jump instruction is encountered.• Processor keeps track of the address of the memory location containing the next instruction to be fetched using Program Counter (PC).*Fetch the contents of the memory location pointed to by the PC. The contents of this location are loaded into the IR (fetch phase).IR <-[[PC]]* Assuming that the memory is byte addressable(4 byte long), increment the contents of the PC by 4 (fetch phase).*PC [PC] +4 *PC Keeps track of execution of a program*Contains the memory address of the next instruction to be fetched and executed. *Constant 4 is used by the processor to increment the contents of PC.• Carry out the actions specified by the instruction in the IR (execution phase).*The data and address lines of external memory connected to the processor bus via MDR and MAR• Register MDR has 2 inputs and 2 outputs.:*Contains data to be written into or read out of the addressed location.*Data can be loaded into MDR either from memory bus or from internal processorbus.• MAR Holds the address of the location to be accessed* I/P of MAR is connected to Internal bus and an O/p to external bus.• The instruction decoder and control logic circuit is responsible for implementing the actions specified by the instruction loaded in IR*The decoder generate the control signals needed to select the registers*The registers, ALU and interconnecting bus are collectively referred to as the data path.*MUX*Select either the output of the register Y or a constant value 4 to be provided as input A of the ALU.

11. Give the control sequence for execution of instruction Add[R3],R1 using a single bus organization?

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

12. Write down the sequence of actions needed to fetch and execute the instruction?LOAD 10(R2),RI.* 1. Pcout, MARin. Read Select4,Add, Zin *2.Zout, Pein, Yin, WMFC *3.MDRout, IRin*4. R2out,MARin, Read*5. WMFC, MDRout*6. IR10out, Yin*7. Select Y. Add, Zin*8. Zout, R1 in, End

13. Write down the sequence of actions needed to fetch and execute the instruction?

STORE RI, 10(R2) *1. Peout, MARin, Read Select4,Add, Zin*2. Zout, Pein, Yin, WMFC*3.MDRout, IRin*4. R2out,MARin, Read*5. WMFC, MDRout*6. IR 10out, Yin*7. Select Y, Add. Zin*8. Zout, MARin*9. R1 out, MDRin, Write*10. WMFC, End

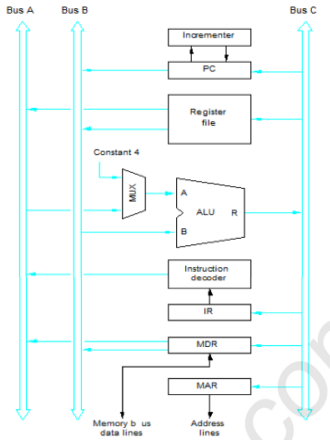
14. Give the sequence of control steps required to perform the operation ADD NUM, RI?

1.Pcout, MARin, Read Select4,Add. Zin*2.Zout, Pein, Yin, WMFC*3.MDRout, Rin*4. IRNUMout, Yin,*5. SelectY, RTout, Add, Zin*6. Zout, Rlin,End.

17. Illustrate the advantages of using multiple bus organization over single bus organization.?

*Multiple Bus Organization Improves Efficiency.*Additional Buses Allow Expansion.*More Buses Means More Compatibility*Processor speed is high in multiple bus organization

18. Draw the diagram of a multi-bus organization with 3 buses. Write the control sequence for the instruction Add R4, R5, R6 for the above mentioned multi-bus organization?.



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

19. How is a branching instruction executed?

A branch instruction replaces the contents of PC with the branch target address, which is usually obtained by adding an offset X given in the branch instruction. The offset X is usually the difference between the branch target address and the address immediately following the branch instruction *Control Sequence of a branch instruction 1. PC out, MKAR in, Read, Clear Y, Set carry in to ALU,ADD,Zin 2. Z out, PC in, WMFC 3. MDR out, IR in 4. PC out, Y in 5. Offset field of IR out, ADD, Zin 6. Zout, PC in End

MOD2: 20. List the components of register transfer logic.

*The set of registers in the system and their functions: A register also encompasses all type of registers including shift registers, counters and memory units.*The binary-coded information stored in the registers: The binary information stored in registers may be binary numbers, binary coded decimal numbers, alphanumeric characters, control information or any other binary coded information.*The operations performed on the information stored in the registers: The operations performed on data stored in registers are called micro operations. Examples are shift, count, add, clear and load*The control functions that initiate the sequence of operations: The control functions that initiate the sequence of operations consists of timing signals that sequence the operations one at a time

21. What is microoperation?

*Micro-operations: operations executed on data stored in one or more registers. For any function of the computer, a sequence of micro-operations is used to describe it Example: Shift, count, clear, add & load

22. What are the different types of micro operations?

Register transfer microoperations b. Arithmetic microoperations (on numeric data stored in the registers)*c. Logic microoperations (bit manipulations on non-numeric data) *d. Shift microoperations

23. How do you represent register in resister transfer logic.?

• Rectangular box with name of the register inside,*The individual cells is assigned a letter with a subscript number,*The numbering of cells from right to left can be marked on top of the box.*16 bit register is partitioned into 2 parts, bits 1 to 8 are assigned the letter (for low) and bits 9 to 16 are assigned the letter for high**).**

Components of Register

Transfer Logic 1. The set of registers in the system and their functions:A register also encompasses all type of registers including shift registers, counters and memory units.

2. The binary-coded information stored in the registers: The binary information stored in registers may be binary numbers, binary coded decimal numbers, alphanumeric characters, control information or any other binary coded information.

3. The operations performed on the information stored in the registers:

The operations performed on data stored in registers are called micro operations. Examples are shift, count, add, clear and load

4. The control functions that initiate the sequence of operations:

The control functions that initiate the sequence of operations consists of timing signals that sequence the operations one at a time.

Micro-Operation: Operations performed in data stored in registers. Elementary operation that can be performed parallel during one clock pulse period. The result of operation may replace the previous binary information of a register or may be transfered to another register. Example: Shift, count, clear, add & load

Types of Micro-Operations in digital system • Interregister transfer micro-operation: Do not change the information content when the binary information moves from one register to another • Arithmetic operation: Perform arithmetic on numbers stored in registers. • Logic microoperation: Perform operations such as AND and OR on individual pairs of bits stored in registers. • Shift microoperation: Specify operations for shift registers. **Design of Arithmetic Circuit** The basic component of the arithmetic section of an ALU is a parallel adder. A parallel adder is constructed with a number of full-adder circuits connected in cascade. By controlling the data inputs to the parallel adder, it is possible to obtain different types of arithmetic operations. The above figure demonstrates the arithmetic operations obtained when one set of inputs to a parallel adder is controlled externally. The number of bits in the parallel adder may be of any value. The input carry Cin goes to the full-adder circuit in the least significant bit position. The output carry Cout comes from the fulladder circuit in the most significant bit position.

Design of Arithmetic Logic Unit The logic circuit can be combined with the arithmetic circuit to produce one arithmetic logic unit. Selection variables S1 and S0 can be made common to both sections provided, we are using a third selection variable S2, to differentiate between the two. Design steps 1. Design the arithmetic section independent of the logic section. 2. Determine the logic operations obtained from the arithmetic circuit in step 1, assuming that the input carries to all stages are 0. 3. Modify the arithmetic circuit to obtain the required logic operations.

Status Registers The relative magnitude of two numbers may be determined by subtracting one number from the other andthen checking certain bit conditions in the resultant difference. This status bit conditions (often calledcondition-code bits or flag bits) are stored in a status register. Status register is a 4 bit register. The four bits are C (carry), Z (zero),S (sign) and V (overflow).These bits are set or cleared as a result of an operation performed in the ALU. • Bit C is set if the output carry of an ALU is 1. • Bit S is set to 1 if the highest order bit of the result in the output of the ALU is 1. • Bit Z is set to 1 if the output of the ALU contains all O's. • Bit V is set if the exclusive —OR of carries C8 and C9 is 1, and cleared otherwise. This is the condition for overflow when the numbers are in signed 2's complement representation. For an 8 bit ALU, V is set if the result is greater than 127 or less than - 128.

Accumulator Register An accumulator is a register for shortterm, intermediate storage of arithmetic and logic data in a computer's CPU (central processing unit).The most elementary use for an accumulator is adding a sequence of numbers. The numerical value in the accumulator increases as each number is added, exactly as it happens in a simple desktop calculator (but much faster, of course). Once the sum has been determined, it is written to the main memory or to another register.

MOD3: Array Multiplier Binary multiplication can be implemented in a combinational two-dimensional logic array called array multiplier. • The main component in each in each cell is a full adder, FA. • The AND gate in each cell determines whether a multiplicand bit mj , is added to the incoming partial product bit based on the value of the multiplier bit, qi . • Each row i, where 0<= i <=3, adds the multiplicand (appropriately shifted) to the incoming parcel product, PPi , to generate the outgoing partial product, PP(i+1), if qi .=1. • If qi .=0, PPi is passed vertically downward unchanged. PP0 is all 0's and PP4 is the desired product. The multiplication is shifted left one position per row by the diagonal signal path.

Sequential Circuit Multiplier

Multiplication is performed as a series of (n) conditional addition and shift operation such that if the given bit of the multiplier is 0 then only a shift operation is performed, while if the given bit of the multiplier is 1 then addition of the partial products and a shift operation are performed. The combinational array multiplier uses a large number of logic gates for multiplying numbers. Multiplication of two n-bit numbers can also be performed in a sequential circuit that uses a single n bit adder

Algorithm: (1) The multiplier and multiplicand are loaded into two registers Q and M. Third register A and C are cleared to 0. (2) In each cycle it performs 2 steps: (a) If LSB of the multiplier qi =1, control sequencer generates Add signal which adds the multiplicand M with the register A and the result is stored in A. (b) If qi =0, it generates Noadd signal to restore the previous value in register A. (3) Right shift the registers C, A and Q by 1 bit

The Booth Algorithm Algorithm

1. Multiplicand is placed in BR and Multiplier in QR 2. Accumulator register AC, Qn+1 are initialized to 0 3. Sequence counter SC is initialized to n (number of bits). 4. Compare Qn and Qn+1 and perform the following 01 –> AC=AC+BR 10 –> AC=AC+BR'+1 00 –> No arithmetic operation 11-> No arithmetic operation 5. ASHR-Arithmetic Shift right AC,QR 6.

Decrement SC by 1 The final product will be store in AC, QR

CLASSIFICATION OF PIPELINE PROCESSORS 1. Arithmetic Pipelining: The arithmetic logic units of a computer can be segmented for pipeline operations in various data formats. 2. Instruction Pipelining: The execution of stream of instructions can be pipelined by overlapping the execution of current instruction with the fetch, decode and execution of subsequent instructions. This technique is known as instruction lookahead 3. Processor Pipelining: Pipeline processing of the same data stream by a cascade of processors, each of which processes a specific task. The data stream passes the first processor with the results stored in memory block which is also accessible by the second processor. The second processor then passes the refined results to the third and so on.

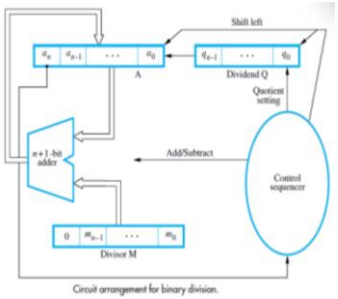
PIPELINING Pipelining is a technique of decomposing a sequential process into sub operations, with each sub process being executed in a special dedicated segment that operates concurrently with all other segments. A pipeline can be visualized as a collection of processing segments through which binary information flows. Each segment performs partial processing dictated by the way the task is partitioned A pipeline processor may process each instruction in 4 steps: F Fetch: Read the instruction from the memory D Decode: Decode the instruction and fetch the source operands E Execute: Perform the operation specified by the instruction W Write: Store the result in the destination location

ARITHMETIC PIPELINES An arithmetic pipeline divides an arithmetic operation into sub operations for execution in the pipeline segments. Pipeline arithmetic units are usually found in very high speed computers. They are used to implementfloating point operations, multiplication of fixed point numbers, and similar computations encountered in scientific problems

INSTRUCTION PIPELINE An instruction pipeline operates on a stream of instructions by overlapping the fetch, decode, and execute phases of instruction cycle. An instruction pipeline reads consecutive instructions from memory while previous instructions are being executed in other segments. This causes the instruction fetch and executes phases to overlap and perform simultaneous operations.

PIPELINE HAZARDS DETECTION AND RESOLUTION Pipeline hazards are caused by resource usage conflicts among various instructions in the pipeline. Such hazards are triggered by inter instruction dependencies when successive instructions overlap their fetch, decode and execution through a pipeline processor, inter instruction dependencies may arise to prevent the sequential data flow in the pipeline There are three classes of data dependent hazards, according to various data update patterns: 1. Write After Read hazards (WAR) 2. Read After Write hazards (RAW) 3. Write After Write hazards (WAW) We use resource object to refer to working registers, memory locations and special flags. The contents of these resource objects are called data objects. Each instruction can be considered a mapping from a set of data objects to a set of data objects. The domain D(I) of an instruction I is a set of resource objects whose data objects may affect the execution of instruction I. The range of an instruction R(I) is the set of resource objects whose data objects may be modified by the execution of instruction I. Obviously, the operands to be used in an instruction execution are retrieved (read) from its domain and the results will be stored (written) in its range.

Restoring Division

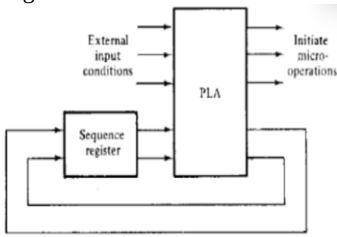


Circuit arrangement for binary division.

Figure shows a logic circuit arrangement that implements the restoring division algorithm just discussed. An n-bit positive divisor is loaded into register M and an nbit positive dividend is loaded into register Q at the start of the operation. Register A is set to 0. After the division is complete, the n-bit quotient is in register Q and the remainder is in register A. The required subtractions are facilitated by using 2'scomplement arithmetic. The extra bit position at the left end of both A and M accommodates the sign bit during subtractions. The following algorithm performs restoring division. Do the following three steps n times: 1. Shift A and Q left one bit position. 2. Subtract M from A, ie; (A-M) and place the answer back in A. 3. If the sign of A is 1, set q0 to 0 and add M back to A (that is, restore A); otherwise, set q0 to 1.

MOD4: Micro-program Control The purpose of control unit is to initiate a series of sequential steps of microoperations. At any given time certain operations are to be initiated while all others remain idle. The control variable at any given time can be represented by a string of 1's and 0's called control word. The control words can be programmed to initiate the various components in the system in an organized manner. A control unit whose control variables are stored in a memory called a micro-programmed control unit. Each control word of memory is called Microinstruction and Sequence of microinstructions is called Micro-program. Control memory is usually ROM since an alteration of micro-program is seldom needed. The use of micro-program involves placing all control variables in words of the ROM for use by the control unit through successive read operations.The content of the word in the ROM at a given address specifies the micro-operations for the system.

PLA control The external sequence register establishes the present state of the control circuit. The PLA outputs determine which microoperations should be initiated depending on the external input conditions and the present state of the sequence register. At the same time other PLA outputs determine the next state of the sequence register.

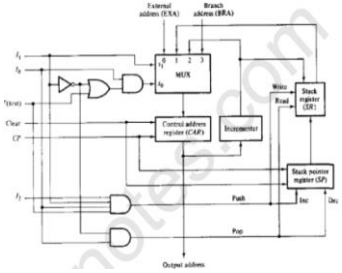


HARDWIRED CONTROL

The control hardware can be viewed as a state machine that changes from one state to another in every clock cycle, depending on the contents of the instruction register, the condition codes and the external inputs. The outputs of the state machine are the control signals. Control logic derived in this section is a hardwired control of the one flip-flop per state method. The design of hardwired control is carried out in 5 consecutive steps 1. The problem is stated 2. An initial equipment configuration is assumed 3. An algorithm is formulated 4. The data

processor part is specified 5. The control logic is designed.

Micro-program Sequencer Micro program sequencer is a control unit which does the tasks of Micro-program sequencing.There are two important factors must be considered while designing the micro programsequencer. o The size of the microinstruction o The address generation time Micro-program sequencer is attached to the control memory.It inspects certain bits in the microinstruction to determine the next address for controlmemory.A typical sequencer has the following address sequencing capabilities. 1. Increments the present address of control memory 2. Branches to an address which will be specified in the bits of microinstruction 3. Branches to a given address if a specified status bit is equal to 1. 4. Transfers control to a new address as specified by an external source 5. Has a facility for subroutines calls and returns.



The output from CAR provides the address for control memory.The contents of CAR are incremented and applied to the multiplexer and to the stack registerfile. The register selected in the stack is determined by stack pointer. Inputs (I0-I2) specify theoperation for the sequencer and input T is the test point for a status bit**.**

Horizontal Micro-Instructions The scheme of micro-instruction by assigning one bit position to each control signal is called horizontal microinstructions. Example: 011101001101001110 In a horizontal microinstruction every bit in the control field attaches to a controller.Horizontal microinstructions represent several microoperations that are executed at the same time.

Vertical Micro-Instructions We can reduce the length of the horizontal micro-instruction so easily by implementing another method known as vertical micro-instructions. In this case, Most signals are not needed simultaneously and many others are mutually exclusive Example: In a vertical microinstruction, a code is used for each action to be performed andthe decoder translates this code into individual control signals. The vertical microinstruction resembles the conventional machine language format comprising one operation and a few operands.

MOD5: DIRECT MEMORY ACCESS • It is a technique used for high speed I/O device. • Here, the device interface transfer data directly to or from the memory without continuous involvement by the processor • A special control unit may be provided to allow the transfer of large block of data at high speed directly between the external device and main memory, without continuous intervention by the processor. This approach is called DMA. • DMA transfers are performed by a control circuit called the DMA Controller. To initiate the transfer of a block of words , the processor sends, ¬ Starting address ¬ Number of words in the block ¬ Direction of transfer. When a block of data is transferred , the DMA controller increment the memory address for successive words and keep track of number of words and it also informs the processor by raising an interrupt signal. While DMA control is taking place, the program requested the transfer cannot continue and the processor can be used to execute another program. After DMA transfer is completed, the processor returns to the program that requested the transfer. R/W - Determines the direction of transfer. R/W =1, DMA controller read data from memory to I/O device. R/W =0, DMA controller perform write operation. Done Flag=1, the controller has completed transferring a block of data and is ready to receive another command. IE=1, it causes the controller to raise an interrupt (interrupt Enabled) after it has completed transferring the block of data. IRQ=1, it indicates that the controller has requested an interrupt.

Use of DMA controllers in a computer system

Burst Mode: The DMA controller may be given exclusive access to the main memory to transfer a block of data without interruption. This is known as Burst/Block Mode Bus Master: The device that is allowed to initiate data transfers on the bus at any given time is called the bus master. Bus Arbitration: It is the process by which the next device to become the bus master is selected and the bus mastership is transferred to it

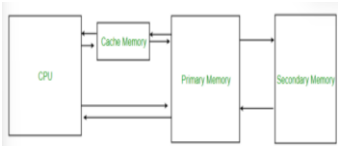
Cycle Stealing: Requests by DMA devices for using the bus are having higher priority than processor requests. Top priority is given to high speed peripherals such as , ¬ Disk ¬ High speed Network Interface and Graphics display device. Since the processor originates most memory access cycles, the DMA controller can be said to steal the memory cycles from the processor. This interviewing technique is called Cycle stealing

SEMICONDUCTOR RAM MEMORIES Semiconductor memories are available in a wide range of speeds. Their cycle times range from 100ns to less than 10 ns **Static Memories (SRAM)** Static memories are the memories that consist of circuits capable of retaining their state as long as power is applied. Two transistor inverters are cross connected to implement a basic flip-flop. The cell is connected to one word line and two bits lines by transistors T1 and T2. When word line is at ground level, the transistors are turned off and the latch retains its state. Read operation: In order to read state of SRAM cell, the word line is activated to close switches T1 and T2. Sense/Write circuits at the bottom monitor the state of b and b‟. Write operation: During the 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

Dynamic Memories (DRAM) Static RAMs are fast, but the cost is too high because their cells require several transistors.Less expensive RAMs can be implemented if simpler cells are used. Such cells don't retain their state indefinitely; hence they are called dynamic RAMs Read Operation: Transistor turned on, Sensor check voltage of capacitor. If voltage is less than Threshold value, Capacitor discharged and it represents logical „0‟ else if voltage is above Threshold value, Capacitor charged to full voltage and it represents Logical „1‟ Write Operation - Transistor is turned on and a voltage is applied/removed to the bit line.

CACHE MEMORIES Cache Memory is a special very high-speed memory. It is used to speed up and synchronizing with high-speed CPU. Cache memory is costlier than main memory or disk memory but economical than CPU

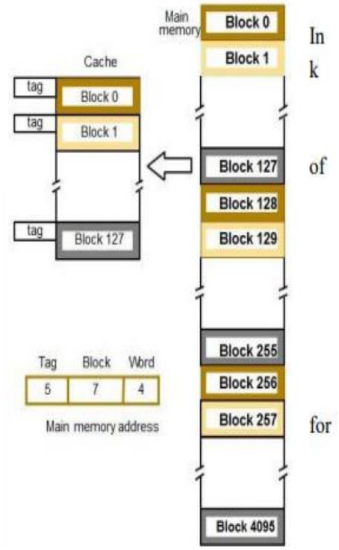
registers. Cache memory is an extremely fast memory type that acts as a buffer between RAM and the CPU. It holds frequently requested data and instructions so that they are immediately available to the CPU when needed.



Existence of a cache is transparent to the processor. The processor issues Read and Write requests in the same manner. If the data is in the cache, it is called a Read or Write hit. Read hit: The data is obtained from the cache. Write hit: Cache has a replicaof the contents of the main memory. Contents of the cache and the main memory may be updated simultaneously.This is the write-through protocol. Update the contents of the cache, and mark it as updated by setting a bit knownas the dirty bit or modified bit Read miss: Block of words containing this requested word is transferred from the memory Write-miss: Write-through protocol is used, and then the contents of the main memory areupdated directly.

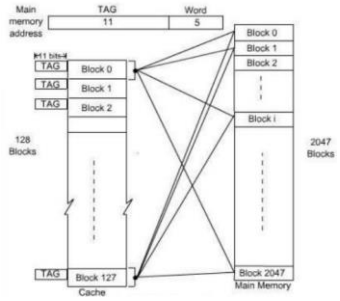
MAPPING FUNCTIONS The mapping functions are used to map a particular block of main memory to a particular block of cache Three mapping functions: • Direct mapping. • Associative mapping. • Set-associative mapping.

Direct Mapping A particular block of main memory can be brought to a particular block of cache memory. So, it is not flexible. The simplest way of associating main memory blocks with cache block is the direct mapping technique. In this technique, block k of main memory maps into block k modulo m of the cache, where m is the total number of blocks in cache.



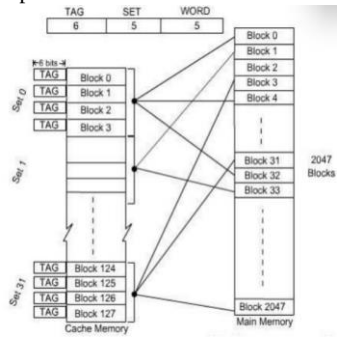
Associative mapping

In the associative mapping technique, a main memory block can potentially reside in any cache block position. In this case, the main memory address is divided into two groups, a low-order bit identifies the location of a word within a block and a high-order bit identifies the block



Set-Associative mapping

This mapping technique is intermediate to the previous two techniques. Blocks of the cache are grouped into sets, and the mapping allows a block of main memory to reside in any block of a specific set. Therefore, the flexibility of associative mapping is reduced from full freedom to a set of specific blocks. This also reduces the searching overhead, because the search is restricted to number of sets, instead of number of blocks. Also the contention problem of the direct mapping is eased by having a few choices for block replacement.



CONTENT ADDRESSABLE MEMORY (CAM)/ ASSOCIATIVE MEMORY

The established way to search a table is to store all items where they can be addressed in sequence. The search procedure is a strategy for choosing a sequence of addresses, reading the content of memory at each address, and comparing the information read with the item being searched until a match occurs. The number of accesses to memory depends on the location of theitem and the efficiency of the search algorithm. The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address. A memory unit accessed by content is called an associative memory or Content Addressable Memory (CAM) This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location