Computer Architecture and Organization

**Unit-IV**

**BUS INTERFACE**

The dillining in transferring information among the units of a computer largely depends on the physical distance separating them. There are two cases:

**1. Intra system communication:-** It occurs within a single computer system and involves information transfer over distance less than a memory. Intra system communication is primarily implemented by groups of coaxial wires called buses, which support parallel, i.e., word data transmission.

**2. Inter system communication:** - It involves communication overmuch longer distances. Intersystem communication is one using physical medium including cables, optical fibers and wireless links.

The various process-level components, CPU, caches, main memory and IO devices within a computer system communication buses. The term bus covers not only the physical links among the components bur also the mechanism for controlling the exchanges of signals over the bus.

Control Address System Data Bus

CPU Cache/main

memoryIO

**---** device 1

IO

device n

**Figure 4.1 – Communication within a computer via a single shared bus.**

**Que. 4 shows the most basic computer bus structure.**

Here a single bus, the system bus, handles all intra-system communication. The units share the system bus, therefore at any time only two units can associate with each other. A typical system bus traction is a memory read (load) operation that involves the transfer of one or more data words over the system bus form the memory cache or (main) M to the CPU. A memory writes (store) operation transfer data over the system bus in the opposite direction.

Input-Output operation normally involves data transfer between an IO device and M. In all the preceding operation M in a slave device with respect to system bus transactions, whereas the CPU can actively control the system bus, that is serve as a bus master. IO devices are normally thought of as slave units, but they can be made into bus master via control units

M.Sc. Semester – II (Computer Science) 1

Computer Architecture and Organization

such as specialized IO controllers of general purpose IO processors. In figure 4.1 indicates, the system bus consist of three main groups of lines:

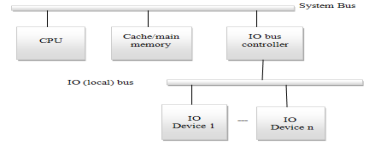
• **Address line** – The address lines, typically 8 to 12 in numbers, transmit addresses of data items stored in the system’s main memory or IO address space.

• **Data line** – The data lines, typically 16 to 128 in number, transmit data words over the bus.

• **Control line** – The control lines perform such functions as identifying the transaction type (memory read, memory write, IO interrupt and so on) and synchronizing communication between fast and slow units.

The characteristics of a system bus tend to closely match of its CPU and vary widely between differently microprocessor families and even between members of the same family. The evolution of CPU’s in speed and word size often matched by a corresponding evolution in their system buses. For example, the first member of Intel’s 80\*86 family, the 8086 microprocessors has interred data and address word sizes of 16 to 20 bits with the 80286 microprocessors; both the same 32 bits with the 80386. The data and address sizes used inside the CPU are often, but not always, the same as those found in the external system bus.

The principle use of the system bus is high-speed data transfer between the CPU and M. most IO devices are slower than the CPU or M and present an external interface that is different from that of the system bus. For example, magnetic disk units and other secondary memory transfer data serially. Therefore, they need to be connected to the system bus via interface circuits called IO controllers that perform series-to-parallel and parallel-to-series format conversions and other control functions. A single IO controller on interface many IO devices to the system bus. As shown in figure 4.2 IO devices are connected to a separate bus called an IO bus.

**Figure 4.2 – Computer with separate system and IO buses.**

M.Sc. Semester – II (Computer Science) 2

Computer Architecture and Organization

Controller manufactures and standards organizations have standardized various IO bus types. For example, the Small Computer System Interface known as the SCSI bus was adopted as a standard by the American National Standard Institute (ANSI) in 1986. This bus connects IO devices such as hard disk units and refers to personal computers. SCSI was originally designed to transfer data a byte at a time rates up to 5 MB/s.

Another bus with a role similar to SCSI is called Industry Standard Architecture (ISA) bus originally developed by the Intel for the IBM PC.

Bus system can distinguished by the manner in which data transfer over the bus are timed.

**Synchronous Bus-** In synchronous bus, each item is transferred during a time slot known in advance to both the source and destination units. Synchronization units. Synchronization can be achieved by driving both units from a common clock source. Synchronized signals must then be transmitted periodically between the communication devices in order to keep their clocks in step with each other. Synchronous communicated has the disadvantages that the time, slot used for information transfer is largest determined by the slowest unit in the system.

**Asynchronous bus –**In asynchronous bus, each item brain transferred is accompanied by a separate control signals to indicate its performance the destination unit. The destination unit may respond with another control signal to acknowledge receipt of the information. **DATA RANSFER TECHIQUES (DTT)**

In a computer, data transfer tables place between two such as CPU and memory, CPU and I/O devices and namely to I/O devices. Usually memory is so designed to be compatible with microprocessor while I/O are not. A computer is interfaces within number of I/O devices of different speeds. In such cases, a slow I/O device may not be ready to give data when processor issues an instruction. Therefore to solve this problem of speed mismatching, a number of data transfer techniques are defined. These techniques are classified as

1. Programmed Data Transfer Technique

2. Direct Memory Access (DMA) Data Transfer Technique.

Data transfer takes place using bus. Buses are usually divided into three functional groups:

**Data lines** – The data lines are designed to transmit all bits of an n bit work in parallel. The data bus size n is usually a multiple of 8.

**Address lines** – Address lines are used to identify a unit or port of a unit to be used in data transfer to be given access to the bus.

M.Sc. Semester – II (Computer Science) 3

Computer Architecture and Organization

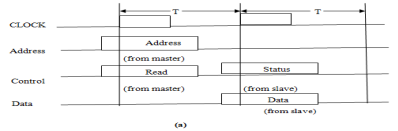
**Control lines**– The control lines or control signals are used to transfer timing signals and status information about the units in the system.

**Programmed Data Technique (DTT)**

In a programmed DTT, The data transfer takes place between the CPU and I/O devices under the control of a program which resides in memory. This program is executed by the CPU. Here the data are transferred when I/O devices are ready for the same. This technique is used when small amount of data is to be transfer. There are three different types that are included under this DTT. They are;

**I. Synchronous DTT**

When the processor and I/O devices match in speed; this type of DTT is used. The data can be transferred from microprocessor to the I/O device using suitable instructions such in IN and OUT. The IN instruction is used to transfer data from input device or from an input port to the processor. Similarly, OUT instruction is used to send data from processor to an output device or to output port. As the speed of I/o devices and processor match i.e. there is synchronization between their speeds, the I/O devices are away ready when processor requires data. Therefore there is no speed mismatch. Figure 4.3 illustrates some typical signal exchanges between slave and master units. In figure 4.3(a), data transfer begins when the bus master places one or more predetermined signals on the control lines specifying the desired bus transaction, for instance read from memory (load) or read from IO device (input). At the same time, the master places the address of the desired slave on the bus’s address lines. All potential slave units then examine the active control and address signals. The slave with as address matching that on the bus responds in the next clock cycle by placing the requested data word on the bus’s data lines; it can also optionally placed information, for example no error occurred, on certain control line. A synchronous write operation is similar except that the bus master rather than the slave is the data source shown in figure 4.3(b). Note that both edges CLOCK can be used as reference points in a bus transaction and the read or write transactions can be designed to take place during one clock cycle of period 2T.

 M.Sc. Semester – II (Computer Science) 4

Computer Architecture and Organization

T T

CLOCK

Address

Address

(from master)

Control

Write

Data

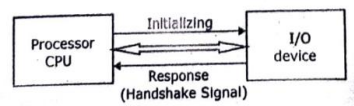
(from master) (from slave) Data

(from master) **(b)**

**Figure 4.3- synchronous data transfer (a) read and (b) write.**

**II. Asynchronous DTT**

This data transfer technique is used when the speeds of I/O devices and the microprocessor do not match and changing characteristic of the I/O devices is not predictable. Thus the microprocessor waits all the device becomes ready to transfer the data. This method is used then I/O devices are slow as compared to microprocessor. In this method, the precious time of the processors is wasted in waiting. Due to this reason this techniques is supposed to be an insufficient method of data transfer. Here two signals are used before an actual data transfer starts. First is that the microprocessor sends an initializing signal to the I/o devices consider to get ready for data transfer. When the data is ready to transfer, I/O device sends a signal to processor. This type of signals is called handshake signals and the mode is called handshake mode of data transfer. The handshaking signal prevents the processors from writing new data before the device has accepted the previous data and become ready to accept new data.

 Asynchronous timing eliminates the bus’s clock signal and replaces it with timing control signals like ACK, which are generated by the bus units. These units are self-timed

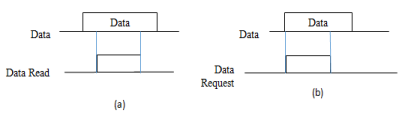
M.Sc. Semester – II (Computer Science) 5

Computer Architecture and Organization

and units with quite different data-transfer rates can communication asynchronously. There are two cases.

**a) One-way asynchronous data a transfer**.

Here one of the two communicating devices supplies all timing signals. If one-way asynchronous is employed, a single signal controls each address or data transfer. This signal can be activated by the source and destination unit, either one of which can be the bus master. Figure 4.4(a) shows a source-initiated data transfer. The source places the data word on the data bus. After a brief delay the source activates the control line with the genetic name DATA READY. The delay is to prevent the DATA READY signal from searching the destination before the word. Alternatively, the source can activate DATA READY also place the data on the data bus at the same time. Figure 4.4(b) shows a data transfer initiated by the destination unit. In this case destination begins the data transfer by activating the control line DATA REQUEST. The source responds by placing the required word on the data lines. Again the data must remain active state long enough to allow the destination unit to read it. Often the DATA REQUEST signals are used to load the data from the source unit to the bits or from the bus to the destination unit. Such control signals are called strove signals are said to strobe data to or from the bus

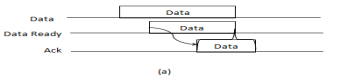
**Figure 4.4 One-way synchronous data transfer timing: (a) source initiated and (b) Destination initiated. b) Two-way asynchronous data transfer –**

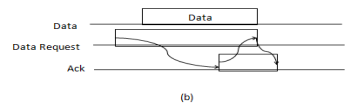
Here both devices can generate the timing signals. The disadvantage of one-way data transfer is that it does not verify that the data transfer has been successfully completed. For example, in a source-initiated data transfer, the source unit receives no indication that the destination unit has actually received the data transmitted to it. If the destination unit is unexpectedly slow in responding to a DATA READY signal, the data may be lost. This problem is eliminated by introduced a second control line that allows the destination unit to send a reply signal to the source when, it receives a DATA READY signal. This control line is known as DATA ACKNOWLEDGE or ACK. Figure 4.5 (a) shows the exchanges of signals, often called handshaking, which accompanies a source –controlled transfer in this

M.Sc. Semester – II (Computer Science) 6

Computer Architecture and Organization

case. The source unit maintains the data on the bus unit it receives the ACK signals. The destination activates ACK after copying the data from the bus. This sequence allows delays of arbitrary length to occur during the data transfer. Figure 4.5(b0 shows same technique for destination-initiated communication. The source unit activates ACK to indicate that the requested data is available on the bus’s data lines. The source maintains the data on the bus until the destination unit deactivates DATA REQUEST, an action that confirms successful receipt of the data at its destination.



**Figure 4.5 – Asynchronous and transfer (handshaking ): (a) source initiated and (b) Destination initiated.**

**c) Interrupt driven DTT**

In this type of DTT, the processor first initiate the I/O device to get ready and then goes on executing its originals program instead of wasting its time in continuously checking the status of I/O device. Now when I/O device becomes ready to send or receive data, it informs the processor through a specific control line called interrupt line. When the processor receives an interrupt, the processor completes the execution of the current instruction at hand. Thereafter, instead of executing the next instruction of the program, CPU takes necessary steps to transfer data from I/O device. First of all, it saves all the content of program counter in the stack then the processor enters in subroutine called Interrupt Service Subroutine(ISS). The ISS saves the status of the processor in the stack and then performs data transfer from the I/O device which interrupted the processor. After completing the data transfer, it restores the processor status and then returns to the original program which the processor was executing before the interrupt signal was received. This interrupt driven DTT is used to interface relatively slow I/O devices such as character printer, A-D connectors etc. In this technique, the valuable time of processor is efficiently utilized as compare to asynchronous data transfer. This improves the overall performance of a system.

M.Sc. Semester – II (Computer Science) 7

Computer Architecture and Organization

**Direct Memory Access (DMA) Data Transfer Technique –**

In DMA data transfer, the data are directly transferred from I/O device to the memory or vice-versa without going through the microprocessor. The CPU does not participate in this type of data transfer. This technique is generally used when bulk amount of data has to be transferred. If bulk amount of data is to be transferred through the microprocessor, it would be a time consuming process.

The microprocessor holds on when data transfer takes place between memory and I/O device using DMA technique. The I/O device which uses DMA technique of data transfer sends on HOLD signal to microprocessor. Having received an HOLD signal from which it sends in HLDA to an I/O device to indicate that HOLD request has received, and the data and address buses have been released. In other words, the microprocessor transfers the control of buses to an I/O device. After gaining control over buses, the I/O device transfers of gain data from memory without involving CPU. This technique of data transfer is used by mass storage devices like HOD, FDD, high speed printers etc. When data transfer is over, the CPU again gains the control over the buses. This mode of data transfer is called as Burst mode data transfer.

In DMA transfer technique, data transfer takes place under the control of an I/O device and therefore an I/O device must contain its own register to store memory address and byte count. It must also contain electronic circuit to generate control signals required for DMA data transfer. Generally I/O devices are not equipped with there. To solve this problem, manufactures have developed single chip programmable DMA controller to interface between I/O devices to the microprocessor. For DMA data transfer, such chips are INTEL 8237A, 82C37AA-5, 82380, 82307 etc.

**IO AND SYSTEM CONTROL**

IO system includes IO devices, control units, which require special software to carry out operation. IO operations are distinguished by the extent to which the CPU is involved in their execution. IO operation means data transfer between an IO device and M, or between an IO device and the CPU. If such operations are completely controlled by CPU, i.e. the CPU executes programs that initiate, direct and terminate the IO operations, the computer is said to be using programmed IO.

This type of IO control can be implemented with little or no special hardware, bus cases the CPU to spend a lot of time performing relatively trivial IO-related functions. One such function is testing the status on devices to determine if they require servicing by the CPU.

M.Sc. Semester – II (Computer Science) 8

Computer Architecture and Organization

Modern computer hardware enables an IO devices transfer a block of information to or from M without CPU intervention. This task required the IO device to generate memory addresses and transfer and transfer data to or local bus connecting it to M via its interface controller. The CPU is still responsible for initiating each block transfer. The IO device interface controller can then carry out the transfer without further program execution by the CPU and IO controller interact only when the CPU controls the system bus to the IO controller in response to the requests. This level of IO controller is called Direct Memory Access. (DMA), and the IO device interface control circuit is called DMA controller.

The DMA controller can also be provided with circuits enabling it to request service from the CPU, i.e. execution of a specific program to services an IO device. This type of request is called an interrupt and it frees the CPU from the task of periodically testing the status of IO devices. After the interrupt has been encountered. Most computers have DMA and interrupt facilities, which are supported by special DMA and interrupt

control units.

**Programmed IO**

• It is a method available every computer for controlling IO operations. • It is most useful in small, low-speed systems where hardware costs must be minimized.

• Programmed IO requires the all IO operations be executed under the direct control of the CPU; in other words, every data-transfer operation involving an IO device requires the execution of an instruction b the CPU.

• Type all the transfer is between two programmable registers; one a CPU register and other attached to the IO device.

• The IO device to M requires the CPU to execute several instructions, including an input instruction to transfer a word from the IO device to the CPU and a store instruction to transfer the word from the CPU to M.

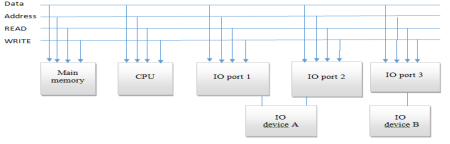
**IO Addressing** – In systems employing programmed IO, the CPU, and M and IO devices usually communicate via system bus. The address lines of the system bus that are used to select memory locations can also be used to select IO devices. An IO device is connected to the bus via an IO port which from the CPU.

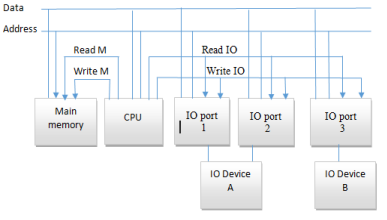
Technique used in may machines to assign a port of the main-memory address space to IO ports. This technique is called memory-mapped IO. A memory-referencing

M.Sc. Semester – II (Computer Science) 9

Computer Architecture and Organization

that causes data to be fetched from or stored at address. X automatically becomes an IO instruction if X is made the address of an port. The usual memory load and store instructions are needed. Figure 4.6 shows the essential structure of a computer with this type of IO addressing. The control lines READ and WRITE which are activated by the CPU when processing a memory reference instruction are used to initiate either a memory access cycle or an IO transfer.

**Figure 4.6 Programmed IO with shared memory and IO address space (Memory-trapped IO)** In the organization shown in figure 4.7 known as IO-mapped IO, the memory and IO address spaces are separate. A memory referencing instruction activates the READ M or WRITE M control line which does not affect the IO devices. The CPU must execute separate IO instruction a activate the READ IO and WRITE IO lines, which cause a word to be transferred between the addressed IO port and the CPU. An IO device and a memory location can have the same address bit pattern without conflict.

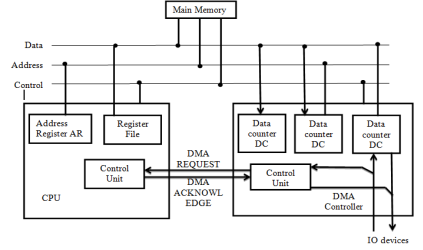
**Figure 4.7 – Programmed IO with separate memory and IO address spaces (IO-mapped IO)**

M.Sc. Semester – II (Computer Science) 10

Computer Architecture and Organization

**Direct Memory Access (DMA)**

The hardware needed to implement DMA is shown in figure 4.8, assuming that all **access** to main memory is via a shared bus. The IO device is connected to the system bus via a special interface circuit known as DMA controller, which contains a data buffer register IODR, as in the programmed IO. It also controls an address register IOAR and a data count register DC. These registers enable the DMA controller to transfer data to or from a contiguous region of memory. IOAR stores the address of the next word to be transferred. It is automatically incremented or decremented after each transfer and nested for zero. When the data count reaches zero, the DMA transfer halts. The DMA controller is normally provided with an interrupt capability, in which case it sends an interrupt to the CPU to signal the end of the IO data transfer. The logic necessary to control DMA can easily be placed to a single IC with other IO control circuits. A DMA controller can be designed to supervise DMA transfers involving several IO devices, each with a different priority of access to the system bus.

**Figure 4.8- Circuitry required for DMA**

DMA transfers proceed as follows for the system shown in figure 4.8.

1. The CPU executes two IO Instructions, which load the DMA register IOAR and DC with their initial values. IOAR should contain the base address of the memory region to be used in the data transfer. DC should contain the number of words to be transferred to or from that region.

M.Sc. Semester – II (Computer Science) 11

Computer Architecture and Organization

2. When the DMA controller is ready to transmit or receive data, it activates the DMA REQUEST line to the CPU. The CPU waits for the next DMA breakpoint. It then relinquishes control of the data and address lines and activates DMA ACKNOWLWDGE. Here DMA REQUEST and DMA ACKNOLEDGE is essentially BUS REQUEST and BUS FRANT lines for control of the system bus.

3. The DMA controller now transfers data directly to or from main memory. After a word is transferred, IOAR and DC are updated.

4. If DC has not yet reached zero but the IO device is not ready to send or receive the next batch of data, the DMA controller releases the system bus to the CPU by deactivating the DMA REQUEST line. The CPU responds by deactivating DMA ACKNOLEDGE and resuming control of the system bus.

5. If DC is decremented to zero, the DMA controller again relinquishes control of the system bus; it may also send an interrupt request signal to the CPU. The CPU responds by halting the IO device or by initiating a new DMA transfer. **I/O INTERRUPTS**

The word interrupt is an infrequent or exceptional event that causes a CPU to temporarily transfer control from its current program to another program that services the event. Interrupts are the primary means by which IO devices obtain the services of thee CPU. They significantly improve a computer’s IO performance by giving IO devices direct and rapid access to the CPU and by freeing CPU from the need to check the status of its IO devices. There are two sources for interrupts.

**I. External interrupts-**IO interrupts are external requests to the CPU to initiate or terminate an IO operation, such as a data transfer with a hard disk. Such interrupts caused by min memory. Interrupts are also produced by hardware or software error-detection circuits that invoke error-handling routine within the operating system. A power-supply failure can generate an interrupt that request execution of an interrupt handler designed to save critical data about the system’s state.

**II. Internal interrupts -** An interrupts by an instruction to divide by zero, or to execute a privileged instruction when not in the privileged state, is example of software generated interrupts. An operating system will also interrupt user program that has exceeded its allotted time. The basic method of interrupting the CPU is by activating a control line with the generic name INTERRUPT REQUEST that connects the interrupt source to the CPU. An interrupt indicator is then stored in a CPU register that the CPU tests periodically, usually at the end of every instruction cycle. On recognizing the presence of the interrupt, the CPU

M.Sc. Semester – II (Computer Science) 12

Computer Architecture and Organization

executes a specific interrupt-handling program. Normally, each interrupt source requires execution of a different program, so the CPU must determine or be given the address of the interrupt program to be used.

The presence of two or more interrupt requests at the same time causes a further problem. The CPU responds to an interrupt request by a transfer of control to an interrupt handler in a manner similar to a subroutine call. The following steps are taken:

1) The CPU identifies the source to the interrupt, for example, by polling IO devices. 2) The CPU obtains the memory address of the required interrupt handler. This address can be provided by the interrupting device along with its interrupt request. 3) The program counter PC and other CPU status information are saved as in a subroutine call.

4) The PC is loaded with the address of the interrupt handler. Execution proceeds until a return instruction are encountered, which transfers control back to the interrupted program.

Instruction sets usually include instructions to selectively disable or mark interrupts requests, thereby causing the CPU to ignore certain interrupts. Without such control, an IO device that generates interrupts rapidly might require too much of the CPU’s time and interface with the CPU’s other tasks. Where a high priority interrupt is being serviced, it is desirable that all interrupts of lower priority be disabled.

**Interrupt Selection**

The problem of selecting one IO device to service from several that have generated interrupts strongly resembles the arbitration process for bus control.

There are two methods of interrupt selection

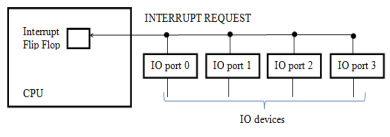
**1. Single-line Interrupt selection**

The interrupt selection method requiring the least hardware is the single-line method that appears in figure 4.9. All IO ports share a single INTERRUPT REQUEST line. On responding to an interrupt request, the CPU must scan all the IO devices to determine the source of the interrupt. This procedure requires activating an INTERRUPT ACKNOWLEDGE line that is connected in daily chain fashion to all IO devices. The connection sequence of this line determines the interrupt priority of each device.

Alternatively the CPU can execute a program that polls each IO device in turn requesting interrupt status information.

M.Sc. Semester – II (Computer Science) 13

Computer Architecture and Organization



**Figure 4.9- Single-line Interrupt system**

**2. Multiple-line or multilevel interrupt selection**

Figure 4.10 shows another common interrupt selection method called multiple-line or multilevel interrupts, which amounts to independent requesting of interrupt service. Each interrupt request line is assigned a unique priority. The source of the interrupt is immediately known to the CPU, thus eliminating the need for a hardware or software scan of the IO ports. The CPU may still have to execute a program that fetches the address of the interrupt-service program to be used. This step can be eliminating by another technique called vectoring of interrupts.

INT REQ 3

Interrupt

INT REQ 2

register

INT REQ 1

INT REQ 0

IO port 0 IO port 1 IO port 2 IO port 3

CPU

IO devices

**Figure 4.10 – Multiple-line interrupt system**

**IO PROCESSOR OR CHANNELS (523 Hayes)**

The 10 Processor (IOP) is a logical extension of the IO control method. An IOP has the ability to execute instruction, which gives it complete control IP operations. An IOP is an Instruction-set processor out it has a more restricted instruction set. IOPs are primarily communication Control units designed to link IO devices to a computer. They have also been called peripheral processing unit (PPU) or channels. The instruction secured by the IOP is

M.Sc. Semester – II (Computer Science) 14

Computer Architecture and Organization

called channel command words (CCW) and have the format shown in figure 4.11. They are of three types:

**1. Data transfer instructions-** These include input (read), output (write), and sensor status. They cause the number of bytes in the data count filed to be transferred between the specified memory region and the previously selected IO device.

**2. Branch instructions-** These cause the 1OP to fetch the next CCW from the specified memory address rather than from the next sequential location.

**3. IO device control instruction-**These are transmitted to the IO device and specify

functions to that device.

0 8 32 37 48 63

| Opcode | Memory address | Flags |  | Data count (bytes) |
| --- | --- | --- | --- | --- |

**Figure 4.11 – An IOP channel**

**IOP organization**

The structure of a system containing an IQP appears in figure 4.12. The IOP and CPU share access to a common memory M via the system bus. M stores separate programs for execution by the CPU and the IOP. It also contains a communication region IOCR for passing information in the form of messages between the two processors. The CPU can place there the parameters of an 10 task, for example, the addresses of the IO programs to be executed, and the identity of the IO devices to be used. The CPU and IOP also communicate with each other directly via control lines. Standard DMA or bus grant/acknowledge lines are used for arbitration of the system bus between the two processors.

M.Sc. Semester – II (Computer Science) 15

Computer Architecture and Organization

Main Memory CPU programs

IOCR

IOP programs

CPU-IOP Communication Region

System bus

DMA REQUEST

DMA ACKNOWLEDGE

CPU IOP

INTERRUPT REQUEST

ATTENTION

IO bus

---

IO devices

**Figure 4.12 System organization of computer containing an IOP**

The CPU an attract the IOP’s attention when executing an IO instruction like START IO, by activating the ATTENTION line. In response the IOP begins execution an IOP program whose specifications have been placed in the IOCR communication area. Similarly the IOP attracts the CPU's attention by activating an INTERRUPT REQUEST line, causing the CPU to execute an interrupt handler that typically responds to the IOP by identifying a new IO program for the IOP to execute. Following algorithm shows the overall behaviour of the IOP and its interaction with the CPU.

Algorithm showing interaction between CPU and IOP

WAIT: If ATTENTION = 1 then

Begin

Fetch parameters from IOCR

SETUP: Set up DMA control register;

Begin IO program execution;

M.Sc. Semester – II (Computer Science) 16

Computer Architecture and Organization

Send commands to IO device;

SEND:Transmit data word;

If transmission error then go to EXIT;

If not end of data then go to SEND;

If not end of IO program then go to SETUP:

EXIT: Place termination status in IOCR:

End;

Go to WAIT;

**BUS ARBITRATION**

If several units generate requests for bus access simultaneously, the bus controller must have a method for selecting one of the units. This election process is called bus arbitration. There are three main arbitration schemes.

1) Daisy-chaining scheme

2) Polling scheme

3) Independent requesting scheme

These schemes differ in the number of control lines they require and in the speed with which the bus controller on respond to bus-access requests of different priorities. Some bus systems combine s distinct arbitration techniques.

**Daisy-chaining Scheme**

Figure 4.13 shows daisy-chaining arbitration. This method involves three control signals to which some generic names are assigned like BUS REQUEST GRANT and BUS BUSY. All the bus unit are……..to the BUS REQUEST line. When activated, it merely serves to indicate that one or more units are requesting use of the bus. The bus controller

responds to a BUS REQUEST signal only if BUS BUSY is inactive. This response takes the form of a signal placed on the BUS GRANT line. On receiving the SUS GRANT signal, a requesting unit enables its physical bus controller and activates BUS BUSY for the duration of its new bus activity.

The man distinguish feature of daisy chaining is the way the BUS GRANT signal distributed; it is connected serially from unit to unit as shown in figure. When the first-unit requesting access to the bus receives BUS GRANT, it blocks further propagation of that signal, activates BUS BUSY, and begins to use the bus. When a non-requesting unit receives the BUS GRANT signal, it forwards the signal to the next unit. Thus if two units simultaneously request bus access, the one closer to the bus controller, i.e. the one that receives BUS GRANT first, gains access to the bus.

M.Sc. Semester – II (Computer Science) 17

Computer Architecture and Organization

BUS GRANT -----

Bus

Controller

BUS REQUEST

U1 U2 Un

BUS BUSY Bus

**Figure 4.13 – Bus arbitration using daisy chaining**

**Polling Scheme**

In this scheme, the BUS GRANT line of the daisy-chain method with a set of poll count lines that are connected directly to all units on the bus or shown in figure 4.14. The units request access to the bus via a common BUSREQUEST line. In response to a signal on BUS REQUEST, the bus controller proceeds to generate a sequence of numbers on the poll count line, Each unit compares these numbers, which may be thought of as unit address, to a unique address assigned to that unit. When a requesting unit U, finds that its address matches the number on the poll-count lines. The bus controller responds by terminating the polling process and Ui, connects to the bus.

----

U1 U2 Un

Poll

Count

----- ----

BUS REQUEST Bus

Bus

Controller

BUS BUSY ----

| ---- |
| --- |

**Figure 4.14 – Bus arbitration using polling**

The priority if a bus unit is determined by the position of its address in the polling sequence. This sequence can be programmed if the poll-count lines are connected to a programmable register; hence selection priority can be altered und software control. A further advantage of polling over daisy chaining is that in polling a failure in one unit need not affect the other units. This flexibility is achieved at the cost of more control lines. Also, the number

M.Sc. Semester – II (Computer Science) 18

Computer Architecture and Organization

of units that can share the bus is limited by the addressing capability of the poll-count lines. **Independent Requesting Scheme**

It has separate BUS REQUEST and BUS GRANT lines for every unit sharing the bus. This approach is shown in figure 4.15, provides the bus controller with immediate identification of all requesting units and enables it to respond rapidly to requests for bus access. The bus-control unit determines priority, which is programmable. The main drawback of bus control by independent requesting is the fact that 2n BUS REQUEST and BUS GRANT lines must be connected to the bus controller in order to control n devices. In contrast, daisy chaining requires two such lines, while going requires approximately long lines.

----

U1 U2 Un

BUS GRANT 1

~~BUS REQUEST 1~~

BUS GRANT 2

Bus

Controller

BUS REQUEST 2 ----

|  |  |
| --- | --- |
|  | ---- |

BUS GRANT n

BUS REQUEST n

BUS BUSY

----

**Figure 4.15 – Bus arbitration using Independent Requesting**

**TRANSACTION PROCESSING BENCHMARK**

In computing, a benchmark is the act of running a computer program, set programs, or other operations, In order to assess the relative performance of am of object, normally by running a number of standard tests and trials against it. The term 'benchmark' is also mostly utilized for the purposes of elaborately designed benchmarking programs themselves.

Benchmarking is usually associated with assessing performance characteristics of computer hardware, for example, the floating point operation performance of a CPU, but there are circumstances when the technique is also applicable to software. Software benchmarks are,

M.Sc. Semester – II (Computer Science) 19

Computer Architecture and Organization

for example, run against compilers or database management systems. Benchmarks provide a method of comparing the performance of various subsystems across different chip/system architectures. As computer architecture advanced, it became more difficult to compare the performance of various computer systems simply by looking at their specifications.

Therefore, tests were developed that allowed comparison of different architectures For example, Pentium 4 processors generally operate at a higher clock frequency than Athlon XP processors, which does not necessarily translate to more computational power. A slower processor, with regard to clock frequency, can perform as well as a processor operating at a higher frequency.

Benchmarks are designed to mimic a particular type of workload on component or system. Synthetic benchmarks to this by specially created programs that impose the workload on the component. Application benchmarks run real word programs on the system. While application benchmarks usually give much better measure of real word performance on give system, synthetic benchmarks are useful for testing individual components, like a hard disk or networking device There are three types of popular benchmarks; kernel, synthetic, and application.

**Kernel** benchmarks are based on analysis and the knowledge that in most cases 10 percent of the code uses 80 percent of the CPU resources. Performance analysts have extracted these code fragmented and have used them as benchmarks. Examples of kernel benchmarks are Livemore Loops and Linpack benchmarks. The fundamental problems with kernel benchmarks are that they usually small, fit in cache, are prone to attack by compilers, and measure only CPU performance.

**Synthetic** benchmarks are based on performance analysts experience and knowledge of instruction mix. Examples of synthetic benchmarks are Dhrvstone and Whetstone. New technologies and instruction set architectures make some older assumptions regarding instruction mix obsolete. Synthetic benchmarks offer problems similar to those of kernel benchmarks.

From a user’s perspective, the best benchmark is the user’s own application program. Examples of application benchmarks are Spice (circuit designers) and GNU compiler (software developers using GNU environments). Unfortunately, there are thousands of applications, and many of them are proprietary. A benchmarks suite with a large number of applications is also impractical because of difficulties in porting and evaluation and long runtime.

M.Sc. Semester – II (Computer Science) 20

Computer Architecture and Organization

**Transaction Processing Performance Council** (TPC) is a non-profit organization founded in 1988 to define transaction processing and database benchmarks and to disseminate objective, verifiable TPC performance data to the industry. TPC benchmarks are used in evaluating the performance of computer system; the results are published on the TPC web site. A measure of transaction processing power is needed to measure and compare the thought and price/performance of various transaction processing systems.

Vendors of transaction processing system quota. Transaction per Second (TPS) rates for their systems. But there isn’t a standard transaction, so it is difficult to verify or compare these TPS clams in additions, there is no accepted way to price a system supporting a desired TPS rate. This makes it impossible to compare the price/performance of different systems.

The performance of transaction processing system depends heavily on the system input/output architecture, data communications architecture and even more importantly on the efficiency of the system software. The benchmark is adequate because:

The interactive transaction forms the basis for the TPS rating. It is also a litmus test for transaction processing systems. It requires the system have at least minimal presentation services, transaction recovery, and data managements. The mini-batch transaction tells the IO performance available to the COBOL programmer. It tells us how fast the end-user IO software is.

The utility program is included to show what a really tricky programmer can sequence out of the system. It tells us how fast the real IO architecture is. On most systems, the utilities trick the IO software into giving the raw IO device performance with almost no software overhead. In other words, we believe these three benchmarks indicates the performance of a transaction processing system because the utility benchmarks gauges the IO hardware, the mini-batch benchmarks gauges the IO software, and the interactive transaction gauges the performance of the online transaction processing system.

The particular program chosen here have become part of the folklore of computing. Increasingly, they are being used to compare system performance from released to released and in some cases to compare the price/performance different vendor’s transaction processing systems. The basic benchmarks are:

**Debit-card :** A banking transaction interacts with a book mode terminal connected via x.25.The system does presentation service to map the input for a Cobol programming which in turn uses a database system to debit bank account, do the standard double-entry bookkeeping and then reply to the terminal. 95% of the transaction must provide one-second responses into relevant measures are throughput and cost.

M.Sc. Semester – II (Computer Science) 21

Computer Architecture and Organization

**Scan:** A mini-batch COBOL transaction sequentially scans and updates one thousand records. A duplexer transaction is eg. automatically mentioned for transaction recovery. Relevant measures elapsed time and cost.

**Sort**: A disk sort of one million records the source and target files are sequential. Relevant measures are express time and cost.

A benchmark is charge for resources it uses rather than the entire system cost. For example, if the benchmarks run for an hour we chose it for an hour. This in turn requires view a measure system cost/hour rather than just system cost.

To make this specific it’s compute the cost of a sort benchmarks which runs for an hour, uses 2 megabytes of memory and two, discs and their controllers.

| **Package** | **Package cost** | **Per hour cost** | **Benchmarks cost** |
| --- | --- | --- | --- |
| Processor | 10K$ | 1.8$ | 1.8$ |
| Memory | 15K$ | .3$ | .3$ |
| Disk | 50K$ | 1.1$ | 1.1$ |
| Software | 50K$ | 1.1$ | 1.1$ |

So the cost is 4.3 $ per sort.

The people who run the benchmarks are free to configure it for minimum cost or minimum time. They may pick a fast processor, add or drop memory, channels, or other accelerators. In general the minimum-elapsed-time system is not the minimum cost system. For example, the minimum cost Tandem system for Sort is a one processor two disc system. Sort takes about 30 minutes at a cost of 1.5$. On the other hand, we believe a 16 processor two disc Tandem system with 8 Mbytes per processor could do sort within ten minutes for about 15$ - six times faster and 10 times as expensive. In the IBM world, minimum cost generally comes with model 4300 processors, minimum time generally comes wih 308. Processors.

To normalize the TPS measures, most of the transactions must have less than a specified response time. To eliminate, the issue of communication line speed and delay, response time is defined as the time interval between the arrival of the last bit from the communications line and the sending of the first bit to the communications line. This is the metric used by most teleprocessing stress testers.

Hence the Transaction per Second (TPS) unit is defined as :

TPS: peak Debit-Credit transaction per second with 95% of the transactions having less than one second response time.

M.Sc. Semester – II (Computer Science) 22

Computer Architecture and Organization

**PERFORMANCE EVALATION: SPECmarks**

Standard Performance Evaluation Corporation (SPEC) was founded in October 1988, by Apollo, Hewlett-Packard, MIPS Computer Systems and Sun Microsystems in cooperation with E. E. Times. SPEC is non-profit construction of 22 major computer vendors whose common goals are “to provide the industry with a realistic yardstick to measure the performance of advanced computer systems” and to maintains, distributes and endorses a standardized set of application-oriented programs to be used as benchmarks.

A computer system’s performance cannot be characterized by a single number or single benchmarks. Many users (Decision makers), however, are looking for a single-number performance metrics; characterization. The customer faces a plethora of confusing performance metrics and relevance by the press to publish complete information on performance and configuration. Both the press to customer, however, must be informed about the danger and the folly of relying on either a single performance number on single benchmarks.

Unfortunately computer vendors have, until recently, been unable or unwilling to agree on a set standards benchmarks, which has made it virtually impossible for customer to evaluate and compare competing systems. The lack of standards in benchmarks has also created problems for vendors and hardware software and system designers. There is no absolute objective truth in comparing benchmarking results.

SPEC starts to address these issues by selecting and developing real application benchmarks that would exercise major system components, SPEC’s members struggled with questions like,

• So we need one benchmarks or many benchmarks.

• What workload should be used to show a particular system in the best light? • What is system performance? And

• How can we compare different computer systems in an objective manner? SPEC wanted to compare system performance across many different technologies, architectures, implementations, memory systems, I/O subsystems, operating systems, clock rates, bus protocols, compilers libraries, and application software. Additional problems, such as scaling system performance with multiples processors and peripherals, had to be considered. And other factors, like graphics and networking, complicated matters. The raw hardware performance depends on many components; CPUs, floating-point units (FPUs) I/O, graphics and network accelerators, peripherals, and memory systems.

M.Sc. Semester – II (Computer Science) 23

Computer Architecture and Organization

However, system performance as seen by the user depends on the efficiency of the operating system, compiler, libraries, algorithms, and application software.

With so many variables, SPEC’s major goal was that the same source code (machine independent) would run on all members’ machines, which required that all benchmarks be ported to SPEC member’s machines. This turned out to be a nontrivial task. Portability conflicts are resolved by SPEC member at SPEC bench-a-thons. A bench-a-thon is an intense five-day workshop during which engineers from SPECs member companies develop benchmarks and tools that are portable across operating systems and platforms.

SPEC chose a simple measure, elapsed time, to run the benchmarks. A simple speed metric and machine-independent code was key to providing a comprehensive and fair comparison between comparisons between competing machines. SPEC measured performance by determining the times required to run a suite of applications and then comparing the time for completion of each with the time of a reference machine. (DEC VAX 70). The individual results are called the SPEC rations for that particular machine. (There is general consensus that the use of a single number is not the best way to represent the results of a suite of benchmarks. Unfortunately pressures exits that force the development of just such a number. By definition all the metrics for DEC VAX 780 are one.

**Books:**

1. Computer Architecture and Organization by Tenenbaum

2. Computer architecture and Organization by J.P. Hayes.

3. Parallel processing Hwang

4. Computer Organization by Hamacher, Vranesic, Zaky(TMH)

**SPEC MARK**

1. SPEC stands for Standard Performance Evaluation Corporation. SPEC is a non-profit corporation formed to establish, maintain and endorse a standardized set of relevant benchmarks that can be applied to the newest generation of high performance computers. Benchmarks are individual program or a mixture of programs that are run on target computer to measure the overall performance of the system, or to measure specific aspects of the performance.

2. SPEC was formed founded in 1988 and the founder this organization believes that user community will benefit greatly from the objective series of application oriented test which serve as common reference points and be considered during the evaluation process.

M.Sc. Semester – II (Computer Science) 24

Computer Architecture and Organization

3. SPEC member organizations includes leading computer and software manufactures. Today, it has more than 60 member companies including major industry players such as Dell, Intel, Microsoft and Unisys. SPE headquartered in Warrenton, Virginia.

4. The goal of SPEC is to ensure that the market place has a fair and useful set of metrics to differentiate candidate systems. The path chosen in an attempt to balance between requiring strict compliance and allowing vendors to demonstrate their advantages. The belief is that a good test that is reasonable to utilize will leads to a greater availability of results in the marketplace.

5. The basic SPEC methodology is to provide the bench maker with a standardized suite of source code based upon existing application that has already been ported to a wide variety of platforms by its membership. The benchmarks then take this source codes, compiles it for the system in questions and then can tune the system for the best result.

**SPEC basically performs two functions.**

1. SPEC develops suits of benchmarks intended to measure computer performance. These suits are packed with source code and tools and are extensively tested for portability before release. They are available to the public for a covering development and administrative cost. By license agreement SPEC members and customers agree to run and report results are specified in each benchmarks suite’s documentation.

2. SPEC publishes news and benchmarks results in the SPEC NEWS LETTER & GPC QUATERLY both are available electronically through http://www.SPEC.org. This provides a centralize source of information for SPEC benchmarks results.

Members companies may publish in free SPEC NEWS LETTER through there is a fee for non-members. SPEC consists of three major branches or subgroups: • The **Open Systems Group (OSG),** the original SPEC committee develops test standards and procedures desktop computer, workstations and servers.

• The **High Performance Group (HPG),** develops tests standards and procedures for supercomputers and system that use parallel processing, symmetric multiprocessing (smp) and other advanced technologies.

• The **Graphics Performance Characterization Group (GPC)** develops tests standards and procedures for widely used graphics and image application.

M.Sc. Semester – II (Computer Science) 25

Computer Architecture and Organization

For general purpose computers, a suite of benchmark program was selected in 1989. It was modified somewhat and published in 1995 and again in 2000. The programs selected range from playing, compiler, and database application to numerically intensive program and quantum chemistry. In each case, the program is compiled for the computer under test, and the running time on a real computer is measured. The same program is compiled and run on one computer selected as a referenced. For SPEC95, the reference is the SUN SPARC station 10/40. For SPEC2000, the reference is an ultraSPARC 10 workstation with a 300-MHz. The SPEC rating is computed as follows

Running time on the reference computer

SPEC = -------------------------------------------------------

Running time on the computer under test

SPEC benchmark is widely used today in evaluating the performance of computer system. In addition to developing and updating new and existing benchmark, SPEC publishes the results of tests conducted using those benchmarks. The results are published on the SPEC web site. Results are informally referred to as **“SPECmarks”**, results or even just **“SPEC”.**

SPEC benchmark are written in a platform neutral programming language (usually C or Fortran), and the interested parties may compile the code using whatever compiler they prefer for their platform, but may not change the code. Manufactures have been known to optimize their compilers to improve performance of various SPEC benchmarks.

**Transaction processing**

In computer science, transaction processing is information processing that is divided into individual, indivisible operations, called transactions. Each transaction must succeed or fall as a complete unit; it cannot remain in an intermediate state. Contents [hide] **1. Description**

Transaction processing is designed to maintain a computer system (typically a database or some modern file systems) in a known, consistent state, by ensuring that any operations carried out on the system that are interdependent are either all completed successfully or all canceled successfully.

For example, consider a typical banking transaction that involves moving $700 from a customer's savings account to a customer's checking account. This transaction is a single operation in the eyes of the bank, but it involves at least two separate operations in computer terms: debiting the savings account by S700, and crediting the checking account by $700. If the debit operation succeeds but he credit does not (or vice versa), the books of the bank will not balance at the end of the day. There must therefore be a way to ensure that either both

M.Sc. Semester – II (Computer Science) 26

Computer Architecture and Organization

operations succeed or both, so that there is never any inconsistency in the bank's database as a whole. Transaction processing is designed to provide this.

Transaction processing allows multiple individual operations to be linked together automatically as a single, indivisible transaction. The transaction-processing system ensures that either all operations in a transaction are completed without error or none of them are. If some of the operations are completed but errors occur when the others are attempted, the transaction- processing system "rolls back" all of the operations of the transaction (including the successful ones), thereby erasing all traces of the transaction and restoring the system to the consistent, known state that it was in before processing of the transaction began. If all operations of a transaction are completed successfully, the transaction is committed by the system, and all changes to the database are made permanent; the transaction cannot be rolled back once this is done.

Transaction processing guards against hardware and software errors that might leave a transaction partially completed, with the system left in an unknown, inconsistent state. If the computer system crashes in the middle of a transaction, the transaction processing system guarantees that all operations in any uncommitted (i.e., not completely processed) transactions are cancelled. Transactions are processed in a strict chronological order. If transaction n+1 intends to touch the same portion of the database as transaction n, transaction n+1 does not begin until transaction n is committed. Before any transaction is committed, all other transactions affecting the same part of the system must also be committed; there can be no "holes" in the sequence of preceding transactions.

**2. Methodology**

The basic principles of all transaction-processing systems are the same. However, the terminology may vary from one transaction-processing system to another, and the terms used below are not necessarily universal.

**3. Rollback**

Transaction-processing systems ensure database integrity by recording intermediate states of the database as it is modified, then using these records to restore the database to a known state if a transaction cannot be committed. For example, copies of information on the database prior to its modification by a transaction are set aside by the system before the transaction can make any modifications (this is sometimes called a before image). If any part of the transaction fails before it is committed, these copies are used to restore the database to the state it was in before the transaction began.

M.Sc. Semester – II (Computer Science) 27

Computer Architecture and Organization

**4. Rollforward**

It is also possible to keep a separate journal of all modifications to a database (sometimes called after images); this is not required for rollback of failed transactions, but it is useful for updating the database in the event of a database failure, so some transaction processing systems provide it, If the database fails entirely, it must be restored from the most recent back-up. The back-up will not reflect transactions committed since the back-up was made. However, once the database is restored, the journal of after images can be applied to the database (rollforward) to bring the database up to date. Any transactions in progress at the time of the failure can then be rolled back. The result is database in a consistent, known state that includes the results of all transactions committed up to the moment of failure.

**5. Deadlocks**

In some cases, two transactions may, in the course of their processing, attempt to access the same portion of a database at the same time, in a way that prevents them from proceeding. For example, transaction A may access portion X of the database, and transaction B may access portion Y of the database. If, at that point, transaction A then tries to access portion Y of the database while transaction B tries to access portion X, a deadlock occurs, and neither transaction can move forward, Transaction-processing systems are designed to detect these deadlocks when they occur. Typically both transactions will be cancelled and rolled back, and then they will be started again in a different order, automatically, so that the deadlock doesn't occur again. Or sometimes, just one of the deadlocked transactions will be cancelled, rolled back, and automatically re-started after a short delay.

Deadlocks can also occur between three or more transactions. The more transactions involved, the more difficult they are to detect, to the point that transaction processing systems find there is a practical limit to the deadlocks they can detect.

**6. Compensating transaction**

In systems where commit and rollback mechanisms are not available or undesirable, a Compensating transaction is often used to undo failed transactions and restore the system to a previous state.

**ACID criteria (Atomicity, Consistency, Isolation, Durability)**

**Transaction processing has these benefits:**

It allows sharing of computer resources among many users It shifts the time of job processing to when the computing resources are less busy. It avoids idling the computing resources without minute-by-minute human interaction and supervision. It is used on

M.Sc. Semester – II (Computer Science) 28

Computer Architecture and Organization

expensive classes of computers to help amortize the cost by keeping high rates of utilization of those expensive resources. A transaction is an atomic unit of processing. **Benchmark (computing)**

In computing, a benchmark is the act of running a computer program, a set of programs, or other operations, in order to assess the relative performance of an object, normally by running a number of standard tests and trials against it. The term 'benchmark' is also mostly utilized for the purposes of elaborately-designed benchmarking programs themselves. Benchmarking is usually associated with assessing performance characteristics of computer hardware, for example, line floating point operation performance of a CPU, but there are circumstances when the technique is also applicable to software. Software benchmarks are, for example, run against compilers or database management systems. Another type of test program, namely test suites or validation suites, are intended to assess the correctness of software.

Benchmarks provide a method of comparing the performance of various subsystems across different chip/system architectures.

**1. Purpose**

As computer architecture advanced, it became more difficult to compare the performance of various computer systems simply by looking at their specifications. Therefore, tests were developed that allowed comparison of different architectures. For example, Pentium 4 processors generally operate at a higher clock frequency than Athlon XP processors, which does not necessarily translate to more computational power. A slower processor, with regard to clock frequency, can perform as well as a processor operating at a higher frequency. See BogoMips and the megahertz myth.

Benchmarks are designed to mimic a particular type of workload on a component or system. Synthetic benchmarks do this by specially created programs that impose the workload on the component. Application benchmarks run real-world programs on the system. While application benchmarks usually give a much better measure of real-world performance on a given system, synthetic benchmarks are useful for costing individual components, like a hard disk or networking device. Benchmarks are particularly important in CPU design, giving processor architects the ability to measure and make tradeoffs in micro architectural decisions. For example, if a benchmark extracts the key algorithms of an application, it will contain the performance-sensitive aspects of that application. Running this much smaller snippet on a cycle-accurate simulator can give clues on how to improve performance.

M.Sc. Semester – II (Computer Science) 29

Computer Architecture and Organization

Prior to 2000, computer and microprocessor architects used SPEC to do this, although SPEC's Unix-based benchmarks were quite lengthy and thus unwieldy to use intact. Computer manufacturers are known to configure their systems to give unrealistically high performance on benchmark tests that are not replicated in real usage. For instance, during the 1980s some compilers could detect a specific mathematical operation used in a well-known floating-point benchmark and replace the operation with a faster mathematically-equivalent operation. However, such a transformation was rarely useful outside the benchmark until the mid-1990s, when RISC and VLIW architectures emphasized the importance of compiler technology as it related to performance. Benchmarks are now regularly used by compiler companies to improve not only their own benchmark scores, but real application performance.

CPUs that have many execution units - such as a superscalar CPU, a VLIW CPU, or a reconfigurable computing CPU - typically have slower clock rates than a sequential CPU with one or two execution units when built from transistors that are just as fast. Nevertheless, CPUs with many execution units often complete real-world and benchmark tasks in less time than the supposedly faster high-clock-rate CPU.

Given the large number of benchmarks available, a manufacturer can usually find at least one benchmark that shows its system will outperform another system; the other systems can be shown to excel with a different benchmark. Manufacturers commonly report only those benchmarks (or aspects of benchmarks) that show their products in the best light. They also have been known to mis-represent the significance of benchmarks, again to show their products in the best possible light. Taken together, these practices are called bench

marketing. Ideally benchmarks should only substitute for real applications if the application is unavailable, or too difficult or costly to port to a specific processor or computer system. If performance is critical, the only benchmark that matters is the target environment's application suite.

**2. Challenges**

Benchmarking is not easy and often involves several iterative rounds in order to arrive at predictable, useful conclusions. Interpretation of benchmarking data is also extraordinarily difficult. Here is a partial list of common challenges:

Vendors tend to tune their products specifically for industry-standard benchmarks. Norton SysInfo (SI) is particularly easy to tune for, since it mainly biased toward the speed of multiple operations. Use extreme caution in interpreting such results. Some vendors have been accused of "cheating" at benchmarks-doing things that give much higher benchmark

M.Sc. Semester – II (Computer Science) 30

Computer Architecture and Organization

numbers, but make things worse on the actual likely workload. Many benchmarks focus entirely on the speed of computational performance, neglecting other important features of a computer system, such as:

Qualities of service, aside from raw performance. Examples of unmeasured qualities of service include security, availability, reliability, execution integrity, serviceability, scalability (especially the ability to quickly and no disruptively add or reallocate capacity), etc. There are often real trade-offs between and among these qualities of service, and all are important in business computing. Transaction Processing Performance Council Benchmark specifications partially address these concerns by specifying ACID property tests, database scalability rules, and service level requirements.

In general, benchmarks do not measure Total cost of ownership. Transaction Processing Performance Council Benchmark specifications partially address this concern by specifying that a price/performance metric must be reported in addition to a raw performance metric, using a simplified TCO formula. However, the costs are necessarily only partial, and vendors have been known to price specifically (and only) for the benchmark, designing a highly specific "benchmark special" configuration with an artificially low price. Even a tiny deviation from the benchmark package results in a much higher price in real world experience.

Facilities burden (space, power, and cooling). When more power is used, a portable system will have a shorter battery life and require recharging more often. A server that consumes more power and/or space may not be able to fit within existing data center resource constraints, including cooling limitations. There are real trade-offs as most semiconductors require more power to switch faster. See also performance per watt.

In some embedded systems, where memory is a significant cost, better code density can significantly reduce costs. Vendor benchmarks tend to ignore requirements for development, test, and disaster recovery computing capacity. Vendors only like to report what might be narrowly required for production capacity in order to make their initial acquisition price seem as low as possible. Benchmarks are having trouble adapting to widely distributed servers, particularly those with extra sensitivity to network topologies. The emergence of grid computing, in particular, complicates benchmarking since some workloads are "grid friendly", while others are not.

Users can have very different perceptions of performance than benchmarks may suggest. In particular, users appreciate predictability- servers that always meet or exceed service level agreements. Benchmarks tend to emphasize mean scores (IT perspective) rather

M.Sc. Semester – II (Computer Science) 31

Computer Architecture and Organization

than low standard deviations (user perspective). Many server architectures degrade dramatically at high (near 100%) levels of usage-"fall off a cliff"- and benchmarks should (but often do not) take that factor into account. Vendors, in particular, tend to publish server benchmarks at continuous at about 80% usage - an unrealistic situation -- and do not document what happens to the overall system when demand spikes beyond that level.

M.Sc. Semester – II (Computer Science) 32