Q2 (a) Define processor clock and clock rate.

Answer Page Number 12, 16 of Textbook

Q2 (b) Explain instruction sequencing and give an illustration.

Answer Page Number 42-46 of Textbook

Q2 (c) Explain input, output and arithmetic logic unit of a computer.

Answer Page Number 2-6 of Textbook

Q3 (b) Explain Auto increment and Auto decrement mode. Where are these used?

Answer Page Number 57 of Textbook

Q4 (a) Describe the hardware mechanism for handling multiple interrupt requests.

Answer
The device asks for an interrupt, the processor grants that request at a convenient time and informs the device. One straightforward mechanism uses two wires at the physical level. One wire is directed from the device to the CPU (we can call it the Interrupt-Request) whereas the second is directed from the CPU to the device (we can call it Interrupt-Granted). When the device wants to request an interrupt it asserts the interrupt-request signal. When the CPU grants the interrupt, it responds by asserting the Interrupt-Granted signal. This process of "asking" and "granting" is often called a hand-shake.

Multiple devices sharing an interrupt line (of any triggering style) all act as spurious interrupt sources with respect to each other. With many devices on one line the workload in servicing interrupts grows in proportion to the square of the number of devices. It is therefore preferred to spread devices evenly across the available interrupt lines. Shortage of interrupt lines is a problem in older system designs where the interrupt lines are distinct physical conductors. Message-signaled interrupts, where the interrupt line is virtual, are favored in new system architectures (such as PCI Express) and relieve this problem to a considerable extent.

Some devices with a badly designed programming interface provide no way to determine whether they have requested service. They may lock up or otherwise misbehave if serviced when they do not want it. Such devices cannot tolerate spurious interrupts, and so also cannot tolerate sharing an interrupt line. ISA cards, due to often cheap design and construction, are notorious for this problem. Such devices are becoming much rarer, as hardware logic becomes cheaper and new system architectures mandate shareable interrupts.

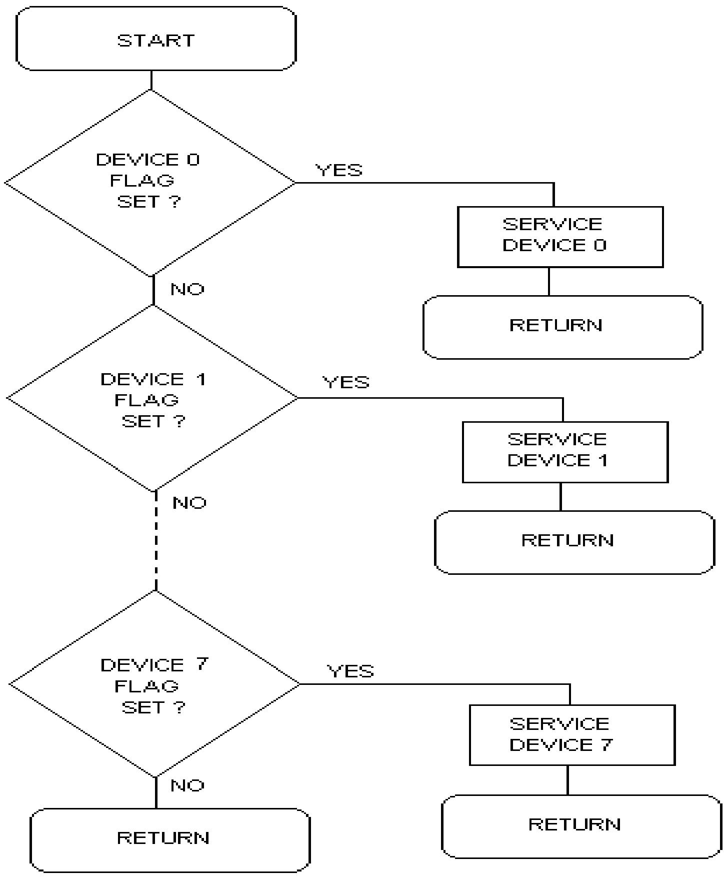
Multiple Interrupts

If more than one device is connected to the interrupt line, the processor needs to know which device service routine it should branch to. The identification of the device requesting service can be done in either hardware or software, or a combination of both. The three main methods are:

1. Software Polling,
2. Hardware Polling, (Daisy Chain),
3. Hardware Identification (Vectored Interrupts).

Software Polling Determination of the Requesting Device

A software routine is used to identify the device requesting service. A simple polling technique is used; each device is checked to see if it was the one needing service.



SOFTWARE POLLING FLOWCHART

Having identified the device, the processor then branches to the appropriate interrupt-handling-routine address for the given device. The order in which the devices appear in the polling sequence determines their priority.

Summary of Software Polled I/O

Polling is the most common and simplest method of I/O control. It requires no special hardware and all I/O transfers are controlled by the CPU programme. Polling is a synchronous mechanism, by which devices are serviced in sequential order.

The polling technique, however, has limitations.

1) It is wasteful of the processors time, as it needlessly checks the status of all devices all the time,

2) It is inherently slow, as it checks the status of all I/O devices before it comes back to check any given one again,

3) When fast devices are connected to a system, polling may simply not be fast enough to satisfy the minimum service requirements,

4) Priority of the device is determined by the order in the polling loop, but it is possible to change it via software.

Q4 (b) What are handshaking signals? Explain the handshake control of data transfer during input and output operation.

Answer

Predetermined hardware or software activity designed to establish or maintain two machines or programs in synchronization. Handshaking often concerns the exchange of messages or packets of data between two systems with limited buffers. A simple handshaking protocol might only involve the receiver sending a message meaning "I received your last message and I am ready for you to send me another one." A more complex handshaking protocol might allow the sender to ask the receiver if he is ready to receive or for the receiver to reply with a negative acknowledgement meaning "I did not receive your last message correctly, please resend it" (e.g. if the data was corrupted en route).

Hardware handshaking uses voltage levels or pulses on wires to carry the handshaking signals whereas software handshaking uses data units (e.g. ASCII characters) carried by some underlying communication medium.

These are dedicated lines used to coordinate data transfer. They are used to signal ready/not ready, or to acknowledge or request. This back and forth handshaking is needed in a sync comms to avoid sending data until the receiver is ready for it.

The MPU and peripherals operate at different speeds, the MPU being faster than the peripherals (like printers and data converters). To avoid overlapping of data during data input (by preventing MPU reading same data twice before i/p peripheral writes next data) or data output (by preventing MPU from writing over the data before o/p device has had chance to accept it), handshake signals are used between MPU & the peripherals. These signals are generally provided by programmable devices.

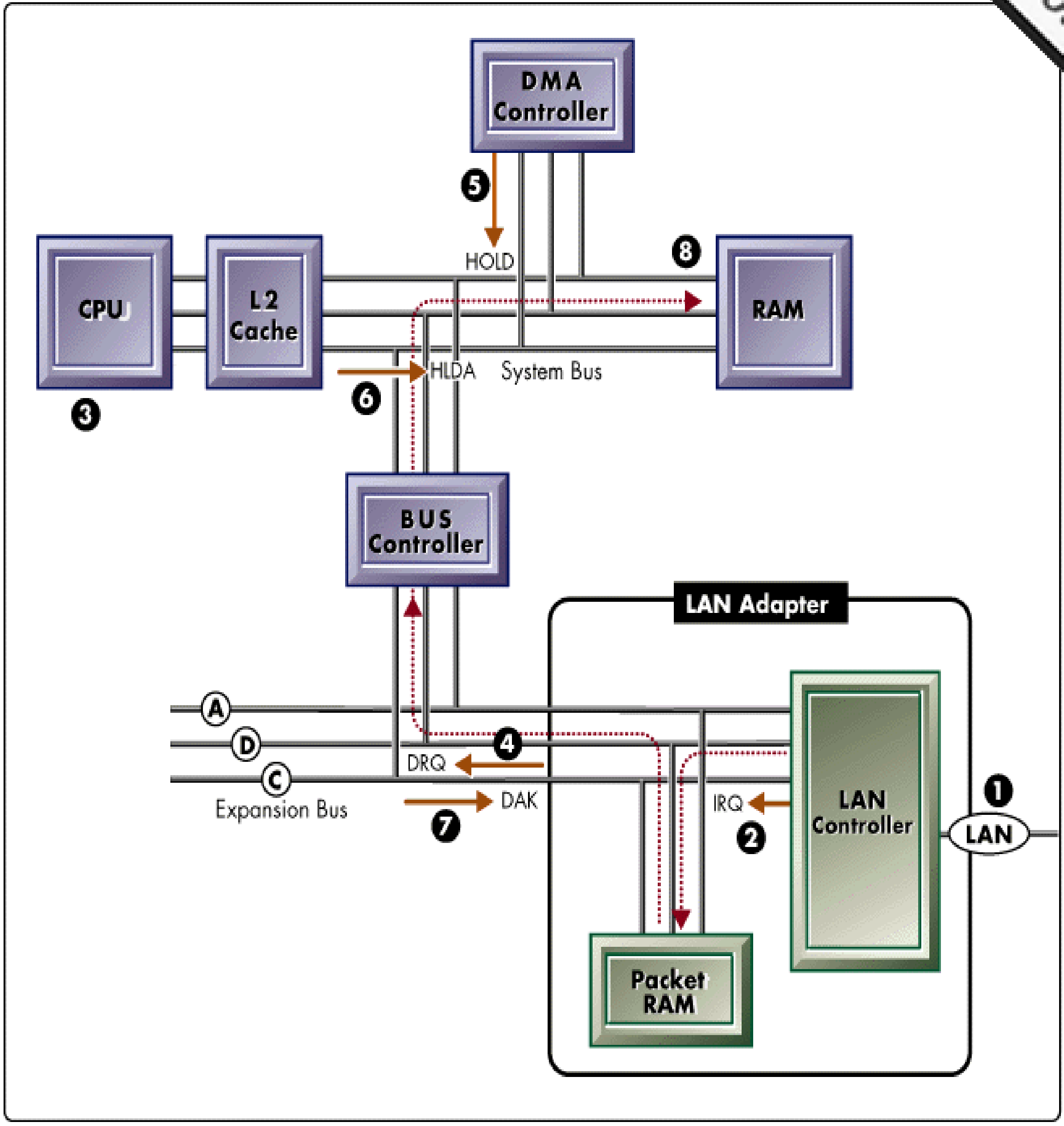
Q4(c) Draw the typical block diagram of a DMA controller and explain how it is used for direct data transfer between memory and peripherals.

Answer

Direct Memory Access (DMA) is one of several methods for coordinating the timing of data transfers between an input/output (I/O) device and the core processing unit or memory in a computer. DMA is one of the faster types of synchronization mechanisms, generally providing significant improvement over interrupts, in terms of both latency and throughput. An I/O device often operates at a much slower speed than the core.
DMA allows the I/O device to access the memory directly, without using the core. DMA can lead to a significant improvement in performance because data movement is one of the most common operations performed in processing applications. There are several advantages of using DMA, rather than the core, in the DSP56300 family:
DMA saves core MIPS because the core can operate in parallel.
DMA saves power because it requires less circuitry than the core to move data.
DMA saves pointers because core AGU pointer registers are not needed.
DMA has no modulo block size restrictions, unlike the core AGU.
Traditionally, DMA uses the same internal address and data buses as the core.
Consequently, when DMA performs one or more word transfers, it can cause the core to temporarily halt activity for one or more cycles while DMA moves the data. With this type of architecture, the core and DMA cannot both perform data moves in the same core clock cycle. To overcome data movement restrictions imposed by sharing resources with the core, the DMA system in the DSP56300 family contains its own dedicated internal address and data buses. Internal memory is partitioned so that the Program Control Unit (PCU) and DMA can both perform internal memory accesses in the same core clock cycle, as long they are accessing different memory partitions. Also, if one of these two controllers (PCU or DMA) is accessing internal memory, the other controller can perform an external memory access in the same core clock cycle.

In DMA as the name suggest the memory can be accessed directly by I/O module. Thus overcome the drawback of programmed I/O and interrupt driven I/O where the CPU is responsible for extracting data from the memory for output & storing data in memory for input. DMA provides different information.
i) Which operation (read/write) to be performed.
ii) The address of I/O device which is to be used.

The DMA controller is among the other components in a computer system. The CPU communicates with the DMA through the address and data buses with any interface unit. The DMA has its own address, which activates with Data selection and one the DMA receives the start control command, it can start the transfer between the peripheral device and CPU.



Q5 (a) What are the needs for input-output interface? Explain the functions of a typical 8-bit parallel interface in detail.

Answer

Input Output Interface:

Input-output interface provides a method for transferring information between internal storage and external I/O devices. Peripherals connected to a computer need special communication links for interfacing them with the central processing unit. The purpose of the communication link is to resolve the differences that exist between the centr computer and each peripheral. The major differences are:

1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory, which are electronic devices. Therefore, a conversion of signal values may be required.

2. The data transfer rate of peripherals is usually slower than the transfer rate of the CPU, and consequently, a synchronization mechanism may be needed.

3. Data codes and formats in peripherals differ from the word format in the CPU and memory.

4. The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

To resolve these differences, computer systems include special hardware components between the CPU and peripherals to supervise and synchronize all input and output transfers. These components are called interface units because they interface between the processor bus and the peripheral device.

Need for I/O interface

1. Peripherals are electromechanical devices. But CPU and Memory are electronic devices. Therefore conversion of signal values may be required.

2. Data codes and formats in peripherals differ from the word format in CPU and memory.
3. Data transfer rate of peripherals are slower than CPU, so synchronization may be needed.

The operating modes of peripherals are different. So they must be controlled so as not to disturb the operation of other peripherals that are connected to CPU.

Q5 (b) Describe the USB architecture with the help of a neat diagram.

Answer

Universal Serial Bus (USB) is a set of interface specifications for high speed wired communication between electronics systems peripherals and devices with or without PC/computer. The USB was originally developed in 1995 by many of the industry leading companies like Intel, Compaq, Microsoft, Digital, IBM, and Northern Telecom.

The major goal of USB was to define an external expansion bus to add peripherals to PC in easy and simple manner. The new external expansion architecture, highlights,

1. PC host controller hardware and software
2. Robust connectors and cable assemblies
3. Peripheral friendly master-slave protocols
4. Expandable through multi-port hubs.

USB offers users simple connectivity. It eliminates the mix of different connectors for different devices like printers, keyboards, mice, and other peripherals. That means USB-bus allows many peripherals to be connected using a single standardized interface socket. Another main advantage is that, in USB environment, DIP-switches are not necessary for setting peripheral addresses and IRQs. It supports all kinds of data, from slow mouse inputs to digitized audio and compressed video.

USB also allows hot swapping. The "hot-swapping" means that the devices can be plugged and unplugged without rebooting the computer or turning off the device. That means, when plugged in, everything configures automatically. So the user needs not worry about terminations, terms such as IRQs and port addresses, or rebooting the computer. Once the user is finished, they can simply unplug the cable out, the host will detect its absence and automatically unload the driver. This makes the USB a plug-and-play interface between a computer and add-on devices.

The loading of the appropriate driver is done using a PID/VID (Product ID/Vendor ID) combination. The VID is supplied by the USB Implementer's forum

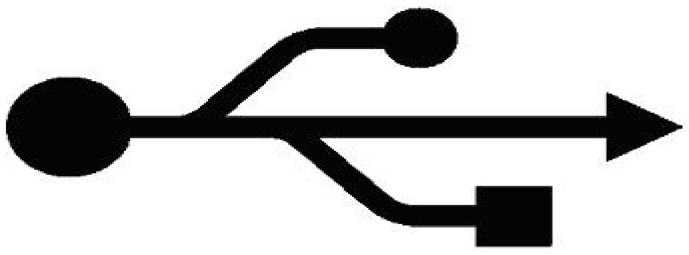


Fig 1: The USB "trident" logo

The USB has already replaced the RS232 and other old parallel communications in many applications. USB is now the most used interface to connect devices like mouse, keyboards, PDAs, game-pads and joysticks, scanners, digital cameras, printers, personal media players, and flash drives to personal computers. Generally speaking, USB is the most successful interconnect in the history of personal computing and has migrated into consumer electronics and mobile products.

USB sends data in serial mode i.e. the parallel data is serialized before sends and de-serialized after receiving. The benefits of USB are low cost, expandability, auto-configuration, hot-plugging and outstanding performance. It also provides power to the bus, enabling many peripherals to operate without the added need for an AC powe adapter.

Various versions USB:

As USB technology advanced the new version of USB are unveiled with time. Let us now try to understand more about the different versions of the USB.

USB1.0: Version 0.7 of the USB interface definition was released in November 1994. But USB 1.0 is the original release of USB having the capability of transferring 12 Mbps, supporting up to 127 devices. And as we know it was a combined effort of some large players on the market to define a new general device interface for computers. This USB 1.0 specification model was introduced in January1996. The data transfer rate of this version can accommodate a wide range of devices, including MPEG video devices, data gloves, and digitizers. This version of USB is known as full-speed USB.

Since October-1996, the Windows operating systems have been equipped with USB drivers or special software designed to work with specific I/O device types. USB got integrated into Windows 98 and later versions. Today, most new computers and peripheral devices are equipped with USB.

USB1.1: USB 1.1 came out in September 1998 to help rectify the adoption problems that occurred with earlier versions, mostly those relating to hubs.

USB 1.1 is also known as full-speed USB. This version is similar to the original release of USB; however, there are minor modifications for the hardware and the specifications. USB version 1.1 supported two speeds, a full speed mode of 12Mbits/s and a low speed mode of 1.5Mbits/s. The 1.5Mbits/s mode is slower and less susceptible to EMI, thus reducing the cost of ferrite beads and quality components.

USB2.0: Hewlett-Packard, Intel, LSI Corporation, Microsoft, NEC, and Philips jointly led the initiative to develop a higher data transfer rate than the 1.1 specifications. The USB 2.0 specification was released in April 2000 and was standardized at the end of 2001. This standardization of the new device-specification made backward compatibility possible, meaning it is also capable of supporting USB 1.0 and 1.1 devices and cables.

Supporting three speed modes (1.5, 12 and 480 megabits per second), USB 2.0 supports low-bandwidth devices such as keyboards and mice, as well as high-bandwidth ones like high-resolution Web-cams, scanners, printers and high-capacity storage systems.

USB 2.0, also known as hi-speed USB. This hi-speed USB is capable of supporting a transfer rate of up to 480 Mbps, compared to 12 Mbps of USB 1.1. That's about 40 times as fast! Wow!

USB3.0: USB 3.0 is the latest version of USB release. It is also called as Super-Spee USB having a data transfer rate of 4.8 Gbit/s (600 MB/s). That means it can deliver over 10x the speed of today's Hi-Speed USB connections.

The USB 3.0 specification was released by Intel and its partners in August 2008. Products using the 3.0 specifications are likely to arrive in 2009 or 2010. The technology targets fast PC sync-and-go transfer of applications, to meet the demands of Consumer Electronics and mobile segments focused on high-density digital content and media.

USB 3.0 is also a backward-compatible standard with the same plug and play and other capabilities of previous USB technologies. The technology draws from the same architecture of wired USB. In addition, the USB 3.0 specification will be optimized for low power and improved protocol efficiency.

USB system overview:

The USB system is made up of a host, multiple numbers of USB ports, and multiple peripheral devices connected in a tiered-star topology. To expand the number of USB ports, the USB hubs can be included in the tiers, allowing branching into a tree structure with up to five tier levels.

The tiered star topology has some benefits. Firstly power to each device can be monitored and even switched off if an over current condition occurs without disrupting other USB devices. Both high, full and low speed devices can be supported, with the hub filtering out high speed and full speed transactions so lower speed devices do not receive them.

The USB is actually an addressable bus system, with a seven-bit address code. So it can support up to 127 different devices or nodes at once (the "all zeroes" code is not a valid address). However it can have only one host: the PC itself. So a PC with its peripherals connected via the USB forms a star local area network (LAN).

On the other hand any device connected to the USB can have a number of other nodes connected to it in daisy-chain fashion, so it can also form the hub for a mini-star sub-network. Similarly it is possible to have a device, which purely functions as a hub for other node devices, with no separate function of its own. This expansion via hubs is possible because the USB supports a tiered star topology. Each USB hub acts as a kind of traffic cop. for its part of the network, routing data from the host to its correct address and preventing bus contention clashes between devices trying to send data at the same time.

On a USB hub device, the single port used to connect to the host PC either directly or via another hub is known as the upstream port, while the ports used for connecting other devices to the USB are known as the downstream ports. USB hubs work transparently as far as the host PC and its operating system are concerned. Most hubs provide either four or seven downstream ports or less if they already include a USB device of their own.

The host is the USB system's master, and as such, controls and schedules a communications activities. Peripherals, the devices controlled by USB, are slaves responding to commands from the host. USB devices are linked in series through hubs. There always exists one hub known as the root hub, which is built in to the host controller.

A physical USB device may consist of several logical sub-devices that are referred to as device functions. A single device may provide several functions, for example, a web-cam (video device function) with a built-in microphone (audio device function). In short, the USB specification recognizes two kinds of peripherals: stand-alone (single function units, like a mouse) or compound devices like video camera with separate audio processor. The logical channel connection host to peripheral-end is called pipes in USB. A USB device can have 16 pipes coming into the host controller and 16 going out of the controller.

The pipes are unidirectional. Each interface is associated with single device function and is formed by grouping endpoints.

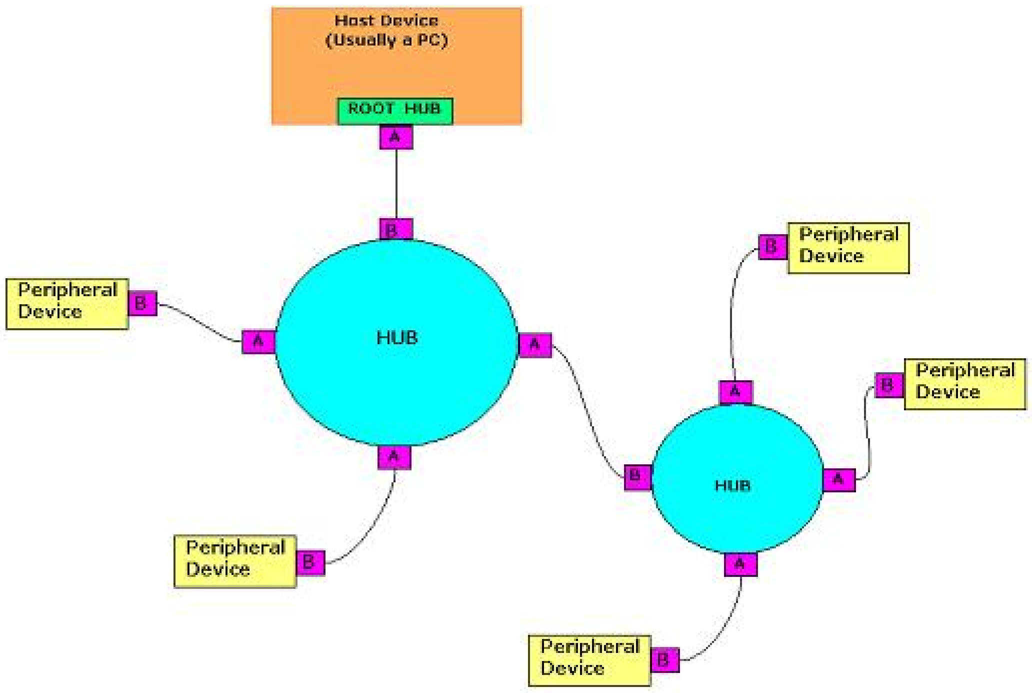


Fig2: The USB "tiered star" topology

The hubs are bridges. They expand the logical and physical fan-out of the network. A hub has a single upstream connection (that going to the root hub, or the next hub closer to the root), and one to many downstream connections.

Hubs themselves are considered as USB devices, and may incorporate some amount intelligence. We know that in USB users may connect and remove peripherals without powering the entire system down. Hubs detect these topology changes. They also source power to the USB network. The power can come from the hub itself (if it has a built-in power supply), or can be passed through from an upstream hub.

USB connectors & the power supply:
Connecting a USB device to a computer is very simple -- you find the USB connector on the back of your machine and plug the USB connector into it. If it is a new device, the operating system auto-detects it and asks for the driver disk. If the device has already been installed, the computer activates it and starts talking to it.

The USB standard specifies two kinds of cables and connectors. The USB cable will usually have an "A" connector on one end and a "B" on the other. That means the USB devices will have an "A" connection on it. If not, then the device has a socket on it that accepts a USB "B" connector.

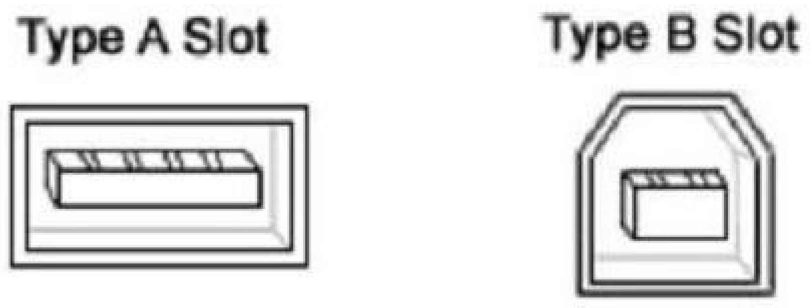


Fig 3: USB Type A & B Connectors

The USB standard uses "A" and "B" connectors mainly to avoid confusion:

1. "A" connectors head "upstream" toward the computer.
2. "B" connectors head "downstream" and connect to individual devices.

Q6 (b) Briefly explain asynchronous and synchronous DRAMs.

Answer Page Number 299 and 303 of Text Book

Q7 (a) Design a fast adder. What are the variations in a fast adder?

Answer

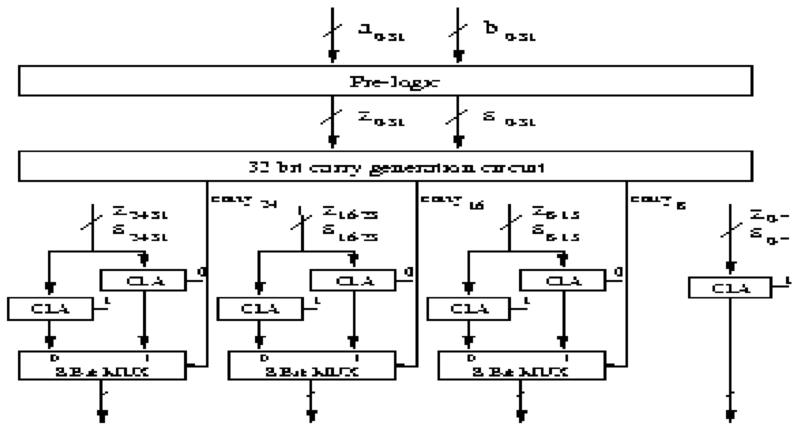


Fig. 1: The hybrid carry lookahead/carry select logic

Q7 (b) Explain how the virtual address is converted into real address in a paged virtual memory system. Give an example.

Answer

In a virtual memory system, the program memory is divided into fixed sized pages and allocated in fixed sized physical memory frames. The pages do not have to be contiguous in memory. A page table keeps track of where each page is located in physical memory. This allows the operating system to load a program of any size into any available frames. Only the currently used pages need to be loaded. Unused pages can remain on disk until they are referenced. This allows many large programs to be executed on a relatively small memory system. A *resident* flag in the page table indicates whether or not the page is in memory. The page table also includes several other flags to keep track of memory usage. A *use* flag is set whenever the page is referenced. A *dirty* bit is set whenever the page is changed to inform the operating system that the page in memory is different than the page on disk.

There are several virtual memory parameters set by a system designer:

Maximum Virtual Address space	The size of a program address is determined by the maximum size of the virtual address space. The number of bits in a virtual address is the log base 2 of this value.
Maximum Physical Address space	The amount of real memory that the system can support determined the number of bits needed to address the physical memory. The size of a physical address is log base 2 of this value.
Size of a page	This is the size of a virtual memory page and a physical memory frame. It is always a power of 2.

The addresses that appear in programs are the virtual addresses or program addresses. Fo every memory access, either to fetch an instruction or data, the CPU must translate the virtual address to a real physical address. A virtual memory address can be considered to be composed of two parts: a page number and an offset into the page. The page number determines which page contains the information and the offset specifies which byte within the page. The size of the offset field is the log base 2 of the size of a page.

Consider an example system with:

16MB Maximum Virtual Address space (24 bits)

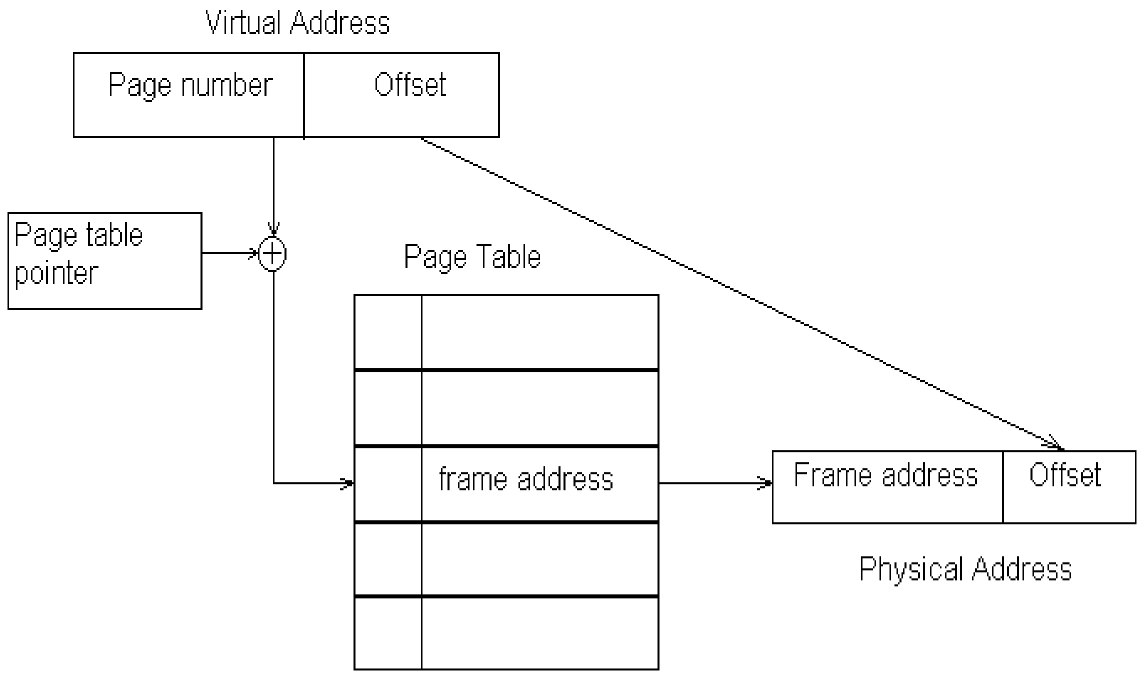
8MB Maximum Physical Address space (23 bits)

1024byte Page size (10 bits)

The virtual addresses can be represented as

13 bits	10 bits
page number	offset

To convert a virtual address into a physical address, the CPU uses the page number as an index into the page table. If the page is resident, the physical frame address in the page table is concatenated in front of the offset to create the physical address.



Q8 (a) Give the algorithm for multiplication of signed 2's complement numbers and illustrate with an example.

Answer

Assume $n = 7$ bits available. Multiply $B = 22 = (0010110)_2$ by $A = -34 = -(0100010)_2$. First represent both operands and their negation in signed 2's complement:

$$\begin{array}{llll} 22: & 0010110, & -22: & 1101010 \\ 34: & 0100010, & -34: & 1011110 \end{array}$$

Then carry out the multiplication in the hardware:

		[M]	0010110			
q_iq_{i-1}	Action	[A]	0000000	[Q]	1011110	0
00	right shift		0000000		0101111	0
10	-B	+	1101010			
			1101010		0101111	0
	right shift		1110101		0010111	1
11	right shift		1111010		1001011	1
11	right shift		1111101		0100101	1
11	right shift		1111110		1010010	1
01	+B	+	0010110			
			0010100		1010010	1
	right shift		0001010		0101001	0
10	-B	+	1101010			
			1110100		0101001	0
	right shift		1111010		0010100	1

The upper half of the final result $1111010 \quad 0010100$ is in register [A] while the lower half is in register [Q]. The product is given in signed 2's complement and its actual value is negative of the 2's complement:

$$B \times A = -\overline{11110100010100} = -00001011101100 = -748_{10}$$

Also note that:

- As the operands are in signed 2's complement form, the *arithmetic shift* is used for the right shifts above, i.e., the MSB bit (sign bit) is always repeated while all other bits are shifted to the right. This guarantees the proper sign extension fo both positive and negative values represented in signed 2's complement.

- When the multiplicand is negative represented by signed 2's complement, it needs to be complemented again for subtraction (when the LSB of multiplier is 1 and the extra bit is 0, i.e., the beginning of a string of 1's).

Q8 (b) Explain the representation of floating point numbers in detail. Give the IEEE standard double precision floating point format.

Answer
Decimal Cases

- $3.141592653589\cdots$
- $2.71828\cdots$

  6.023×10^{23} N_A
- ( )

  6.626×10^{-32}
- ($\hbar$)

In programming, a floating point number -123.45×10^{-6} is expressed as $-123.45E-6$. In general, a floating-point number can be written as

$$\pm M \times B^{E}$$

where

- M is the fraction *mantissa* or *significand*.
- E is the exponent.
- B is the base, in decimal case $B = 10$.

Binary Cases

As an example, a 32-bit word is used in MIPS computer to represent a floating-point number:

S	E	M

1 bit..... 8 bits............. 23 bits

$$(-1)^{S} \times M \times 2^{E}$$

representing:

- The *implied* base is 2 (not explicitly shown in the representation).
- The exponent can be represented in signed 2's complement (but also see biased notation later).
- The *implied* decimal point is between the exponent field E and the significant field M.
- More bits in field E mean larger range of values represent able.
- More bits in field M mean higher precision.
- Zero is represented by all bits equal to 0: $0000 \cdots 000_2$

Normalization

To efficiently use the bits available for the significand, it is shifted to the left until all leading 0's disappear (as they make no contribution to the precision). The value can be kept unchanged by adjusting the exponent accordingly.

Moreover, as the MSB of the significant is always 1, it does not need to be shown explicitly. The significant could be further shifted to the left by 1 bit to gain one more bit for precision. The first bit 1 before the decimal point is implicit. The actual value represented is

$$(-1)^S \times (1. + M) \times 2^E$$

However, to avoid possible confusion, in the following the default normalization does not assume this implicit 1 unless otherwise specified.

Zero is represented by all 0's and is not (and cannot be) normalized.

Example: A binary number $x = 0.0001101001101$ can be represented in 14-bit floating-point form in the following ways (1 sign bit, a 4-bit exponent field and a 9-bit significant field):

- $x = 0.0001101001101 \times 2^0 \; \boxed{0} \; \boxed{0000} \; \boxed{000110100}$
- $x = 0.001101001101 \times 2^{-1} \; \boxed{0} \; \boxed{1111} \; \boxed{001101001}$
- $x = 0.01101001101 \times 2^{-2} \; \boxed{0} \; \boxed{1110} \; \boxed{011010011}$
- $x = 0.1101001101 \times 2^{-3} \; \boxed{0} \; \boxed{1101} \; \boxed{110100110}$
- $x = 1.101001101 \times 2^{-4} \; \boxed{0} \; \boxed{1100} \; \boxed{101001101}$ with an implied 1.0:

By normalization, highest precision can be achieved.

Q9 (a) What are the advantages and disadvantages of hardwired and micro programmed control?

Answer

A hardwired control unit has a processor that generates signals or instructions to be implemented in correct sequence. This was the older method of control that works through the use of distinct components, drums, a sequential circuit design, or flip chips. It is implemented using logic gates & flip flops. It is faster, less flexible & limited in complexity

A micro programmed control unit on the other hand makes use of a micro sequencer from which instruction bits are decoded to be implemented. It acts as the device supervisor that controls the rest of the subsystems including arithmetic and logic units, registers, instruction registers, off-chip input/output, and buses.

It is slower, more flexible & greater complexity

The Hard-Wired Control Unit

Figure 2 is a block diagram showing the internal organization of a hard-wired control unit for our simple computer. Input to the controller consists of the 4-bit opcode of the instruction currently contained in the Instruction Register and the negative flag from the accumulator. The controller's output is a set of 16 control signals that go out to the various registers and to the memory of the computer, in addition to a HLT signal that is activated whenever the leading bit of the op-code is one. The controller is composed of the following functional units: A ring counter, an instruction decoder, and a control matrix.

The ring counter provides a sequence of six consecutive active signals that cycle continuously. Synchronized by the system clock, the ring counter first activates its T0 line, then its T1 line, and so forth. After T5 is active, the sequence begins again with T0. Figure 3 shows how the ring counter might be organized internally.

The instruction decoder takes its four-bit input from the op-code field of the instruction register and activates one and only one of its 8 output lines. Each line corresponds to one of the instructions in the computer's instruction set. Figure 4 shows the internal organization of this decoder.

The most important part of the hard-wired controller is the control matrix. It receives input from the ring counter and the instruction decoder and provides the proper sequence of control signals. Figure 5 is a diagram of how the control matrix for our simple machine might be wired. To understand how this diagram was obtained, we must look carefully at the machine's instruction set (Table 1). Table 2 shows which control signals must be active at each ring counter pulse for each of the instructions in the computer's instruction set (and for the instruction fetch operation). The table was prepared by simply writing down the instructions in the left-hand column. (In the circuit these will be the output lines from the decoder). The various control signals are placed horizontally along the top of th table. Entries into the table consist of the moments (ring counter pulses T0, T1, T2, T3, T4, or T5) at which each control signal must be active in order to have the instruction executed. This table is prepared very easily by reading off the information for each instruction given in Table 1. For example, the Fetch operation has the EP and LM control signals active at ring count 1, and ED, LI, and IPC active at ring count 2. Therefore the first row (Fetch) of Table 2 has T0 entered below EP and LM, T1 below R, and T2 below IP, ED, and LI.

Once Table 2 has been prepared, the logic required for each control signal is easily obtained. For each an AND operation is performed between any active ring counter (Ti) signals that were entered into the signal's column and the corresponding instruction contained in the far left-hand column. If a column has more than one entry, the output of the ANDs are ORed together to produce the final control signal. For example, the LM column has the following entries: T0 (Fetch), T3 associated with the LDA instruction, and T3 associated with the STA instruction. Therefore, the logic for this signal is:

LM = T0 + T3*LDA + T3*STA

This means that control signal LM will be activated whenever any of the following conditions is satisfied: (1) ring pulse T0 (first step of an instruction fetch) is active, or (2) an LDA instruction is in the IR and the ring counter is issuing pulse 3, or (3) and STA instruction is in the IR and the ring counter is issuing pulse 3.

The entries in the JN (Jump Negative) row of this table require some further explanation. The LP and EI signals are active during T3 for this instruction if and only if the accumulator's negative flag has been set. Therefore the entries that appear above these signals for the JN instruction are T3*NF, meaning that the state of the negative flag must be ANDed in for the LP and EI control signals.

Figure 6 gives the logical equations required for each of the control signals used on our machine. These equations have been read from Table 2, as explained above. The circuit diagram of the control matrix (Figure 5) is constructed directly from these equations.

It should be noticed that the HLT line from the instruction decoder does not enter the control matrix, Instead this signal goes directly to circuitry (not shown) that will stop the clock and thus terminate execution.

A Micro-programmed Control Unit

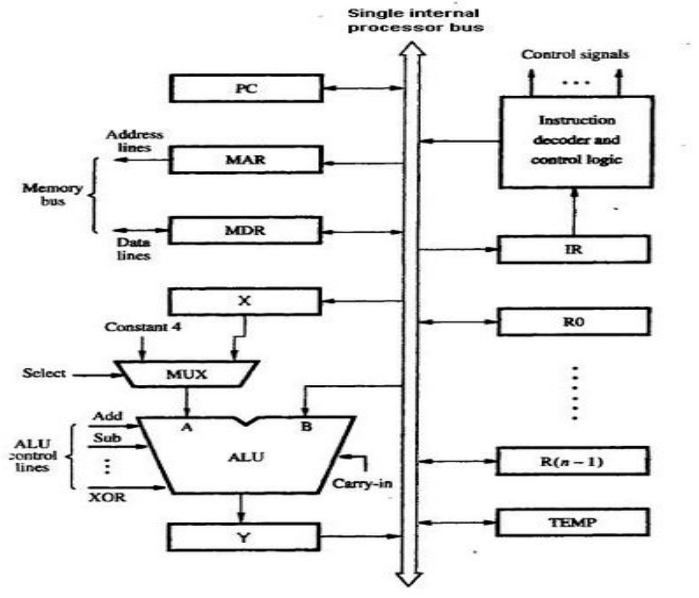
As we have seen, the controller causes instructions to be executed by issuing a specific set of control signals at each beat of the system clock. Each set of control signals issued causes one basic operation (micro-operation), such as a register transfer, to occur within the data path section of the computer. In the case of a hard-wired control unit the control matrix is responsible for sending out the required sequence of signals.

An alternative way of generating the control signals is that of micro-programmed contro In order to understand this method it is convenient to think of sets of control signals that cause specific micro-operations to occur as being "microinstructions" that could be stored in a memory. Each bit of a microinstruction might correspond to one control signal. If the bit is set it means that the control signal will be active; if cleared the signal will be inactive. Sequences of microinstructions could be stored in an internal "control" memory. Execution of a machine language instruction could then be caused by fetching the proper sequence of microinstructions from the control memory and sending them out to the data path section of the computer. A sequence of microinstructions that implements an instruction on the external computer is known as a micro-routine. The instruction set of the computer is thus determined by the set of micro-routines, the "micro program," stored in the controller's memory. The control unit of a micro program-controlled computer is essentially a computer within a computer.

Q9 (b) Draw neat diagram of single bus organization of CPU showing ALU, all types of registers and the data paths among them. Compare it with multiple bus organization of CPU.

Answer

A single 12-bit-wide bus provides for exchange of information between pairs of registers within the data path section. The registers and the 256 X 12 bit RAM memory are controlled by 16 control signals. Most of the registers have Load (L) and Enabled (E) signals. An active L signal to a register causes the contents of the bus to be clocked into that register on the next rising pulse from the system clock. An active E signal enables the tristate outputs of the register, thereby making its contents available to the bus. Therefore, a register transfer from, for example, register A to register B would require active EA and LB control signals.



Single Bus Organization

Processing of data is done by the Arithmetic-Logic-Unit (ALU), a circuit that is capab of adding or subtracting the 12-bit numbers contained in its two input registers: the accumulator (ACC) and register B. The operation performed by the ALU is selected by the Add (A) or Subtract (S) control signals. The accumulator also contains a single flip-flop that is set whenever its contents are negative (i.e., whenever the leading bit is set--meaning a negative 2's complement number). The value of this "negative flag" provides input to the controller/sequencer, and, as we shall see, permits implementation of conditional branching instructions.

The machine's RAM memory is accessed by first placing the 8-bit address in the Memory Address Register (MAR). An active Read (R) control signal to the RAM will then cause the selected word from the RAM to appear in the Memory Data Register (MDR). An active Write (W) signal, on the other hand, will cause the word contained in the MDR to be stored in the RAM at the address specified by the MAR. Since there are no input or output ports in this simple computer, all I/O is memory mapped. In other words, several memory locations are reserved for input/output devices. Memory reads from any of those locations will cause data from the corresponding input device to appear in the MDR; memory writes to them will cause data in the MDR to be sent to the corresponding output device. A word stored in any given memory location may be data to be manipulated by the computer or a coded instruction that specifies an action to be taken.

The data path section also contains a Program Counter (PC) whose function it is to point to the address in RAM of the next instruction to be executed. The Increment Program Counter (IP) control signal causes the contents of the PC to increase by one. Since, as we shall see, instructions on this machine are one word long, this provides a simple mechanism for sequential instruction execution. In addition there is an Instruction Register (IR) which holds the instruction that is about to be execute and provides its opcode to the controller/sequencer.

<u>Text Book</u>

Computer Organization, Carl Hamacher, Zvonko Vranesic, Safwat Zaky, 5th Edition, TMH, 2002