Q.2 a. Explain in detail the Processor technology?

Answer:

Embedded systems contain processing cores that are either microcontrollers digital signal (DSP). A processor is an important unit in the embedded system hardware. It is the heart of the embedded system.

The key characteristic, however, is being dedicated to handle a particular task. Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Physically, embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

Embedded processors can be broken into two broad categories. Ordinary microprocessors (μP) use separate integrated circuits for memory and peripherals. Microcontrollers (μC) have many more peripherals on chip, reducing power consumption, size and cost. In contrast to the personal computer market, many different basic CPU architectures are used, since software is custom-developed for an application and is not a commodity product installed by the end user. Both Von Neumann as well as various degrees of Harvard architectures are used. RISC as well as non-RISC processors are found. Word lengths vary from 4-bit to 64-bits and beyond, although the most typical remain 8/16-bit. Most architectures come in a large number of different variants and shapes, many of which are also manufactured by several different companies.

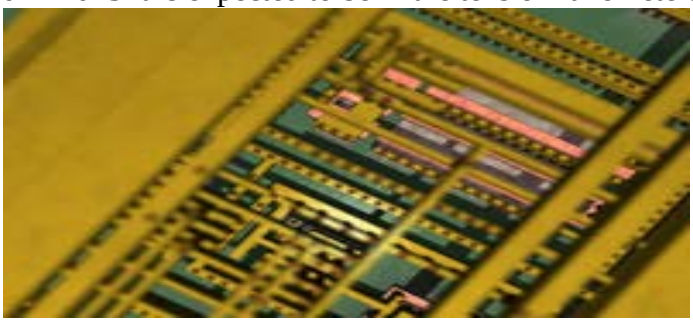
b. Compare Processor Technology with IC technology, give three examples.

Answer:

An **integrated circuit** or **monolithic integrated circuit** (also referred to as an **IC**, a **chip**, or a **microchip**) is a set of electronic circuits on one small plate ("chip") of semi conductor material, normally silicon. This can be made much smaller than a discrete circuit made from independent components.

Integrated circuits are used in virtually all electronic equipment today and have revolutionized the world of electronics. computers, mobile phones, and other digital home appliances now inextricable parts of the structure of modern societies, made possible by the low cost of producing integrated circuits.

ICs can be made very compact, having up to several billion transistors and other electronics an area the size of a fingernail. The width of each conducting line in a circuit can be made smaller and smaller as the technology advances, in 2008 it dropped below 100nanometers and in 2013 it is expected to be in the tens of nanometers.



Synthetic detail of an integrated circuit through four layers of planarized copper interconnect, down to the polysilicon (pink), wells (greyish), and substrate (green)
ICs were made possible by experimental discoveries showing that semiconductor could perform the functions of vacuum tubes and by mid-20th-century technology advancements in semiconductor device fabrication. The integration of large numbers of tiny transistors into a small chip was an enormous improvement over the manual assembly of circuits using discrete electronic components. The integrated circuit's mass capability, reliability, and building-block approach to circuit design ensured the rapid adoption of standardized Integrated Circuits in place of designs using discrete transistors.
There are two main advantages of ICs over discrete circuits: cost and performance. Cost is low because the chips, with all their components, are printed as a unit by photoliography rather than being constructed one transistor at a time. Furthermore, much less material is used to construct a packaged IC die than to construct a discrete circuit. Performance is high because the components switch quickly and consume little power (compared to their discrete counterparts) as a result of the small size and close proximity of the components. As of 2012, typical chip areas range from a few square millimeters to around 450 mm2, with up to 9 million transistors per mm2.

Q.3 a. What is the combinational logic?

Answer:

In digital circuit theory, combinational logic (sometimes also referred to as combinatorial logic or time-independent logic) is a type of digital logic which is implemented by boolean circuits, where the output is a pure function of the present input only. This is in contrast to sequential logic, in which the output depends not only on the present input but also on the history of the input. In other words, sequential logic has memory while combinational logic does not.
Combinational logic is used in computer circuits to do boolean algebra on input signals and on stored data. Practical computer circuits normally contain a mixture of combinational and sequential logic. For example, the part of an arithmetic logic unit, or ALU, that does mathematical calculations is constructed using combinational logic. Other circuits used in computers, such as half adders, full adders, half subtractors, full subtractors, multiplexers, demultiplexers, encoders and decoders are also made by using combinational logic.

b. Explain optimizing of data path & FSM

Answer:

• Unlike combinational logic circuits, the output of sequential logic circuits not only depends on current inputs but also on the past sequence of inputs. • Sequential circuits are constructed using combinational logic and a number of memory elements with some or all of the memory outputs fed back into the combinational logic forming a feedback path or loop.
• A very simple sequential circuit with no inputs created using inverters to form a feedback loop.
Sequential circuit = Combinational logic + Memory Elements
A state variable in a sequential circuit represents the single-bit variable Q stored in a memory element in circuit.

– Each memory element may be in state 0 or state 1 depending on the current value stored in the memory element.
• The State of A sequential Circuit:
– The collection of all state variables (memory element stored values) that at any time contain all the information about the past necessary to account for the circuit's future behavior.
– A sequential circuit that contains n memory elements could be in one of a maximum of 2n states at any given time depending on the stored values in the memory elements.
– Sequential Circuit State transition: A change in the stored values in memory elements thus changing the sequential circuit from one state to another.

Q.4 a. Explain in detail with the help of diagram processor architecture.
Answer:
All computers run using very low-level commands which do some very basic functions, such as reading data, writing data, jumping to addresses, and calculating basic arithmetic. (The complete list of commands that can be run by a CPU is known as that computer's instruction set). Instruction sets are relatively small; most higher-order programming languages, such as C++, Ada, Fortran, or Visual Basic, must be compiled (or translated, or interpreted) into these low level commands in order for a program to run.
These low-level commands are run in a series of steps, which are synchronized with the computer's clock. (One apt analogy would be an internal combustion engine. In an engine, the pistons, valves, and fuel systems must all run in a very synchronized manner, so likewise a computer runs – with precise timing dictating when instructions are fetched and executed, and when data is read and written. If an engine runs in a cycle: intake, compression, ignition, and exhaust; so does a computer's CPU: fetch, load, execute, write.)
CPU architects strive for designs that are compact and efficient, thus forcing many tradeoffs to be considered during design. A 32-bit architecture can move more data than a 16-bit architecture in each cycle (thereby making it faster), but the data bus is also twice as wide, which takes up more area on the limited space of a chip. Despite these challenges, continual advances in VLSI design have made it possible for computer processors to steadily grow exponentially more powerful over the past few decades.

The 8051 architecture provides many functions (CPU, RAM, ROM, I/O, interrupt logic, timer, etc.) in a single package

- 8-bit ALU, Accumulator, 8-bit Registers and 8-bit data bus; hence it is an 8-bit microcontroller
- Boolean processor
- Multiply, divide and compare instructions
- 4 register banks (memory mapped)
- Fast interrupt with register bank switching
- Interrupts with selectable priority
- Dual 16-bit address bus – It can access 2 x 216 memory locations – 64 kB (65536 locations) each of RAM and ROM
- On-chip RAM – 128 bytes (data memory)
- On-chip ROM – 4 kB (program memory)

- Four byte bi-directional input/output port
- UART (serial port)
- Two 16-bit Counter/timers
- Power saving mode (on some derivatives)

b. What is the selection criteria for microprocessors?

Answer:

A microprocessor-based controller is a package of control loops and functions into which a number of inputs are connected, and from which a number of outputs are delivered to the final control devices. The microprocessor evaluates the incoming primary variables and incorporates them into certain computations that then determine the magnitude and direction of change to the manipulated control device.

For example, the signal(s) from the flue gas analyzer(s) are connected to the inputs of the controller where they interface with calculated variables to produce the trim action required to achieve the optimum fuel/air ratio.

Today's technology allows for many control advantages to be obtained by using the microprocessor based controller. Different manufacturers offer different options with their controllers. Items to consider when selecting a new controller include:

1. boiler size
2. number of boilers to be operated
3. normal loads and load variations
4. rate of load change

a. type of fuels

5. existing measurement devices
6. final control elements.
7. Also important are the interactions between the individual loops of the controller:
8. bumpless transfers between the individual controlling loops
9. bumpless transfers between auto/manual modes
10. verification of control signals
11. isolation of input signals
12. fault diagnostics
13. fail-safe action when fault occurs
14. adaptability to changing conditions

Q.5 a. Explain the working of watch dog timers.

Answer:

A watchdog timer (sometimes called a computer operating properly or COP timer, or simply a watchdog) is an electronic timer that is used to detect and recover from computer malfunctions. During normal operation, the computer regularly restarts the watchdog timer to prevent it from elapsing, or "timing out". If, due to a hardware fault or program error, the computer fails to restart the watchdog, the timer will elapse and generate a timeout signal. The timeout signal is used to initiate corrective action or actions. The corrective actions typically include placing the computer system in a safe state and restoring normal system operation.

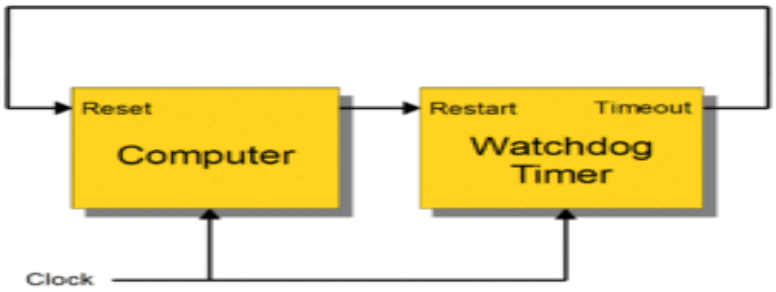
Watchdog timers are commonly found in computer-controlled equipment where humans cannot easily access the equipment or would be unable to react to faults in a timely

manner. Many embedded systems cannot depend on a human to reboot them if the software hangs; they must be self-reliant. For example, remote embedded systems such as space probes are not physically accessible to human operators; these could become permanently disabled if they were unable to autonomously recover from faults. A watchdog timer is usually employed in cases like these.

Watchdog timers may also be used when running untrusted code in a sandbox, to limit the CPU time available to the code and thus prevent some types of denial-of-service attacks.

Architecture and operation

Watchdog timers come in many configurations, and many allow their configurations to be altered. Microcontrollers often include an integrated, on-chip watchdog. In other computers the watchdog may reside in a nearby chip that connects directly to the CPU, or it may be located on an external expansion card in the computer's chassis. The watchdog and CPU may share a common clock signal, as shown in the block diagram below, or they may have independent clock signals.



The act of restarting the watchdog timer is commonly referred to as "kicking the dog" or other similar terms; this is typically done by writing to a watchdog control port. Alternatively, in microcontrollers that have an integrated watchdog timer, the watchdog is sometimes kicked by executing a special machine language instruction. An example of this is the CLRWDT (clear watchdog timer) instruction found in the instruction set of some PIC microcontrollers.

b. What are the LCD controllers?

Answer:

These LCD screens are limited to monochrome text and are often used in copiers,fax machines, laser printers, industrial test equipment, networking equipment such as routers and storage devices.

The screens come in a small number of standard configurations. Common sizes are 8x1 (one row of eight characters), 16×2, 20×2 and 20×4. Larger custom sizes are made with 32, 40 and 80 characters and with 1, 2, 4 or 8 lines. The most commonly manufactured larger configuration is 40×4 characters, which requires two individually addressable HD44780 controllers with expansion chips as a single HD44780 chip can only address up to 80 characters. A common smaller size is 16×2, and this size is readily available as surplus stock for hobbyist and prototyping work.

Character LCDs can come with or without backlights, which may be LED, fluorescent, or electroluminescent.

Character LCDs use a standard 16 contact interface, commonly using pins or card edge connections on 0.1 inch / 2.54mm centers. Those without backlights may have only 14 pins, omitting the final two pins powering the light. The pinout is as follows:

Amber backlight on a HD44780 display

1. Ground
2. VCC (+3.3 to +5V)
3. Contrast adjustment (VO)
4. Register Select (RS). RS=0: Command, RS=1: Data
5. Read/Write (R/W). R/W=0: Write, R/W=1: Read
6. Clock (Enable). Falling edge triggered
7. Bit 0 (Not used in 4-bit operation)
8. Bit 1 (Not used in 4-bit operation)
9. Bit 2 (Not used in 4-bit operation)
10. Bit 3 (Not used in 4-bit operation)
11. Bit 4
12. Bit 5
13. Bit 6
14. Bit 7
15. Backlight Anode (+)
16. Backlight Cathode (-)

The nominal operating voltage for LED backlights is 5V at full brightness, with dimming at lower voltages dependent on the details such as LED color. Non-LED backlights often require higher voltages.

Q.6 a. What are the common memory types? Explain in detail.

Answer:

Volatile Memory

A primary distinction in memory types is volatility. Volatile memories only hold their contents while power is applied to the memory device. Examples of volatile memories include static RAM (SRAM), synchronous static RAM (SSRAM), synchronous dynamic RAM (SDRAM), and FPGA on-chip memory.

Non-Volatile Memory

Non-volatile memories retain their contents when power is switched off, making them good choices for storing information that must be retrieved after a system power-cycle. Processor boot-code, persistent application settings, and FPGA configurationdataaretypicallystoredinnon-volatilememory.Althoughnon-volatile

memory has the advantage of retaining its data when power is removed, it is typically much slower to write to than volatile memory, and often has more complex writing and erasing procedures. Non-volatile memory is also usually only guaranteed to be erasable a given number of times, after which it may fail. Examples of non-volatile memories include all types of flash, EPROM and EEPROM.

Most modern embedded systems use some type of flash memory for non-volatile storage. Many embedded applications require both volatile and non-volatile memories because the two memory types serve unique and exclusive purposes. The following sections discuss the use of specific types of memory in embedded systems.

On-Chip Memory

On-chip memory is the simplest type of memory for use in an FPGA-based embedded system. The memory is implemented in the FPGA itself; consequently, no external connections are necessary on the circuit board. To implement on-chip memory in your design, simply select On-Chip Memory from the Component Library on the System Contents tab in SOPC Builder. You can then specify the size, width, and type of on-chip memory, as well as special on-chip memory features such as dual-port access.

ED51008-1.2

Advantages

On-chip memory is the highest throughput, lowest latency memory possible in an FPGA-based embedded system. It typically has a latency of only one clock cycle.

Memory transactions can be pipelined, making a throughput of one transaction per clock cycle typical.

Some variations of on-chip memory can be accessed in dual-port mode, with separate ports for read and write transactions. Dual-port mode effectively doubles the potential bandwidth of the memory, allowing the memory to be written over one port, while simultaneously being read over the second port.

Another advantage of on-chip memory is that it requires no additional board space or circuit-board wiring because it is implemented on the FPGA directly. Using on-chip memory can often save development time and cost.

Finally, some variations of on-chip memory can be automatically initialized with custom content during FPGA configuration. This memory is useful for holding small bits of boot code or LUT data which needs to be present at reset.

Disadvantages

While on-chip memory is very fast, it is somewhat limited in capacity. The amount of on-chip memory available on an FPGA depends solely on the particular FPGA device being used, but capacities range from around 15 K Bytes in the smallest Cyclone II device to just under 2 M Bytes in the largest Stratix III device.

Because most on-chip memory is volatile, it loses its contents when power is disconnected. However, some types of on-chip memory can be initialized automatically when the FPGA is configured, essentially providing a kind of non-volatile function.

Best Applications
The following sections describe the best uses of on-chip memory.

Cache
Because it is low latency, on-chip memory functions very well as cache memory for microprocessors. The Nios II processor uses on-chip memory for its instruction and data caches. The limited capacity of on-chip memory is usually not an issue for caches because they are typically relatively small.

Preliminary
Tightly Coupled Memory
The low latency access of on-chip memory also makes it suitable for tightly coupled memories. Tightly coupled memories are memories which are mapped in the normal address space, but have a dedicated interface to the microprocessor, and possess the high speed, low latency properties of cache memory.

Look Up Tables
For some software programming functions, particularly mathematical functions, it is sometimes fastest to use a LUT to store all the possible outcomes of a function, rather than computing the function in software. On-chip memories work well for this purpose as long as the number of possible outcomes fits reasonably in the capacity of on-chip memory available.

FIFO
Embedded systems often need to regulate the flow of data from one system block to another.

Poor Applications
On-chip memory is poorly suited for applications which require large memory capacity. Because on-chip memory is relatively limited in capacity, avoid using it to store large amounts of data; however, some tasks can take better advantage of on-chip memory than others. If your application utilizes multiple small blocks of data, and not all of them fit in on-chip memory, you should carefully consider which blocks to implement in on-chip memory. If high system performance is your goal, place the data which is accessed most often in on-chip memory.

On-Chip Memory Types
Depending on the type of FPGA you are using, several types of on-chip memory are available. For details on the different types of on-chip memory available to you, refer to the device handbook for the particular FPGA family you are using.

b. Explain with the help of diagram the concept of cache memory.

Answer:

The cache is a small mirror-image of a portion (several "lines") of main memory.

- cache is faster than main memory ==> so we must maximize its utilization
- cache is more expensive than main memory ==> so it is much smaller

How do we keep that portion of the current program in cache which maximizes cache utilization.

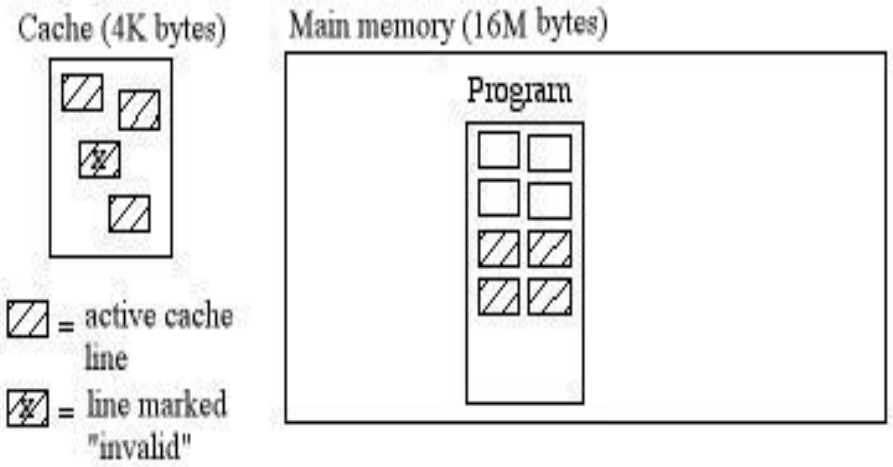
Locality of reference

The principle that the instruction currently being fetched/executed is very close in memory to the instruction to be fetched/executed next. The same idea applies to the data value currently being accessed (read/written) in memory.

If we keep the most active segments of program and data in the cache, overall execution speed for the program will be optimized. Our strategy for cache utilization should maximize the number of cache read/write operations, in comparison with the number of main memory read/write operations.

Example

A line is an adjacent series of bytes in main memory (that is, their addresses are contiguous). Suppose a line is 16 bytes in size. For example, suppose we have a 212 = 4K-byte cache with 28 = 256 16-byte lines; a 224 = 16M-byte main memory, which is 212 = 4K times the size of the cache; and a 400-line program, which will not all fit into the cache at once.



Each active cache line is established as a copy of a corresponding memory line during execution. Whenever a memory write takes place in the cache, the "Valid" bit is reset (marking that line "Invalid"), which means that it is no longer an exact image of its corresponding line in memory.

Cache Dynamics

When a memory read (or fetch) is issued by the CPU:

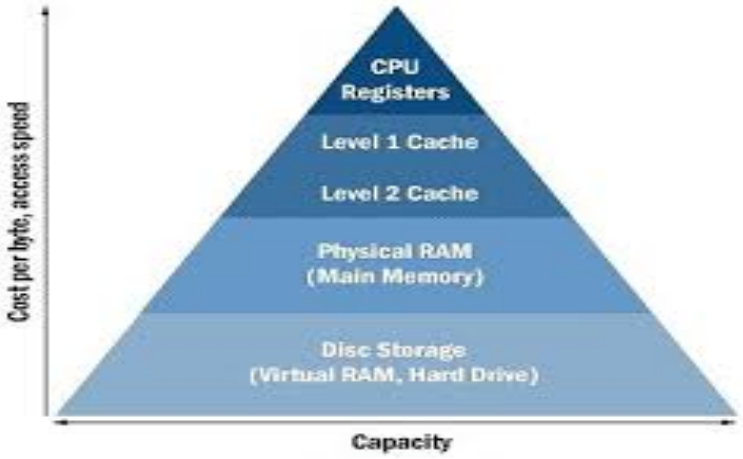
1. If the line with that memory address is in the cache (this is called a cache hit), the data is read from the cache to the MDR.

2. If the line with that memory address is not in the cache (this is called a miss), the cache is updated by replacing one of its active lines by the line with that memory address, and then the data is read from the cache to the MDR.

When a memory write is issued by the CPU:
1. If the line with that memory address is in the cache, the data is written from the MDR to the cache, and the line is marked "invalid" (since it no longer is an image of the corresponding memory line).

2. If the line with that memory address is not in the cache, the cache is updated by replacing one of its active lines by the line with that memory address. The data is then written from the MDR to the cache and the line is marked "invalid."

Cache updating is done in the following way.
1. A candidate line is chosen for replacement using an algorithm that tries to minimize the number of cache updates throughout the life of the program run. Two algorithms have been popular in recent architectures:
- Choose the line that has been least recently used - "LRU" for short (e.g., the PowerPC)
- Choose the line randomly (e.g., the 68040)
2. If the candidate line is "invalid," write out a copy of that line to main memory (thus bringing the memory up to date with all recent writes to that line in the cache).
3. Replace the candidate line by the new line in the cache.



Q.7 a. Difference between serial and parallel protocols.
Answer:
The communication links across which computers—or parts of computers—talk to one another may be either serial or parallel. A parallel link transmits several streams of data simultaneously along multiple channels (e.g., wires, printed circuit tracks, or optical fibers); a serial link transmits a single stream of data.
Although a serial link may seem inferior to a parallel one, since it can transmit less data per clock cycle, it is often the case that serial links can be clocked considerably faster than parallel links in order to achieve a higher data rate. A number of factors allow serial to be clocked at a higher rate:

- Clock skew between different channels is not an issue (for unclocke asynchronous serial communication links).
- A serial connection requires fewer interconnecting cables (e.g., wires/fibres) and hence occupies less space. The extra space allows for better isolation of the channel from its surroundings.
- Crosstalk is less of an issue, because there are fewer conductors in proximity.

In many cases, serial is a better option because it is cheaper to implement. Many ICs have serial interfaces, as opposed to parallel ones, so that they have fewer pins and are therefore less expensive

Serial and Parallel Communication

Data can be transmitted between a sender and a receiver in two main ways: serial and parallel.

Serial communication is the method of transferring one bit at a time through a medium.

0	1	0	0	0	0	1	0

Parallel communication is the method of transferring blocks, eg: BYTEs, of data at the same time.

0
1
0
0
0
0
1
0

As you can appreciate parallel communication is faster than serial. For this reason, the internal connections in a computer, ie: the busses, are linked together to allow parallel communication. However, the use of parallel communication for longer distance data communication is unfeasible for economic and practical reasons, eg: amount of extra cable required and synchronisation difficulties. Therefore, all long distance data communications takes place over serial connections.

b. Explain with the help of diagram Direct Memory Access.

Answer:

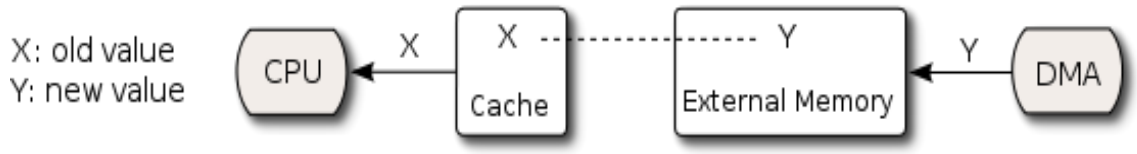
Direct memory access (DMA) is a feature of modern computers that allows certain hardware subsystems within the computer to access system memory independently of the central processing unit (CPU).

Without DMA, when the CPU is using programmed input/output, it is typically fully occupied for the entire duration of the read or write operation, and is thus unavailable to perform other work. With DMA, the CPU initiates the transfer, does other operations while the transfer is in progress, and receives an interrupt from the DMA controller when the operation is done. This feature is useful any time the CPU cannot keep up with the rate of data transfer, or where the CPU needs to perform useful work while waiting for a relatively slow I/O data transfer. Many hardware systems use DMA, including disk drive

controllers, graphics cards, network cards and sound cards. DMA is also used for intra-chip data transfer in multi-core processors. Computers that have DMA channels can transfer data to and from devices with much less CPU overhead than computers without a DMA channel. Similarly, a processing element inside a multi-core processor can transfer data to and from its local memory without occupying its processor time, allowing computation and data transfer to proceed in parallel.

DMA can lead to cache coherency problems. Imagine a CPU equipped with a cache and an external memory that can be accessed directly by devices using DMA. When the CPU accesses location X in the memory, the current value will be stored in the cache. Subsequent operations on X will update the cached copy of X, but not the external memory version of X, assuming a write-back cache. If the cache is not flushed to the memory before the next time a device tries to access X, the device will receive a stale value of X.

Similarly, if the cached copy of X is not invalidated when a device writes a new value to the memory, then the CPU will operate on a stale value of X.



Q.8 a. Explain the task states in RTOS with the help of two examples.

Answer:

Task States:

– Running

– Ready (possibly: suspended, pended)

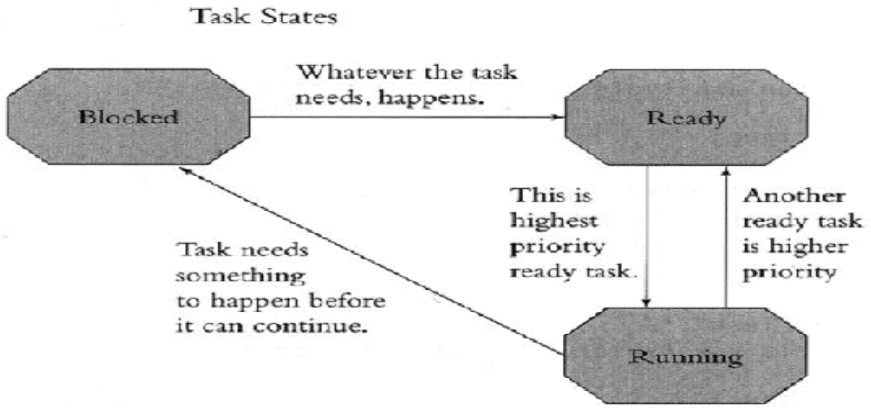
– Blocked (possibly: waiting, dormant, delayed)

– Scheduler – schedules/shuffles tasks between

Running and Ready states

– Blocking is self-blocking by tasks, and moved to Running state via other tasks' interrupt signaling (when block-factor is removed/satisfied)

– When a task is unblocked with a higher priority over the 'running' task, the scheduler 'switches' context immediately



b. Write short notes on semaphores and shared data.

Answer: Page Number 173-174 of Text Book II

Q.9 Discuss the case study for sending application layer byte streams on a TCP/IP network using RTOS Vx works.

Answer: Page Number 537 of Text book

TEXT BOOKS

1. Embedded System Design, A Unified Hardware/Software Introduction, Frank Vahid / TonyGivargis, 2006 reprint, John Wiley Student Edition.

2. An Embedded Software Primer, David .E. Simon, Fourth Impression 2007, Pearson Education.

3. Embedded Systems, Raj Kamal, 13th reprint 2007, Tata-McGrawHill Publications.