Jyoti Nivas College Autonomous Post Graduate Centre

Tech on Tap

E-Journal MCA DEPARTMENT

Issue: 7 April 2017

MICROPROCESSOR CHIPS INFORMATION

ANSHU (15MCA02)

SHWETHA (152MCA39)

Microprocessor chips (MPU) are silicon devices that serve as the central processing unit (CPU) in computers. They contain thousands of electronic components and use a collection of machine instructions to perform mathematical operations and move data from one memory location to another. Microprocessors contain an address bus that sends addresses to memory, read and write lines, and a data bus that can send data to memory or receive data from memory. They also include a clock line that enables a clock pulse to sequence the processor and a reset line that resets the program counter and restarts execution.

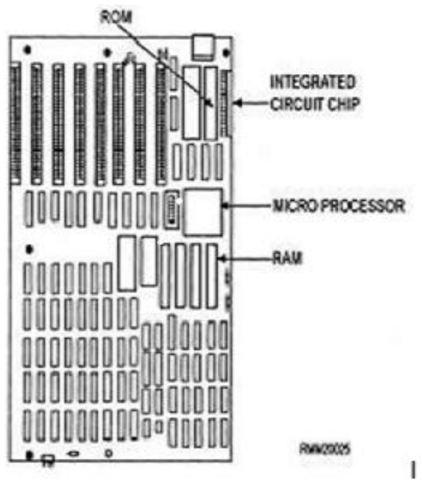
System Architectures

Basic microprocessor chip components include one or more arithmetic logic units (ALU) and shift registers. There are two system architectures for microprocessor chips.

Devices that use a reduced instruction set computer (RISC) design process a few simple instructions instead of many complex ones in order to speed operations.

By contrast, devices that use a complex instruction set computer (CISC) design provide variable length instructions, multiple addressing forms, and contain only a small number of general-purpose registers.

Input/output (I/O) ports and interfaces are connections that provide a data path between microprocessor chips (MPU) and external devices such as keyboards, displays, and readers. The number of I/O ports is equal to the number of input, output, and general-purpose ports (lines) combined. Communication controllers manage data inputs and outputs. They also convert data outputs for transmission over communication lines and perform all of the necessary control functions, error checking, and synchronization.



There are some Interfaces for microprocessor chips include: Transport control protocol/internet protocol (TCP/IP),Serial peripheral interface (SPI),Inter-IC (I2C) bus, infrared data association (IrDA),Synchronous data link control (SDLC),High-level data link control (HDLC),Pulse width modulation (PWM),Microprocessor chips (MPU) that use system management bus (SMBus),Control area network bus (CANbus),Universal serial bus (USB) ports

Important specifications to consider when selecting microprocessor chips (MPU) include:

Data bus - Most microprocessor chips are available with an 8-bit, 16-bit, 24-bit, 32-bit, 64-bit, 128-bit, or 256-bit data bus.

Microprocessor family - Products from many proprietary microprocessor families are commonly available.

Supply voltage - Supply voltages range from - 5 V to 5 V and include intermediate voltages such as - 4.5 V, - 3.3 V, - 3 V, 1.2 V, 1.5 V, 1.8 V, 2.5 V, 3 V, 3.3 V, and 3.6 V.

Clock speed - Clock speed, the frequency that determines how fast devices connected to the system bus operate, is generally expressed in megahertz (MHz).

Random access memory (RAM) - RAM is usually expressed in kilobytes (kB) or megabytes (MB).

Power dissipation - Power dissipation, the device's total power consumption, is generally expressed in watts (W) or milliwatts (mW).

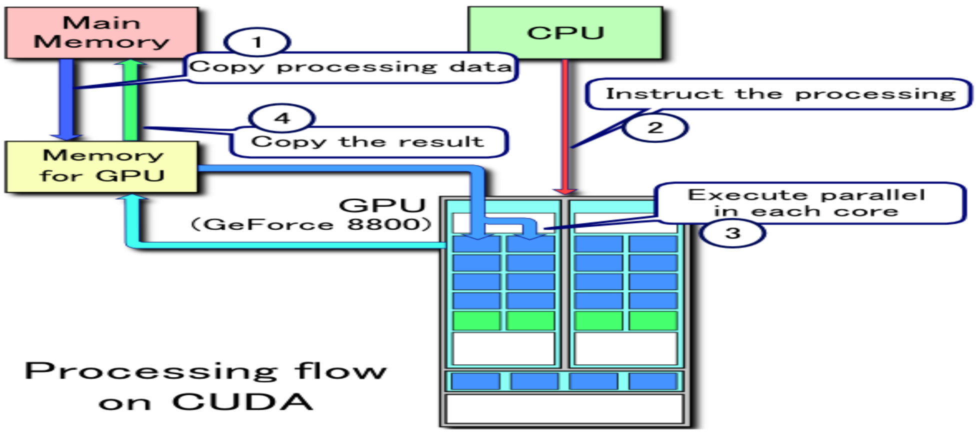
Operating temperature - Operating temperature is a full-required range.

CUDA

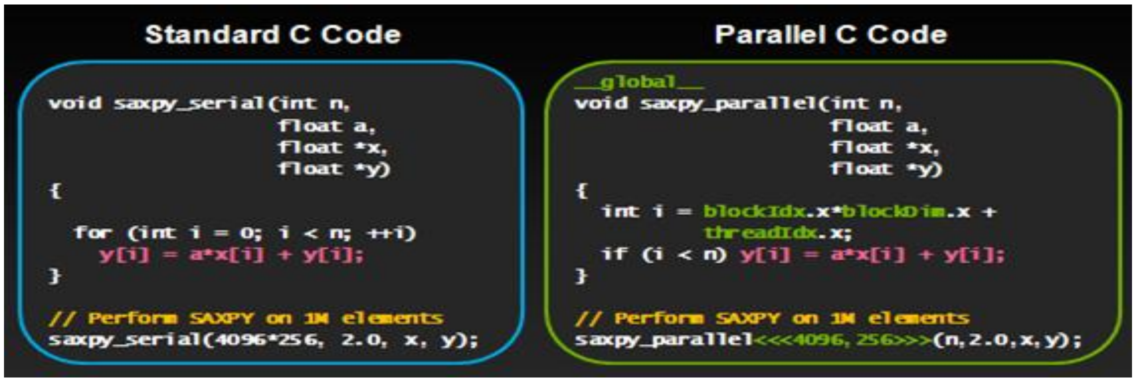
NEETU KUMARI (15MCA10)

LAVANYA M(15MCA13)

CUDA stands for Compute Unified Device Architecture. It's a technology developed by NVIDIA that accelerates GPU (graphics processing units) computation processes. It allows software developers and software engineers to use a CUDA-enable graphics processing unit (GPU) for general purpose processing units – an approach termed GPGPU (General-Purpose computing on Graphics Processing Units). The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute Kernals.



Writing programs with CUDA: One important thing to remember is that your entire program does not need to be written in CUDA. If you're writing a large application, complete with a user interface, and many other functions, and most of your code will be written in C++ or whatever your language of choice is. Then, when something extremely computationally intense is needed, your program can simply call the CUDA kernel function you wrote. So the main idea is that CUDA should only be used for the most computationally intense portions of your program.



Standard C Code

```
void saxpy_serial(int n,
                  float a,
                  float *x,
                  float *y)
{
  for (int i = 0; i < n; ++i)
    y[i] = a*x[i] + y[i];
}

// Perform SAXPY on 1M elements
saxpy_serial(4096*256, 2.0, x, y);
```

Parallel C Code

```
__global__
void saxpy_parallel(int n,
                    float a,
                    float *x,
                    float *y)
{
  int i = blockIdx.x*blockDim.x +
          threadIdx.x;
  if (i < n) y[i] = a*x[i] + y[i];
}

// Perform SAXPY on 1M elements
saxpy_parallel<<<4096,256>>>(n,2.0,x,y);
```

CUDA without a graphics card: While CUDA is specifically meant to run on nVidia’s graphics cards, it can also run on any CPU. Albeit, the program will never be able to run nearly as fast on a CPU, it will still work.

Choosing a Video Card With CUDA: In short, a higher number of CUDA cores typically means that the video card provides faster performance overall. However, the number of CUDA cores is only one of several things to consider when choosing a video card.

CUDA is well suited for large datasets: Most modern CPUs have a couple megabytes of L2 cache because most programs have high data coherency.

Parallel Computing with CUDA: Computing is evolving from "central processing" on the CPU to "co-processing" on the CPU and GPU.

POWER PC

ANGEL CHRISTINA. T(152MCA31)

SOUMYA.PH(15MCA27)

In February 1990, IBM introduced RS/6000 microprocessor based on POWER architecture with UNIX operating system. PowerPC was second generation POWER architecture. It has Reduced Instruction Set Computer (RISC) architecture. RISC architecture tries to keep the processor as busy as possible. Salient features of RISC architecture are :

- Fixed length instructions (4 byte instructions). This allows single decoding mechanism.
- Single cycle instruction execution.
- Less number of instructions.

PowerPC was created in 1991 by Apple-IBM-Motorola alliance. Originally intended for personal comput PowerPC CPUs have since become popular embedded and high-performance processors as well. It is larg based and compatible with POWER microprocessor. Design features of PowerPC are as follows :

- Broad range implementation.
- Simple processor design.
- Superscalar architecture.
- Multiprocessor features.
- 64-bit architecture.
- PowerPC can switch from one mode to another at run time.
- Separate set of Floating Point Registers (FPRs) for floating-point instructions.

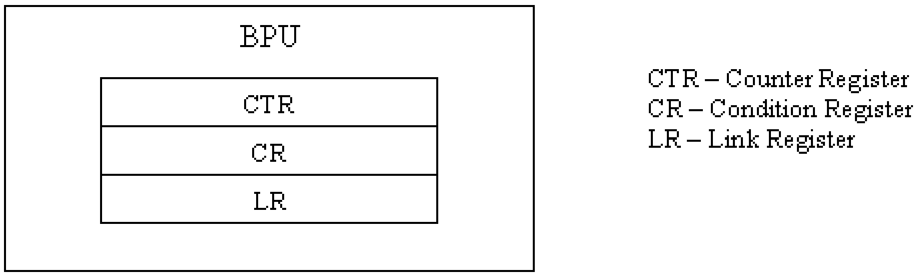
Motorola PowerPC 601 was the first PowerPC. Few of its features were -

1. 64-bit microprocessor.
2. 32-bit address lines.
3. Can handle integer data of 8, 16 and 32 bits.
4. RISC architecture with 4 byte instruction length.
5. PC 601 has virtual memory addressing of 4 penta byte.

Apart from the changes to the instruction set, the most significant changes in PowerPC were in the memory model and the memory management definition. In the POWER Architecture, the processor did not maintain data memory consistent with either I/O accesses or instruction fetches. Software had to manage memory consistency for both these areas. Before copying an area of memory to disk, software had to ensure that any modified copies of the memory area that were in the data cache had been written to main memory. Before starting a read from disk, software had to ensure that the data cache did not contain a copy of any part of the memory area, and software had to invalidate any copy of the memory area in the instruction cache before restarting the program that requested the operation. POWER processors always accessed main memory through the caches.

PowerPC memory model, however, provides greater flexibility. It implements processor-enforced data memory consistency, relieving software of the responsibility for the consistency of memory with respect to I/O operations. The model allows speculative access to any page unless it has an attribute indicating that it contains I/O or it exhibits other volatile characteristics. It also makes it possible to map I/O into the main memory space.

As in the POWER memory model, the PowerPC memory model requires software to maintain instruction memory consistent with data memory. Programs that modify or generate instructions must ensure that cached copies of a memory area containing the new instructions are consistent with the main memory before attempting to execute those instructions.



Branch Processing Unit of PowerPC

- The Branch Processing Unit (BPU) looks at lower four instructions in instruction queue to bring the branch instruction in advance.
- The jump instruction is analyzed and the next instruction is brought and executed till the write-back stage. With this the branch takes single cycle.
- A branch instruction has a Jump Prediction Bit associated with which tells whether there is likelihood of jump or not.
- In case a jump is predicted new instructions may be brought in for the entire instruction queue. Later, if the prediction comes out to be true then the execution continues normally and we have considerable amount of performance gain.
- However, if branch prediction turns out to be false then we have something called Branch Folding. In branch folding all instructions executed after the prediction are discarded and the execution resumes just after branch instruction. We have loss of instruction cycles in this case.
- The PowerPC Architecture permits a range of implementations from low-cost controllers through high-performance processors.
- It allows the implementation of processors targeted for desktop and notebook systems, yet it contains features to support the efficient implementation of processors for use in a range of multiprocessor systems.

ARCHITECTURAL SUPPORT FOR SOFTWARE DEBUGGING

KAVYASHREE.C (15MCA09)

MARY RESHMA JENNIFER.J (15MCA15)

Debugging plays an important role in software development. As computers go everywhere and find new applications, the request for reliable software increases. It was approximated that software bugs damaged only US economy about 59.5 billion dollars.

The program testing techniques generally classified into the following trends:

• Static program analysis techniquesfind bugs by source code analysis. They try to extract facts about behavior of program and reason about inconsistencies in it data flow n testing, code inspection and control flow analysis are few examples for this category. These methods suffer from un decidability of program analysis problems

• Dynamic test techniquesattempt to find the bugs, by feeding data (called "test cases") to the running program and observing its outputs. The major challenge here is design of test suits to manifest maximum bugs structure based testing, experienced based techniques, boundary value analysis are some of branches under this approach. The success of these techniques, depends on the test coverage that test suits provide.

• Run-time monitoring instruments the program with assertions to verify the execution of software. Since, it works within program, it accesses data and control states.

Majority of the current architectures for debugging fall in two following categories:

1. Replay architecturesfor effective recording of the traces of execution, specially ones lead to an error. The traces can be analyzed and replayed by developers for postmortem analysis of software . Debugging requires enough understanding of error. Reproducing of error by step-wise replaying of error scenario provides such understanding. It helps to find error manifestation conditions, and consequently to fix it. In case of parallel program, it is more valuable, because non-deterministic nature of execution causes bugs to manifest in special threads interleaving some software-only techniques records the program execution, while it is running. These techniques suffer from performance overhead. replay architectures which continuously record the production run (i.e. program execution steps) in anticipation of a "trigger". Trigger usually represents an error These logs can be used for reproduction of error or other postmortem analysis.
Concurrency and non-repeatable input introduce non-determinism to the program. Thus, a replay architecture must guarantee that replay of logs leads to correct manifestation of the error. In other word, the replay must be faithful. For this reason, the architecture has to capture synchronization

races (the order of execution of instructions of threads) and memory race (the order of access to shared data). Observe that, the number events in production runs is usually huge. Thus, the major challenge in these architectures is, to minimize the set of stored events while guaranteeing the faithful replay of log.

2. Lifeguard architecturesto provide hardware support for program monitoring. This approach modifies the architecture to facilitates efficient implementation of run-time monitors. This monitors often augmented with handlers to prevent the program from malicious behavior.

By this we glanced at the approach of architectural support for software debugging as an emerging filed. The hardware architecture is the final place that software resides on for a long time. Thus, architecture has access to valuable and information about the software, such as memory accesses, information flow, thread interleaving and etc. This information helps to improve the quality of software. This approach tries to incorporate the processors to capture this information, to address software defects. We categorized this approach into two broad categories, replay architectures and lifeguard architectures.

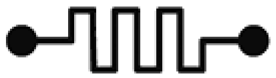
MEMRISTOR

SHINY ELIZABETH.S(15MCA26)

SAI JYOTHI(15MCA30)

Definition of Memristor:

"The memristor is formally defined as a two-terminal element in which the magnetic flux Φm between the terminals is a function of the amount of electric charge q that has passed through the device."



In other Words, A memristor is an electrical component that limits or regulates the flow of electrical current in a circuit and remembers the amount of charge that has previously flowed through it. Memristors are important because they are non-volatile, meaning that they retain memory without power.

History of Memristor

Memristor theory was formulated and named by Leon Chua in a 1971 paper. Chua strongly believed that a fourth device existed to provide conceptual symmetry with the resistor, inductor, and capacitor. This symmetry follows from the description of basic passive circuit elements as defined by a relation between two of the four fundamental circuit variables.

A device linking charge and flux (themselves defined as time integrals of current and voltage), which would be the memristor, was still hypothetical at the time. However, it would not be until thirty-seven years later, on April 30, 2008, that a team at HP Labs led by the scientist R. Stanley Williams would announce the discovery of a switching memristor. Based on a thin film of titanium dioxide, it has been presented as an approximately ideal device.

Memristor

A memristor is one of four basic electrical circuit components, joining the resistor, capacitor, and inductor. The memristor, short for "memory resistor" The reason that the memristor is radically different from the other fundamental circuit elements is that, unlike them, it carries a memory of its past. When you turn off the voltage to the circuit, the memristor still remembers how much was applied before and for how long. That's an effect that can't be duplicated by any circuit combination of resistors, capacitors, and inductors, which is why the memristor qualifies as a fundamental circuit element.

The known three fundamental circuit elements as resistor, capacitor and inductor relates four fundamental circuit variables as electric current, voltage, charge and magnetic flux. In that we were missing one to relate charge to magnetic flux. That is where the need for the fourth fundamental element comes in. This element has been named as memristor.

Memristance (Memory + Resistance) is a property of an Electrical Component that describes the variation in Resistance of a component with the flow of charge. Any two terminal electrical component that exhibits Memristance is known as a Memristor An ideal memristor is a passive two-terminal electronic device that is built to express only the property of memristance (just as a resistor expresses resistance and an inductor expresses inductance). However, in practice it may be difficult to build a 'pure memristor,' since a real device may also have a small amount of some other property, such as capacitance (just as any real inductor also has resistance). A common analogy for a resistor is a pipe that carries water. The water itself is analogous to electrical charge, the pressure at the input of the pipe is similar to voltage, and the rate of flow of the water through the pipe is like electrical current. Just as with an electrical resistor, the flow of water through the pipe is faster if the pipe is shorter and/or it has a larger diameter.

Types of Memristors:
• SpintronicMemristor
• Spin Torque Transfer Magneto resistance
• Titanium dioxide memristor
• Polymeric memristor
• Spin memristive systems
• Magnetite memristive systems
• Resonant tunneling diode memristor

Applications of memristors:
Non-volatile memory applications: • Memristors can retain memory states, and data, in power-off modes. • Non-volatile random access memory, or NVRAM, is much the first application. • There are already 3nm Memristors in fabrication now
Low-power and remote sensing applications: coupled with memcapacitors and meminductors, the complementary circuits to the memristor which allow for the storage of charge, memristors can possibly allow for Nano-scale low power memory and distributed state storage, as a further extension of NVRAM capabilities. These are currently all hypothetical in terms of time to market.

Advantages of memristors:

- Has properties which can not be duplicated by the other circuit elements (resistors, capacitors, and inductors
- Capable of replacing both DRAM and hard drives
- Smaller than transistors while generating less heat
- Works better as it gets smaller which is the opposite of transistors
- Devices storing 100 gigabytes in a square centimeter have been created using memristors
- Quicker boot-ups
- Requires less voltage (and thus less overall power required)

Disadvantages of memristors:

- Not currently commercially available
- Current versions only at 1/10th the speed of DRAM
- Has the ability to learn but can also learn the wrong patterns in the beginning
- Since all data on the computer becomes non-volatile, rebooting will not solve any issues as it often times can with DRAM
- Suspected by some that the performance and speed will never match DRAM and transistors

DATA PREFETCHING TECHNIQUES

SEEMA DABADI (15MCA24)

JANUKA CHHETRI (15MCA07)

In computer architecture, **instruction prefetch** is a technique used in central processor units to speed up the execution of a program by reducing wait states.

Prefetching of data occurs when a processor requests an instruction or data block from main memory before it is actually needed. Once the block comes back from memory, it is placed in a cache. When the instruction/data block is actually needed, it can be accessed much more quickly from the cache than if it had to make a request from memory. Thus, prefetching hides memory access latency and hence, it is a useful technique for addressing the memory wall issue.

Since programs are generally executed sequentially, performance is likely to be best when instructions are prefetched in program order. Alternatively, the prefetch may be part of a complex branch prediction algorithm, where the processor tries to anticipate the result of a calculation and fetch the right instructions in advance. In the case of dedicated hardware (like a Graphics Processing Unit) the prefetch can take advantage of the spatial coherence usually found in the texture mapping process. In this case, the prefetched data are not instructions, but texture elements that are candidates to be mapped on a polygon.

Types of prefetching

Data or instruction prefetching: As the name implies, the prefetching can be performed for either data blocks or instruction blocks. Since data access patterns show less regularity than instruction patterns, accurate data prefetching is generally more challenging than instruction prefetching.

Hardware or software prefetching: Prefetching can be performed in either hardware or software. Hardware prefetchers may use some storage to detect access patterns and based on it, prefetch instructions are issued. Software prefetchers insert prefetch instructions in program source-code based on knowledge of program control flow.

Application: A prefetch operation is useful or useless depending on whether the item brought by it removes or does not remove a future cache miss. A prefetch operation is harmful if the item brought by it replaces a useful block and thus, possibly increases the cache misses. Harmful prefetches lead to cache pollution. A prefetch operation is redundant if the data-block brought by it is already present in cache.

CLUSTER COMPUTING

SANTOSINI DAS(15MCA22)

LOKESHWARI V(15MCA14)

DEFINITION:-

Cluster is a system comprising two or more computers or systems **(called nodes)** which work together to execute applications or perform other tasks, so that users who use them, have the impression that only a single system responds to them, thus creating an illusion of a single resource **(virtual machine)**. This concept is called transparency of the system. A computer cluster consists of a set of loosely or tightly connected computers that work together so that, in many respects, they can be viewed as a single system. Unlike grid computers, computer clusters have each node set to perform the same task, controlled and scheduled by software.

TYPES:-

a) High Availability Clusters

HA Clusters are designed to ensure constant access to service applications. The clusters are designed to maintain redundant nodes that can act as backup systems in the event of failure. The minimum number of nodes in a HA cluster is two – one active and one redundant – though most HA clusters will use considerably more nodes. HA clusters aim to solve the problems that arise from mainframe failure in an enterprise. Rather than lose all access to IT systems, HA clusters ensure24/7 access to computational power. This feature is especially important in business, where data processing is usually time-sensitive.

b) Load-balancing Clusters

Load-balancing clusters operate by routing all work through one or more load-balancing front-end nodes, which then distribute the workload efficiently between the remaining active nodes. Load-balancing clusters are extremely useful for those working with limited IT budgets. Devoting a few nodes to managing the workflow of a cluster ensures that limited processing power can be optimised.

c) High-performance Clusters

HPC clusters are designed to exploit the parallel processing power of multiple nodes. They are most commonly used to perform functions that require nodes to communicate as they perform their tasks – for instance, when calculation results from one node will affect future results from another.

ADVANTAGES OF CLUSTER COMPUTING:-

1. It helps in reducing cost.

2. It improves network technology.

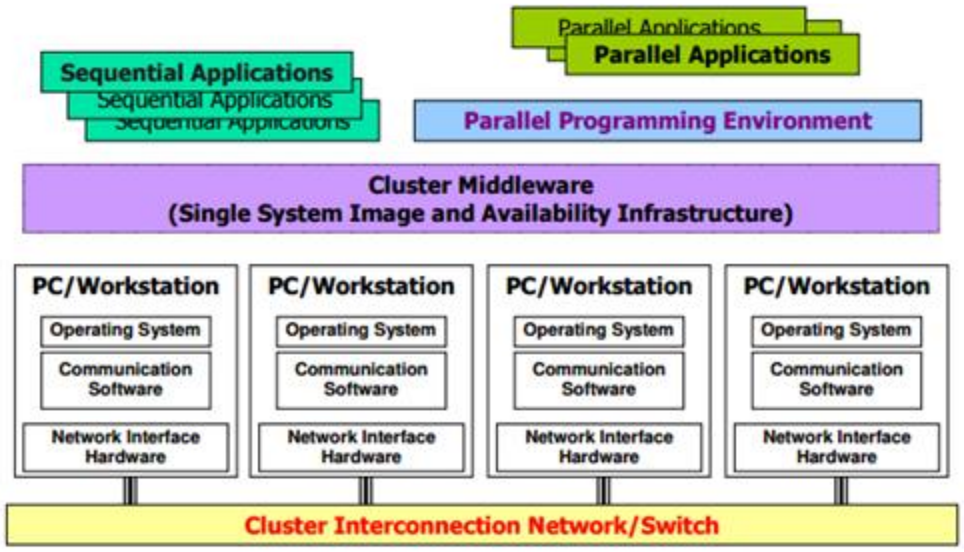
3. It is always available in all possible situations.

DISADVANTAGES OF CLUSTER COMPUTING:-

1) It is difficult for developing software for distributed system.

2) It can be easily accessed and applied to secret data.

ARCHITECTURE:



BENEFITS OF COMPUTER CLUSTER:

Computer clusters offer a number of benefits over mainframe computers, including:

1. Reduced Cost
2. Processing Power
3. Improved Network Technology
4. Scalability
5. Availability

X86 ARCHITECTURE

NANCY ESTHER DEVA KUMAR(152MCA34)

VINOLIYA GRACE E(152MCA35)

The x86 architecture is an instruction set architecture (ISA) series for computer processors. Developed by Intel Corporation, x86 architecture defines how a processor handles and executes different instructions passed from the operating system (OS) and software programs. The “x” in x86 denotes ISA version. Designed in 1978, x86 architecture was one of the first ISAs for microprocessor-based computing. The key features are: provides a logical framework for executing instructions through a processor, allows software programs and instructions to run on any processor in the Intel 8086 family, provides procedures for utilizing and managing the hardware components of a central processing unit (CPU).

The x86 architecture has 8 General-Purpose Registers(GPR), 6 Segment Registers, 1 Flags Register and an Instruction Pointer.

The General-Purpose Registers are EAX: Accumulator, EBX: Base index(for use with arrays), ECX: Counter (for use with loops and strings), EDX: Extend the precision of the accumulator, ESI: Source index for string operations, EDI: Destination index for string operations, EBP: Stack base pointer for holding the address of the current stack frame and ESP: Stack pointer for top address of the stack.
The Segment Registers are CS: Code Segment Register, DS: Data Segment Register, SS: Stack Segment Register, ES: Extra Segment Register, FS: Extra Segment Register #2 and GS: Extra Segment Register#3.
The Flags Register is EFLAGS: Reports on the status of the program being executed and the Instruction Pointer Register is EIP: Holds the program counter, the current instruction address.

The operating modes of x86 architecture: Real Mode, Protected Mode, Virtual 8086 Mode, Unreal Mode, System Management Mode and Long Mode.

Real Mode: A highly restricted operating mode in which only the first 1 MB of physical memory is directly accessible and provides no support for memory protection, multitasking, or code privilege levels, **Protected Mode**: The normal mode of operation and allows system software to use features like virtual memory, paging and safe multi-tasking, **Virtual 8086 mode**: A virtual real mode which allows execution of real mode applications that are incapable of running directly in protected mode, **Unreal Mode**: A variant of real mode in which one more data segment registers have been loaded with 32-bit addresses and limits but it is not a separate addressing mode that the x86 processors can operate on, **System Management Mode**: An operating mode of x86 CPUs in which all normal execution, including the operating system is suspended, **Long Mode**: A mode where a 64-bit operating system can access 64-bit instructions and registers but the real mode or virtual 8086 mode programs cannot be run in long mode.

The addressing modes include: Register Addressing Mode, Immediate Addressing Mode, Direct Addressing Mode, Register Indirect Addressing Mode, Indexed Addressing Mode, Base Relative Addressing Mode and Base Indexed Addressing Mode.

Register Addressing Mode: Data transfer using registers and operand value is present in register (MOV AL, BL);**Immediate Addressing Mode**: Data is stored in code segment and operand value is present in the instruction (MOV AX, 12345);**Direct Addressing Mode**: Direct memory address is supplied as part of the instruction and operand offset value is given in instruction (MOV AX, [1234]);**Register Indirect Addressing Mode**: Registers used are BX, SI-Source Index, DI-Destination Index or BP-Base Pointer and operand offset value is given in CPU register (MOV [BX],AX);**Indexed Addressing Mode**: Operand offset is given by a sum of a value held in either SI or DI register and constant displacement is specified by an operand (MOV SI, 3);**Base Relative Addressing Mode**: Operand offset is given by a sum of a value held either in BP or BX and a constant offset specified as an operand (MOV AX, [BP+1]);**Base Indexed Addressing Mode**: Operand offset is given by sum of either BX or BP with either SI or DI (MOV AX, [BX+SI]).

The x86 architecture supports 8 types of instructions: Data Transfer Instructions, Arithmetic Instructions, Bit Manipulation Instructions, String Instructions, Branch and Loop Instructions, Processor Control Instructions, Iteration Control Instructions and Interrupt Instructions.

Data Transfer Instructions are used to transfer the data from the source operand to the destination operands; **Arithmetic Instructions** are used to perform arithmetic operations like addition, subtraction, multiplication, division, etc.; **Bit Manipulation Instructions** are used to perform operations where data bits are involved, i.e. operations like logical, shift, etc.; **String Instructions** are a group of bytes/words and their memory is always allocated in a sequential order; **Branch and Loop Instructions** are used to transfer/branch the instructions during an execution; **Processor Control Instructions** are used to control the processor action by setting/resetting the flag values; **Iteration Control Instructions** are used to execute the given instructions for number of times and **Interrupt Instructions** are used to call the interrupt during program execution.

x86 started out as a 16-bit instruction set for 16-bit processors (the 8086 and 8088 processors), then was extended to a 32-bit instruction set for 32-bit processors (80386 and 80486), and now has been extended to a 64-bit instruction set for 64-bit processors.

As of 2017, the majority of personal computers and laptops sold are based on the x86 architecture. At the high end, x86 continues to dominate compute-intensive workstation and cloud computing segments.

BRANCH PREDICTOR IN COMPUTER ARCHITECTURE

G.PRIYADHARSHINI(15MCA19)

T .MARY CHROSINE(15MCA29)

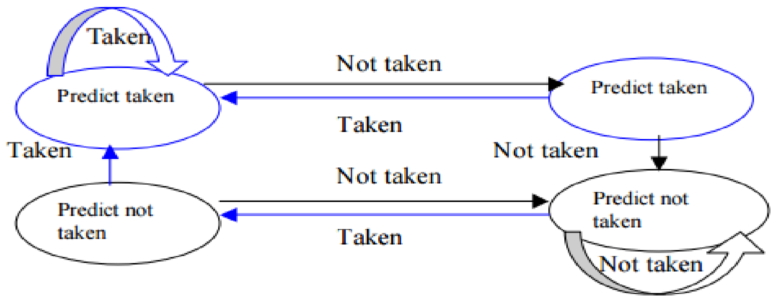
Introduction:

Branches change the normal flow of control in programs in unexpected manner which interrupts the normal fetch and issue operation of instructions. To resume the normal fetch it takes considerable number of cycles until the outcome of branches and new target is known. Branches are very frequent in general purpose programs – around 20% of the instructions are branches – and simplest solution to deal with branches is to stall the pipeline. Stalling pipeline doesn't violate the correctness of program. However, it does degrade the overall IPC (Instruction Level Parallelism).which is even severe in longer pipelines. In order to maintain the high throughput branch prediction in modern high performance microarchitecture has become essential.

Schemes of Dynamic Branch prediction:

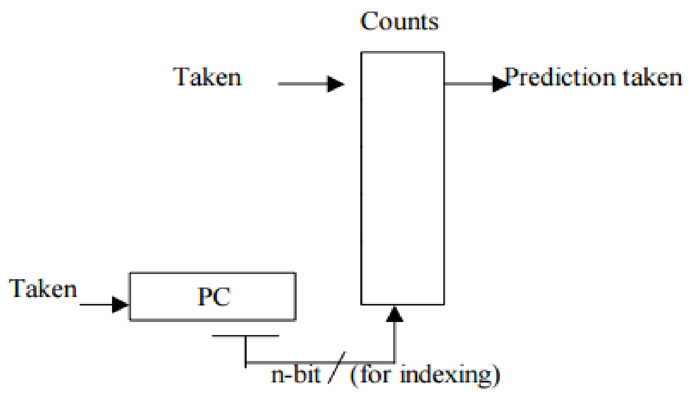
Two-bit Counter Branch Prediction Scheme:

The two-bit counter scheme always assigns a two-bit counter to the prediction cache buffer for each entry when each conditional branch occurs. The counter value is between 0 and 3. The counter is incremented when branch is predicted as taken; otherwise the counter is decremented if the prediction is not taken. Therefore, a prediction must be wrong twice before it is changed. The most significant bit determines the taken decision of prediction. The buffer is responsible for collecting history information, and applies the information to making the prediction. The state of branch's entry in the buffer is therefore changed dynamically when the branch instructions are executed.



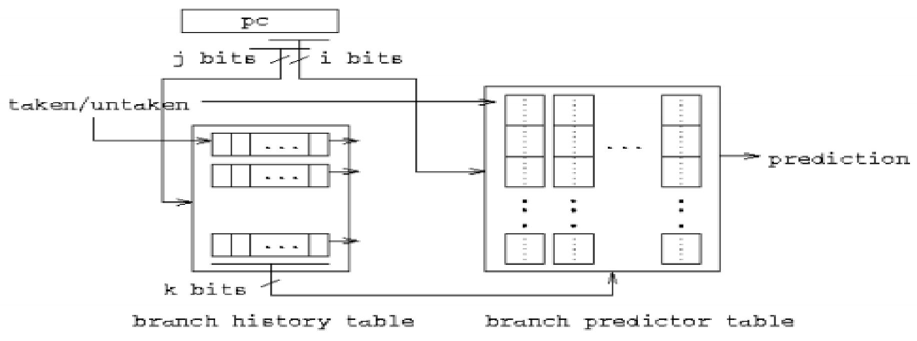
Bimodal Branch Prediction Scheme:

Bimodal scheme takes advantage of two-bit counter branch behavior. The state of counters is stored in a counter table that records all the branches history. Each branch will then map to a unique counter. The branch history table is indexed by some bits of branch address. The mapping will not be any problem if the counter table is large enough for all branches, says more than 128K bytes; but for the smaller tables, the performance of accurate prediction can be impeded by sharing same counter with multiple branches. However, the smaller size of predictor still performs more effective than the two-bit scheme.



Correlated Branch Prediction Scheme:

One of solutions for the problem of bimodal branch on the mapping collisions is the scheme of correlated prediction schemes. This correlated scheme uses two branch history tables, one is for keeping the recent branches history records and the other one is for keeping the state of branches in each entry contained 2-bit counter. They are so-called history table and counter table. In other word, the correlated scheme takes advantage of relationship between different branch instructions that is certain repetitive branch pattern of several consecutive branches. The local, global, and global selection branch predictors are all considered as correlated branch prediction scheme.



MULTITHREADING IN COMPUTER ARCHITECTURE

SUREKHA(15MCA28)

RANJITHA C (15MCA21)

In computer architecture,is the ability of a single core in a multi-core processor to execute multiple processes concurrently, appropriately supported by the operating system.This approach differs from multiprocessing, as with multithreading the processes and threads share the resources of a single or multiple cores.

Types of Multi Threading

- Coarse-grained multithreading
- Interleaved multithreading
- Simultaneous multithreading
- Implementation specifics

Coarse-grained multithreading

The simplest type of multithreading occurs when one thread runs until it is blocked by an event that normally would create a long-latency stall. Such a stall might be a cache miss that has to access off-chip memory, which might take hundreds of CPU cycles for the data to return. Instead of waiting for the stall to resolve, a threaded processor would switch execution to another thread that was ready to run. Only when the data for the previous thread had arrived, would the previous thread be placed back on the list of ready-to-run threads.

Interleaved multithreading

The purpose of interleaved multithreading is to remove all data dependency stalls from the execution pipeline . Since one thread is relatively independent from other threads, there is less chance of one instruction in one pipelining stage needing an output from an older instruction in the pipeline. Conceptually, it is similar to preemptive multitasking used in operating systems; an analogy would be that the time slice given to each active thread is one CPU cycle.

Simultaneous multithreading

The most advanced type of multithreading applies to superscalar processors. Whereas a normal superscalar processor issues multiple instructions from a single thread every CPU cycle, in simultaneous multithreading (SMT) a superscalar processor can issue instructions from multiple threads every CPU cycle. Recognizing that any single thread has a limited amount of instruction-level parallelism, this type of multithreading tries to exploit parallelism available across multiple threads to decrease the waste associated with unused issue slots.

Implementation specifics

A major area of research is the thread scheduler that must quickly choose from among the list of ready-to-run threads to execute next, as well as maintain the ready-to-run and stalled thread lists. An important subtopic is the different thread priority schemes that can be used by the scheduler. The thread scheduler might be implemented totally in software, totally in hardware, or as a hardware/software combination.

Advantages

- If a thread gets a lot of cache misses, the other threads can continue taking advantage of the unused computing resources, which may lead to faster overall execution as these resources would have been idle if only a single thread were executed
- If several threads work on the same set of data, they can actually share their cache, leading to better cache usage or synchronization on its values
- Multiple threads can interfere with each other when sharing hardware resources such as caches or translation lookaside buffers (TLBs). As a result, execution times of a single thread are not improved but can be degraded, even when only one thread is executing, due to lower frequencies or additional pipeline stages that are necessary to accommodate thread-switching hardware.

Disadvantages

- Overall efficiency varies; Intel claims up to 30% improvement with its Hyper-Threading Technology, while a synthetic program just performing a loop of non-optimized dependent floating-point operations actually gains a 100% speed improvement when run in parallel. On the other hand, hand-tuned assembly language programs using MMX or AltiVec extensions and performing data prefetches (as a good video encoder might) do not suffer from cache misses or idle computing resources. Such programs therefore do not benefit from hardware multithreading and can indeed see degraded performance due to contention for shared resources.
- From the software standpoint, hardware support for multithreading is more visible to software, requiring more changes to both application programs and operating systems than multiprocessing. Hardware techniques used to support multithreading often parallel the software techniques used for computer multitasking. Thread scheduling is also a major problem in multithreading

APPLICATIONS

A case study with EARTH–MANNA

Multithreading offers benefits with respect to the formulation of irregular dynamic programs and their dynamic scheduling, load balancing and interaction. Furthermore, low-cost communication on distributed-memory machines by remote-memory access is provided by some systems for efficient communication. EARTH is one of the few systems which combines both, while most other systems either focus on communication or provide multithreading in shared-memory environments. Dynamic irregular applications are often awkward to parallelize on distributed memory when using SPMD style programming via MPI and show different requirements for formulation. In addition, dynamic irregular applications also may show a fairly tight data coupling. Systems like EARTH are beneficial then, because they specifically support large number of small data exchanges by providing short startup times and the tolerance of even small latencies (offering very fine-grain threads). However, static regular applications with tight data coupling are supported too.

ALIAS ANALYSIS

RAKSHA (15MCA20)

H SRINAVYA B (15MCA06)

Alias analysis is a technique in compiler theory, used to determine if a storage location may be accessed in more than one way. Two pointers are said to be aliased if they point to the same location.

Alias analysis techniques are usually classified by flow-sensitivity and context-sensitivity. They may determine may-alias or must-alias information. The term **alias analysis** is often used interchangeably with points-to analysis, a specific case.

Alias analyzers intend to make and compute useful information for understanding aliasing in programs.

There are three possible alias cases here:

1. The variables p and q cannot alias (i.e., they never point to the same memory location).
2. The variables p and q must alias (i.e., they always point to the same memory location).
3. It cannot be conclusively determined at compile time if p and q alias or not

In alias analysis, we divide the program's memory into *alias classes*. Alias classes are disjoint sets of locations that cannot alias to one another. For the discussion here, it is assumed that the optimizations done here occur on a low-level intermediate representation of the program. This is to say that the program has been compiled into binary operations, jumps, moves between registers, moves from registers to memory, moves from memory to registers, branches, and function calls/returns.

Type-based Alias Analysis

If the language being compiled is type safe, the compiler's type checker is correct, and the language lacks the ability to create pointers referencing local variables, (such as ML, Haskell, or Java) then some useful optimizations can be made. There are many cases where we know that two memory locations must be in different alias classes:

1. Two variables of different types cannot be in the same alias class since it is a property of strongly typed, memory reference-free (i.e., references to memory locations cannot be changed directly) languages that two variables of different types cannot share the same memory location simultaneously.
2. Allocations local to the current stack frame cannot be in the same alias class as any previous allocation from another stack frame. This is the case because new memory allocations must be disjoint from all other memory allocations.

3. Each record field of each record type has its own alias class, in general, because the typing discipline usually only allows for records of the same type to alias. Since all records of a type will be stored in an identical format in memory, a field can only alias to itself.
4. Similarly, each array of a given type has its own alias class.

Flow-based Alias Analysis

Analysis based on flow, unlike type based analysis, can be applied to programs in a language with references or type-casting. Flow based analysis can be used in lieu of or to supplement type based analysis. In flow based analysis, new alias classes are created for each memory allocation, and for every global and local variable whose address has been used. References may point to more than one value over time and thus may be in more than one alias class. This means that each memory location has a set of alias classes instead of a single alias class.

Advantage

1. Fast time bound and accuracy.
2. Increase the program performance.

Disadvantage

1. Many alias analyses are prohibitively slow and thus impractical for production use.
2. The alias analyses in the literature require the entire program (or some representation of it), which inhibits separate compilation and compiling libraries.
3. Most alias analyses have been evaluated only statically, and thus we do not know the effectiveness of these algorithms with respect to the optimizations that use them.

NANOCOMPUTING

CHITHRA.M.S(15MCA04)

NETHRAVATHI.M(15MCA17)

A nanocomputer is a **computer** whose physical dimensions are microscopic. The field of **nanocomputing** is part of the emerging field of nanotechnology. Several types of nanocomputers have been suggested or proposed by researchers and futurists. Nanocomputing describes computing that uses extremely small, or nanoscale, devices (one nanometer [nm] is one billionth of a meter). In 2001, state-of-the-art electronic devices could be as small as about 100 nm, which is about the same size as a virus. The **integrated circuits**(IC) industry, however, looks to the future to determine the smallest electronic devices possible within the limits of computing technology.

If such a new device or computer architecture were to be developed, this might lead to million fold increases in computing power. Such circuits would consume far less power per function, increasing battery life and shrinking boxes and fans necessary to cool circuits. Also, they would be remarkably fast and able to perform calculations that are not yet possible on any computer. Benefits of significantly faster computers include more accuracy in predicting weather patterns, recognizing complex figures in images, and developing **artificial intelligence (AI)**. Potentially, single-chip memories containing thousands of gigabytes of data will be developed, capable of holding entire libraries of books, music, or movies.

Nano Computing is an advanced stage of development due to recent nanotechnology advances in these areas. These computers promise to be so fast and powerful that our modern supercomputers will be little more than toys in years to come. The problem is that here sheer capacity for processing data means they will likely be able to readily crack even the most advanced encryption methods through brute force. Nanocomputing can be justified as the usage of Nano-particles and Nano principles in the Computing domain which can be really very useful.

Nano technology in computer science is named as Nano computing .which is divided into categories- Electronic Nano computing, Mechanical. Nano computing, Quantum Nanocomputing etc. The application and use of nano materials in electronic devices, in optical and magnetic components are the economically most important parts of the nanotechnology permeate the design of artefacts at nowadays and likely in the near future. Advances in technology repeatedly allow software to lower and lower levels.

Future nanocomputers could be evolutionary, scaled-down versions of today's computers, working in essentially the same ways and with similar but nanoscale devices. Current nanocomputing research involves the study of very small electronic devices and molecules, their fabrication, and architectures that can benefit from their inherent electrical properties.

Nanocomputing technology has the potential for revolutionizing the way that computers are used. However, in order to achieve this goal, major progress in device technology, computer architectures, and IC processing must first be accomplished. It may take decades before revolutionary nanocomputing technology becomes commercially feasible.

E-BALL PC TECHNOLOGY

MAYA.A.U(15MCA16)

AMALA JACOB(15MCA01)

The E-Ball concept pc is a sphere shaped computer which is the smallest design among all the laptops and desktops.

This PC concept features all the traditional elements like mouse, keyboard, large screen display, DVD recorder, etc, all in an innovative manner. E-Ball is designed to be placed on two stands, opens by simultaneously pressing and holding the two buttons located on each side. After opening the stand and turning ON the PC, pressing the detaching mouse button will allow you to detach the optical mouse from the PC body. This concept features a laser keyboard that can be activated by pressing the particular button. E-Ball is very small, it is having only 6 inch diameter sphere. It is having 120×120mm motherboard.

Elements of E-Ball

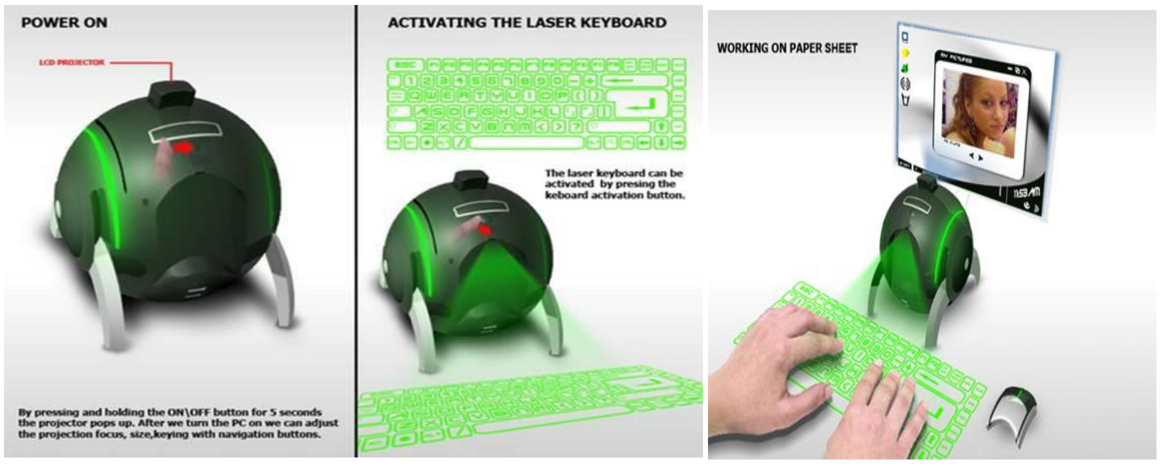
It contains wireless optical mouse and laser keyboard, and LCD projector.It has around 350-600GB of Hard Disk Drive. It contains 5GB RAM. It has two 50W speakers. It has LAN and WLAN card and a Web cam. When you want to carry it around you can easily “pack it” into a ball. This is a futuristic concept, and this is how the future computers will look like.

This device has an optical keyboard and an holographic display. So don’t have a physical keyboard and no monitor! Still, the mouse is physical but it fits in to the computer when we want to carry it around. The bad thing about using a virtual keyboard is that it need a smooth surface. It is strange enough to call this device a computer, because it is so small and it have a powerful battery for great stand by time.

Working of E-Ball

E Ball concept pc don't have any external display unit, it has a button when you press this button a projector will pop and it focus the computer screen on the wall which can be adjusted with navigation keys. If there is no wall then it has a paper sheet holder that divides into three pieces like an umbrella just after popping up, and it will show desktop on the paper sheet. Also, the E-Ball PC supports a paper holder and the paper sheet on the holder could act like a screen where you can watch movies or something. This concept PC will measure 160mm in diameter and it was designed for Microsoft Windows OS.E-Ball concept pc has a laser keyboard that is fully a concept keyboard that is visible when the pc is in working. The keyboard is not physical - it is interpreted by lasers that appear after you press the respective button. It recognizes your fingers with the help of an IR sensor when you are typing at a particular place, while the mouse is a pop out wonder making this an exiting piece of technology.

The software interface of E-Ball concept pc is highly stylized with icons that can be remembered easily that support all type of windows operating system. E-Ball concept pc work very easy while you are making video presentations, listening music watching large screen movies, and chatting on the net.


POWER ON
LCD PROJECTOR
By pressing and holding the ON\OFF button for 5 seconds the projector pops up. After we turn the PC on we can adjust the projection focus, size,keying with navigation buttons.
ACTIVATING THE LASER KEYBOARD
The laser keyboard can be activated by presing the keboard activation button.
WORKING ON PAPER SHEET

ADVANTAGES AND LIMITATIONS

E-Ball is portable. It has large memory. Useful for making video presentations. These have greater speed. Supports user-defined keyboard layouts. It is efficient. It is very easy to useIt is more secure than other computer. But normal OS can't work in these computers. Also cost of E-Ball is very high and it is difficult to understand if any problems occur in hardware components.

CONCLUSION

As years passes, the computer size is becoming smaller. Today's technology is at its peak point beyond what we could ever imagine. New inventions and innovations are emerging on daily basis. Our imaginations have dressed into reality and today it has become possible to have a whole computer in our pocket all the time. This ball computer has taken the computer technology to new horizons.

PARALLEL COMPUTING

KAVYA SURESH(15MCA08)

ROOPA M S(15MCA22)

Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously. Large problems can often be divided into smaller ones, which can then be solved at the same time.

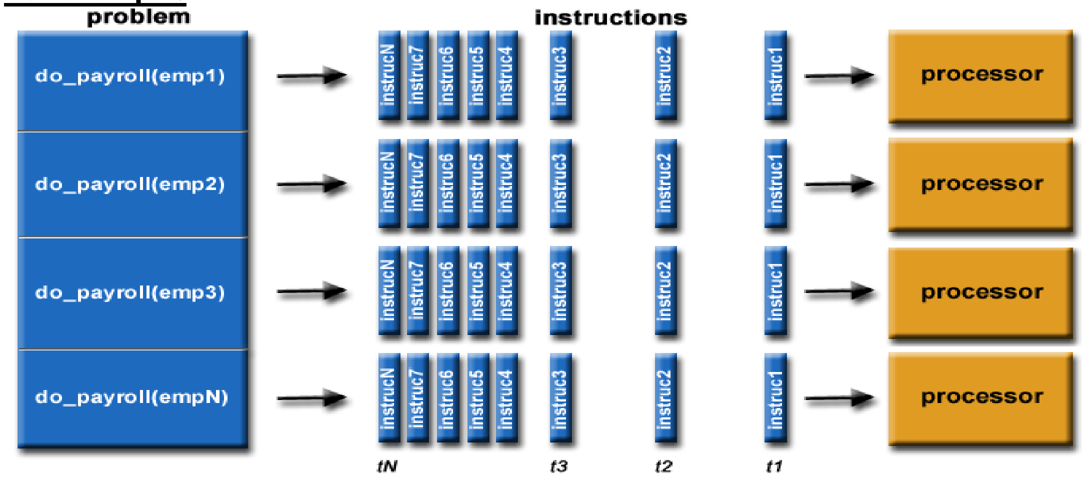
There are several different forms of parallel computing: bit-level, instruction-level, data, and task parallelism. Parallelism has been employed for many years, mainly in high-performance computing, but interest in it has grown lately due to the physical constraints preventing frequency scaling.As power consumption (and consequently heat generation) by computers has become a concern in recent years, parallel computing has become the dominant paradigm in computer architecture, mainly in the form of multi-core processors.

In parallel computing, a computational task is typically broken down in several, often many, very similar subtasks that can be processed independently and whose results are combined afterwards, upon completion. Parallel computers can be roughly classified according to the level at which the hardware supports parallelism, with multi-core and multi-processor computers having multiple processing elements within a single machine.

Parallel Computing: In the simplest sense, *parallel computing* is the simultaneous use of multiple compute resources to solve a computational problem:

- A problem is broken into discrete parts that can be solved concurrently
- Each part is further broken down to a series of instructions
- Instructions from each part execute simultaneously on different processors
- An overall control/coordination mechanism is employed

For example:



- The computational problem should be able to:

  - Be broken apart into discrete pieces of work that can be solved simultaneously;
  - Execute multiple program instructions at any moment in time;
  - Be solved in less time with multiple compute resources than with a single compute resource.

- The compute resources are typically:

  - A single computer with multiple processors/cores
  - An arbitrary number of such computers connected by a network

Advantages of parallel computing

- One advantage to parallel computing is the ability to process information
- Parallel programming saves time, allowing the execution of applications in a shorter wall-clock time.

Disadvantages of parallel computing

- Maintaining the system is difficult because it is complex.
- Are harder to implement, they're harder to debug or prove correct.

Applications of Parallel Computing:

- Data bases, Data mining.
- Networked videos and Multimedia technologies.
- Medical imaging and diagnosis.
- Advanced graphics and virtual reality.
- Collaborative work environments.

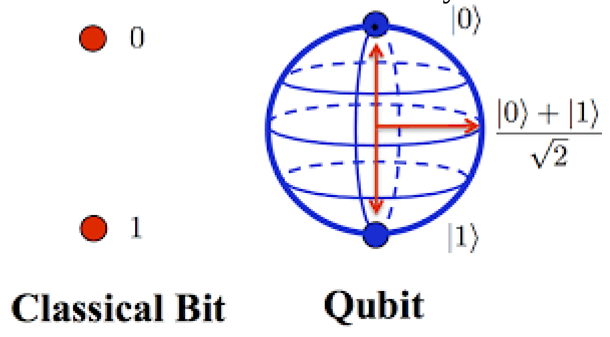
QUANTUM MECHANICS IN COMPUTER ARCHITECTURE

KUMARI POONAM(15MCA12)

SRISHTI PAL(15MCA11)

What is a Quantum Computer?
Quantum computer is a machine that performs calculation based on the law of quantum mechanics, which is the behavior of particles at the sub- atomic level. The Quantum computers work with the physics of quantum mechanics at its core. A quantum computer is built atom by atom.
Quantum computers replace traditional bits that are used in digital communications with quantum bits, or qubits. Potential applications can be found in a variety of fields, from medicine to space travel.

How Does Quantum Computing Work?
The quantum computers use atoms (quanta) as its physical system. Unlike in regular computing where the information is carried in either 0 or 1 bit, the quantum mechanics allows an atom to be in both 0 and 1 state simultaneously. This bit of data is called a qubit.



So,if there are equal number of qubits and the regular bits, then the qubits will hold twice the information, i.e. if there are n qubits in the supercomputer, then it will have 2^n different states. So experimentally, it can hold more information as compared to regular digital bits thereby increasing the speed of the system exponentially. Calculation is based on the law of quantum theory mechanism.

Quantum Theory
Quantum theory's development began in 1900 with a presentation by Max Planck to the German Physical Society, in which he introduced the idea that energy exists in individual units (which he called "quanta"), as does matter.

The Essential Elements of Quantum Theory:
- Energy, like matter, consists of discrete units, rather than solely as a continuous wave.
- Elementary particles of both energy and matter, depending on the conditions, may behave like either particles or waves.
- The movement of elementary particles is inherently random, and, thus, unpredictable.

- The simultaneous measurement of two complementary values, such as the position and momentum of an elementary particle, is inescapably flawed; the more precisely one value is measured, the more flawed will be the measurement of the other value.

The Quantum computer, by contrast, can work with a two-mode logic gate: XOR and a mode we'll call QO1 (the ability to change 0 into a superposition of 0 and 1, a logic gate which cannot exist in classical computing). In a quantum computer, a number of elemental particles such as electrons or photons can be used (in practice, success has also been achieved with ions), with either their charge or polarization acting as a representation of 0 and/or 1. Each of these particles is known as a quantum bit, or qubit, the nature and behavior of these particles form the basis of quantum computing. The two most relevant aspects of quantum physics are the principles of superposition and entanglement.

Superposition

Superposition is interference peaks from an electron wave in a double-slit experiment. An electron has a dual nature. Example is a quantum logical qubit state, as used in quantum information processing, which is a linear superposition of the "basis states."

Entanglement

Quantum entanglement is a physical phenomenon that occurs when pairs or groups of particles are generated or interact in ways such that the quantum state of each particle cannot be described independently of the others, even when the particles are separated by a large distance—instead, a quantum state must be described.

Application :

Physics

Materials Science
&
Engineering

Advantages

- Could process massive amounts of complex data.
- Ability to solve scientific and commercial problems.
- Process data in a much faster speed.
- Capability to convey more accurate answers.

Disadvantages

- Security and Privacy Issues.
- Ability to crack down password (s).
- Capability to break every level of encryption.
- Moral, ethical, social, and economic issues.

TRANSPORT TRIGGERED ARCHITECTURE (TTA)

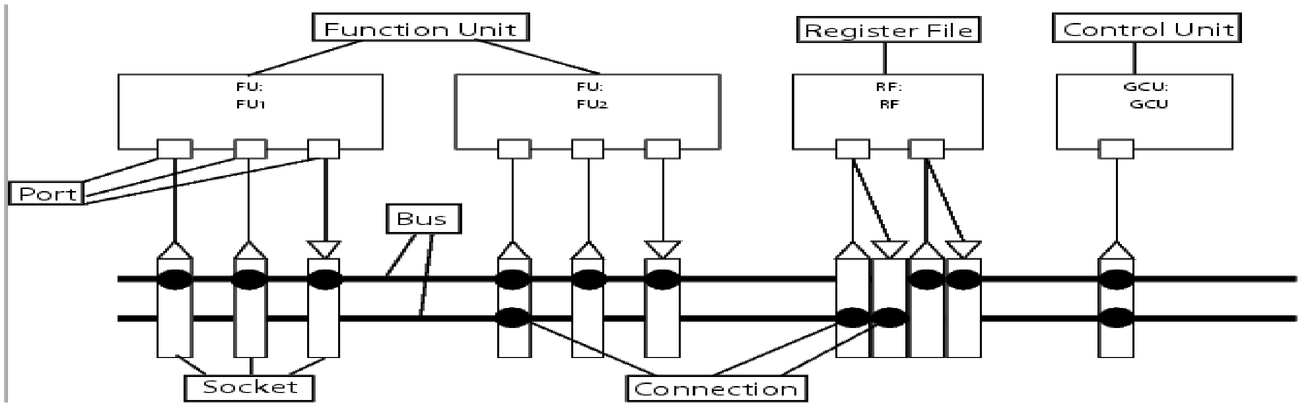
ASHA V (15MCA03)

NIKITHA J (15MCA18)

INTRODUCTION

In Computer Architecture, Transport Triggered Architecture (TTA) is a processor design philosophy where the processors internal data paths are exposed in the instruction set. All operation parameter reads and result writes are explicitly stated in the instruction set. Due to its modular structure, TTA is an ideal processor template for application-specific instruction-set processors (ASIP) with customized data path but without the inflexibility and design cost of fixed function hardware accelerators. A TTA instruction word is composed of multiple slots, one slot per bus, and each slot determines the data transport that takes place on the corresponding bus. The fine-grained control allows some optimizations that are not possible in a conventional processor. For example, software can transfer data directly between functional units without using registers.

STRUCTURE OF TTA



TTA processors are built of independent function units and register files, which are connected with transport buses and sockets.

FUNCTION UNIT

Each function unit implements one or more operations, which implement functionality ranging from a simple addition of integers to a complex and arbitrary user-defined application-specific computation..Each function unit may have an independent pipeline. In case a function unit is fully pipelined, a new operation that takes multiple clock cycles to finish can be started in every clock cycle..Function units that implement memory accessing operations and connect to a memory module are often called load/store units.

CONTROL UNIT

Control unit is a special case of function units which controls execution of programs. Control unit has access to the instruction memory in order to fetch the instructions to be executed. In order to allow the executed programs to transfer the execution (jump) to an arbitrary position in the executed program, control unit provides control flow operations. A control unit usually has an instruction pipeline, which consists of stages for fetching, decoding and executing program instructions.

REGISTER FILES

Register files contain general purpose registers, which are used to store variables in programs. Like function units, also register files have input and output ports. The number of read and write ports, that is, the capability of being able to read and write multiple registers in a same clock cycle, can vary in each register file.

TRANSPORT BUSES AND SOCKETS

Interconnect architecture consists of transport buses which are connected to function unit ports by means of sockets. Due to expense of connectivity, it is usual to reduce the number of connections between units (function units and register files). A TTA is said to be fully connected in case there is a path from each unit output port to every unit's input ports.

ADVANTAGES OF TTA

- Extreme simplicity and therefore quick design time.
- Great flexibility and performance scalability.
- Extra fine grain parallelism as a result of separating ALU (Arthematic Logic Unit) functionality into its individual components.

DISADVANTAGE OF TTA

- Low code density.
- This increases the code size, requiring a larger instruction memory to store TTA programs compared to dynamically scheduled processors.

DIGITAL SIGNAL PROCESSOR ARCHITECTURE

SHINY.D 15MCA25

What is a Digital Signal Processor?

A Digital Signal Processor, or DSP, is a specialized microprocessor that has an architecture which is optimized for the fast operational needs of digital signal processing. A Digital Signal Processor (DSP) can process data in real time, making it ideal for applications that can’t tolerate delays. Digital signal processors take a digital signal and process it to improve the signal into clearer sound, faster data or sharper images. Digital Signal Processors use video, voice, audio, temperature or position signals that have been digitized and mathematically manipulate them. A digital signal processor is designed to perform these mathematical functions rapidly. The signals are processed so the information contained in them can be displayed or converted to another type of signal.A digital signal processor (DSP) is an integrated circuit designed for high-speed data manipulations, and is used in audio, communications, image-manipulation, and other data acquisition and data-control applications.

The goal of DSPs is usually to measure, filter or compress continuous real-world analog signals. Most general-purpose microprocessors can also execute digital signal processing algorithms successfully, but dedicated DSPs usually have better power efficiency thus they are more suitable in portable devices such as mobile phones because of power consumption constraints. DSPs often use special memory architectures that are able to fetch multiple data or instructions at the same time.

Working of Digital Signature Processor

Digital technology such as personal computers (PCs), assist us in many ways: writing documents, spell checking, and drawing. Unfortunately, the world is analog, and electronic analog computers are not as versatile as digital computers. Therefore, in order to make use of the tremendous processing power that digital technology offers us, we must do the following:

1) Convert the analog signals into electrical signals, using a transducer (such as a microphone)

2) Digitize these signals (i.e., convert them from analog to digital using an analog-todigital converter (ADC)). Once the signal is in digital form, our computer can easily process it through a digital signal processor. The DSP specializes in processing these signals, which makes it slightly different from microcomputers, microcontrollers, or general-purpose microprocessors. After the DSP has processed the signal, the output signal must be converted back to analog form so that we can sense it.

Why Do We Need Digital Signal Processors

Can we not use a general-purpose microprocessor to process signals as well?

General-purpose microprocessors are quite slow in performing multiply and divide operations. They will typically sequentially execute a series of shift, add, and subtract operations from their microcode to perform a single multiply operation, and may consume many cycles to complete. The DSP performs multiplication in a single cycle by implementing all shifts and add operations in parallel. The circuitry is relatively complex which requires 1 to 3 transistors. The benefit is very fast multiplication, which is required for processing most of the digital signals.

Most DSPs have a specialized instruction that allows them to multiply, add, and save the result in a single cycle. This instruction is usually called MAC (short for Multiply, Accumulate)

Components of a Digital Signal Processor System

DSP systems consist of a DSP chip, memory, possibly an analog-to-digital converter (ADC), a digital-to-analog converter (DAC), and communication channels. Not all DSP systems have the same architecture with the same components. The selection of components in a DSP system depends on the application. For example, a sound system would probably require A/D and D/A converters, whereas an image processing system may not.

Types of Digital Signal Processors

There are many different kinds of programmable digital signal processors. The most common types categorized by Clock Frequency, RAM size, Data Bus Width, ROM Size, Flash size, packaging type, MMAC/MIPS/FLOPS and I/O Voltage

Applications

Since their introduction to the market, DSPs have found a wide variety of applications. They are used in everyday hi-fi systems as well as high-end virtual-reality applications. Generally, DSP is not an expensive technology. Some practical DSP systems are:

- Hi-Fi Equipment
- Toys
- Videophones
- Modems
- Phone Systems
- 3D Graphics Systems
- Image Processing Systems

THE MACHINE

J. MEGHA(152MCA32)

Companies are generating so much data these days that many are seeking new ways to crunch it, even beyond the capabilities of today’s computers.

“The Machine” is being developed by the HP Company also known as HPE (Hewlett-Packard Enterprises). “The Machine” was announced by Hewlett-Packard in Las Vegas in 2014, it presented the project as a near-complete over all of traditional computer architecture. Gone were the CPU-centric architecture, the slow copper communications, and the messy hierarchy of traditional memory. In their place, specialized computing cores, speedy light-carrying photonic connections, and a massive store of dense, energy-efficient memristor memory. The resulting computer, its designers say, will be efficient enough to manipulate petabyte-scale data sets in an unprecedented fashion, expanding what companies and scientists can accomplish in areas such as graph theory, predictive analytics, and deep learning in a way that could improve our daily lives.

HPE’s Machine research project involves a new type of computing design that would alter the way computer servers, a core HPE product line, are built. As detailed in the trade publication The Register, the Machine, unlike a traditional server, uses advanced photonics (the transmission of information by lights) “The Machine” has been built to focus on memory. But that shift means that HPE had to redesign how a server operates including the chips that it uses, the construction, and the overall body, or chassis.

The Machine, if successful, would allow for servers that are hundreds of thousands of times more powerful than today’s devices. Eventually, HPE plans to use an experimental type of memory chip called a memristor in the Machine

The Machine deals with the three main important concepts:-

- Connection
- Security
- Knowledge

A new architecture is needed, we're told, a memory-driven architecture to reignite growth in computing power.Hence HPE's obsession with a new computer architecture that has large amounts of reasonably fast persistent memory as the centerpiece, providing tracts of storage for applications to access directly at high speed to manipulate data.

The basic unit of The Machine is a collection of hardware grouped in a Load Store Domain. A Load Store Domain consists of:

- Multiple independent Compute Nodes
- Independent operating systems
- Local memory
- Load/store access to shared, in-memory storage
- Shared byte-addressable persistent memory:

Non-volatile wrt operating system life cycle

- Global address space
- Hardware access control
- Accessed with standard CPU load/store instructions.

Conclusion:-

The purpose behind **“The Machine”** is to create a new kind of computer that isn’t beholden to the rules and limitations that dictate the performance, power consumption, and form factor of current computers. While no one’s arguing that Intel hasn’t done some amazing work in keeping Moore’s law alive and kicking, it’s clear that the current computing paradigm of transistor-transistor logic, RAM, and miles of copper wire is drawing ever closer to a conclusion. HP hopes that by leveraging new technologies, such as memristors (pictured top) and silicon photonics (optical transistors and wiring), we can essentially hit a big Reset button, giving us another attempt at building computers that are super-fast but still very energy efficient.

SPECULATIVE EXECUTION

KARIMA (152MCA33)

ASMA FARHEEN (152MCA36)

Speculative execution in computer systems is doing work, the result of which may not be needed. This performance optimization technique is used in pipelined processors and other systems. Speculative execution is a performance optimization. The main idea is to do work that may not be needed. The target is to provide more concurrency if extra resources are available. Modern pipelined microprocessors use speculative execution to reduce the cost of conditional branch instructions using schemes that predict the execution path of a program based on the history of branch executions.

It turns out that in order to improve performance and utilization of computer resources, some instructions have to be scheduled ahead of time in a place that is not determined that such instructions have to be executed at all, ahead of branch. In compiler optimization for multiprocessing systems, speculative execution involves an idle processor executing code in next processor block, in case there is no dependency on code that could be running on other processors.

The benefit of this scheme is reducing response time for individual processors and overall system. The compiler is limited in issuing speculative execution instruction, since it requires hardware assistance to buffer effects of speculatively-executed instruction. Without hardware support, compiler could only issue speculative instruction which have no side effects in case of wrong speculation.

TYPES OF SPECULATIVE EXECUTION:

Eager execution:

Eager execution is a form of speculative execution where both sides of the conditional branch are executed; however, the results are committed only if the predicate is true. With unlimited resources, eager execution (also known as *oracle execution*) would in theory provide the same performance as perfect branch prediction. With limited resources eager execution should be employed carefully since the number of resources needed grows exponentially with each level of branches executed eagerly.

Predictive execution:

Predictive execution is a form of speculative execution where some outcome is predicted and execution proceeds along the predicted path until the actual result is known. If the prediction is true, the predicted execution is allowed to commit, however if there is a misprediction, execution has to be unrolled and re-executed.

Lazy execution:

Lazy execution does not speculate. The incorporation of speculative execution into implementations of the Haskell programming language is a current research topic. Eager Haskell is designed around the idea of speculative execution. A 2003 PhD thesis made GHC support a kind of speculative execution with an abortion mechanism to back out in case of a bad choice called *optimistic execution*.It was deemed too complicated.