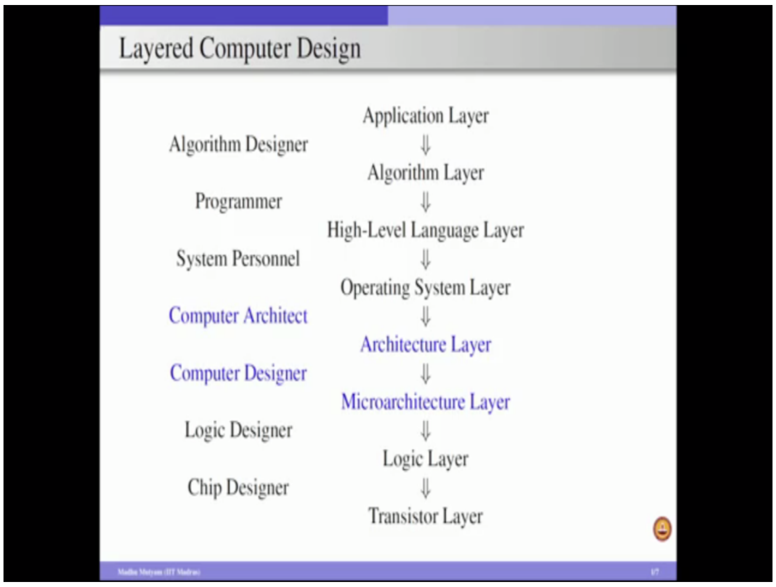
Computer Architecture

Prof. Madhu Mutyam
Department of Computer Science and Engineering
Indian Institute of Technology, Madras.

Lecture -01
Introduction

Welcome to NPTEL's MOOC on computer architecture. So, as part of this module, I am going to discuss the overview of computer architecture course and we define what is Computer Architecture and so what is the syllabus we cover and what are the reference books and so on. So, before defining the computer architecture, I would like to explain the layered view of computer design.

(Refer Slide Time: 00:40)



Layered Computer Design

	Application Layer
Algorithm Designer	⇓
	Algorithm Layer
Programmer	⇓
	High-Level Language Layer
System Personnel	⇓
	Operating System Layer
Computer Architect	⇓
	Architecture Layer
Computer Designer	⇓
	Microarchitecture Layer
Logic Designer	⇓
	Logic Layer
Chip Designer	⇓
	Transistor Layer

If you see a computer design in a layered view, there are several layers starting from the application layer and all the way up to transistor layer. And each of these layers specifies set of operations and for example, if you consider application layer. So, talks about the type of applications that run on a computer. In general, a computer can run any application and so on, but in reality when we design a computer, we design a computer for specific purposes. For example, if we consider super computer. Our target applications for super computer are high end applications, which are the weather modeling, molecular simulations and etcetera.

On the other hand, if we consider a desktop computer our target applications are email, word processing and etcetera. So, depending on the type of applications for which we want to design a computer, we need to define our requirements and we tune our designs for optimizing those requirements. After this application layer, the next layer is algorithmic layer. So, in the algorithmic layer typically once the application is defined.

Now, we need to see, what are the characteristics of the applications. Such as what type of data this application takes, what type of operations we can perform, how the operations sequencing happen and is there any implicit parallelism among the operations, we have to define all these things. And once we define these things, the next step is to come with the step by step procedure to compute the outputs for the application. So once the algorithmic layer is completed, effectively like, we have an algorithm the next one is we need to translate this algorithm into a high level programming language.

Effectively, we code this algorithm into a high level programming language, such as C, C++, Java or Fortran. Once we code a high level program, the next step is, we need to run this high level program on the computer. But to run that we need operating system layer, which effectively specifies the system software which include operating systems, compilers, linkers, assemblers and so on. So, these are the actual routines which help executing our high level program on the real hardware.

And the next comes the architectural layer. So, this architectural layer specifically describes the instruction set architecture. This instruction set architecture is nothing but the set of the instructions, instruction formats, operands, addressing modes, I/O-mechanisms, memory model and all these things. And once we have an ISA, our system software level, especially the complier, converts the program into the machine understandable form by using this ISA.

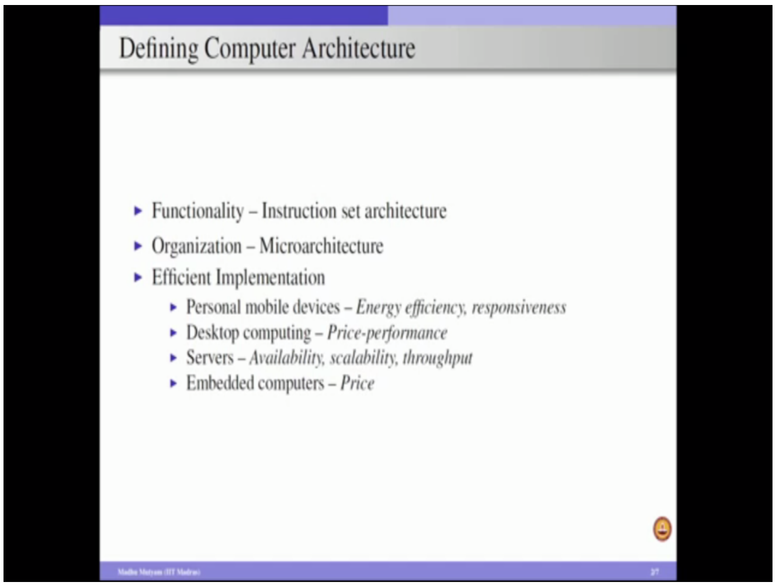
Through this architectural level, all the other layers are actually corresponds to hardware and all the other layers above architectural layer correspond to the software. In other words the architectural layer is the interface between the software and hardware. Once ISA is defined, our next task is to have the functional units which execute the instruction specified by ISA and that comes in microarchitectural layer. In other words microarchitecture or also called as organization, deals with the functional units and the interconnection among these functional units to realize the ISA specified at the architectural layer.

And once the microarchitectural layer is defined, the next one is the logic layer which actually consists of digital circuits both combinational and sequential circuits. And the components in the logic layer are the flip flops and gates. Using these flip flops and gates, we build digital circuits both combinational sequential circuits. And the last layer is the transistor layer which actually consists of transistors, resistors, capacitors and all these are used to build the gates and the transistors.

So, effectively, if you see this, the layered view of computer design, except the application layer, all other layers implement the layer above. For example, a transistor layer implements whatever is available at the logic layer. And the logic layer implements whatever is defined at the microarchitectural layer. And microarchitectural layer defines, implements, whatever is there in architectural layer and so on.

So, because our course is on computer architecture, we mainly focus on two layers - those are architectural layer and microarchitectural layer. So, having discussed this layered view of computer design, now we will define what is computer architecture.

(Refer Slide Time: 06:48)



Defining Computer Architecture

- Functionality – Instruction set architecture
- Organization – Microarchitecture
- Efficient Implementation
  - Personal mobile devices – *Energy efficiency, responsiveness*
  - Desktop computing – *Price-performance*
  - Servers – *Availability, scalability, throughput*
  - Embedded computers – *Price*

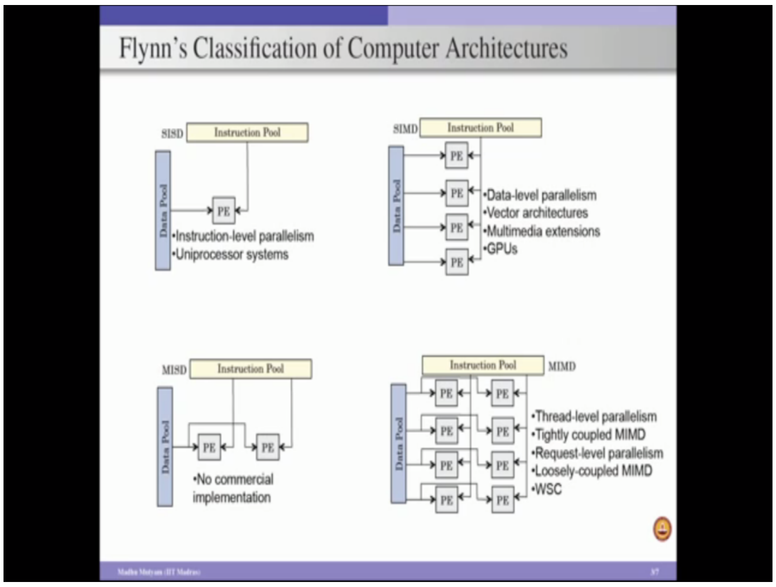
The computer architecture defines mainly the functionality or functional requirements of a computer that we want to design, which is specified by the instruction set architecture. Next one is the organization and also called as the microarchitecture, which is nothing but the interconnection of functional units to realize ISA and finally, the efficient implementation. How do we define one implementation is efficient?

So, to do that first of all we need to see what are the classes of computers because depending on the type of the class of computer our efficiency matrix can be defined. If you see different classes of computers, we have personal mobile devices, desktop computers, servers and embedded computers. And each of these computers have different functional requirements and they need to be designed for optimizing different metrics and so on.

If we consider, for instance personal mobile devices, here the main important thing is responsiveness and predictiveness, and because these personal mobile devices are operated using battery, so our main design requirement is energy efficiency. When we want to design a personal mobile device, we have to design in such a way that the energy efficiency is improved. On the other hand, if we consider servers, because the servers are used by multiple users simultaneously to execute their programs and so on.

So, here the requirements, functional requirements, are availability and scalability. And because these are multiple users are using, so when we design a server type of system, we have to design the system to optimize the throughput and similarly, for the other type of computers.

(Refer Slide Time: 08:50)



Flynn's Classification of Computer Architectures

- •Instruction-level parallelism
- •Uniprocessor systems
- •Data-level parallelism
- •Vector architectures
- •Multimedia extensions
- •GPUs
- •No commercial implementation
- •Thread-level parallelism
- •Tightly coupled MIMD
- •Request-level parallelism
- •Loosely-coupled MIMD
- •WSC

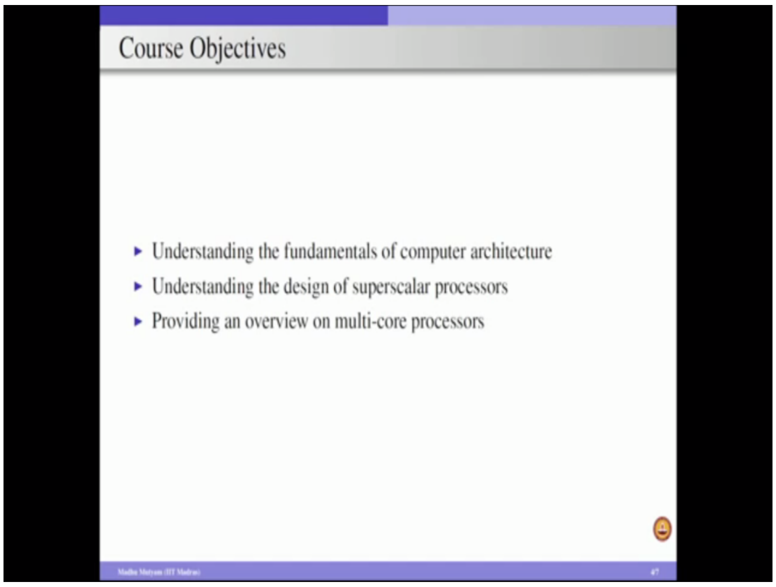
The Flynn's classification of computer architectures, there are four groups of computer architectures starting with SISD, single instruction and single data stream. So, we have instructions are taken from a single instruction stream and then these instructions are operated on single data stream. And here typically we exploit the instruction level parallelism among the instructions of the program.

And the example processors are computers belong to these class of computer architecture are uniprocessor system. The next class of computer architecture is SIMD, single instruction and multiple data. Here, we have instruction from a single instruction stream, but these instructions are operated on multiple data streams. Effectively, we can exploit the parallelism available at the data level. And the example systems that belong to this class of computer architecture are vector architectures, array processors, GPUs, multimedia extensions and so on.

And the third class of computer architecture is MISD, multiple instructions and single data stream, but actually this class of computer architecture is defined just for the completeness sake, but there is no commercially viable implementation of systems belonging to this particular class. And the last class of computer architecture is MIMD, multiple instructions and multiple data. Here instructions are taken from multiple instruction streams and these can be operated on multiple data streams simultaneously.

So, effectively we can exploit parallelism available at thread level. And example systems that belong to this class of computer architecture are multicore processors.

(Refer Slide Time: 11:01)



Course Objectives

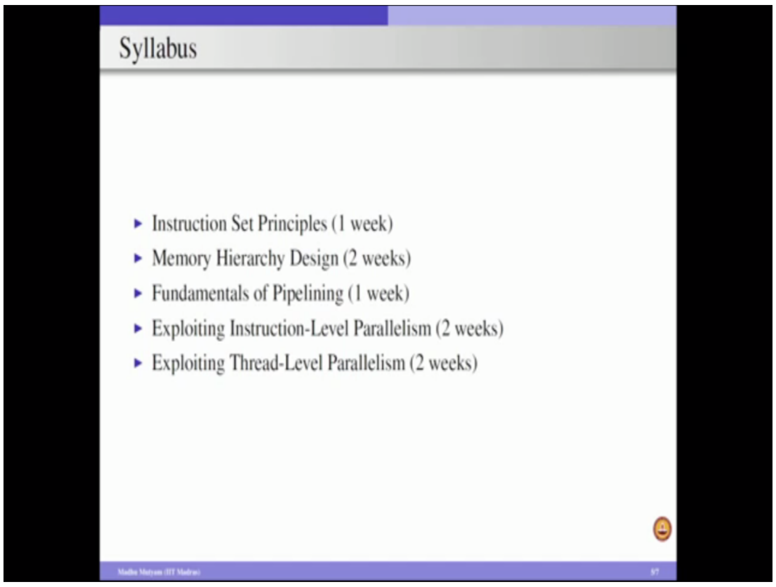
- Understanding the fundamentals of computer architecture
- Understanding the design of superscalar processors
- Providing an overview on multi-core processors

Having defined the computer architecture and discussed the various classes of computer architectures and so on, now we are going to discuss the course objectives. By the way, in this course we mainly focus on two classes of computer architectures. One is SISD, single instruction single data stream and the second one is MIMD, multiple instruction and multiple data streams. Effectively we will consider a unicore processor systems as well as multicore processor systems and discuss the internals in detail.

So, main objectives of the course are three fold. We discuss mainly, the fundamentals of computer architecture, because understanding the fundamentals of computer architecture helps in building efficient computer systems and so on. And next we will discuss in detail, understanding of the Superscalar processor design, because if you consider any of the current day multicore processors, each of the individual cores are internally designed as Superscalar processors.

Here, in this part we discuss how out-of-order execution happens and how dynamic execution happens and so on. And finally, the third objective of the course is to provide an overview on multicore processors. Especially, issues related to multi-core processors such as the cache coherency, synchronization and memory consistency.

(Refer Slide Time: 12:33)



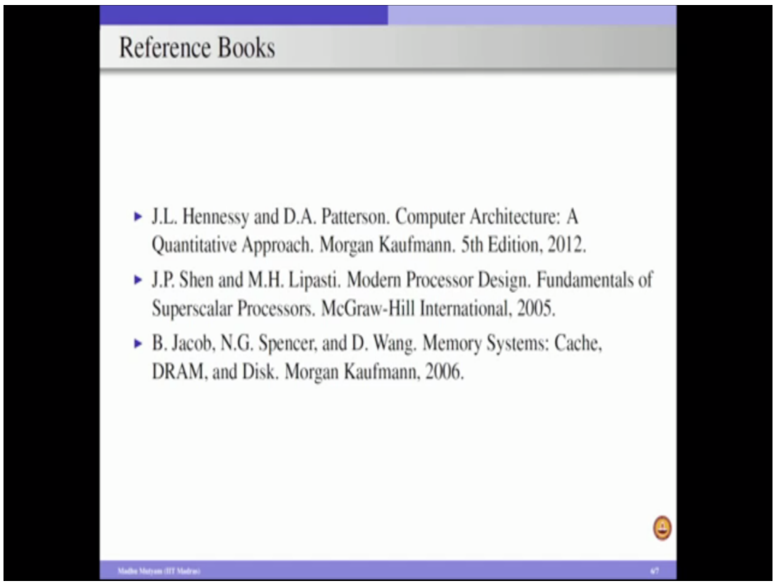
Syllabus

- Instruction Set Principles (1 week)
- Memory Hierarchy Design (2 weeks)
- Fundamentals of Pipelining (1 week)
- Exploiting Instruction-Level Parallelism (2 weeks)
- Exploiting Thread-Level Parallelism (2 weeks)

This course is a twenty hour lecture course and we consider the syllabus for this course like this. One week of lectures on instruction set principles, two weeks of lectures on memory hierarchy designs, specifically concentrating on cache memory design and DRAM based memory design. And one week of lectures on understanding the fundamentals of pipelining, issues related to pipelining such as pipeline hazards and how to resolve those pipeline hazards. And two weeks of lectures on exploiting instruction level parallelism available in SISD type of computer architectures.

And in this the section we also discuss the out-of-order execution, Superscalar processors, dynamic scheduling and all the other things. And finally, the last two weeks of the course, we discuss exploiting thread level parallelism, especially discussing things related to multicore architecture.

(Refer Slide Time: 13:44)



Reference Books

- J.L. Hennessy and D.A. Patterson. Computer Architecture: A Quantitative Approach. Morgan Kaufmann. 5th Edition, 2012.
- J.P. Shen and M.H. Lipasti. Modern Processor Design. Fundamentals of Superscalar Processors. McGraw-Hill International, 2005.
- B. Jacob, N.G. Spencer, and D. Wang. Memory Systems: Cache, DRAM, and Disk. Morgan Kaufmann, 2006.

Madhu Mutyam (IIT Madras) 6/7

So, for this course we use these three reference books "*Computer architecture - A quantitative approach by Hennessy and Patterson"* and "*Modern processor fundamentals of superscalar processors"*. Especially this is for understanding the internals of superscalar processors and this is by "*Shen and Lipasti"*. And the third one is especially for DRAM based memory based design, we consider "*Memory systems*" book by "*Jacobs, Spencer's and Wang*". So, with this brief overview, I would like to stop this module.

Thank you.