

Introduction

DSC 315: Computer Organization & Operating Systems

Dr. Laltu Sardar

School of Data Science,
Indian Institute of Science Education and Research Thiruvananthapuram (IISER TVM)



5th January, 2026



Introduction and Basic Principles



Most difficult to Learn



Most Boring Subject to Teach



Most difficult to Learn



Most Boring Subject to Teach



Heavy Meal



feeling Sleepy

Introduction

Some Historical Revolution

- Agricultural Revolution - Farming
- Industrial Revolution - invention of machines
- Information Revolution - computers

Presence of Computer

- automobile, WWW, search engines, cell phones

Classes of Computers

1 Personal Computer

- Laptop
- Desktop
- Mobile Phone

2 Server

- Desktop with greater power

3 Supercomputer

- High RAM and memory
- Memory in TB range

4 Embedded Computer

- Includes microprocessor
- TV, car, IoT, etc.
- Mainly special purpose small computers

Now - Personal mobile devices and cloud computing

Computer Organization

What is Computer Organization

- Study of the structural and operational aspects of computer systems
- Deals with how hardware components are interconnected
- Focuses on how instructions are executed at the hardware level
- Bridge between computer architecture and hardware implementation

Computer Organization vs Computer Architecture

- Organization: Physical structure, implementation details
- Architecture: Conceptual design, visible to programmers
- Example: Organization specifies cache size; architecture defines ISA

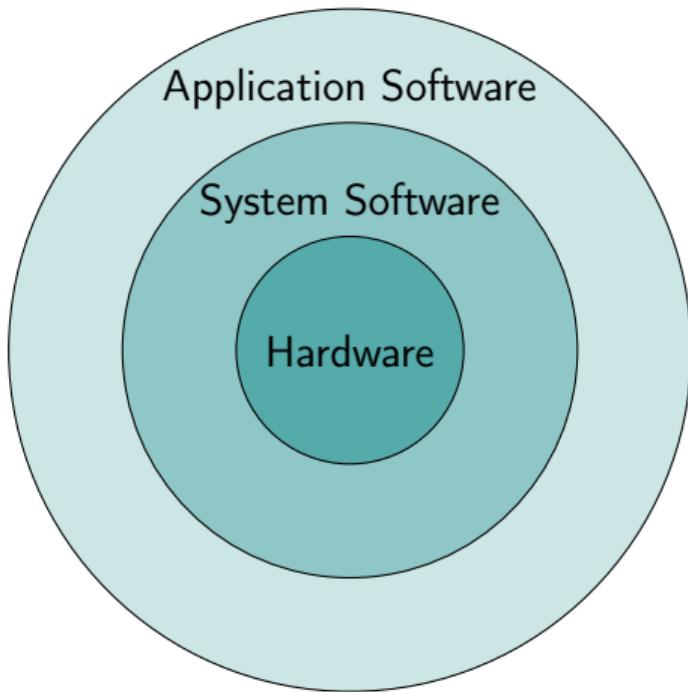
Need for Computer Organization

- Understand system performance
- Optimize software by knowing hardware limits
- Enable efficient system design
- Basis for advanced systems like CPUs, GPUs, embedded devices

Targets of the Course on CO

- 1 How a code executes on hardware
- 2 How software and hardware interact
- 3 How performance of a computer program is measured
- 4 How to improve performance
- 5 Parallel processing in multi-core era

Software and Hardware Layers



Application Software

- Text Editors, music players

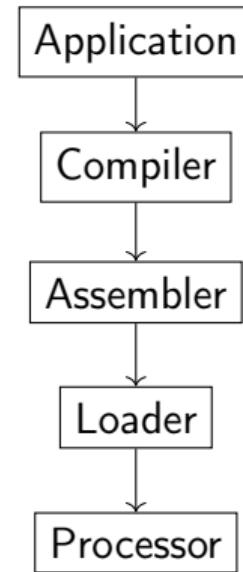
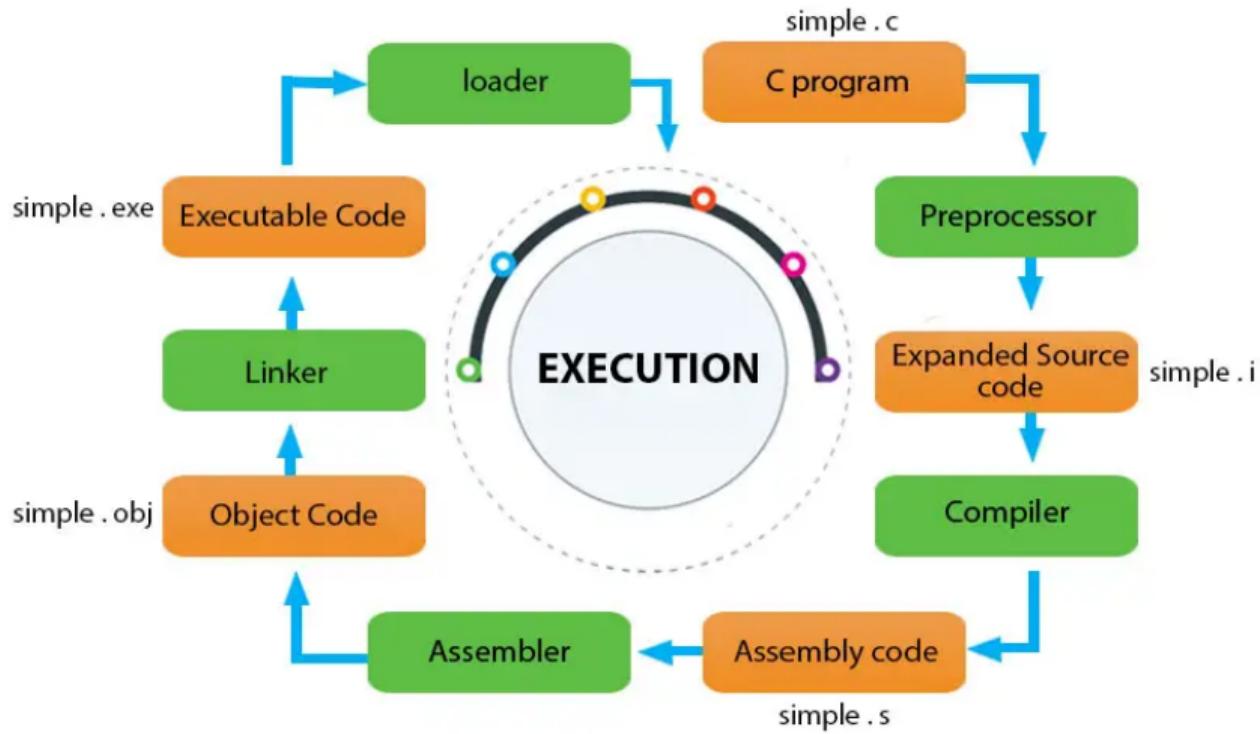
System software includes:

- Operating systems: Linux, iOS, Android, Windows
- Compilers
- Loader
- Assemblers

Application Software

- RAM, SSD, Motherboard, Processor Chip

Program Execution Workflow Diagram



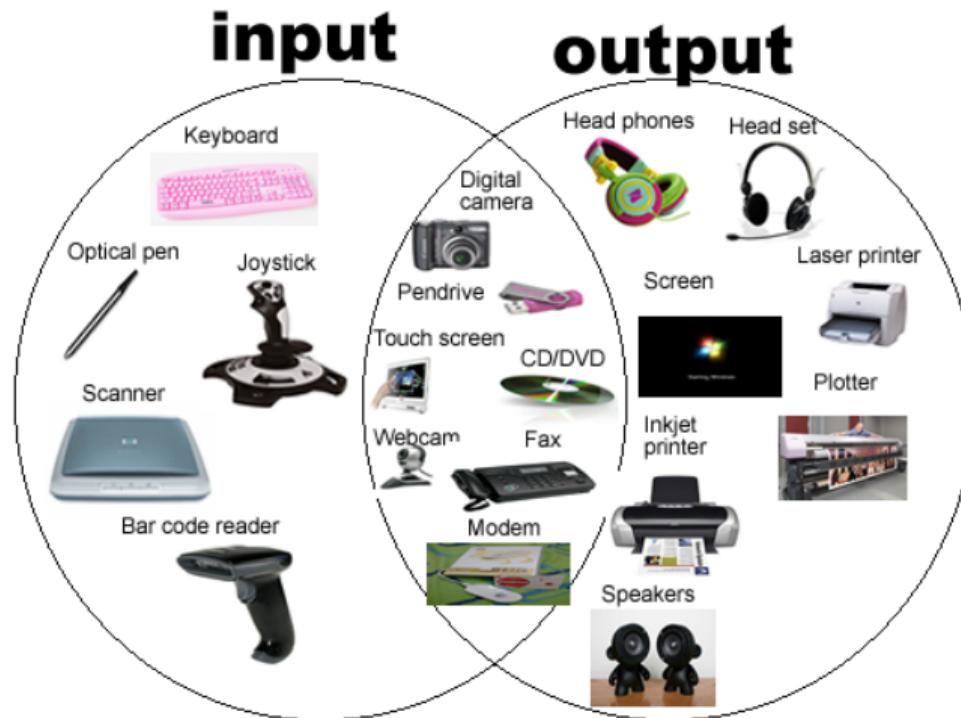
Program Execution Workflow

- 1 Application software or editor writes code
- 2 Compiler compiles and translates code to assembly language
- 3 Assembler converts assembly code to instructions or machine language
- 4 Loader loads instructions into the processor

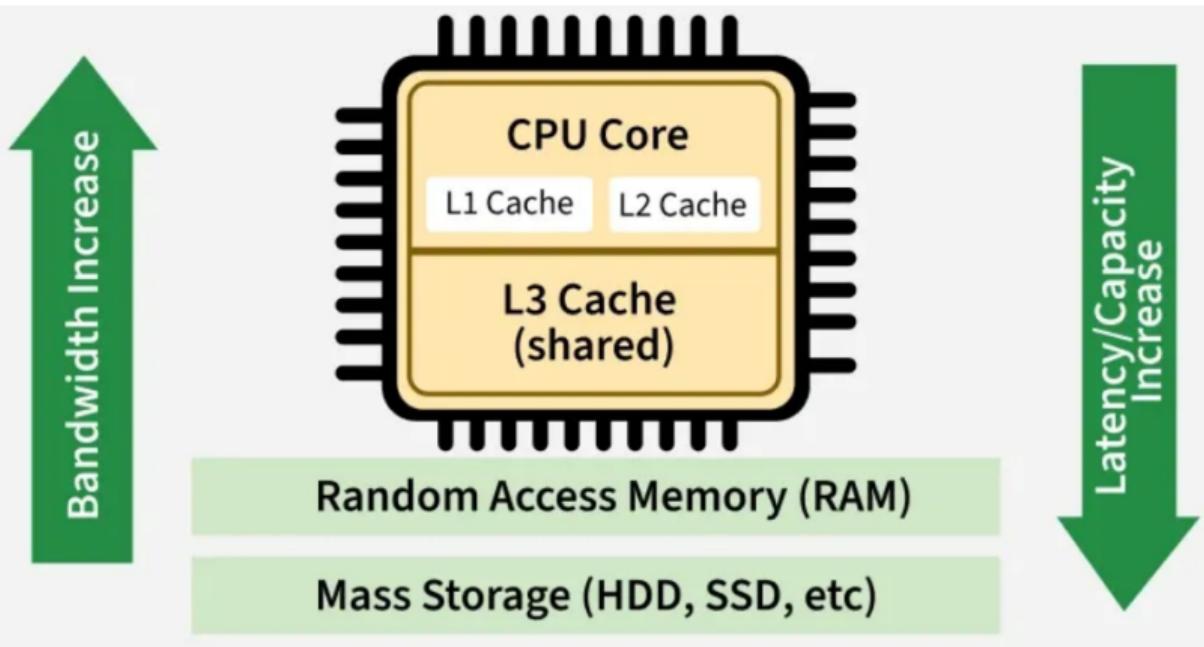
Role of Operating Systems

- Resource management
- Decides which program will run or be executed and when

Computer Hardware Components



Memory and Processing Units



Secondary Memory

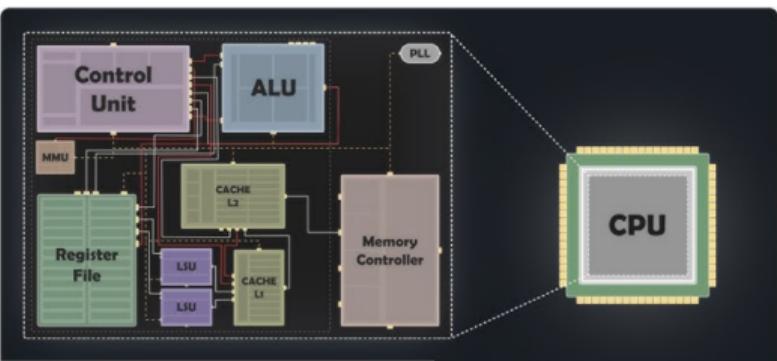
- HDD, SSD
- External hard disk or SSD

Primary Memory

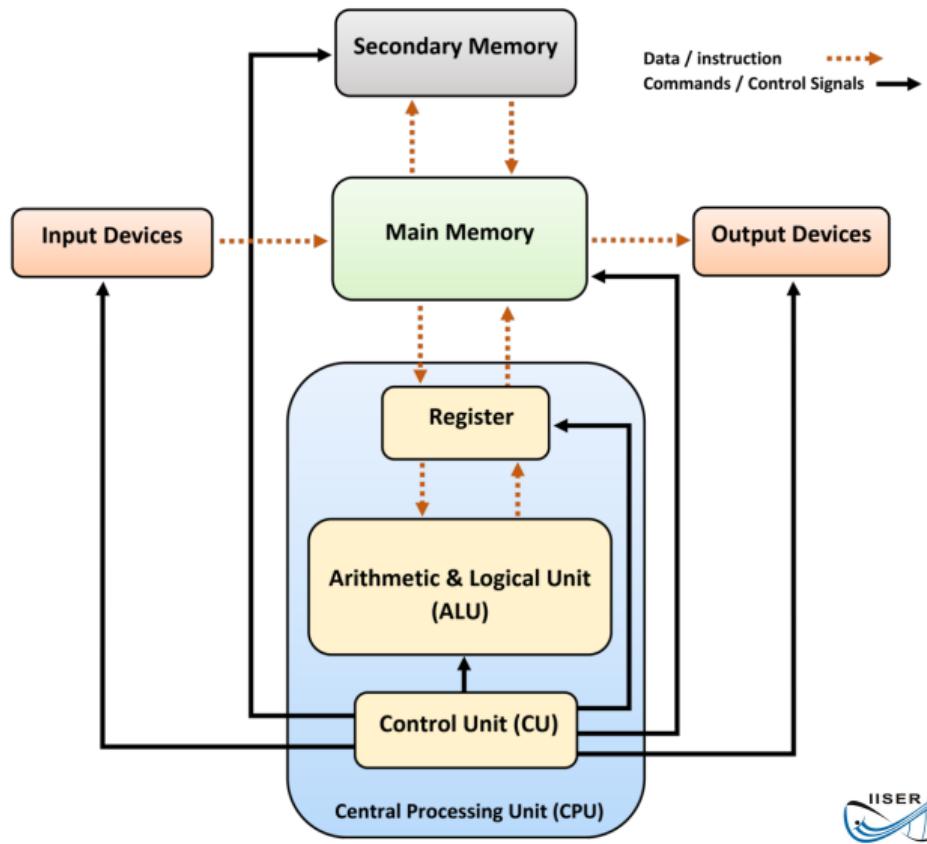
- RAM (volatile)

Processors

- Cache L3
- Cache L2
- Cache L1
- Registers



Computer Processor Architecture



Abstraction

What Is Abstraction

- Abstraction is the technique of managing complexity by hiding unnecessary details
- Each system level exposes only what is required to the level above
- Lower level implementation details are hidden

Why Abstraction Is Needed

- Modern computer systems are extremely complex
- Understanding everything at once is impractical
- Abstraction allows independent design and reasoning

Abstraction in Computer Organization

- Computer systems are structured as hierarchical layers
- Each layer builds on the services of the layer below
- Communication occurs through well defined interfaces

Abstraction Levels

- 1 Digital logic level
- 2 Microarchitecture level
- 3 Instruction Set Architecture (ISA)
- 4 Operating system level
- 5 Programming language level
- 6 Application level

Digital Logic Level

- Lowest abstraction level
- Uses gates, flip flops, and circuits
- Implements basic binary operations

Microarchitecture Level

- Describes internal CPU organization
- Includes registers, ALU, control unit, and pipelines
- Executes instructions defined by the ISA

Instruction Set Architecture

- Interface between hardware and software
- Defines instructions, registers, and addressing modes
- Same ISA can have multiple hardware implementations

Operating System Level

- Manages hardware resources
- Provides abstractions such as processes, memory, and files
- Shields programs from direct hardware access

Programming Language Level

- Provides high level constructs
- Examples include variables, loops, and functions
- Hides machine specific details

Application Level

- End user software
- Solves real world problems
- Requires no knowledge of hardware internals

Benefits of Abstraction

- Reduced system complexity
- Increased productivity
- Improved portability
- Independent hardware and software evolution

Example

- Programmer writes a print statement
- No need to understand memory layout or CPU execution
- Abstraction layers handle all low level operations

Performance Measurement

Why Performance Measurement?

- To evaluate **how fast** a computer system executes programs
- To compare different architectures and design choices
- To identify system bottlenecks
- To guide **optimizations** in hardware and software

The System Clock

- Processor operations are governed by a **system clock**
- Each operation **starts** with a **clock pulse**
- Processor **speed** depends on **clock frequency**
- Clock frequency is measured in **Hertz** (cycles per second)
- $3.5 \text{ GHz} = 3.5 * 10^9$ cycles per second.

Clock Generation and Terminology

- Clock signals are generated using a quartz crystal
- Continuous wave is converted into digital voltage pulses
- **Clock cycle** (clock tick): one pulse
- **Clock rate** (clock speed): pulses per second
- **Cycle time**: time between pulses

Clock Rate Constraints

- Clock rate depends on processor **physical layout**
- Signals **need time to propagate** and stabilize
- Different paths may have different delays
- Operations must be **synchronized** for correct voltage levels

Instructions and Clock Cycles

- Instruction execution involves multiple steps
- Examples: fetch, decode, execute, load, store, etc
- Most instructions require multiple clock cycles
- Some instructions take few cycles, others many
- clock speeds on different processors does not tell the whole story about performance.

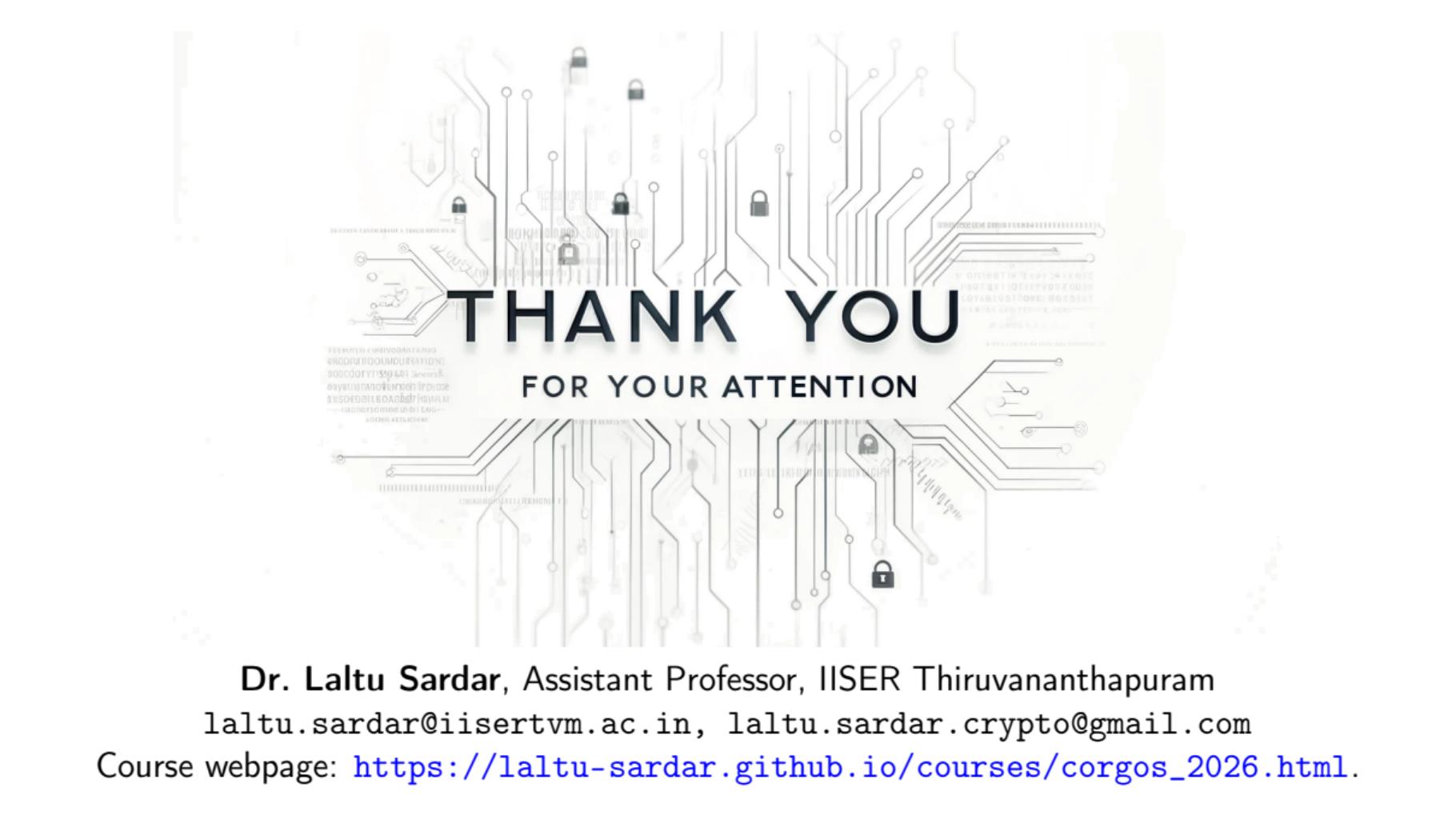
Processor Execution Time

The processor time T required to execute a program can be expressed as:

$$T = I_c \times CPI \times t$$

- I_c : Instruction count
- CPI : Cycles per instruction
- t : Processor clock cycle time

This equation captures the basic factors affecting execution time.



THANK YOU

FOR YOUR ATTENTION

Dr. Laltu Sardar, Assistant Professor, IISER Thiruvananthapuram

laltu.sardar@iisertvm.ac.in, laltu.sardar.crypto@gmail.com

Course webpage: https://laltu-sardar.github.io/courses/corgos_2026.html.