

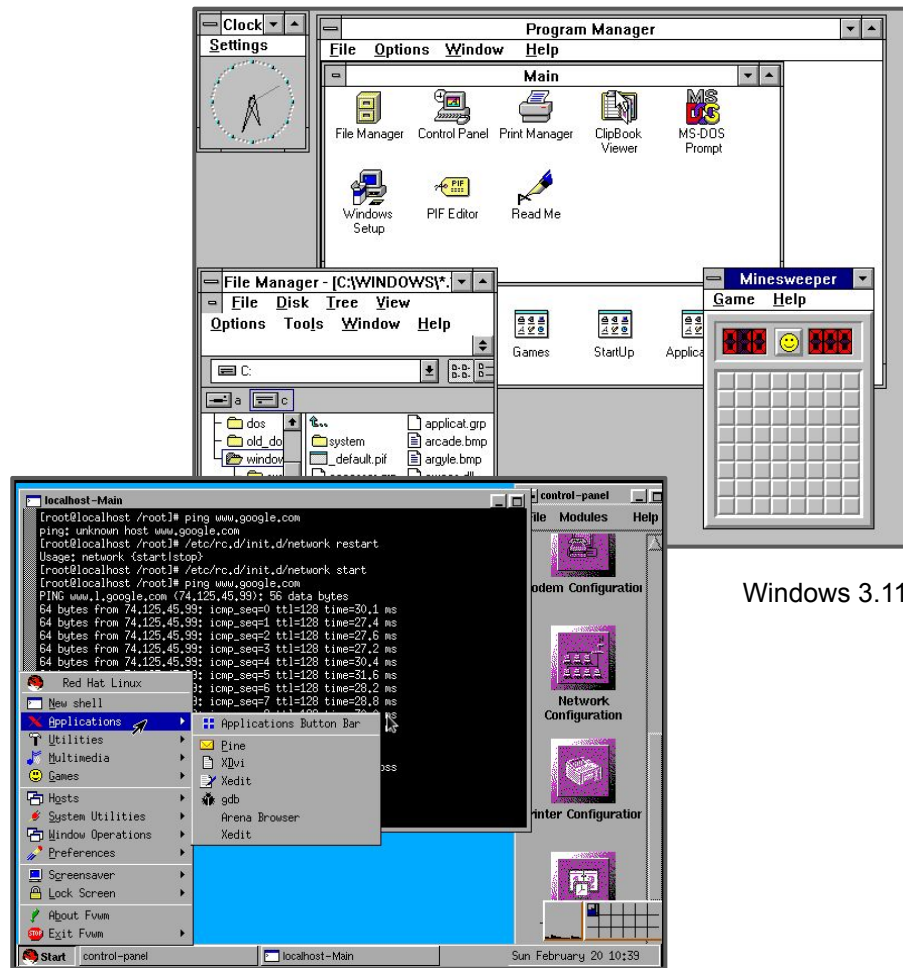
Operating Systems

CS-C3140, Lecture 1

Alexandru Paler

Contents

- Introduction to operating systems
- Practical arrangements of the course
 - Material (what to study)
 - Slides, Readings, Textbook, ...
 - Requirements (how to pass)
 - Exercises and exam
- Introduction to computer systems
 - Basics on computers
 - Processors, memory
 - Larger systems



Windows 3.11

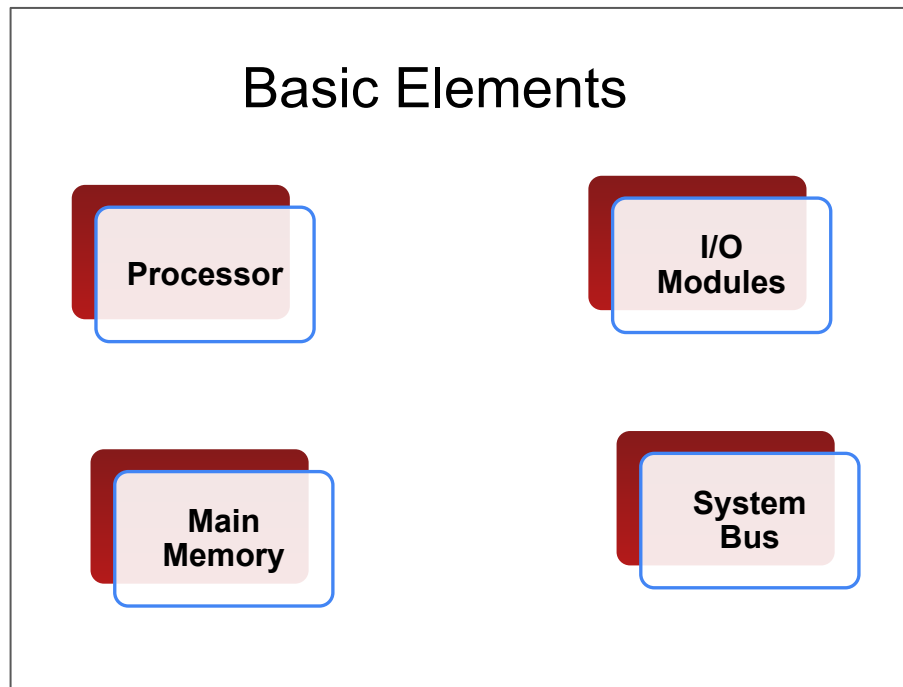
Learning objectives: ... understand the principles of operating system implementation

- Objectives
 - Systems in general (what do we "operate")
 - Software system structures
 - Processing, memory systems, storage systems
 - Concurrency and parallelism
 - Virtualization and distributed systems
- For CS students this is fundamental knowledge
 - For many others, this is essential
 - So, the course goes much beyond traditional issues

On systems and operating them ...

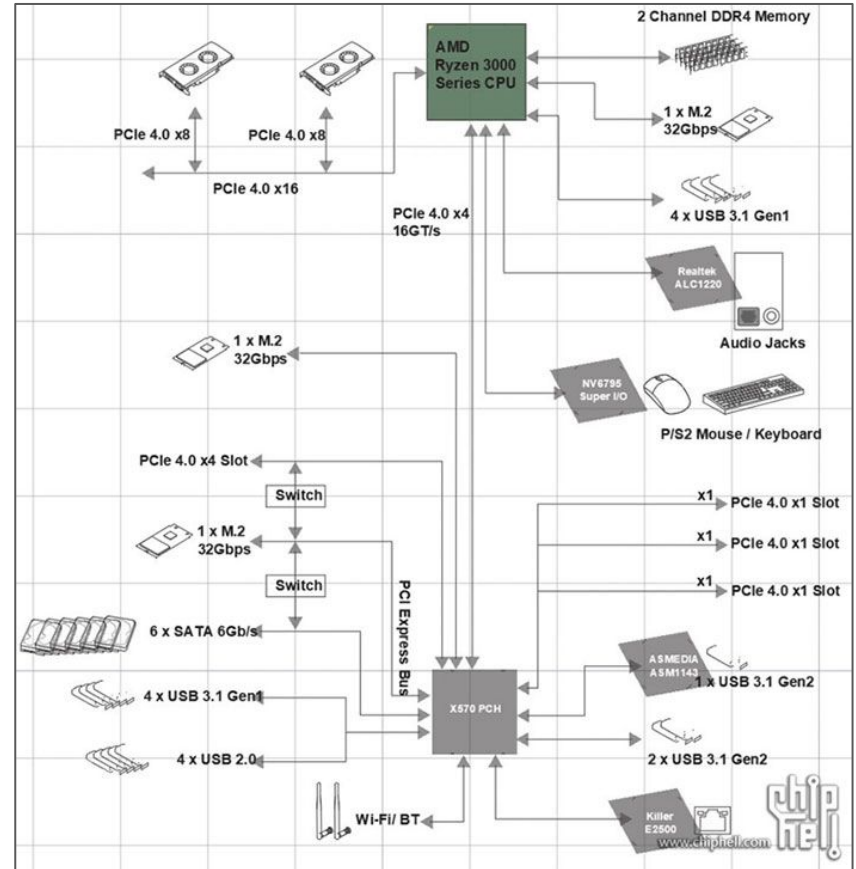
Operating System

- Exploits the **hardware resources** of one or more processors
- Provides a set of services to system **users**
- Manages secondary memory and I/O devices



Operating Systems – What are “systems”?

- “A set of **interrelated components** working **together** toward some **common** objective”
 - Structures
 - Delimitations
 - Subsystems / Components
 - The services
 - Inputs and outputs
 - Interaction with environment
- In practice
 - Requirements, Interfaces between subsystems, Component properties
 - Not all “things” are systems



Operating Systems – What is “operating”?

- Systems are used in a shared way
 - Multiple users
 - Not necessarily “the end users”.
 - Tasks, processes, etc. that use the system resources
 - Multiple components (or “subsystems”)
 - System resources that are used in a shared way
- Typically, the systems are complex
 - Abstraction of the resources are needed
- Sharing resources is complex
 - Management of the shared resources is needed
 - In systems, the resources are typically inter-dependent
- But security, dependability, etc. is also needed...



1960's – IBM System/360

Course arrangements

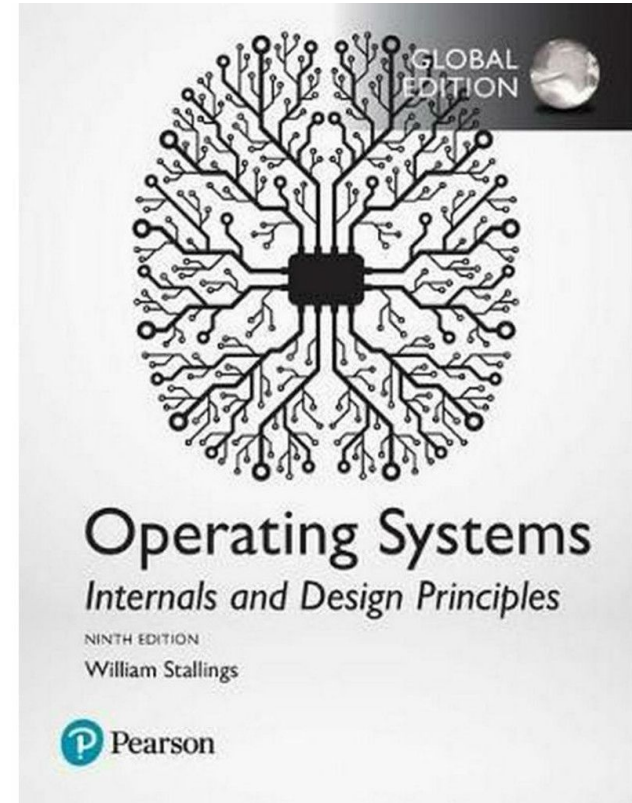
Nature of the course

- This is a standard course
 - A core course for SW system students
 - Similar course in any CS curricula
- Approach
 - The course is based on a textbook => exam
 - Note: the course is a “extended subset” of the book
 - The world is rapidly changing – and OSeS are also
 - Exercises are supporting the basic contents
- Connected to CS history
 - Understand what is in a typical system
 - historical reasons



Material and Requirements

- Materials
 - William Stallings: Operating Systems Internals and Design Principles (any recent edition)
 - Check the material & weekly pages on MyCourses
- Requirements
 - Exercises (mandatory)
 - Testing your understanding and skills
 - Final exam (mandatory)
 - Testing your overall knowledge on the course topics



Arrangements

- Staff

- *Alexandru Paler*, Ioana Moflic, Sneha Saj, Alex Ilov
- Discord server: <https://discord.gg/4rH8juDStu>

- Lectures

- A broad view on operating system concepts
- Weekly: Tuesday 8-10, T1 in CS building (this room)
- Outline slides will appear on MyCourses

- Exercises

- We will use A+ (instructions & readings there, published per round)
 - online each Friday at 22:00 Helsinki time
 - one week deadline: 100% points
 - one week extension (two weeks): 70% points
- Weekly: Q&A session – Tuesday 10-12 (starting next week)
- Weekly: Thursday 10-12 (starting next week)
- Weekly: Friday 10-12 (starting next week)
- The exercises are more on the practice (than the exam & lectures)

Grading

- The grade is based on points
 - The exercises: 8 (?) rounds and 20 points/round (= max. 160 total)
 - The exam points are scaled to max. 160 points total
- To pass the course you have to pass both the exam and the exercises
 - A minimum number of points for the exam (to be announced separately)
 - Per round limits for the exercises (you have to pass 6/8 rounds)

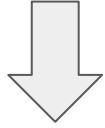
Introduction to computer architecture

Computers and operating systems

- A computer is a device consisting of
 - CPU
 - Memory
 - I/O peripherals: disk, display, network card
- A computer is executing programs
 - Perform arithmetic or logical computations
 - Have control
 - Do input and output (I/O)
- An operating system is a special program
 - Controls access to computer peripherals
 - Enables other programs to run
- How many different operating systems have you used?



Processor and Main Memory



Controls the
operation of the
computer

Performs the data
processing
functions

Referred to as the
*Central Processing
Unit (CPU)*

- Stores data and programs
- Typically volatile
- Contents of the memory is lost when the computer is shut down
- Referred to as real memory or primary memory

I/O Modules and System Bus

Provides for communication among processors, main memory, and I/O modules

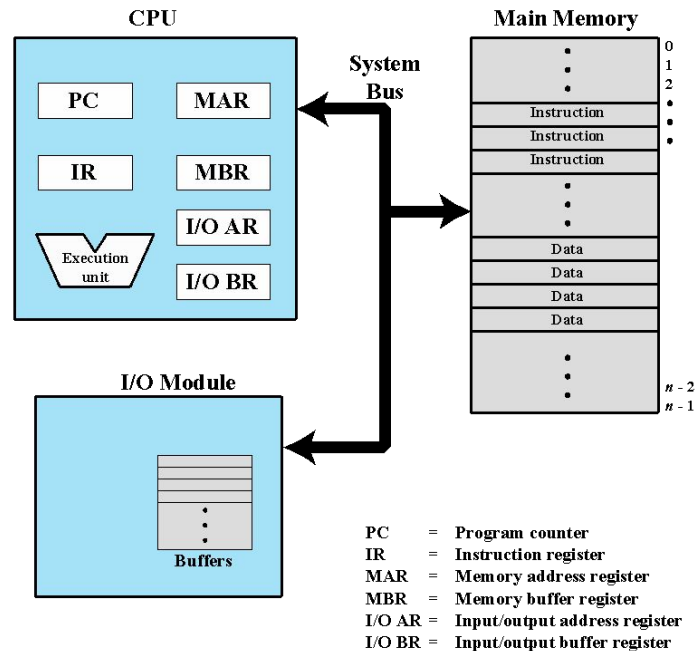
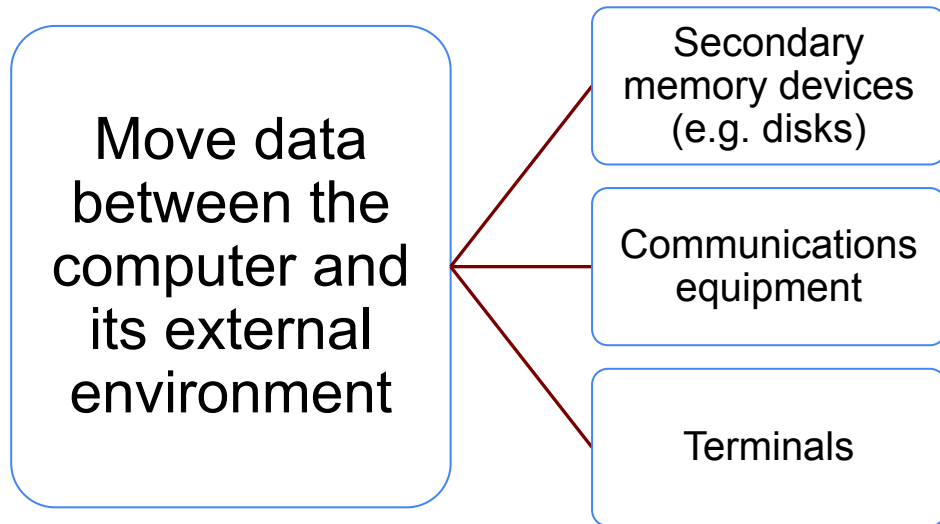
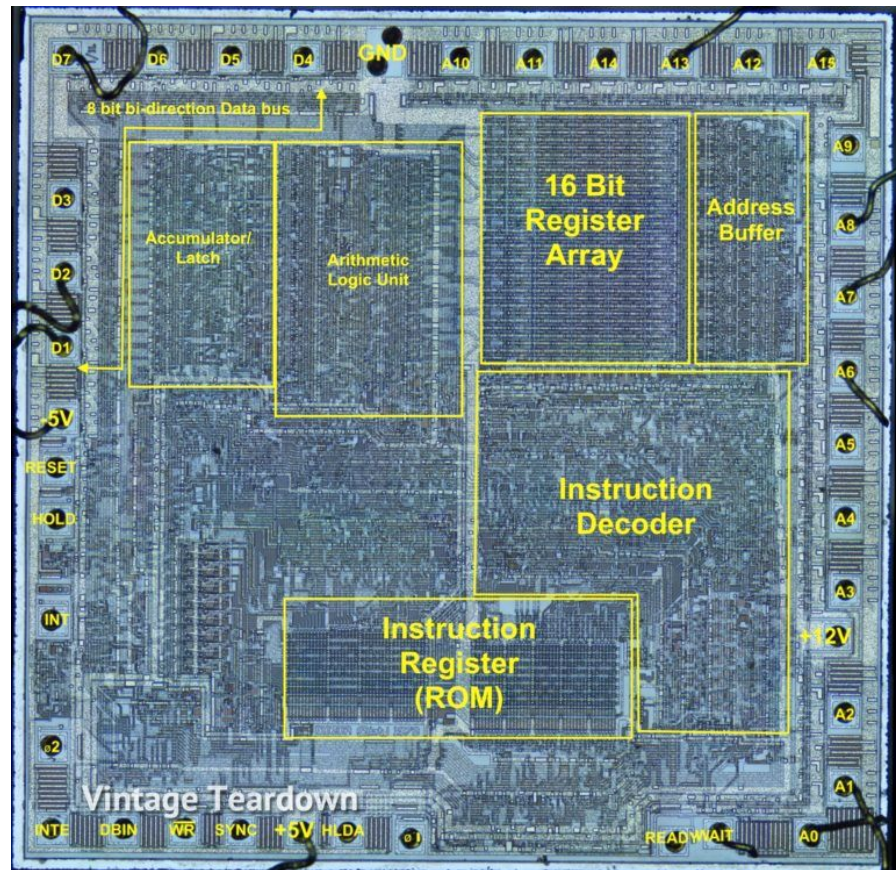


Figure 1.1 Computer Components: Top-Level View

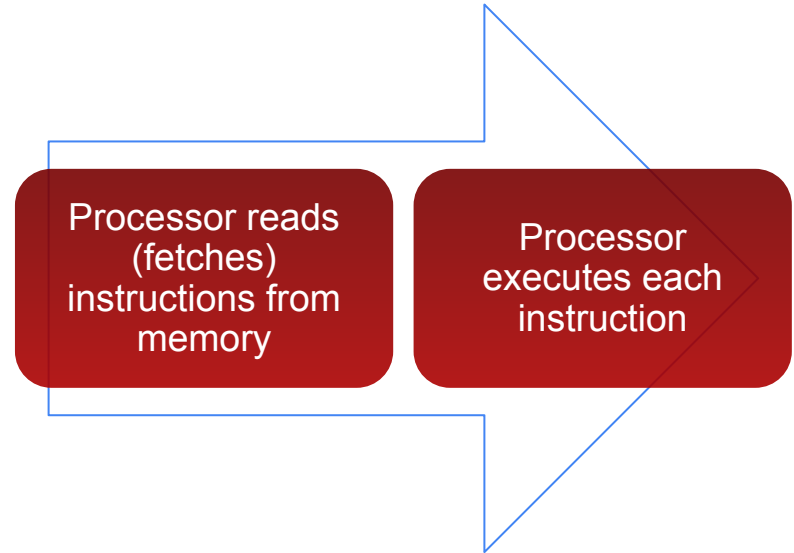
Processor Registers

- User-visible registers
 - available to all user programs or applications
 - data registers and address registers
 - some architectures save/restore all user-visible registers and others require user to do it
- Control and status registers
 - control the operation of the processor
 - usually visible only to the operating system
- Program Counter (PC): the address of the next instruction to execute
- Instruction register (IR): the mostly recently fetched instruction
- Program Status Word (PSW): condition codes, status information, interrupt enable/disable and user/kernel mode flags
- HW can be designed in a way that the OS support (e.g., memory protection/switching between tasks) can be implemented with less effort



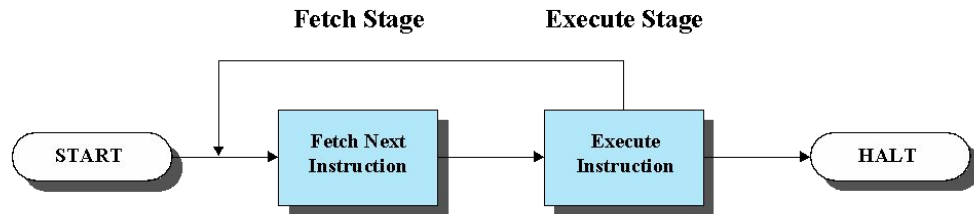
Instruction execution

- A program consists of a set of instructions
- A processor executes instructions in two-stage manner
 1. **Instruction fetch** from the memory
 2. **Instruction execution** which involves various CPU operations
- Program terminates
 - If an explicit instruction for that is issued
 - An unrecoverable error occurs
- Fetched instructions are loaded into Instruction Register IR and can perform
 - Processor-memory data transfer
 - Processor-I/O data transfer
 - Arithmetic or logic operation
 - Changing the program control (jumping to another address)



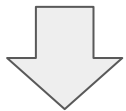
Instruction Fetch and Execute

- CPU fetches an instruction from memory
 - Program counter (PC) holds the address of the next instruction to be fetched
 - PC is incremented after each fetch
- Fetched instruction is loaded into Instruction Register (IR)
- Processor interprets the instruction and performs required action:
 - Processor-memory
 - Processor-I/O
 - Data processing
 - Control



Control Stack and Interrupts

Control stack and Interrupts



- Programs use memory to operate
 - Memory is usually divided into different areas
 - Used in a different way for different purposes
 - Text (program code)
 - Data (often there is a heap)
 - Stack (especially control, but also for data)
- The stack stores frames (data of a function call)
 - Frame pointer (FP) indicates the current frame
 - Access to data
 - Frames are often linked
 - Stack pointer (SP) indicates the top of the stack
 - In addition to call frames, there can be temporaries, etc.
- A method with which a peripheral (memory, disk, network card) can interrupt the CPU execution
 - Improves the processor utilization
 - I/O devices usually much slower than CPU
 - e.g. **accessing hard disk is several orders of magnitude slower than executing a CPU instruction**
 - without interrupts processor would have to wait until the device catches up
- With interrupts processor can execute something else while the I/O is being performed
 - when the external device becomes ready, it sends an interrupt request
 - CPU transfers control to an interrupt handler

Interrupts as events

- The terms exception, interrupt and trap are often used somewhat interchangeably
 - Note that there both HW and SW involved
 - Try not to get confused...
- Exception
 - An exception is a condition that occurs within the processor itself (e.g., division by zero)
- Interrupt
 - An interrupt is a signal from either an external HW
- Trap
 - Basically a software interrupt

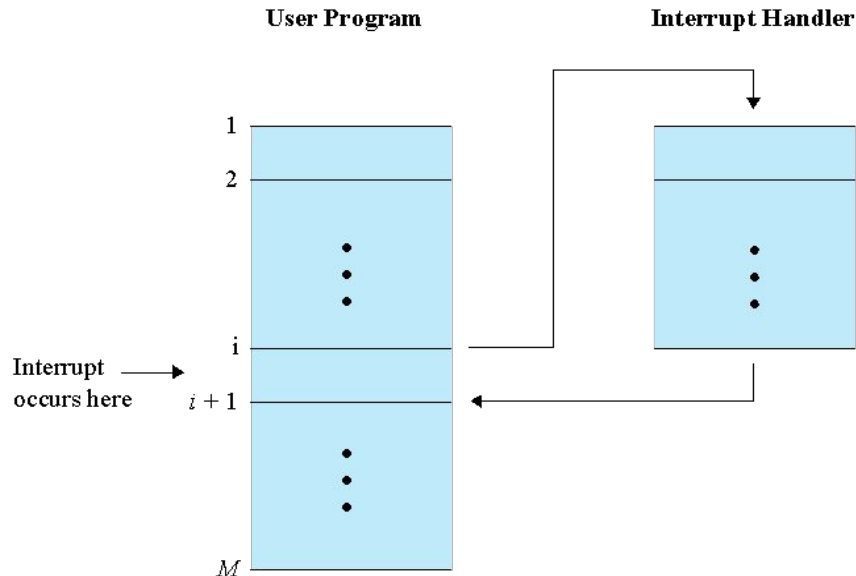


Figure 1.6 Transfer of Control via Interrupts

Multiple interrupts

- More than one device can generate interrupts simultaneously
 - E.g. printer and disk can complete their tasks at the same time
 - Disabling interrupts
 - CPU ignores further interrupts when in interrupt handler
 - Does not take into account relative priority or time-critical needs
- Prioritized interrupts
 - higher priority interrupt can interrupt lower priority interrupt handler
 - lower priority interrupt handler resumes once the higher interrupt has been dealt

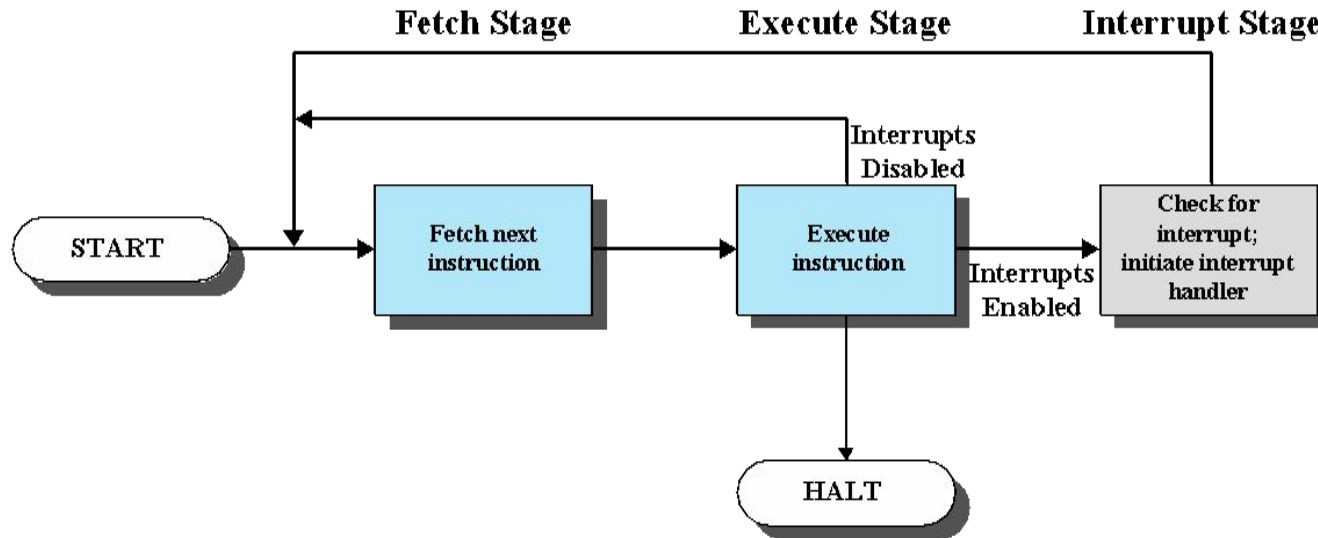


Figure 1.7 Instruction Cycle with Interrupts

Memory

Memory systems

- Trade-off between capacity, access time and cost
 - faster access time, greater cost per bit
 - greater capacity, smaller cost per bit
 - greater capacity, slower access speed
- The solution is to have a memory hierarchy
- When one goes down the hierarchy
 1. per bit cost decreases
 2. capacity increases
 3. access time increases
 4. frequency of access to the memory decreases
- Claims 1–3 apply due to our hardware
- Claim 4 is valid due to the *locality of reference*
 - the memory references of a program tend to cluster
 - applies to both code and data

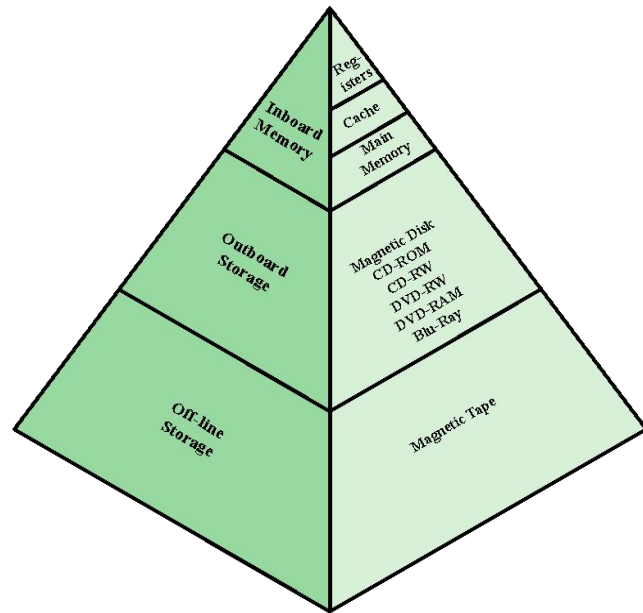
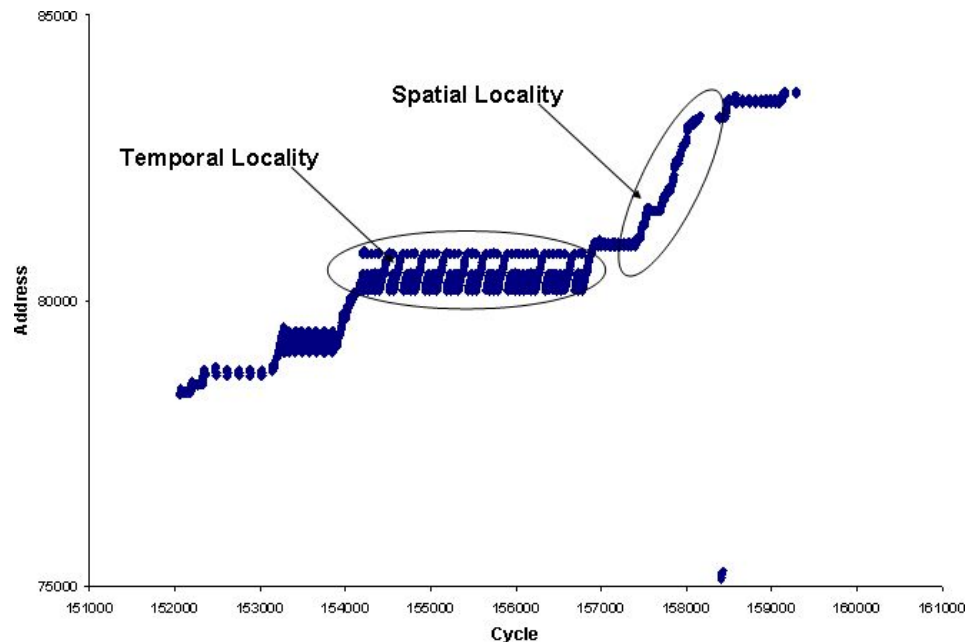


Figure 1.14 The Memory Hierarchy

Principle of Locality

- Memory references by the processor tend to cluster
- Data is organized so that the percentage of accesses to each successively lower level is substantially less than that of the level above
- Can be applied across more than two levels of memory



<https://stackoverflow.com/questions/7638932/what-is-locality-of-reference>

Memory hierarchy

- Typically there are several levels
 - Data path itself (latches, etc., not visible to programmer)
 - Registers (visible to the programmer/compiler)
 - Cache (not visible to the programmer/compiler, but OS handles)
 - L1 (often zero-wait, usually per core)
 - Often separate for code and data
 - L2 (shared)
 - L3 (can be found off-chip)
 - Main memory (usually DRAM)
 - Disk memory
- Differences in sizes and speeds are significant
- Note that the same data can reside in several places

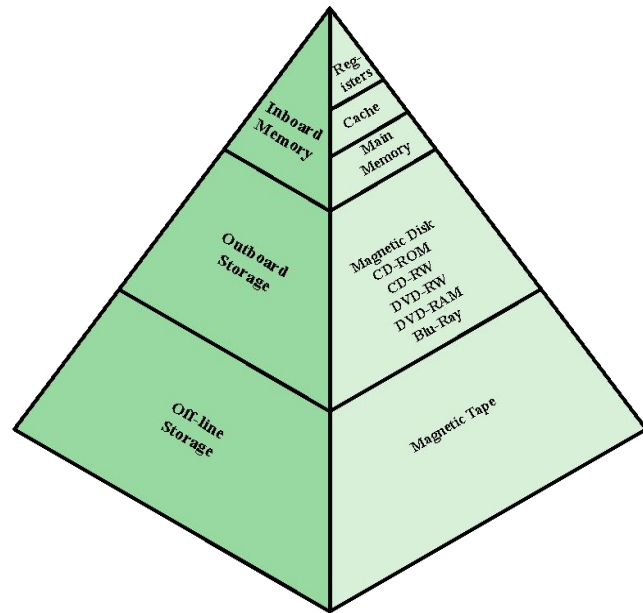
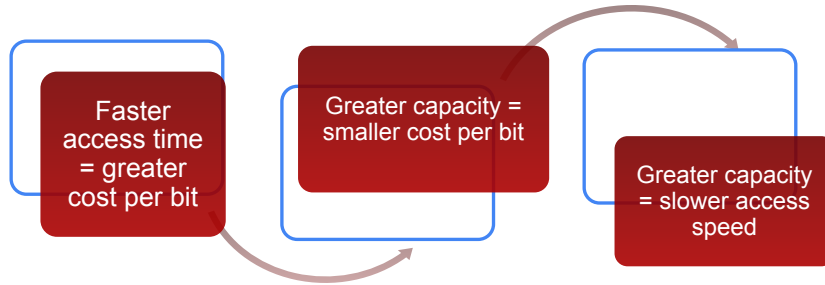


Figure 1.14 The Memory Hierarchy

Memory Relationships



Secondary Memory

Also referred to as auxiliary memory

- External
- Nonvolatile
- Used to store program and data files

Basic cache operation

- Caches are small memories
 - Close to core
 - Invisible to the programmer
 - Storing frequently used data
- Cache organization
 - **Cache lines (small blocks, e.g., 64 bytes)**
 - Cache sets (direct-mapped are common, full assoc. rare)
- Associate memory operation
 - In addition to data, coding of the address
 - **Tag** and index (block bits not needed)
 - Bits for checks, e.g., validity and modifications (dirty bit)
 - Tags are the memory addresses
 - **Cache hits and misses**

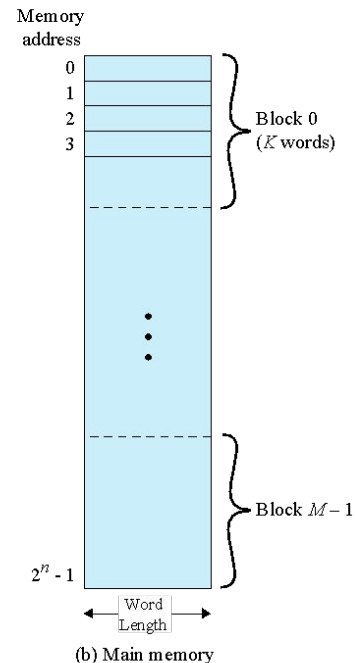
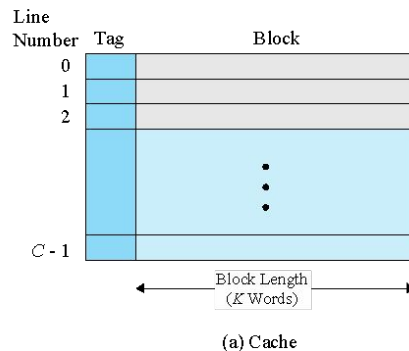
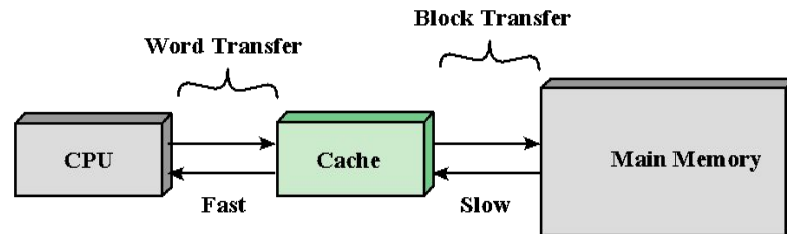


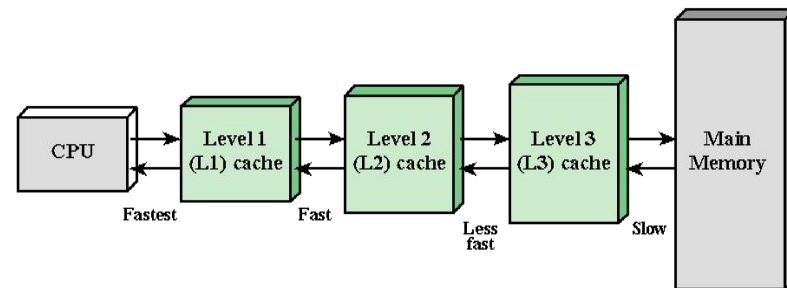
Figure 1.17 Cache/Main-Memory Structure

Cache Memory

- Invisible to the OS
- Interacts with other memory management hardware
- Processor must access memory at least once per instruction cycle
- Processor execution is limited by memory cycle time
- Exploit the principle of locality with a small, fast memory



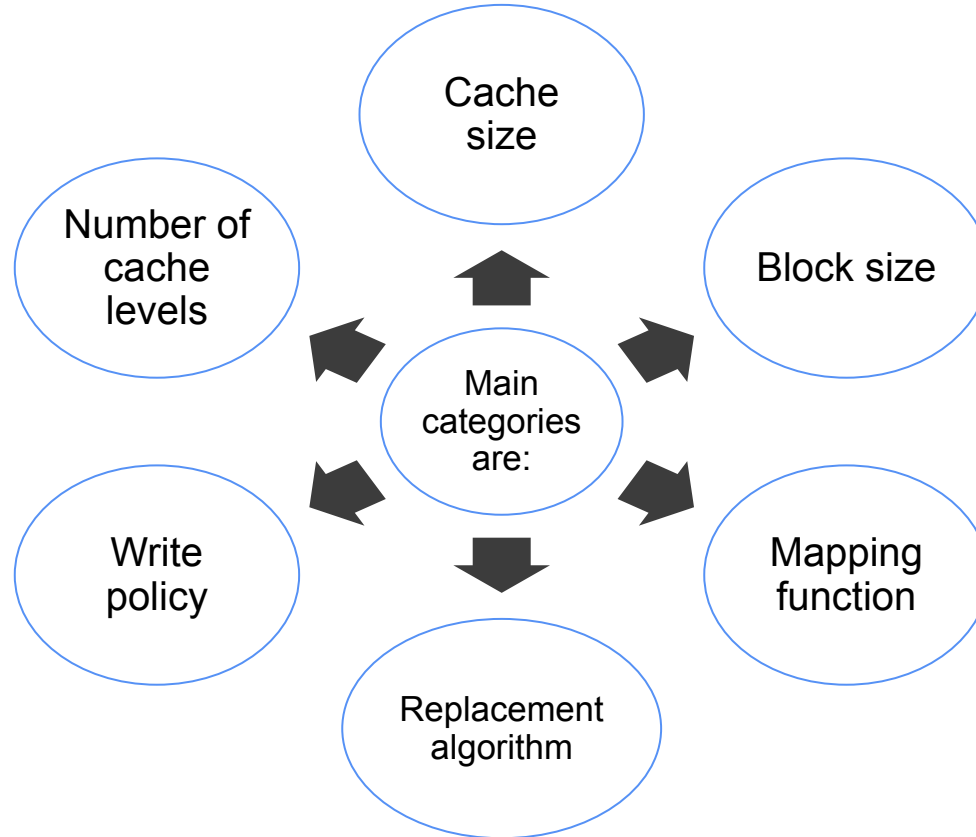
(a) Single cache



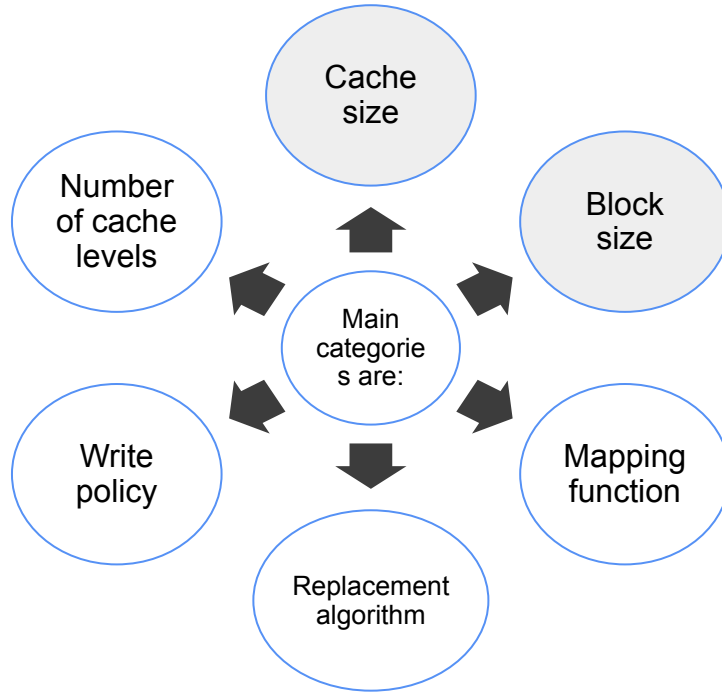
(b) Three-level cache organization

Figure 1.16 Cache and Main Memory

Cache Design



Cache Design



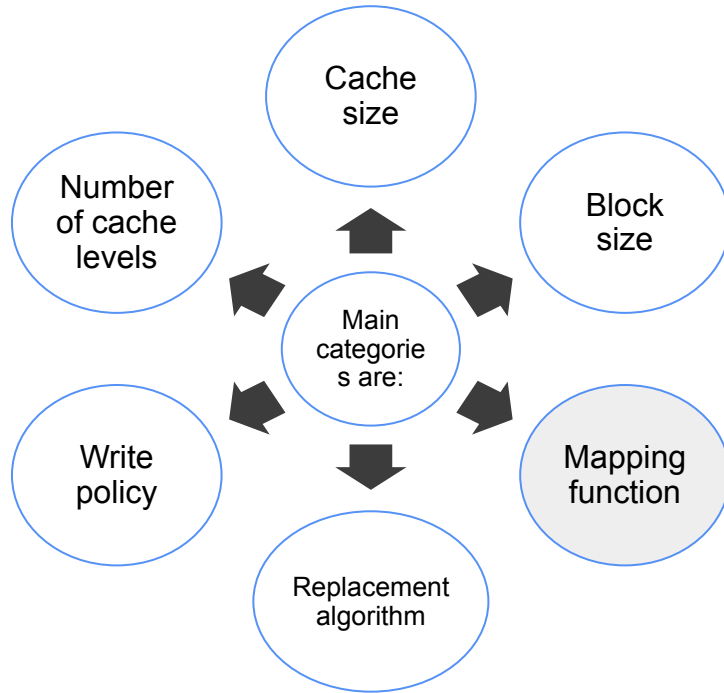
Cache Size

Small caches have significant impact on performance

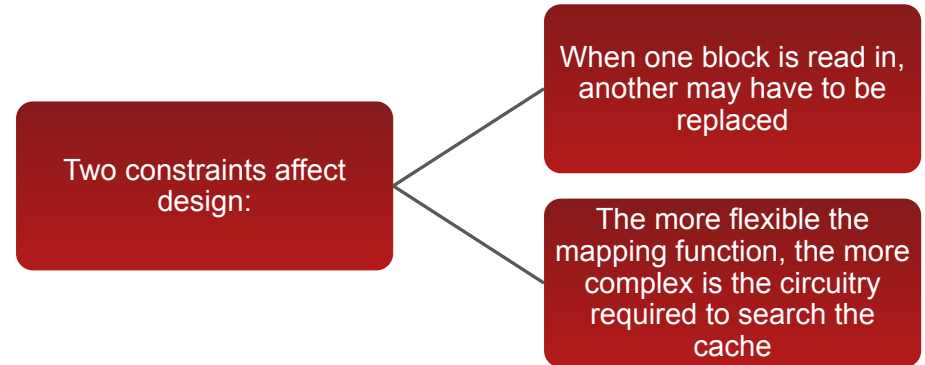
Block Size

The unit of data exchanged between cache and main memory

Cache Design



Determines which cache location the block will occupy



Writes and allocation

- Basic operation
 - Replacement policy (random, LRU), must be fast!
 - Do not use page replacement algorithms here!
- Reads
 - Caches operate (mainly) on-demand
- Writes
 - Write-through and write-back
- Allocation
 - Write allocate (typical with write-back)
 - Datum at the missed-write location is loaded to cache, followed by a write-hit operation
 - Write around (typical with write-through)
 - not loaded to cache, direct writing to memory

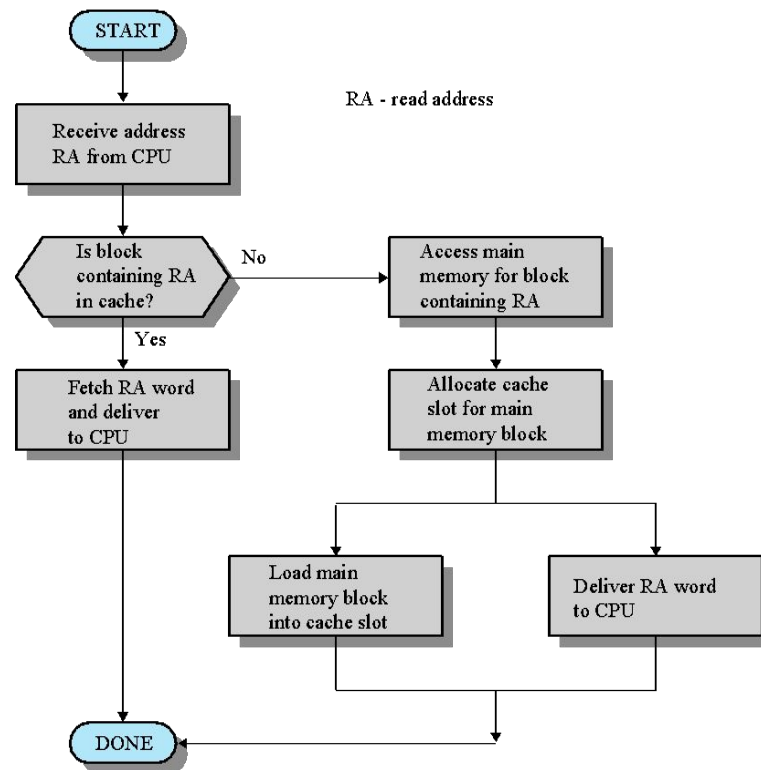


Figure 1.18 Cache Read Operation

Summary

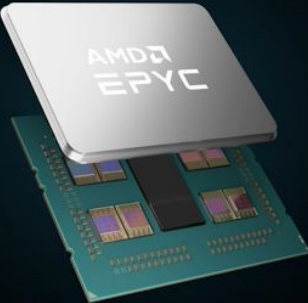
- Basic Elements
 - Interrupts and the instruction cycle
 - Interrupt processing
 - Multiple interrupts
- The memory hierarchy
- Cache memory
 - Motivation
 - Cache principles
 - Cache design

Intel® Pentium® Pro Processor 200 MHz, 1M Cache, 66 MHz FSB	
Performance Specifications	
Total Cores ?	1
Processor Base Frequency ?	200 MHz
Cache ?	1 MB L2 Cache
Bus Speed ?	66 MHz
TDP ?	44 W
VID Voltage Range ?	3.3V±5%/3.2V±0.1V

Intel Pentium Pro, 1995



WD 640MB, 1995



AMD EPYC 7003 processor shown on a green PCB with gold pins.

"ZEN 3" CORES	LEADERSHIP CORE PERFORMANCE Key for single-threaded apps or to maximize per-core licensing
768MB L3 CACHE	LEADERSHIP X86 L3 CACHE: UP TO 96MB / CCD Over 1.5GB of L3 cache in a 2P Server
SP3 SOCKET COMPATIBLE	KEEPS SP3 PLATFORM AND ECOSYSTEM INTEGRITY Just a simple BIOS update will enable existing "Milan-X" servers for "Milan-X"
SECURITY	LEADERSHIP AMD INFINITY GUARD FEATURES* Security features supported by mainstream Linux® distros, VMware®, GCP, and Azure

* 3rd Gen EPYC with AMD 3D-V-Cache Press Overview | Embargoed Until March 21, 2022, 9:00am ET

AMD EPYC 7003, 2022