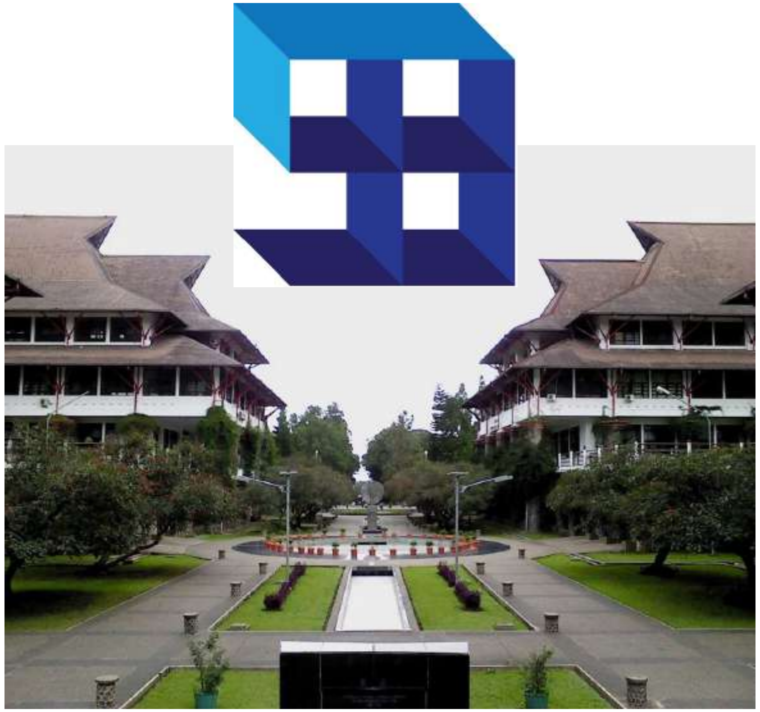


STEI - Institut Teknologi Bandung

Modul 1

Course Introduction

and a Tour of Computer System

EL3011 Arsitektur Sistem Komputer

Introduction

- Time/Place: Wednesday 10:00 - 11:00 (online)

  Friday 9:00 - 11:00 (R. 9018)
- Place: -
- Instructor: - Dr. Kusprasapta Mutijarsa, S.T., M.T.

  - Dr. Reza Darmakusuma, S.T., M.T.
- Course LMS: Edunex

  https://edunex.itb.ac.id/courses/43727/preview
- Attendance

  https://akademik.itb.ac.id

Requisite

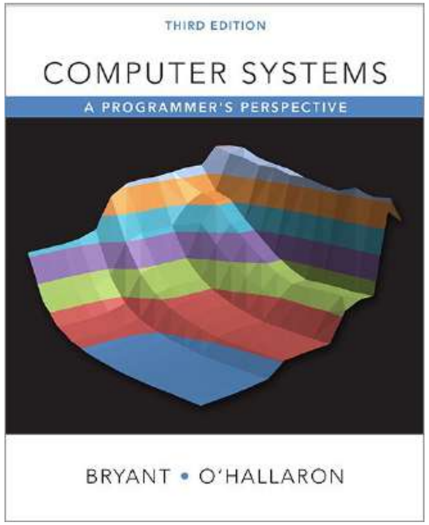
- **Prerequisite**
  - C Programming
  - EL2095 Digital System
- **Corequistite**
  - EL3110 Computer Architecture Laboratory

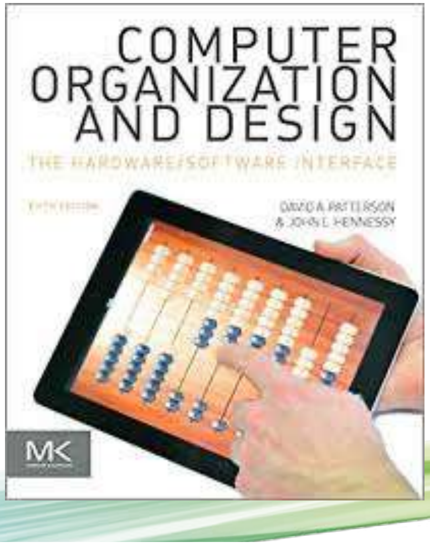
Arsitektur Sistem Komputer

Text Book and Materials

1. Randal E. Bryant, David R. H, Computer Systems A Programmer’s Perpective 3rd ed, 2015
   - URL: http://csapp.cs.cmu.edu

2. John L. Hennessy and David A. Patterson , Computer Organization and Design: The Software Hardware Interface, Morgan Kaufmann Publishers, 5th Edition, 2014

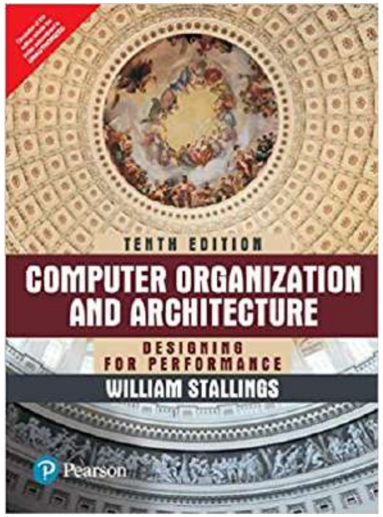


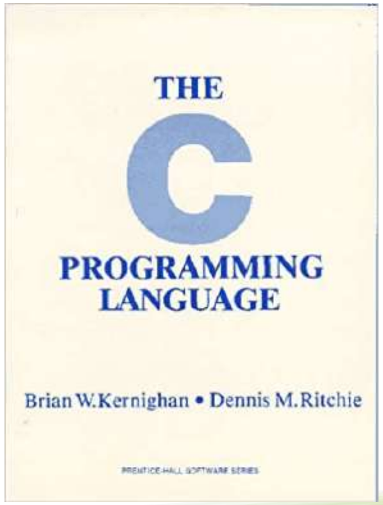

COMPUTER ORGANIZATION AND DESIGN
THE HARDWARE/SOFTWARE INTERFACE
FIFTH EDITION
DAVID A. PATTERSON & JOHN L. HENNESSY
MK

Other References:

1. William Stallings, Computer Organization and Architecture 10th Ed, Pearson, 2016

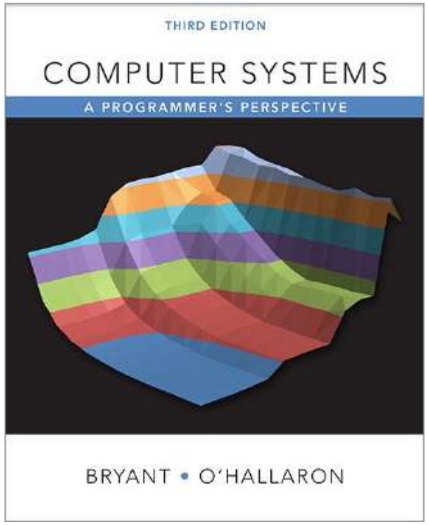
2. Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, 1978


TENTH EDITION
COMPUTER ORGANIZATION AND ARCHITECTURE
DESIGNING FOR PERFORMANCE
WILLIAM STALLINGS
Pearson


THE
C
PROGRAMMING LANGUAGE
Brian W. Kernighan • Dennis M. Ritchie
PRENTICE-HALL SOFTWARE SERIES

Text Book and Materials

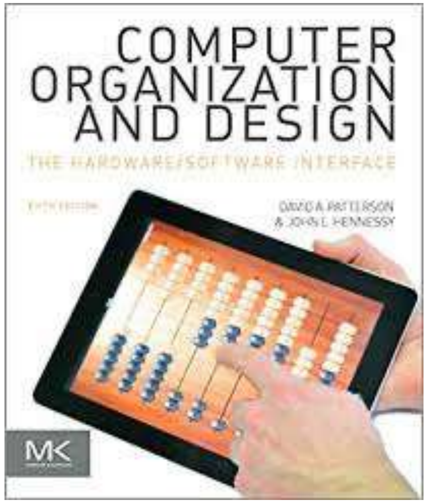
- Randal E. Bryant, David R. H, Computer Systems A Programmer’s Perpective 3rd ed, 2015
  - URL: http://csapp.cs.cmu.edu



- Chapter:
  - Ch 1: Introduction (Tour of Computer Systems)
  - Ch 2: Integer and Floating Point (Representation and Operation)
  - Ch 3: Intel’s ISA (Data Format, ALU Ops, Control, Procedure, Array)
  - Ch 6: Cache Memory
  - Ch 10: Virtual Memory

Text Book and Materials

- John L. Hennessy and David A. Patterson , Computer Organization and Design: The Software Hardware Interface, Morgan Kaufmann Publishers, 5th Edition, 2014



- Chapter:
  - Ch 1: Introduction (History of computer, performance analysis)
  - Ch 2: MIPS ISA
  - Ch 3: ALU
  - Ch 4: MIPS Single Cycle (Data Path and Control, Pipeline)
  - Ch 5: Memory
  - Ch 6: I/O Subsystem

Course schedule

Week	Topics	Reference
1	Course Introduction, Tour of Computer System	[CSAP] Ch1, [P&H] Ch1
2	Bit Representation and Operations	[CSAP] Ch2
3	Integer Representation and Operations	[CSAP] Ch2
4	Floating Point Representation	[CSAP] Ch2
5	Intel Processor – Assembly Language	[CSAP] Ch3
6	Intel Processor – Control Flow	[CSAP] Ch3
7	Intel Processor – Stack and Procedure	[CSAP] Ch3
8	Mid Term Exam	
9	MIPS Instruction Set Architecture	[P&H] Ch3
10	MIPS Arithmetic	[P&H] Ch3
11	MIPS Data Path and Control	[P&H] Ch3
12	MIPS Single Cycle Instruction	[P&H] Ch3
13	Pipeline	[P&H] Ch4
14	Memory Hierarchy	[CSAP] Ch 6
15	Cache Memory	[CSAP] Ch 6
16	Final Exam	

Class Assesment

Homework	30%
Quiz	10%
Exam	60%

Grading Scale

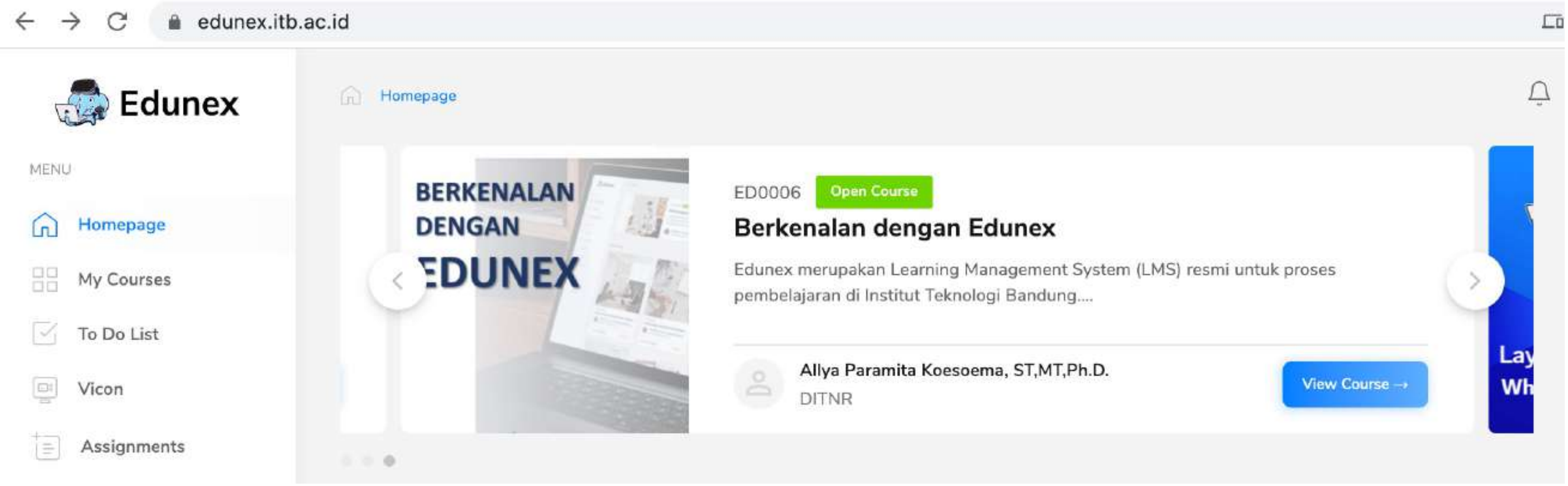
• A	> 90%
• AB	80% - 89%
• B	70% - 79%
• BC	60% - 69%
• C	50% - 59%
• D	40% - 49%
• E	< 39%

Course Objectives

- This course will give you an in-depth understanding of the **inner-workings of modern digital computer systems and tradeoffs present at the hardware-software interface**. You will get an understanding of the design process in the context of a complex hardware system and practical experience with computer-aided design tools. Topics include: Instruction set design, computer arithmetic, controller and datapath design, memory systems, input-output systems, networks interrupts and exceptions, pipelining, performance and cost analysis, computer architecture history, and a survey of advanced architectures.

Course Material

- https://edunex.itb.ac.id/courses/43727/preview



ED0006

Berkenalan dengan Edunex

Edunex merupakan Learning Management System (LMS) resmi untuk proses pembelajaran di Institut Teknologi Bandung....

Allya Paramita Koesoema, ST,MT,Ph.D.

DITNR

A Tour of Computer Systems

Topics

- Understanding of computer system
- Compiler System
- Hardware Organizations
  - Bus, I/O Device, Memory, Processor
- Cache memory
- Operating System
- Virtual memory

Computer System

- Hardware and Operating system works together to execute an application.
- Implementation of a computer can change but not the concept

Why you should be in this class?

- To become knowledgeable about the interaction between software and hardware.
- Learn to avoid numeric error
- Learn to exploit the underlying hardware
- Learn the details of designing a processor (MIPS)

Program `hello`

- Classic first program
- `hello` program is created using a text editor and saved as `hello.c`
  - Source program is a sequence of bits, each with a value 0 or 1, organized into 8 bits called byte
  - Each byte represents a character
  - `hello.c` is stored in a file as a sequence of bytes.

`hello` Program

- Written in high level language C
- Code :

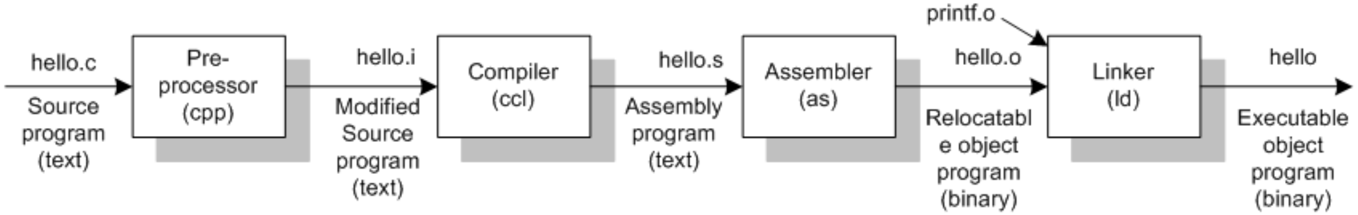
```
#include <stdio.h>

int main(void)
{
    printf(“hello, world\n”);
    return(0);
}
```

`hello` Program

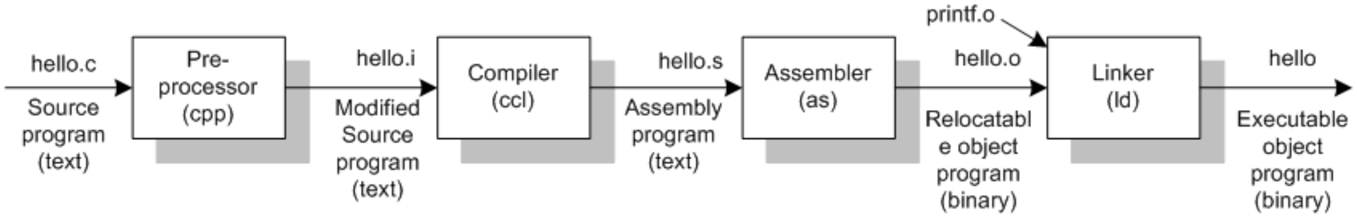
- Every C statement must be translated to machine instructions (in binary)
- These insstructions are then packaged into an executable object program and stored in a binary file
- Translation process is performed by a compiler

Compilation System



- Preprocessing phase
  - preprocessor (cpp) modifies the original C program according to the # directive
  - Example: #include <stdio.h> tells the preprocessor to read the stdio.h file and insert it into the program text.
- Compilation phase
  - compiler (ccl) translates the text file hello.i into the text file hello.s which contains an assembly language program. Each statement in an assembly language represents one machine-language instruction in a text form.

Compilation System



- Assembly phase
  - assembler (as) translates hello.s into machine-language instructions, packages then into a relocatable object program and store the result into a file hello.o
- Linking phase
  - linker (ld) merges hello.o with printf.o and the result is an executable object file

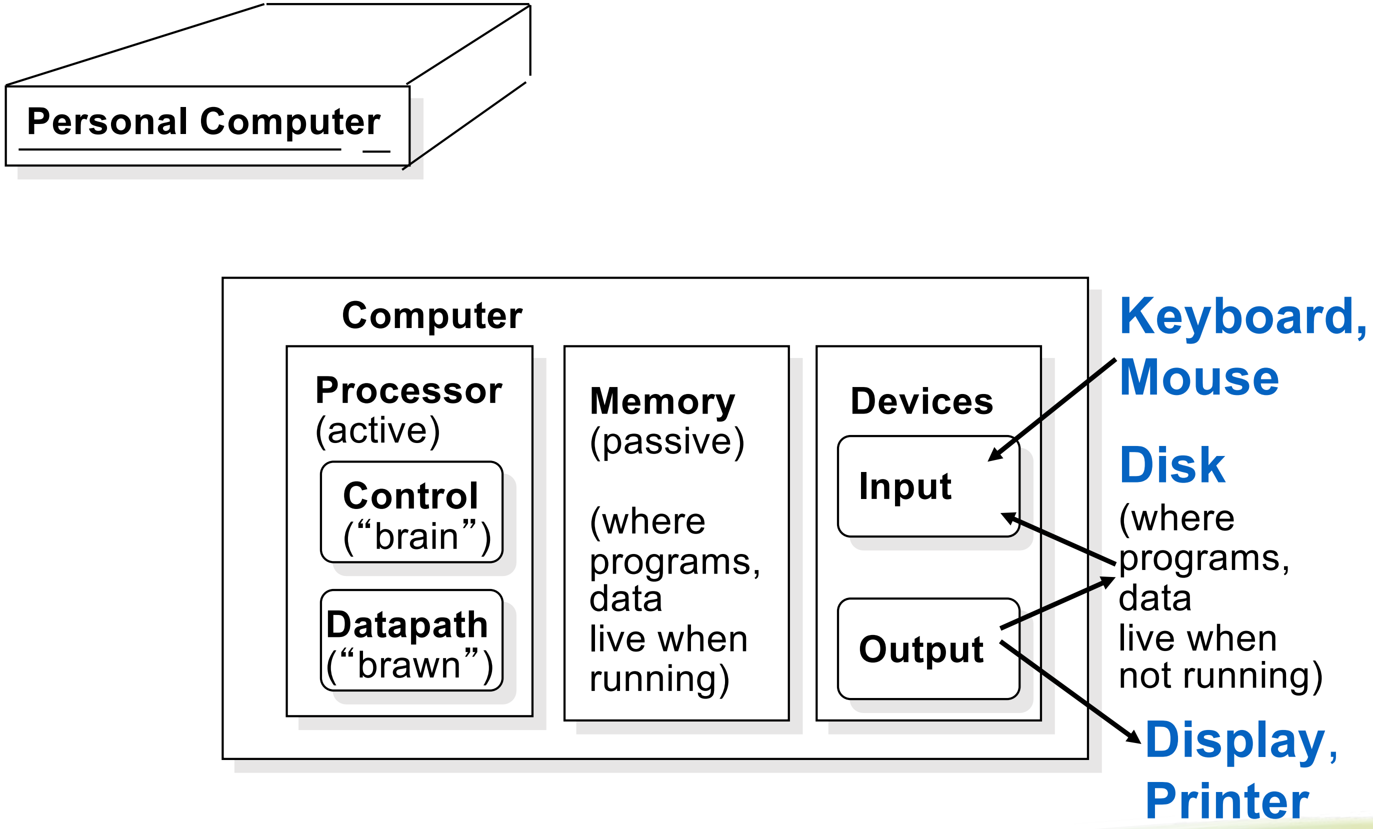
Understand how compilation system works

- Optimizing program performance
- Example :
  - Which one more efficient?
    - switch or if-then-else ?
    - while or do ?
    - Using pointer or array indexes?
  - Which is faster local variable or passed by reference?
- Understanding link time error
  - What is link error?
  - What is static or dynamic library?
- Avoid security holes
  - Buffer overflow bugs

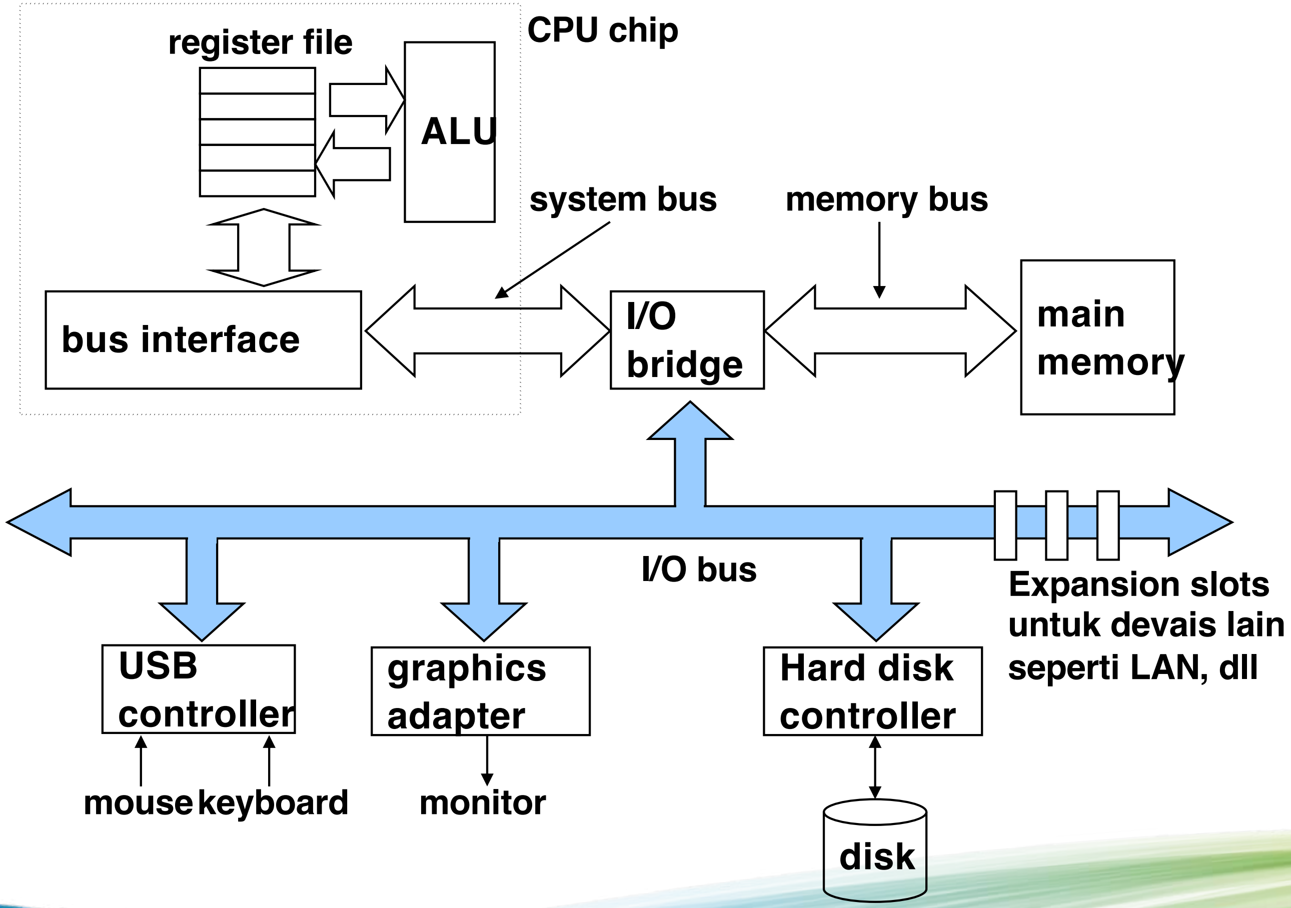
Hardware Organization

- To understand what happens when we run the hello program, we need to know how the hardware is organized.
- In general the component of a computer system consists of :
  - Bus
  - I/O devices
  - Main Memory
  - Processor

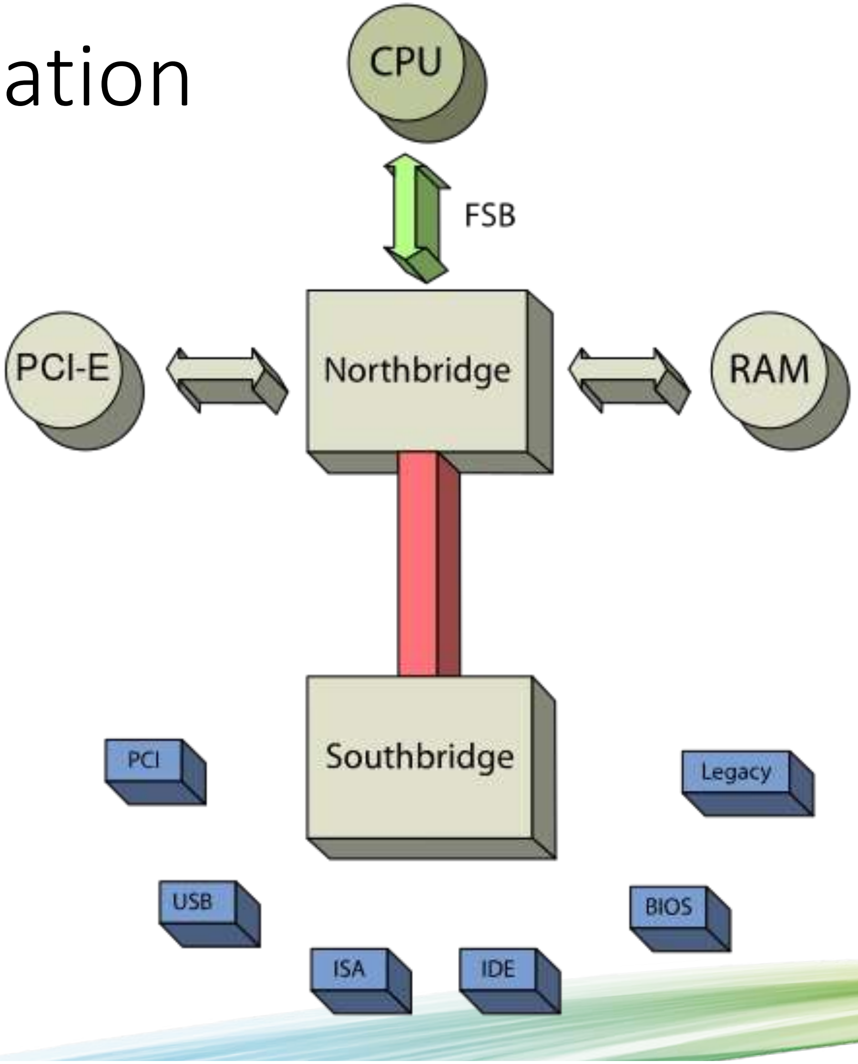
Hardware Organization



Hardware Organization



Hardware Organization



Hardware Organization

- Bus
  - Parallel conduits that carry bytes of information between components.
  - Bus size is usually given in words
    - Intel Pentium, word size = 4 bytes
    - Intel Itanium, word size = 8 bytes
    - Embedded, word size = 1 or 2 bytes
- I/O Devices
  - Connection to the outside world
    - Example: keyboard, mouse, monitor, disk drive (disk)
  - Every I/O device is connected using a controller or adapter
    - Controller : chip set in the device itself or on the motherboard
    - Adapter : card that plugs into to the slot of the motherboard

Hardware Organization

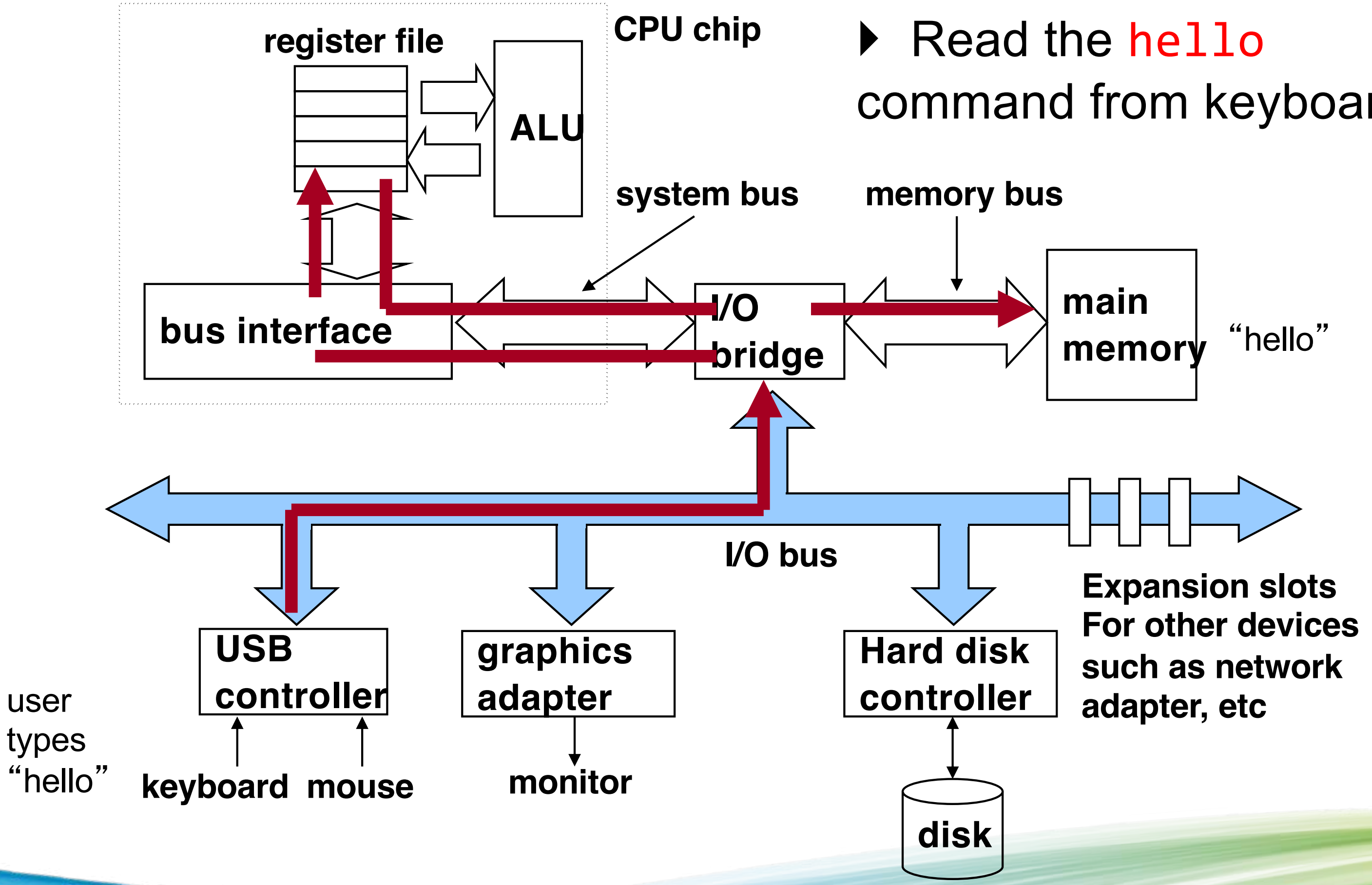
- Main Memory
  - Temporary storage that holds both program and data it manipulates while the processor is running the program.
    - Physically, the main memory is a collection of Dynamic Random Access Memory (DRAM)
    - Logically, the main memory is a organized as a linear array
- Processor (Central Processing Unit % CPU)
  - The engine that executes the instructions stored in the main memory
  - Consists of registers, ALU and program counter (PC)
  - At any point of time the PC (contains address) is always points to an instruction in the main memory.
  - Processor is always doing the same task over and over again
    - Read an instruction from memory
    - Execute it
    - And read the next instruction

Hardware Organization

- Processor
  - Has only a few (?) instructions that revolve around main memory, registers and arithmetic/logic unit ALU
    - **Register** : fast memory but only a few, reside inside the CPU
    - **ALU** : computes new data or address
  - Types of CPU operation:
    - **Load** : copy a byte or a word from main memory to a register
    - **Store** : copy a byte or a word from register to the main memory
    - **Update** : copy the content of two registers to ALU, adds the two words and store the result into a register
    - **I/O Read** : copy a byte or a word from an I/O device to a register
    - **I/O Write** :copy a byte or a word from a register to an I/O device

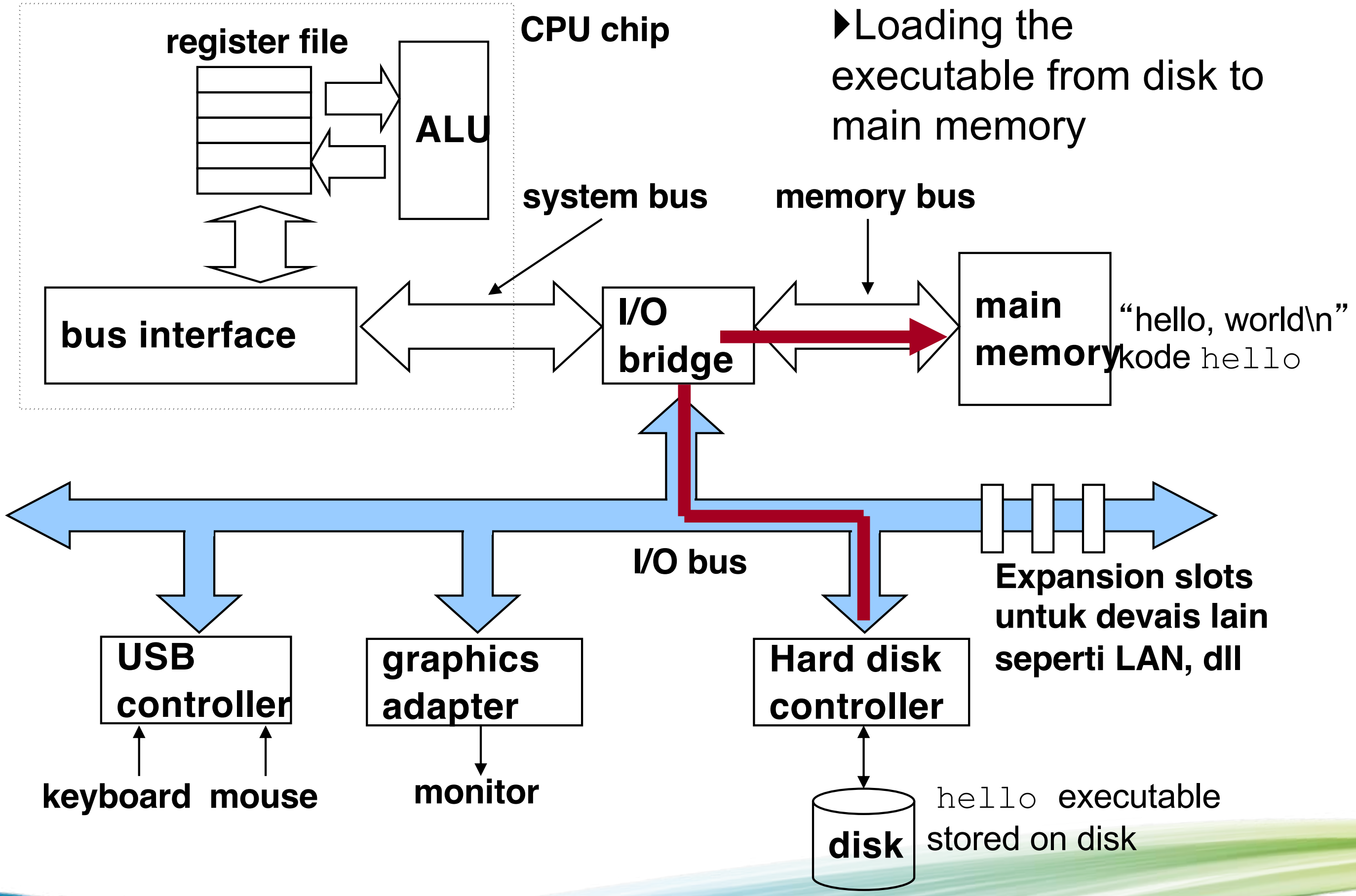
Executing **`hello`** program (1)

- Read the `hello` command from keyboard



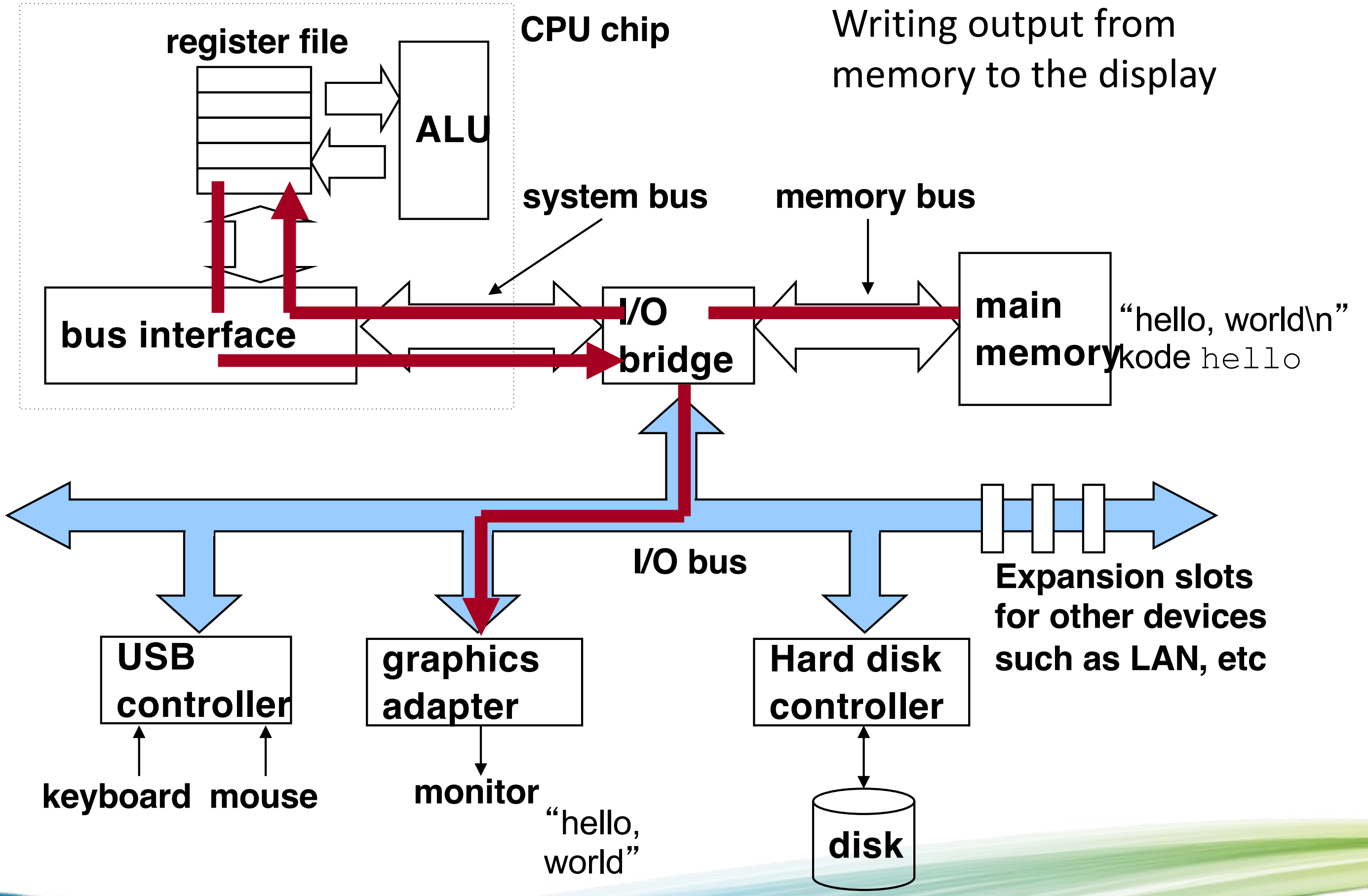
Executing hello program (2)

- Loading the executable from disk to main memory

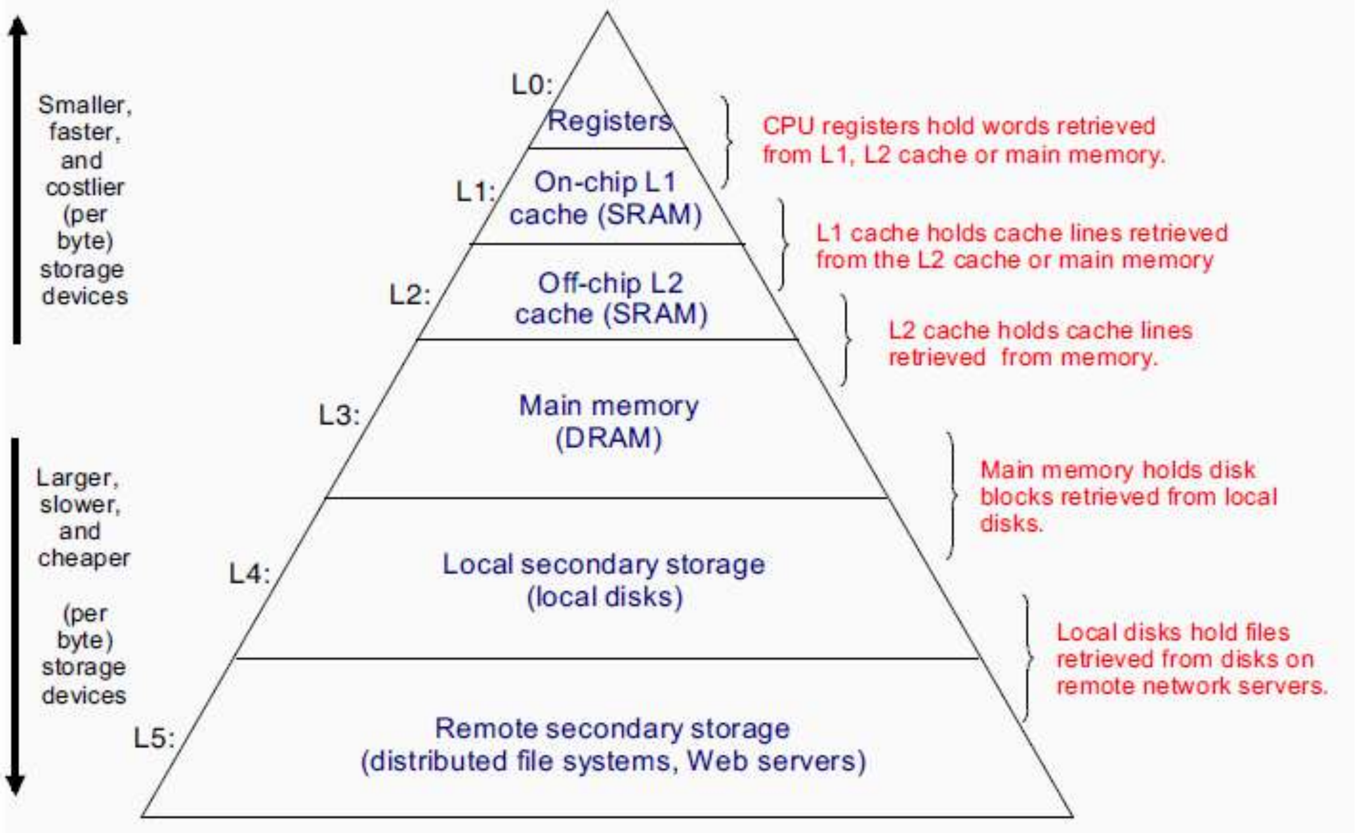


Executing `hello` program (3)

Writing output from memory to the display

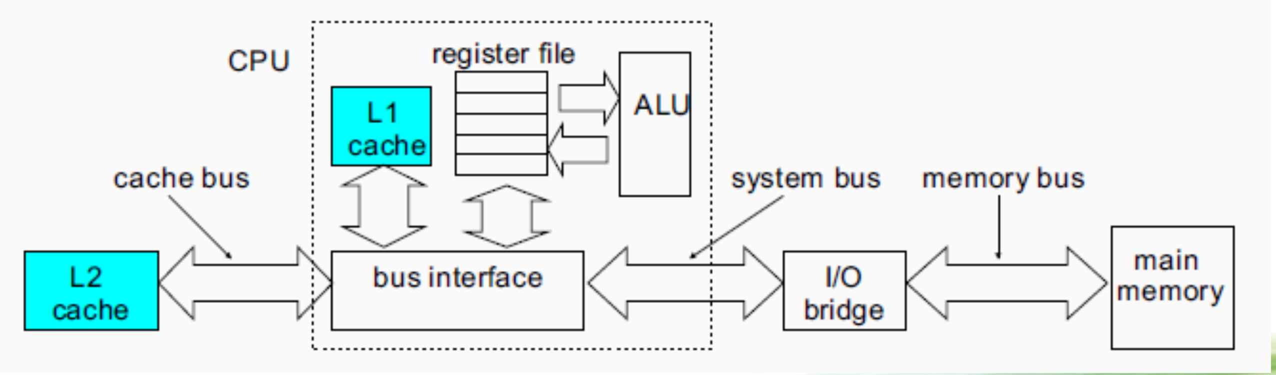


Memory Hierarchy


Smaller, faster, and costlier (per byte) storage devices
Larger, slower, and cheaper (per byte) storage devices
L0:
Registers
L1:
On-chip L1 cache (SRAM)
L2:
Off-chip L2 cache (SRAM)
L3:
Main memory (DRAM)
L4:
Local secondary storage (local disks)
L5:
Remote secondary storage (distributed file systems, Web servers)
CPU registers hold words retrieved from L1, L2 cache or main memory.
L1 cache holds cache lines retrieved from the L2 cache or main memory
L2 cache holds cache lines retrieved from memory.
Main memory holds disk blocks retrieved from local disks.
Local disks hold files retrieved from disks on remote network servers.

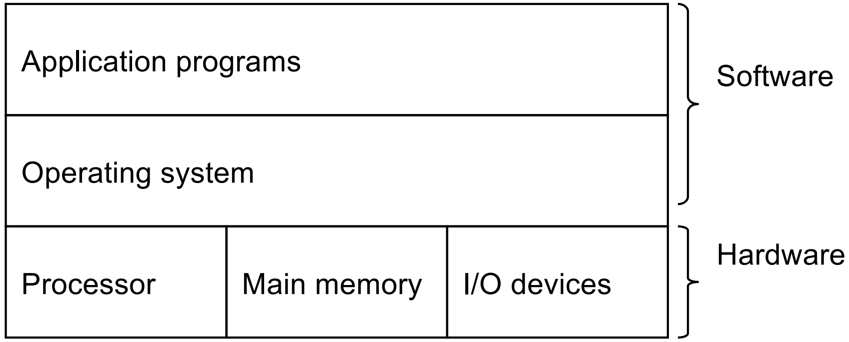
Cache Memory

- Cache memory is needed to solve the problem of speed difference between the processor and main memory
  - Cache is a high speed static RAM (faster than DRAM still slower than registers)
  - Cache holds the most recently accesesed information
  - L1 cache size is about tens thousand to a hundred thousand bytes
  - L2 cache size is about hundred thousand to millions of bytes



The OS manages the HW

- Primary Purpose:
  - Manages all hardware components
  - Provide applications with wimple and uniform mechanism for manipulating hardware



Application programs

Operating system

Processor

Main memory

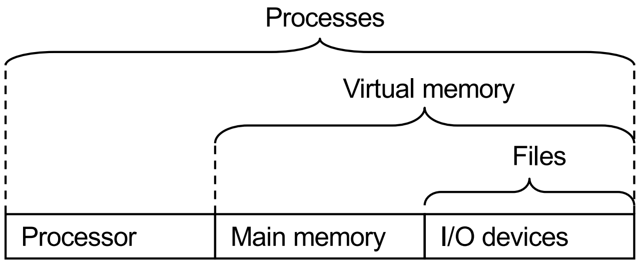
I/O devices

Software

Hardware

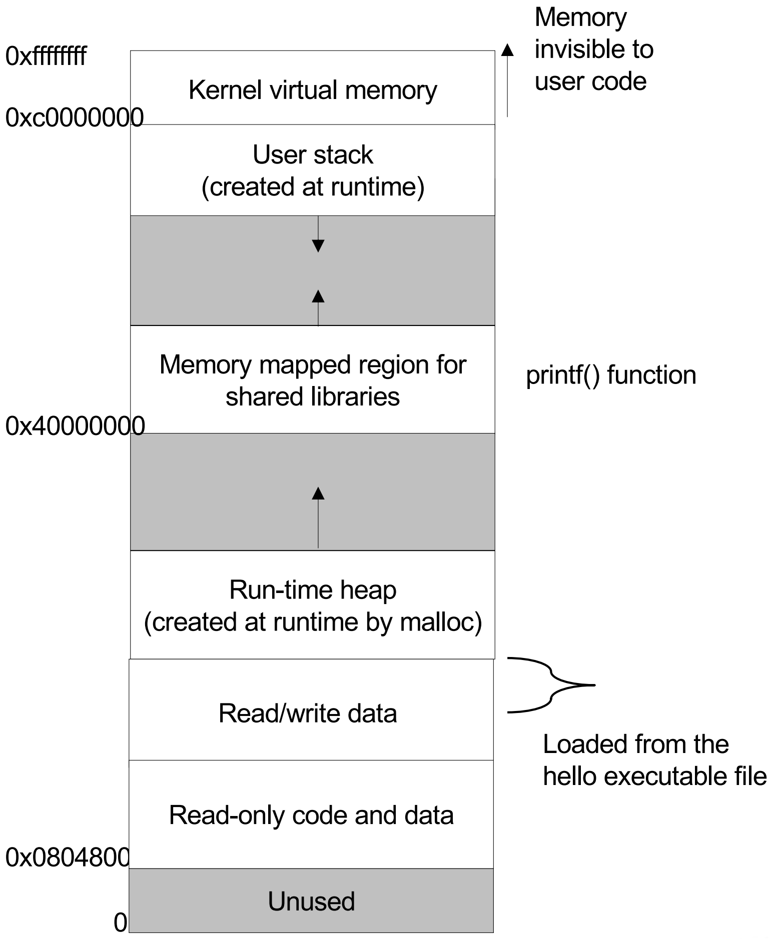
Abstractions

- Process
  - Is the OS abstraction of running program
  - Multiple process can run concurrently
  - OS keeps track of all the state information that the process needs in order to run
- Threads
  - A process can have multiple execution units that can run concurrently
  - Threads shares the same code and global data



Virtual Memory

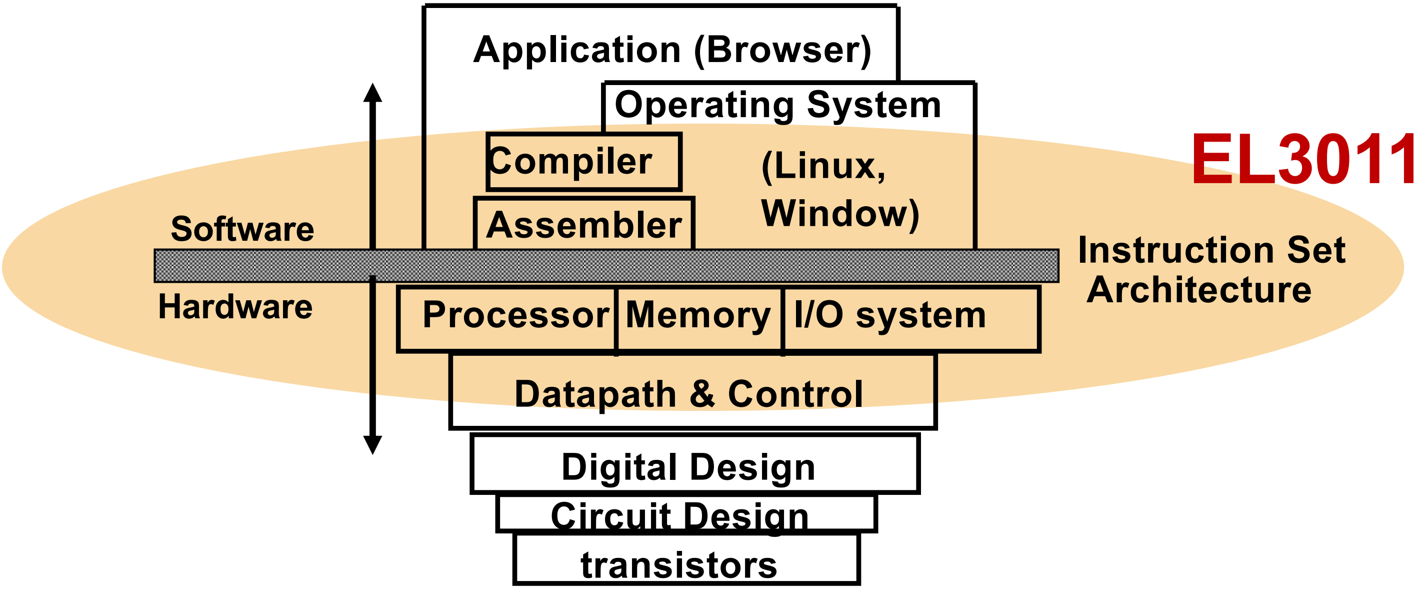
- Is an abstraction that provides an illusion that a process has exclusive use of the main memory
- Each process has the same view of memory (virtual address space)
- Contents
  - Program dan data
  - Heap
  - Shared library
  - Stack
  - Kernel virtual memory


0xffffffff
Kernel virtual memory
Memory invisible to user code
0xc0000000
User stack (created at runtime)
Memory mapped region for shared libraries
printf() function
0x40000000
Run-time heap (created at runtime by malloc)
Read/write data
Read-only code and data
Loaded from the hello executable file
0x0804800
Unused
0

Files

- A sequence of bytes
- Every I/O device is modeled as a file
- All input and output is performed by reading and writing files (Unix I/O)

Computer Abstraction



- Both Software and Hardware consist of hierarchical layers.
- Each lower layer hides the complexity from the layer above
- This abstraction principle is the way to cope with complexity

The Big Picture

Both hardware and software consist of hierarchical layers, with each lower layer hiding details from the level above. *This principle of abstraction is the way both hardware designers and software designers cope with the complexity of computer systems.* One key interface between the levels of abstraction is the instruction set architecture: the interface between the hardware and low-level software. This abstract interface enables many implementations of varying cost and performance to run identical software.

John L. Hennessy

David A. Patterson

Welcome to EL3011 Class

"Big things have small beginnings"
T.E. Lawrence, Lawrence of Arabia
(1962)