Politechnika Wrocławska

Computer Architecture and Organization

Lecture 1

Dr. Radosław Michalski
Department of Computational Intelligence, Faculty of Computer Science and Management, Wrocław University of Science and Technology

Version 1.0, spring 2017

Source and licensing

The most current version of this lecture is here:
https://github.com/rmhere/lecture-comp-arch-org

Overview of this lecture

Course organization

The scope of the course

Architectural classification

Memory hierarchy

Computer architectures

Course organization

Schedule

This is a group course, i.e. you have to pass two parts (laboratory and lecture) in order to pass the course.

Lecture:

- fifteen lectures
- exam at the end

Laboratory:

- introduction
- five logic circuits labs
- five assembly labs
- additional labs

東海道線(上野東京ライン)標準時刻
Timetable of Tōkaidō Line (Ueno-Tōkyō Line)

Mildiou - Newspaper, CC BY 2.0

Course organization

Materials

The course materials are published on BOARD.

- https://eportal.ii.pwr.edu.pl/
- full URL: here

What you will find there:

- grading rules
- PDF files with lectures
- recommended software, books, and MOOCs
- important materials for the lab

Please devote some time to study those.

Course organization

Recommended software and flickrreading

Software:

- Logisim - designing and simulating digital logic circuits (Java)
- Logic Circuit - as above (Windows)
- MARS - MIPS Assembler and Runtime Simulator (Java)

Books:

- D. Patterson, J. Hennessy, *"Computer Organization and Design"*, Elsevier
- W. Stallings, *"Computer Organization and Architecture"*, Pearson (also in Polish)
- D. Patterson, J. Hennessy, *"Computer Architecture – a Quantitative Approach"*, Elsevier
- W. Komorowski, *"Krótki kurs architektury i organizacji komputerów"*, Mikom

Course organization

Massive open online courses (MOOCs)

Coursera:

- Build a Modern Computer from First Principles: From Nand to Tetris

Open Security Training:

- Introductory Intel x86: Architecture, Assembly, Applications, & Alliteration
- Introductory Intel x86-64: Architecture, Assembly, Applications, & Alliteration
- Introduction to ARM

The scope of the course

What will be discussed

Lectures:

- computer architectures
- RISC processors
- assembly programming language
- memory organization
- pipelining
- parallel computing
- history and the future of computer architectures

Laboratory part:

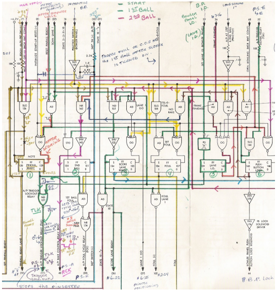
- building simple digital logic circuits
- simple programming in assembly language

The scope of the course

Logic circuits

Please keep in mind that:

- the lecture does not cover logic circuits part
- you have to study it on your own
- please learn or recap:
    - logic gates - see here
    - boolean logic - see here
    - logic simplification - see here
    - Karnaugh maps - see here
    - working with breadboards - see here
- use simulators to prepare yourself
- laboratory materials - see here



Doug Coldwell - Circuit logic diagram, CC BY 2.0

Architectural classification

Introduction to classification

- **What classification actually is?**
  It helps to characterize the object. By basing on a set of features classification allows to assign the object to a given category within a taxonomy.
- **Why should we classify computer architectures?**
  To easily figure out what are the capabilities and limitations of a given architecture. After that we can look into details.

Architectural classification

Most popular classifications

- **Flynn's taxonomy** is looking at the multiplicity of instruction streams and the data in computer systems (1966)
- **Feng's classification** distinguishes between serial and parallel processing (1972)
- **Handler's classification** looks at the degree of parallelism built inside the hardware structure of the computer (1977)

Architectural classification

Flynn's taxonomy

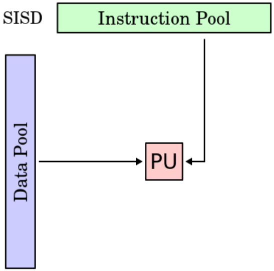
- **Flynn's taxonomy** was defined by Michael J. Flynn in 1966[1]
- it is most commonly referred to when thinking of classifying computer architectures
- it distinguishes computer architectures by basing upon the number of concurrent instruction streams and data streams available in the architecture:
  - **instruction stream** - sequence of instructions performed by the processing unit
  - **data stream** - the exchange of data between the processor and memory

[1] Michael J Flynn. "Some computer organizations and their effectiveness". In: *IEEE transactions on computers* 100.9 (1972), pp. 948–960.

Architectural classification

Flynn's taxonomy - SISD

- **SISD: single instruction stream, single data stream**
  Exhibits no parallelism: fetch a single instruction, then fetch data. Examples: single core personal computers, early mainframes.

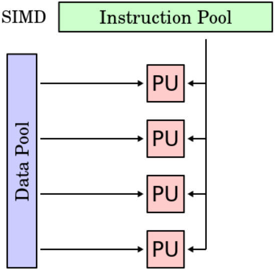


Cburnett - SISD, CC BY-SA 3.0

Architectural classification

Flynn's taxonomy - SIMD

- **SIMD: single instruction stream, multiple data streams**
  The same instruction is being run for multiple data streams. For instance, compute the inverse of many matrices. Examples: GPU computing.

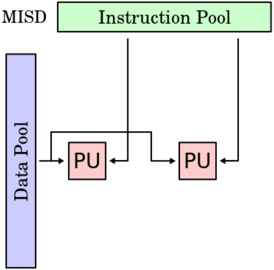


Cburnett - SIMD, CC BY-SA 3.0

Architectural classification

Flynn's taxonomy - MISD

- **MISD: multiple instruction streams, single data stream**
  Multiple instructions operate on one data stream. Needed for fault tolerance (same results expected), yet extremely rare. Met in real-time computing architectures.

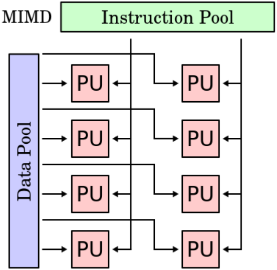


Cburnett - MISD, CC BY-SA 3.0

Architectural classification

Flynn's taxonomy - MIMD

- **MIMD: multiple instruction streams, multiple data streams** Multiple autonomous processors simultaneously executing different instructions on different data asynchronously. A typical parallel approach found in modern computing.



Cburnett - MIMD, CC BY-SA 3.0

Architectural classification

Feng's classification

- **Feng's classification** is a classification proposed by Tse-yun Feng in 1972
- it is based on the use of **degree of parallelism** to classify various computer architectures

Architectural classification

Feng's classification - details

- the **maximum number of bits that can be processed within a unit time** is called the **maximum parallelism degree P**
- now, let us define $\mathbf{P_i}$ as the **number of bits that can be processed within the ith processor cycle**; generally, $P_i \leq P$
- **average degree of parallelism** is expressed as $\mathbf{P_a}$

$$P_a = \frac{\sum_{i=1}^{T} P_i}{T} \tag{1}$$

Architectural classification

Feng's classification - utilization rate

- finally, **utilization rate** of a computer system in T cycles is defined as

$$\mu = \frac{P_a}{P} = \frac{\sum_{i=1}^{T} P_i}{P \cdot T} \tag{2}$$

- if the system is fully utilized, $P_i = P$ for all cycles resulting with $\mu = 1$
- **maximum degree of parallelism P** is given by the product of the number of bits in a word (n) and number of words in parallel (m), namely $P = (m, n)$

Architectural classification

Feng's classification - classes

Based on the approach proposed by Feng, four classes emerge:

- **Word Serial Bit Serial (WSBS)** - one bit of one selected word is processed at a time. This represents serial processing and needs maximum processing time $P = (1, 1)$.
- **Word Serial Bit Parallel (WSBP)** - all bits of a selected word are processed at a time. Bit parallel means all bits of a word $P = (1, *)$.
- **Word Parallel Bit Serial (WPBS)** - one bit from all words are processed at a time. Word parallel signifies selection of all words $P = (*, 1)$.
- **Word Parallel Bit Parallel (WPBP)** - all bits of all words are processed at a time. Results in maximum parallelism $P = (*, *)$.

Architectural classification

Handler's classification

- **Handler's classification** was proposed by Wolfgang Handler in 1977
- it addresses the computer at three levels:
  - PCU - program control unit
  - ALU - arithmetic logic unit
  - ELC - elementary logic circuit
- **PCU** controls ALU
- **ALU** corresponds to **functional unit** or processing element
- **ELC** corresponds to the **logic circuit** needed to perform one-bit operations in ALU

Architectural classification

Handler's classification - details

Handler's classification is describing the computer in a following way:

$$\mathbf{Computer} = (\mathbf{p} \cdot \mathbf{p'}, \mathbf{a} \cdot \mathbf{a'}, \mathbf{b} \cdot \mathbf{b'})$$

Where:

- p – number of PCUs
- p' – number of PCUs that can be pipelined
- a – number of ALUs controlled by each PCU
- a' – number of ALUs that can be pipelined
- b – number of bits in ALU or processing element word
- b' – number of pipeline segments on all ALUs or in a single PE

Architectural classification

Summary

- architectural classification distinguishes computer systems focusing on the degree of parallelism
- each classification evaluates those systems differently
- every computer system fits into a category
- the classifications discussed:
  - Flynn's taxonomy
  - Feng's classification
  - Handler's classification
- if you want to know more, read this document

Architectural classification

Sources & recommended materials

- B. Barney, *"Introduction to Parallel Computing"*, Lawrence Livermore National Laboratory (tutorial)
- S.S. Jadhav, *"Advanced Computer Architecture and Computing"*, Technical Publications Pune (book)
- N. Kandel, *"Feng's Classification"* (presentation)
- M. A. Salih, *"Architectural Classification"*, University of Technology, Iraq (presentation)

Memory hierarchy

Introduction

- memory is the place where instructions and data are located
- everyone wants unlimited amount of memory with low latency
- yet, the decrease in latency comes with the increase in cost
- **thus, the memory hierarchy was introduced**

Matt Kieffer - Crucial 1gb SDRAM (...), CC BY-SA 2.0

Memory hierarchy

Rationale

We are therefore forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible[2].

John von Neumann, public domain

[2] John Von Neumann and Goldstine Brucks. *Preliminary discussion of the logical design of an electronic computing instrument*. 1946.

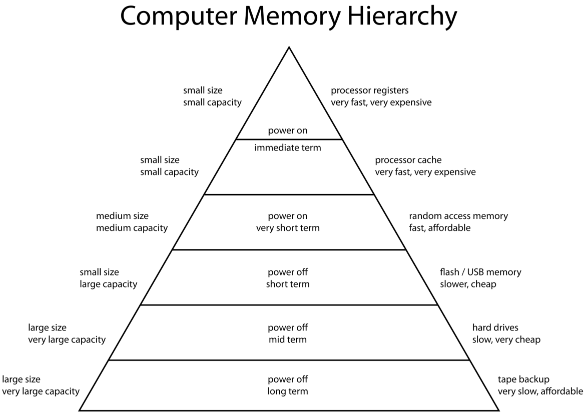
Memory hierarchy

Basic concept

- the **system memory is organized as a hierarchy**
- **entire addressable memory space** is available **in largest** and therefore slowest memory
- **incrementally smaller and faster memories**, each containing a subset of the memory below it, proceed in steps up **toward the processor**
- from the perspective of the **processor**, it **accesses only the fastest memory**

Memory hierarchy

Schema



Computer memory hierarchy, public domain

Memory hierarchy

From the top to the bottom

Processor registers:

- on top of the hierarchy
- the fastest, matched in speed to the CPU
- power-consuming
- small amount of registers

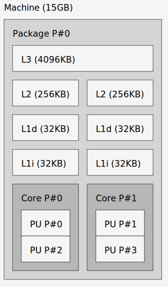
The rest (lower in hierarchy):

- slower
- less power-consuming
- cheaper

Memory hierarchy

Memory hierarchy - example #1

Lenovo™ ThinkPad® X260, Intel® Core™ i7-6500U, 16 GB RAM



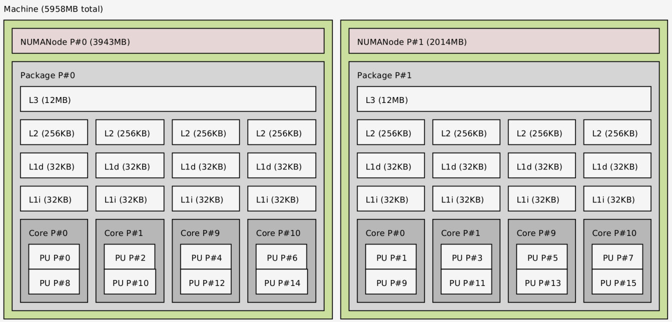
Screenshot from the application lstopo (package Portable Hardware Locality)

Lenovo, ThinkPad and ThinkStation are trademarks of Lenovo in the United States, other countries, or both. Intel, Intel Core and Xeon are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

Memory hierarchy

Memory hierarchy - example #2

Lenovo^{TM} ThinkStation® D20, Intel® Xeon^{TM} E5640, 6 GB RAM



Screenshot from the application lstopo (package Portable Hardware Locality)

Memory hierarchy

Volatility

For how long does data reside in memory?

- **until power off**: registers, cache, RAM
- **until removal**: hard drives, flash drives, tape storage

Yet, keep in mind that the volatile data can remain for a while after powering off the machine[3].

[3] J Alex Halderman et al. "Lest we remember: cold-boot attacks on encryption keys". In: *Communications of the ACM* 52.5 (2009), pp. 91–98.

Memory hierarchy

Locality

How we access data in memory?

- most often - **not randomly**
- **spatial locality** - data is more likely to be accessed if neighboring data is accessed
- **temporal locality** - data is more likely to be accessed if it has been recently accessed

This leads to introducing the concepts of **hot and cold storage** and various **memory management strategies**.

Memory hierarchy

Memory management

How to manage the memory wisely?

- firstly, **become familiar with your computer system memory hierarchy**
- put the **most often access data in the upper ranges** of the memory hierarchy
- put **unimportant data in the lower ranges** of the hierarchy
- understand that there is always **a trade-off between the latency and cost**
- observe **how the computer system is using the memory** in its current organization and introduce changes according to your needs

Memory hierarchy

CPU and cache

How does the CPU use the cache?

- **CPU seeks for data in cache**, if it is not there, it is loaded from the lower levels of hierarchy
- if CPU will find a block in cache, it is **cache hit**, otherwise **cache miss**
- **hit rate** is the fraction of memory access found in the upper level
- **miss rate** equals 1 - hit rate

Memory hierarchy

Summary

- memory hierarchy introduces the order of the memory types
- upper levels - faster but more expensive
- lower levels - slower but less expensive
- memory volatility - lifetime of data
- memory locality - which data will be accessed
- CPU and cache relationship
- if you want to know more, read this document

To do: use the tool **lstopo** from the package Portable Hardware Locality to learn about your computer memory organization.

Memory hierarchy

Sources & recommended materials

- T. Schwarz, *“Introduction to Information Storage Technology”*, Santa Clara University, CA, United States (course materials)
- D. Patterson, J. Hennessy, *“Computer Architecture: A Quantitative Approach”*, Elsevier (book)
- R. Bryant, G. Ganger, *“15-213: Introduction to Computer Systems”*, Carnegie Mellon School of Computer Science, PA, United States (course materials)

Computer architectures

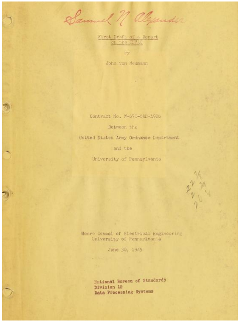
Introduction

- **What is a computer architecture?**
  It is the design of organization and interaction of crucial components of a computer system.
- **Most important computer architectures:**
  - Princeton (von Neumann) architecture
  - Harvard architecture
  - Harvard-Princeton architecture

Computer architectures

Princeton architecture - introduction

- Proposed by John von Neumann in 1945
- Princeton architecture consists of:
  - memory
  - ALU
  - control unit
  - I/O devices



Samuel N Alexander

First Draft of a Report
on the EDVAC

by

John von Neumann

Contract No. W-670-ORD-4926

Between the

United States Army Ordnance Department

and the

University of Pennsylvania

Moore School of Electrical Engineering
University of Pennsylvania

June 30, 1945

National Bureau of Standards
Division 12
Data Processing Systems

EDVAC report, public domain

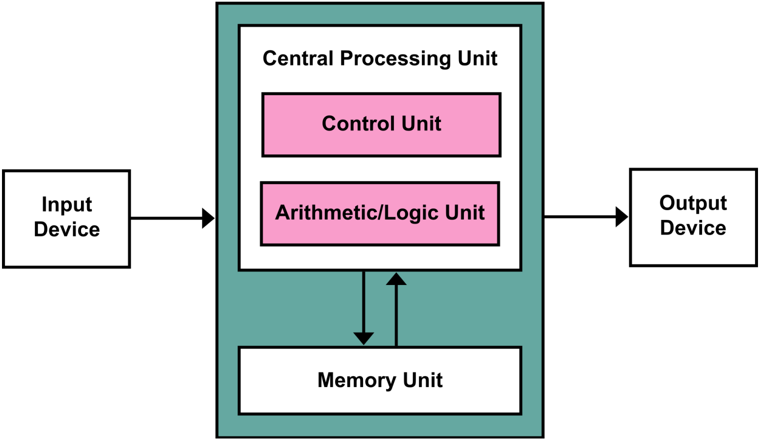
Computer architectures

Princeton architecture - details

- **memory and devices are controlled by the CPU**
- all the components are **connected by a system bus**
- **data** traverses trough system bus in **half duplex mode**
- **memory** holds **program and data** (stored-program concept)

Computer architectures

Princeton architecture - schema



Kapooht - Von Neumann Architecture, CC BY-SA 3.0

Computer architectures

Princeton architecture - advantages and disadvantages

Advantages:

- CU gets the data and instructions in a unified way from one memory
- data from memory and devices is accessed in the same way
- programmers can organize the memory according to their needs

Disadvantages:

- serial instruction processing (no parallelism)
- single system bus is a bottleneck
- as instructions are in the same memory they can be rewritten

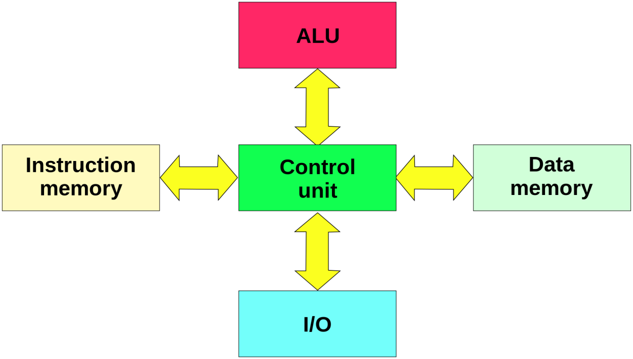
Computer architectures

Harvard architecture - introduction

- Harvard architecture has been introduced in Harvard Mark I computer (1944)
- the main difference: **data and instructions are separate**
- therefore it is possible to access them **simultaneously**
- the **program cannot modify itself**

Computer architectures

Harvard architecture - schema



Nessa los - Harvard architecture, CC BY-SA 3.0

Computer architectures

Harvard architecture - advantages and disadvantages

Advantages:

- parallel access to data and instructions
- data and instructions are accessed the same way
- different sizes of memory cells

Disadvantages:

- spare memory cannot be used by second type of memory
- more expensive because of the need of having and controlling two buses

Computer architectures

Harvard-Princeton architecture

- links both architectures overcoming their limitations
- **separate upper layers of memory** hierarchy and **common the lower ones**
- at least **one level of cache is separated for data and instructions** (remember lstopo results?)
- **fast** due to parallelism introduced by separating this level of memory
- keeping **flexibility for programmers**

Most of general purpose computers follow this architecture (also called modified Harvard architecture).

Computer architectures

Summary

- Harvard and Princeton architectures differ how they access memory for data and instructions
- Harvard has separate buses for memory and data
- Princeton (von Neumann's) has a single bus
- Harvard-Princeton (modified Harvard architecture) is a combination of both approaches
- if you want to know more, read this document

Computer architectures

Sources & recommended materials

- P. Dudzik, A. Guzik, *“Architektury komputerów i procesorów”*, AGH University of Science and Technology, Kraków, Poland (course materials)
- O. Matunga, *“Micro Computer Architecture”* (presentation)
- D. Patterson, J. Hennessy, *“Computer Architecture: A Quantitative Approach”*, Elsevier (book)

Final slide

Questions? Comments?

If you have any ideas on how to improve these lectures,
please submit them as issues in this repository:

https://github.com/rmhere/lecture-comp-arch-org