Hardware Architecture - Digital Circuits Introduction

Lecturer: Guillaume Beslon
(Lecture adapted from Lionel Morel)

3IF - Computer Science and Information Technologies - INSA Lyon

Fall 2023

These slides are available at:
https://moodle.insa-lyon.fr/course/view.php?id=1442

Preamble: Who am I?

Guillaume Beslon (guillaume.beslon@insa-lyon.fr)

- Professor at the INSA-Lyon “Département informatique"
  - Architecture des circuits (IF-3-AC)
  - Tronc Commun Scientifique (IF-5-TCS0) et Sciences computationnelles (IF-5-TCS2)
  - Projet Scientifique, Artistique et Technique (IF-5-P-SAT)
  - Module COINFO (BS-3-COINFO)
- Leader of the “Beagle Team” (INRIA/LIRIS)
  - Computational biology and artificial evolution
  - Artificial life
  - Computational neurosciences
- In charge of the “Lumière et Son” (aka “Tek”) artistic option in the département des Humanités

Beware: my main office is NOT in the computer department building
$\rightarrow$ You’ll find me at the “Centre Inria de Lyon” (CEI-2 building)

Preamble: Gentle Warning (1/2)

THE CONSUMER IN A CONNECTED WORLD

Brain Drain: The Mere Presence of One's Own Smartphone Reduces Available Cognitive Capacity

ADRIAN F. WARD, KRISTEN DUKE, AYELET GNEEZY, AND MAARTEN W. BOS

ABSTRACT Our smartphones enable—and encourage—constant connection to information, entertainment, and each other. They put the world at our fingertips, and rarely leave our sides. Although these devices have immense potential to improve welfare, their persistent presence may come at a cognitive cost. In this research, we test the "brain drain" hypothesis that the mere presence of one's own smartphone may occupy limited-capacity cognitive resources, thereby leaving fewer resources available for other tasks and undercutting cognitive performance. Results from two experiments indicate that even when people are successful at maintaining sustained attention—as when avoiding the temptation to check their phones—the mere presence of these devices reduces available cognitive capacity. Moreover, these cognitive costs are highest for those highest in smartphone dependence. We conclude by discussing the practical implications of this smartphone-induced brain drain for consumer decision-making and consumer welfare.

Ward, A. F., Duke, K., Gneezy, A., & Bos, M. W. (2017). Brain drain: The mere presence of one's own smartphone reduces available cognitive capacity. *Journal of the Association for Consumer Research*, 2(2), 140-154.

Preamble: Gentle Warning (1/2)

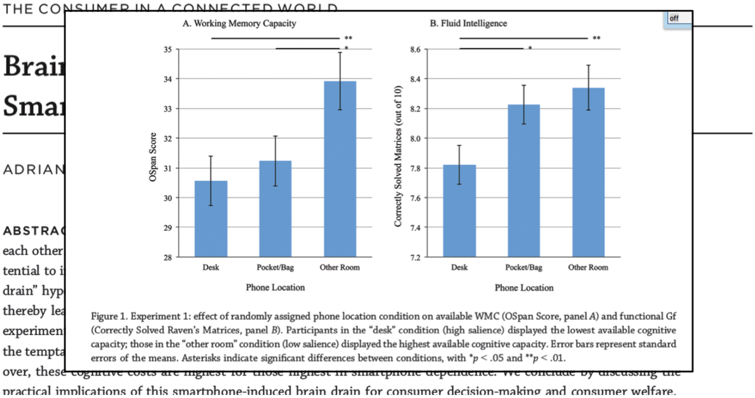


Figure 1. Experiment 1: effect of randomly assigned phone location condition on available WMC (OSpan Score, panel *A*) and functional Gf (Correctly Solved Raven's Matrices, panel *B*). Participants in the "desk" condition (high salience) displayed the lowest available cognitive capacity; those in the "other room" condition (low salience) displayed the highest available cognitive capacity. Error bars represent standard errors of the means. Asterisks indicate significant differences between conditions, with $^{*}p < .05$ and $^{**}p < .01$.

Ward, A. F., Duke, K., Gneezy, A., & Bos, M. W. (2017). Brain drain: The mere presence of one's own smartphone reduces available cognitive capacity. *Journal of the Association for Consumer Research*, 2(2), 140-154.

Preamble: Gentle Warning (2/2)

Psychological Science OnlineFirst, published on May 22, 2014 as doi:10.1177/0956797614524581

ASSOCIATION FOR PSYCHOLOGICAL SCIENCE

Research Article

Psychological Science
1–10

DOI: 10.1177/0956797614524581
pss.sagepub.com
SAGE

The Pen Is Mightier Than the Keyboard: Advantages of Longhand Over Laptop Note Taking

Pam A. Mueller[1] and Daniel M. Oppenheimer[2]
[1]Princeton University and [2]University of California, Los Angeles

Abstract
Taking notes on laptops rather than in longhand is increasingly common. Many researchers have suggested that laptop note taking is less effective than longhand note taking for learning. Prior studies have primarily focused on students' capacity for multitasking and distraction when using laptops. The present research suggests that even when laptops are used solely to take notes, they may still be impairing learning because their use results in shallower processing. In three studies, we found that students who took notes on laptops performed worse on conceptual questions than students who took notes longhand. We show that whereas taking more notes can be beneficial, laptop note takers' tendency to transcribe lectures verbatim rather than processing information and reframing it in their own words is detrimental to learning.

Mueller, P. A., & Oppenheimer, D. M. (2014). The pen is mightier than the keyboard: Advantages of longhand over laptop note taking. *Psychological science*, 25(6), 1159-1168.

Preamble: Gentle Warning (2/2)

Psychological Science OnlineFirst, published on May 22, 2014 as doi:10.1177/0956797614524581

Research Article

The Pen
Advantag
Note Tak

Pam A. Muell
[1]Princeton University a

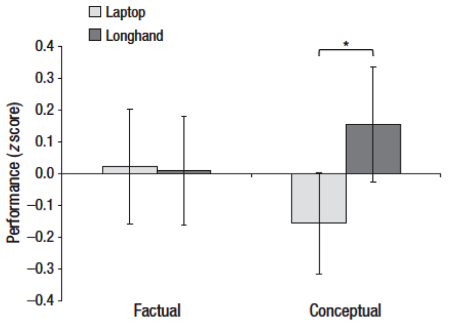


Fig. 1. Mean *z*-scored performance on factual-recall and conceptual-application questions as a function of note-taking condition (Study 1). The asterisk indicates a significant difference between conditions ($p < .05$). Error bars indicate standard errors of the mean.

Abstract
Taking notes on la … ggested that laptop note taking is less … cused on students' capacity for multit … ven when laptops are used solely to … llower processing. In three studies, w … ual questions than students who took … aptop note takers' tendency to transcribe lectures verbatim rather than processing information and reframing it in their own words is detrimental to learning.

Mueller, P. A., & Oppenheimer, D. M. (2014). The pen is mightier than the keyboard: Advantages of longhand over laptop note taking. *Psychological science*, 25(6), 1159-1168.

But it’s your own business, do as you want!

Back to the AC lecture...

Context: Architectures - Systèmes - Réseaux

3IF

- Semester 1:
  - IF-3-AC - Architecture des Circuits (Guillaume Beslon)
  - IF-3-AO - Architecture des Ordinateurs (Lionel Morel)
  - IF-3-PRC - Programmation C (Frédéric Prost)
- Semester 2:
  - IF-3-SYS - Systèmes d'Exploitation (Guillaume Salagnac)
  - IF-3-RE - Bases Techniques pour les réseaux (Frédérique Biennier)

4IF

- Semester 1:
  - IF-4-PR - Programmation réseau (Sara Bouchenak)
- Semester 2:
  - IF-4-SERE - Sécurité Réseau (Lionel Brunie)
  - IF-4-PLD-COMP - Projet Compilateur (Florent de Dinechin)

AC, AO, SYS: Objectifs

Objectives (cf ECTS files)

- AC: “Découvrir les principes théoriques et pratiques qui régissent le fonctionnement des circuits numériques, des portes logiques de base jusqu’à la construction d’un microprocesseur simple.”
- AO: “Comprendre le fonctionnement d’un ordinateur moderne et les fondements de l’exécution d’un programme sur une machine.”
- SYS: “Acquérir une compréhension basique des principes de fonctionnement des systèmes d’exploitation: partage et protection des ressources matérielles, isolation des programmes, interaction avec l’utilisateur.”

People — `first.last@insa-lyon.fr`

In order of appearance:

- Guillaume Beslon (Lectures, TD-TP 3IF1)
- Lucas Chaloyard (TD-TP 3IF2, TP 3IF3)
- Lionel Morel (TD-TP 3IF3, TP 3IF2)
- Florent de Dinechin (TD-TP 3IF4)
- Jonathan Rouzaud-Cornabas (TP 3IF2, 3IF4)
- Guillaume Salagnac (TP 3IF1 + IF-3-SYS)

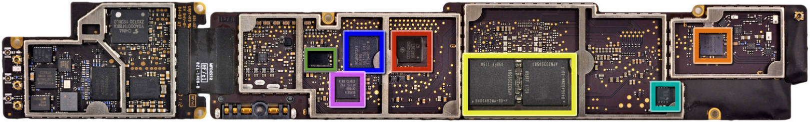
But permutations may (and actually will) happen here and there...

None of us has his office in the computer science department... and we are all very busy!
$\Rightarrow$We organize Q/A sessions every Monday 13h-14h to answer your questions...

What is there in a computer[1]?

[1] source: `http://www.ifixit.com`

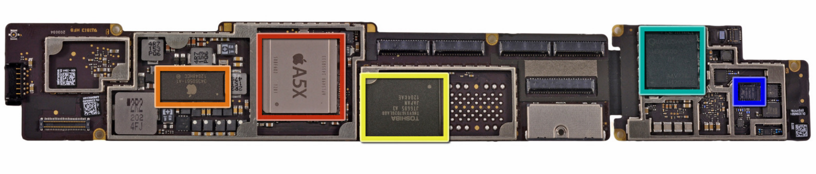
What is there in a computer[2]?



- DRAM
- Voltage conversion booster
- IO controller
- MAC/baseband/radio (FM) transceiver
- Audio codec
- device driver

[2] source: `http://www.ifixit.com`

What is there in a computer[3]?



- NAND Flash
- 3G/4G modem
- RF Amp module
- Power supply IC
- A5X processor

[3] source: http://www.ifixit.com

What is there in a computer?

In AC we will neglect all the “extra-components” (screen, battery, power supply...) to focus on the computation machinery (i.e. mainly on the processor and a bit on the memory)

We will focus on a single question: how are computers organized such that they are able to efficiently execute programs and such that we are able to efficiently control/program them!

BUT remember that:

- 85% of the environmental impact of a computer is due to manufacture and shipping,
- Computers require rare resources for their manufacture (lithium, gold, silver, neodymium...) which extraction has considerable ecological and social impact.

$\rightarrow$ New lecture to come in 4IF (Lionel Morel)

Computer Architecture: What do you know so far?

You (probably) know that computers manipulate 0s and 1s...

0110010001
0101110101
0101001010
0010010010
1011010101
0100101010
1010100101
0010101110
0101001001
0010100100
1100101010
0011110110
0101001001
1101011010
1100100111

Computer Architecture: What do you know so far?

But how can we build computers that execute programs if we only have 0s and 1s?

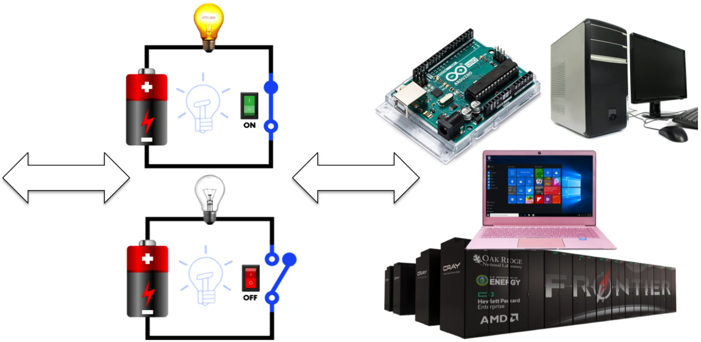
0110010001
0101110101
0101001010
0010010010
1011010101
0100101010
1010100101
0010101110
0101001001
0010100100
1100101010
0011110110
0101001001
1101011010
1100100111

?

Computer Architecture: What do you know so far?

You (probably) know that 0s and 1s are actually electrical levels in wires...

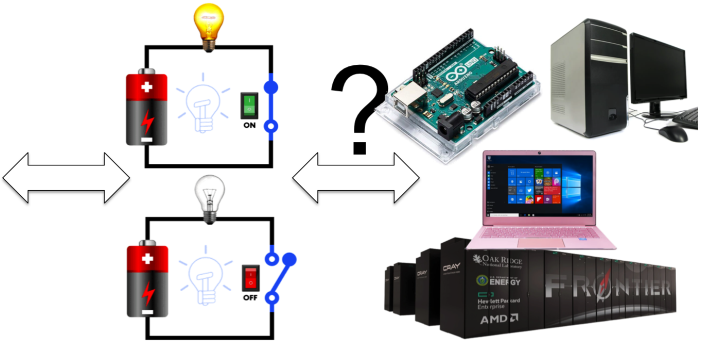
0110010001
0101110101
0101001010
0010010010
1011010101
0100101010
1010100101
0010101110
0101001001
0010100100
1100101010
0011110110
0101001001
1101011010
1100100111



Computer Architecture: What do you know so far?

But the problem holds! How can we build computers "simply" from wires?

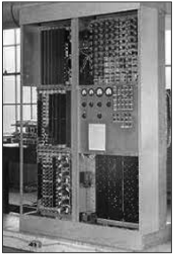
0110010001
0101110101
0101001010
0010010010
1011010101
0100101010
1010100101
0010101110
0101001001
0010100100
1100101010
0011110110
0101001001
1101011010
1100100111



Computer Architecture: What do you know so far?

To answer the question we will focus on a some universal design principles invented in the early times of computer science...

0110010001
0101110101
0101001010
0010010010
1011010101
0100101010
1010100101
0010101110
0101001001
0010100100
1100101010
0011110110
0101001001
1101011010
1100100111



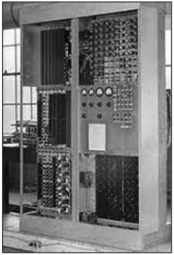
The EDVAC (Electronic Discrete Variable Automatic Computer), 1945

Computer Architecture: What do you know so far?

The following courses (3-IF-AO, 3-IF-SYS...) will then transpose these principles into modern machines...

0110010001
0101110101
0101001010
0010010010
1011010101
0100101010
1010100101
0010101110
0101001001
0010100100
1100101010
0011110110
0101001001
1101011010
1100100111

IF-3-AC

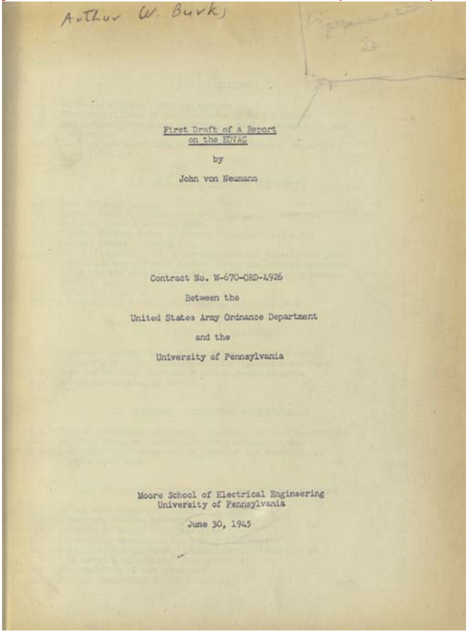


IF-3-AO
IF-3-SYS

The EDVAC (Electronic Discrete Variable Automatic Computer), 1945

John von Neumann and the EDVAC

First Draft Report on the EDVAC (1945)



Arthur W. Burks

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

The considerations which follow deal with the structure of a very high speed ***automatic digital computing*** *system, and in particular with its logical control.*

An automatic computing system is a (usually highly composite) device, which can ***carry out instructions*** *to perform calculations of a considerable order of complexity—e.g. to solve a non-linear partial differential equation in 2 or 3 independent variables numerically.*

bold faces added by us :)

Basic concepts in von Neumann's architecture

Example: Computing the sum of all integers from 1 to 100?

```
int main()
{
  int x,i;
  x = 0;
  i = 0;
  for (i = 1; i<100;i++)
    {
      x = x+1;
    }
  return x;
```

Let’s transform this high level list of orders into elementary instructions

```
int main()
{
  int x,i;
  x = 0;
  i = 0;
  for (i = 1; i<100;i++)
    {
      x = x+1;
    }
  return x;
```

```
0x0000000100000f70 <+0>:     push    %rbp
0x0000000100000f71 <+1>:     mov     %rsp,%rbp
0x0000000100000f74 <+4>:     movl    $0x0,-0x4(%rbp)
0x0000000100000f7b <+11>:    movl    $0x0,-0x8(%rbp)
0x0000000100000f82 <+18>:    movl    $0x0,-0xc(%rbp)
0x0000000100000f89 <+25>:    movl    $0x1,-0xc(%rbp)
0x0000000100000f90 <+32>:    cmpl    $0x64,-0xc(%rbp)
0x0000000100000f94 <+36>:    jge     0x100000fb1 <main()+65>
0x0000000100000f9a <+42>:    mov     -0x8(%rbp),%eax
0x0000000100000f9d <+45>:    add     $0x1,%eax
0x0000000100000fa0 <+48>:    mov     %eax,-0x8(%rbp)
0x0000000100000fa3 <+51>:    mov     -0xc(%rbp),%eax
0x0000000100000fa6 <+54>:    add     $0x1,%eax
0x0000000100000fa9 <+57>:    mov     %eax,-0xc(%rbp)
0x0000000100000fac <+60>:    jmpq    0x100000f90 <main()+32>
0x0000000100000fb1 <+65>:    mov     -0x8(%rbp),%eax
0x0000000100000fb4 <+68>:    pop     %rbp
```

And code theses instructions with numbers...

```
0x0000000100000f70 <+0>:     push   %rbp
0x0000000100000f71 <+1>:     mov    %rsp,%rbp
0x0000000100000f74 <+4>:     movl   $0x0,-0x4(%rbp)
0x0000000100000f7b <+11>:    movl   $0x0,-0x8(%rbp)
0x0000000100000f82 <+18>:    movl   $0x0,-0xc(%rbp)
0x0000000100000f89 <+25>:    movl   $0x1,-0xc(%rbp)
0x0000000100000f90 <+32>:    cmpl   $0x64,-0xc(%rbp)
0x0000000100000f94 <+36>:    jge    0x100000fb1 <main()+65>
0x0000000100000f9a <+42>:    mov    -0x8(%rbp),%eax
0x0000000100000f9d <+45>:    add    $0x1,%eax
0x0000000100000fa0 <+48>:    mov    %eax,-0x8(%rbp)
0x0000000100000fa3 <+51>:    mov    -0xc(%rbp),%eax
0x0000000100000fa6 <+54>:    add    $0x1,%eax
0x0000000100000fa9 <+57>:    mov    %eax,-0xc(%rbp)
0x0000000100000fac <+60>:    jmpq   0x100000f90 <main()+32>
0x0000000100000fb1 <+65>:    mov    -0x8(%rbp),%eax
0x0000000100000fb4 <+68>:    pop    %rbp
```

```
0000f70 55 48 89 e5 c7 45 fc 00 00 00 00 c7 45 f8 00 00
0000f80 00 00 c7 45 f4 00 00 00 00 c7 45 f4 01 00 00 00
0000f90 83 7d f4 64 0f 8d 17 00 00 00 8b 45 f8 83 c0 01
0000fa0 89 45 f8 8b 45 f4 83 c0 01 89 45 f4 e9 df ff ff
0000fb0 ff 8b 45 f8 5d c3 90 90 01 00 00 00 1c 00 00 00
0000fc0 00 00 00 00 1c 00 00 00 00 00 00 00 1c 00 00 00
0000fd0 02 00 00 00 70 0f 00 00 34 00 00 00 34 00 00 00
```

And code theses instructions with numbers...

```
0x0000000100000f70 <+0>:     push   %rbp
0x0000000100000f71 <+1>:     mov    %rsp,%rbp
0x0000000100000f74 <+4>:     movl   $0x0,-0x4(%rbp)
0x0000000100000f7b <+11>:    movl   $0x0,-0x8(%rbp)
0x0000000100000f82 <+18>:    movl   $0x0,-0xc(%rbp)
0x0000000100000f89 <+25>:    movl   $0x1,-0xc(%rbp)
0x0000000100000f90 <+32>:    cmpl   $0x64,-0xc(%rbp)
0x0000000100000f94 <+36>:    jge    0x100000fb1 <main()+65>
0x0000000100000f9a <+42>:    mov    -0x8(%rbp),%eax
0x0000000100000f9d <+45>:    add    $0x1,%eax
0x0000000100000fa0 <+48>:    mov    %eax,-0x8(%rbp)
0x0000000100000fa3 <+51>:    mov    -0xc(%rbp),%eax
0x0000000100000fa6 <+54>:    add    $0x1,%eax
0x0000000100000fa9 <+57>:    mov    %eax,-0xc(%rbp)
0x0000000100000fac <+60>:    jmpq   0x100000f90 <main()+32>
0x0000000100000fb1 <+65>:    mov    -0x8(%rbp),%eax
0x0000000100000fb4 <+68>:    pop    %rbp
```

```
0000f70 55 48 89 e5 c7 45 fc 00 00 00 00 c7 45 f8 00 00
0000f80 00 00 c7 45 f4 00 00 00 00 c7 45 f4 01 00 00 00
0000f90 83 7d f4 64 0f 8d 17 00 00 00 8b 45 f8 83 c0 01
0000fa0 89 45 f8 8b 45 f4 83 c0 01 89 45 f4 e9 df ff ff
0000fb0 ff 8b 45 f8 5d c3 90 90 01 00 00 00 1c 00 00 00
0000fc0 00 00 00 00 1c 00 00 00 00 00 00 00 1c 00 00 00
0000fd0 02 00 00 00 70 0f 00 00 34 00 00 00 34 00 00 00
```

> *At any rate a* ***central arithmetical part*** *of the device* ***will probably have to exist****, and this constitutes the first specific part:* ***CA****.*

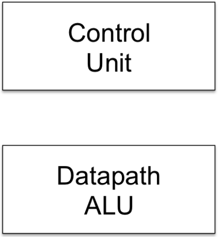
We need a component to execute these elementary instructions $\Rightarrow$ This is the **Datapath.** It contains an **Arithmetic and Logic Unit** able to perform basic numerical computations**.**

von Neumann architecture

> *A distinction must be made between the specific instructions given for and defining a particular problem, and the* ***general control organs*** *which see to it that these instructions—****no matter what they are****—are carried out [...] By the* ***central control*** *we mean this latter function only, and the organs which perform it form the second specific part:* ***CC****.*

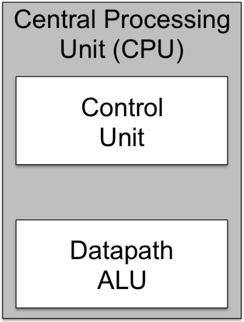
We need a component to sequence the elementary instructions
$\Rightarrow$ This is the **Control Unit.**

von Neumann architecture



von Neumann architecture

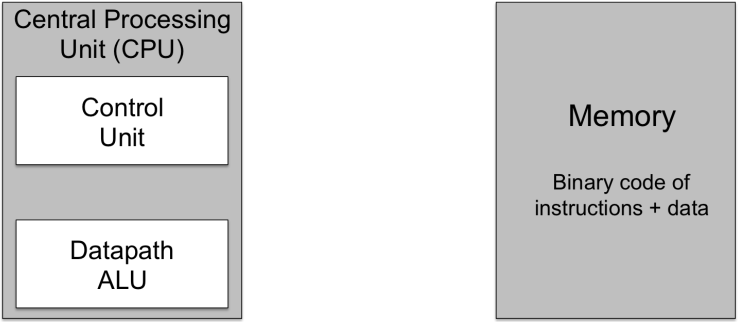
The Datapath and the Control Unit together form the **Central Processing Unit** of the computer



*At any rate the **total memory** constitutes the third specific part of the device: **M**.*

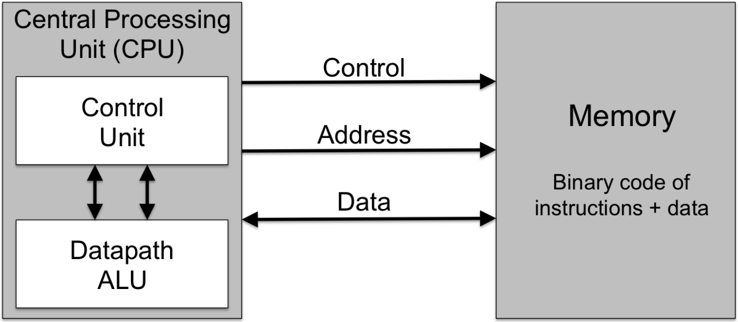
$\Rightarrow$Memory able to store a large number of ... numbers!

von Neumann architecture



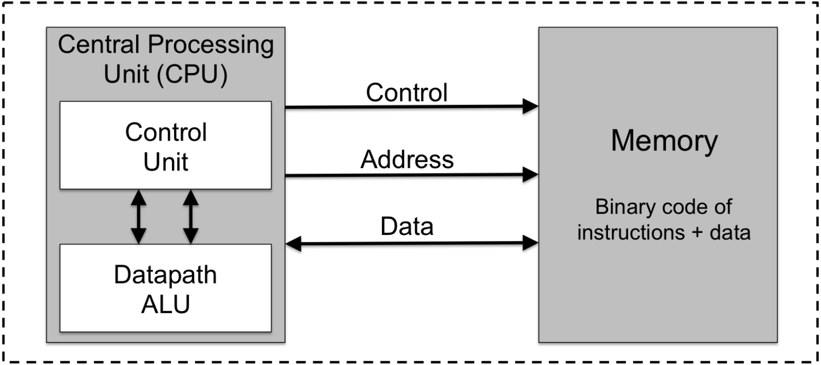
von Neumann architecture

Of course we need all these components to communicate with each others... $\Rightarrow$ **Buses**



von Neumann architecture

75 years after being articulated, the von Neumann architecture is still the basic architecture of most modern computers...



Conclusion: A computer is made of 3 (+1) elements

1. A **Memory** that contains the program and its data (von Neumann architecture).
2. A **Datapath**. It is a computing tool that is able to perform various computations.
3. A **Control Unit**. It reads the program one instruction at a time and controls the datapath such that it computes the result of the current instruction.

+1 **Buses** that enable to exchange data between the three components.

The objective of the AC lecture is to understand how we can build these elements (starting from very basic components) and assemble them to build a (simple) computer.

Coarse-grain plan for AC lecture

In AC, we will take a bottom-up approach with six steps:

1. How information is coded (binary)?
2. How can we process an information to compute other information from it (e.g. simple mathematical functions)?
3. How can we memorize information?
4. How to build machines with "simple" behaviors?
5. How to build machines with "complex" behaviors?
6. How to build von Neumann machines able to execute a sequence of instructions?

- Through the end of the course, we will build a (very) simple programmable machine
- The “Computer Architecture” course will further this discussion towards “real” computers.

IF-3-AC: Expected Skills

- Coding and decoding information in binary
- Building combinatorial circuits from Boolean functions
- Building simple memory elements (registers, memories)
- Modelling simple sequential behaviour with Finite-State Machines (FSM)
- Modelling complex sequential behaviour with Algorithmic-State Machines (ASM)
- Building a von Neumann machine able to execute simple programs
- Understanding basic performance issues of digital circuits

Practical matters

Evaluation

Final grade calculated from:

- 3 moodle quizzes (40% of the final grade):
  - week 43 (self-evaluation)
  - week 46 (evaluated, 20% of the final grade)
  - week 48 (evaluated, 20% of the final grade)
- 1 final exam (60% of the final grade):
  - 1h30
  - Date: Tuesday December 21th, 8:30AM-10:00AM

Q/A sessions

Every Monday from 1 to 2 PM, Room 501.208.
Except November 13th and Novembre 27th !!!

Organization of lab. works

5 labs sessions

- 2 "Travaux Dirigés" (TDs, 2h each)
- 3 "Travaux Pratiques" (TPs, 4h each)

Organization of the Practical Work sessions

- We will use the “Digital” simulation platform (free, multi-OS, digital circuits simulator).
- Booklet with all lab instructions for the module is available on Moodle (no paper-version will be provided).
- Labwork will not be evaluated. Learning is the unique objective ...
- Important: You are asked to start working on the labworks before the lab sessions.
- Final objective: Build a (very simple – 4 instructions!) “computer” able to control a scrolling display.

Demo time

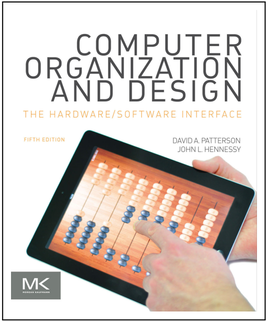
Practical matters - Agenda (tentative)

week	Lecture	Labs	Eval.
W38	lec 1: coding (today;)		
W39	lec 2: basic circuits	TD coding	
W40	lec 3: memories		
W41		TD boolean logic	
W42			
W43	lec 4: FSM		MCQ1
W44	Holidays		
W45	lec 5: complex circuits	TP basic circuits	
W46	lec 6: von Neumann	TP complex circuits	MCQ2
W47		TP Von Neumann machine	
W48			MCQ3
W51	Final Exam		

Readings

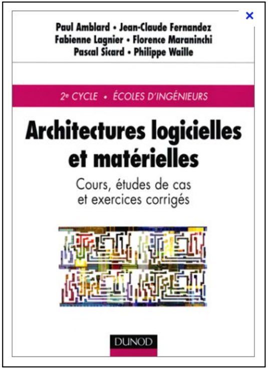
The lecture “poly”

F. de Dinechin



Computer Organization and Design

D. Patterson & J. Hennessy



Architecture Logicielle et Matérielle

P. Amblard et al

All you need is on Moodle

... hmm, actually *will be*...

Accueil Tableau de bord Mes cours Aide Accès rapides

Informatique / Informatique / Informatique / Informatique / IF-3

3IF - Architecture des circuits numériques

Cours Paramètres Participants Notes Rapports Plus

Organisation

Course Info

Code: IF-3-AC

ECTS: 2.0

Lectures Hours: 9hrs (6*1.5hrs)

Lab Hours: 16hrs (2*2hrs + 3*4hrs)

Personal Work: 25hrs

Language: spoken French, lecture slides in English, labworks in French.

ECTS description: EN/FR

`https://moodle.insa-lyon.fr/course/view.php?id=1442`