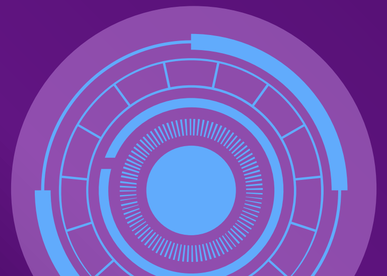
Marcela Gonçalves dos Santos
marcela.goncalvesdossantos@concordia.ca

Computer Architecture

Computer Hierarchy and Laws

Computer Level Hierarchy



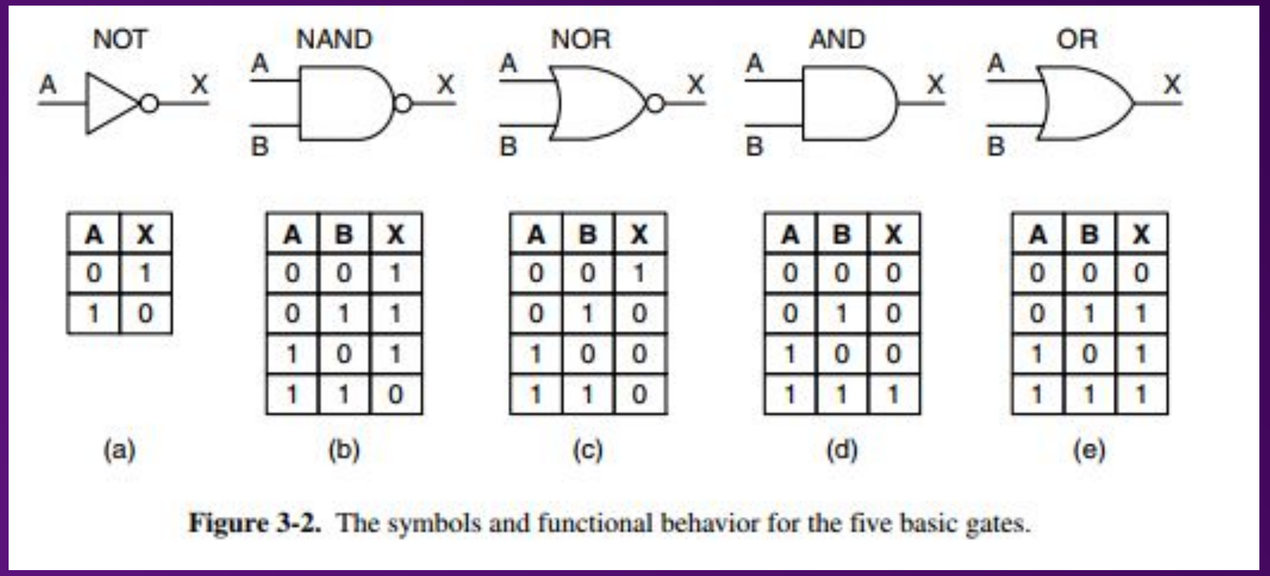
Computer Level Hierarchy

- Abstraction layers
- Divide and conquer approach
- Virtual representations
- Possibility to work entirely on each layer

Level 6	User	Executable Programs
Level 5	High Level Language	C++ , Java
Level 4	Assembly Language	Assembly Code
Level 3	System Software	Operating System
Level 2	Machine	Instruction Set Architecture
Level 1	Control	Microcode
Level 0	Digital Logic	Circuits , Gates

Digital Logic level

- At the lowest level that we will study, the digital logic level, the interesting objects are called gates.



A	X
0	1
1	0

(a)

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)

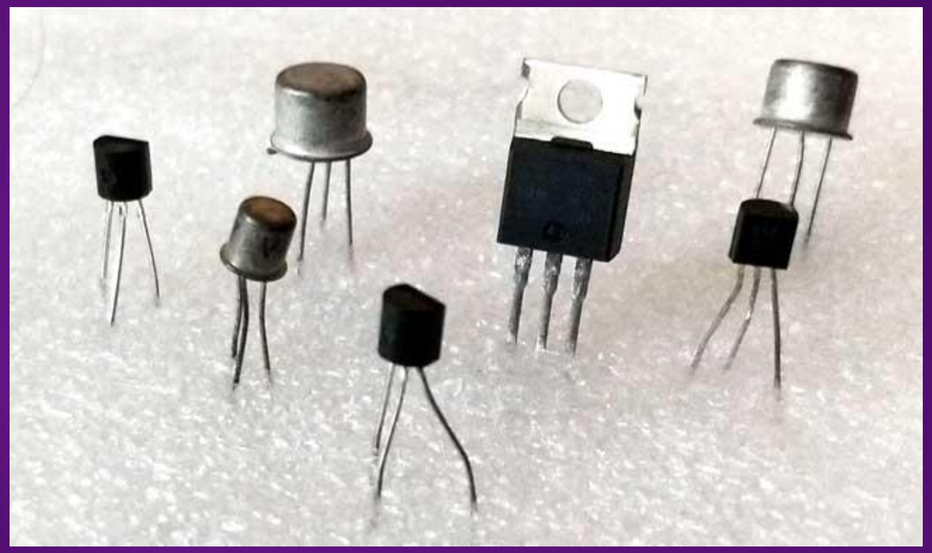
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

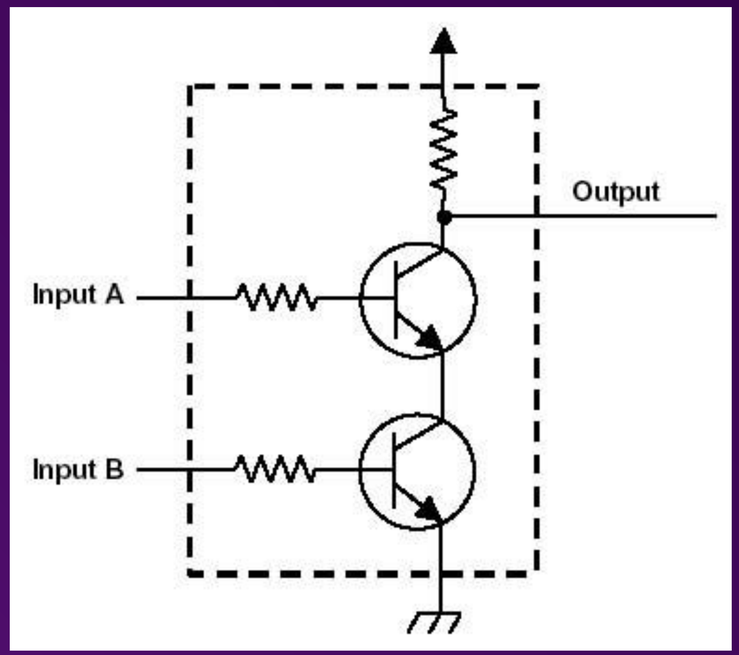
(e)

Figure 3-2. The symbols and functional behavior for the five basic gates.

Digital Logic level

- Although built from analog components, such as transistors, gates can be accurately modeled as digital devices.





Digital Logic level

- Each gate has one or more digital inputs (signals representing 0 or 1) and computes as output some simple function of these inputs, such as AND or OR.

- Each gate is built up of at most a handful of transistors.

Digital Logic level

- A small number of gates can be combined to form a **1-bit memory**, which can store a 0 or a 1.
- The 1-bit memories can be combined in groups of (for example) **16, 32, or 64 to form registers.**
- Each register can hold a single binary number up to some maximum.

Computer Level Hierarchy

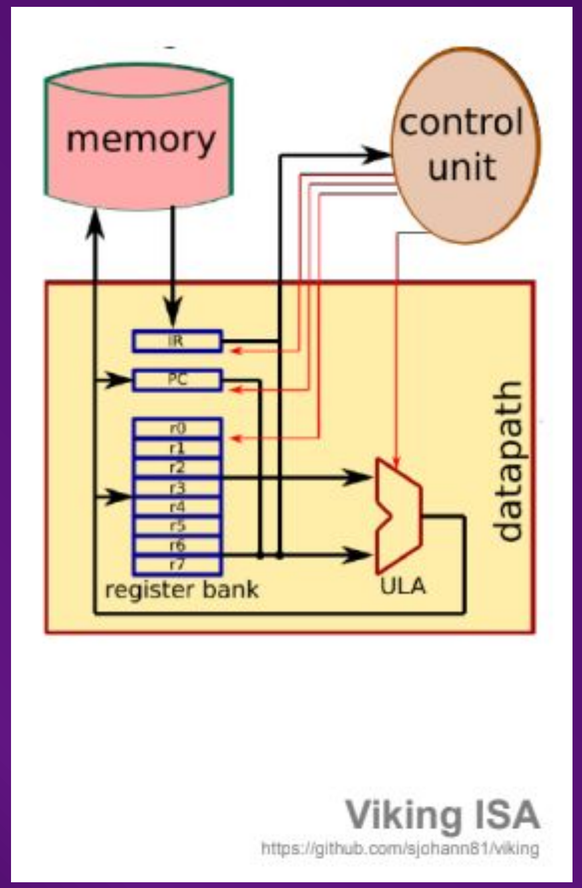
- Abstraction layers
- Divide and conquer approach
- Virtual representations
- Possibility to work entirely on each layer

Level 6	User	Executable Programs
Level 5	High Level Language	C++ , Java
Level 4	Assembly Language	Assembly Code
Level 3	System Software	Operating System
Level 2	Machine	Instruction Set Architecture
Level 1	Control	Microcode
Level 0	Digital Logic	Circuits , Gates

Control Logic level

- Manages signal transactions and data exchange within internal components and I/Os.
- Directly related to CPU Datapath, Bus and Registers
- Hardwired or microprogrammed

Control Logic level - Data Path

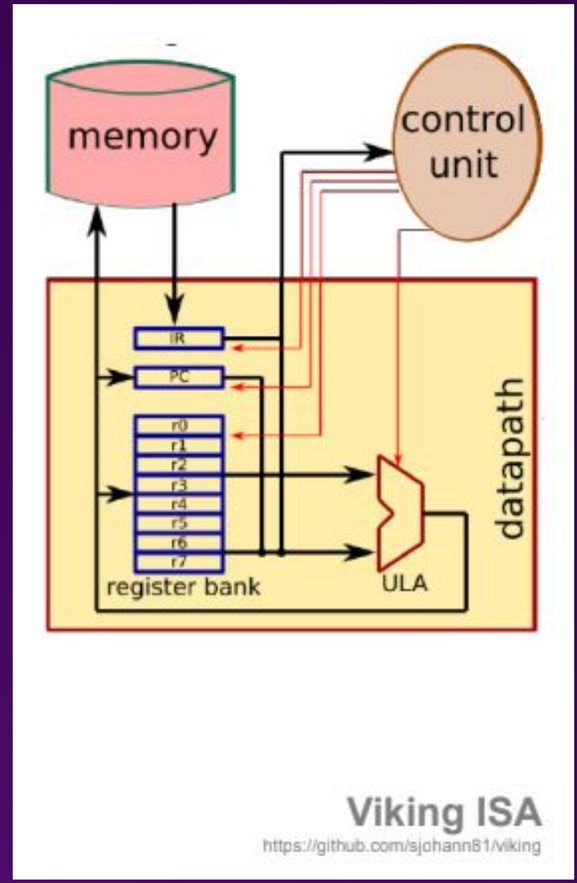


Instruction	Class	Operation
AND	Computation	Logical product
OR	Computation	Logical sum
XOR	Computation	Logical diff
SLT	Computation	Set if less than
ADD	Computation	Add
SUB	Computation	Subtract
LDW / LDB	Load/Store	Load word/byte
STW / STB	Load/Store	Store word/byte
BEZ	Branch	Branch if equal zero
BNZ	Branch	Branch if diff zero
LDI	Pseudo	Load immediate
HCF	Pseudo	Halt and catch fire

Control Logic level - Data Path

- The data path is that part of the CPU containing the ALU, its inputs, and its outputs.

- Execute instructions, store results and keeps processor state



Let's talk more about the registers

MAR	Memory Address Register	*Memory location of data to be accessed*
MDR	Memory Data Register	*Data transferred to or from memory*
AC	Accumulator	*Temporary operations results storage*
PC	Program Counter	*Next instruction address*
CIR	Current Instruction Register	*Current instruction during processing*

Activity 1

Present your understanding of what are the main aspects of the first two abstraction levels seem so far

- What is the digital logic level?
- What is the control level?
- What is datapath?

- Groups of 2 students

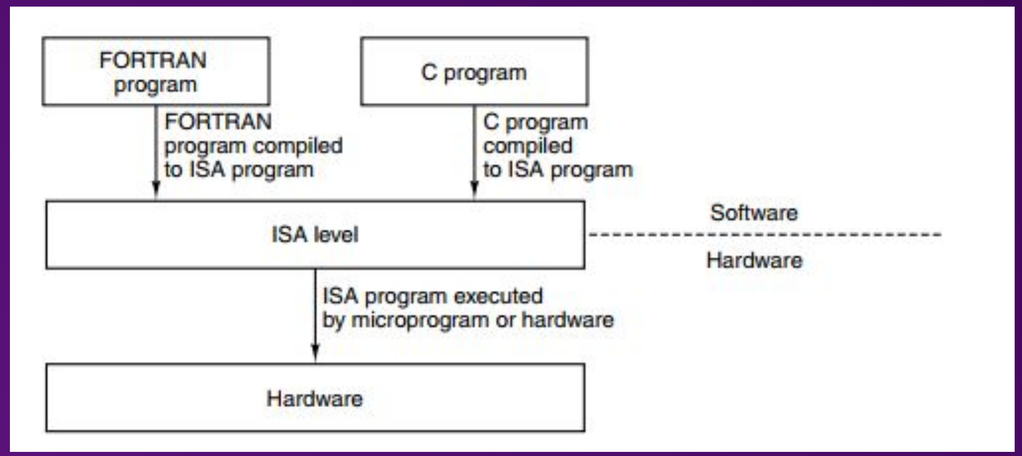
Computer Level Hierarchy

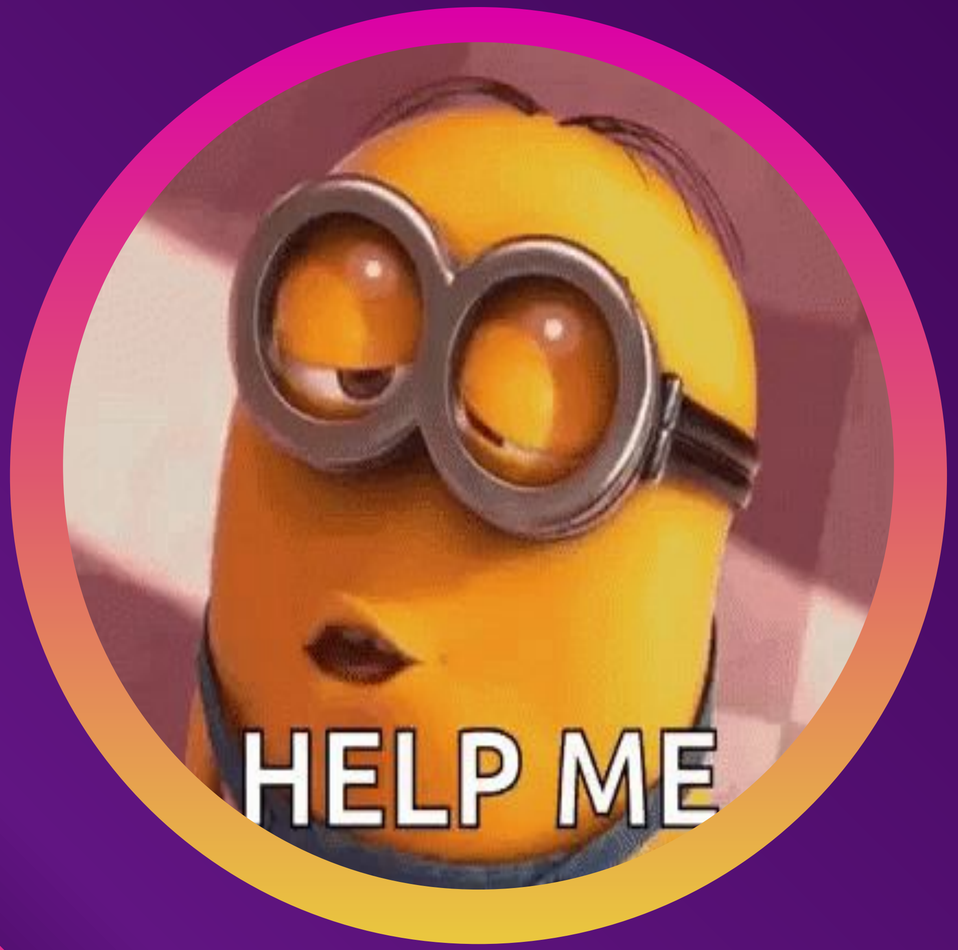
- Abstraction layers
- Divide and conquer approach
- Virtual representations
- Possibility to work entirely on each layer

Level 6	User	Executable Programs
Level 5	High Level Language	C++ , Java
Level 4	Assembly Language	Assembly Code
Level 3	System Software	Operating System
Level 2	Machine	Instruction Set Architecture
Level 1	Control	Microcode
Level 0	Digital Logic	Circuits , Gates

Instruction Set Architecture level

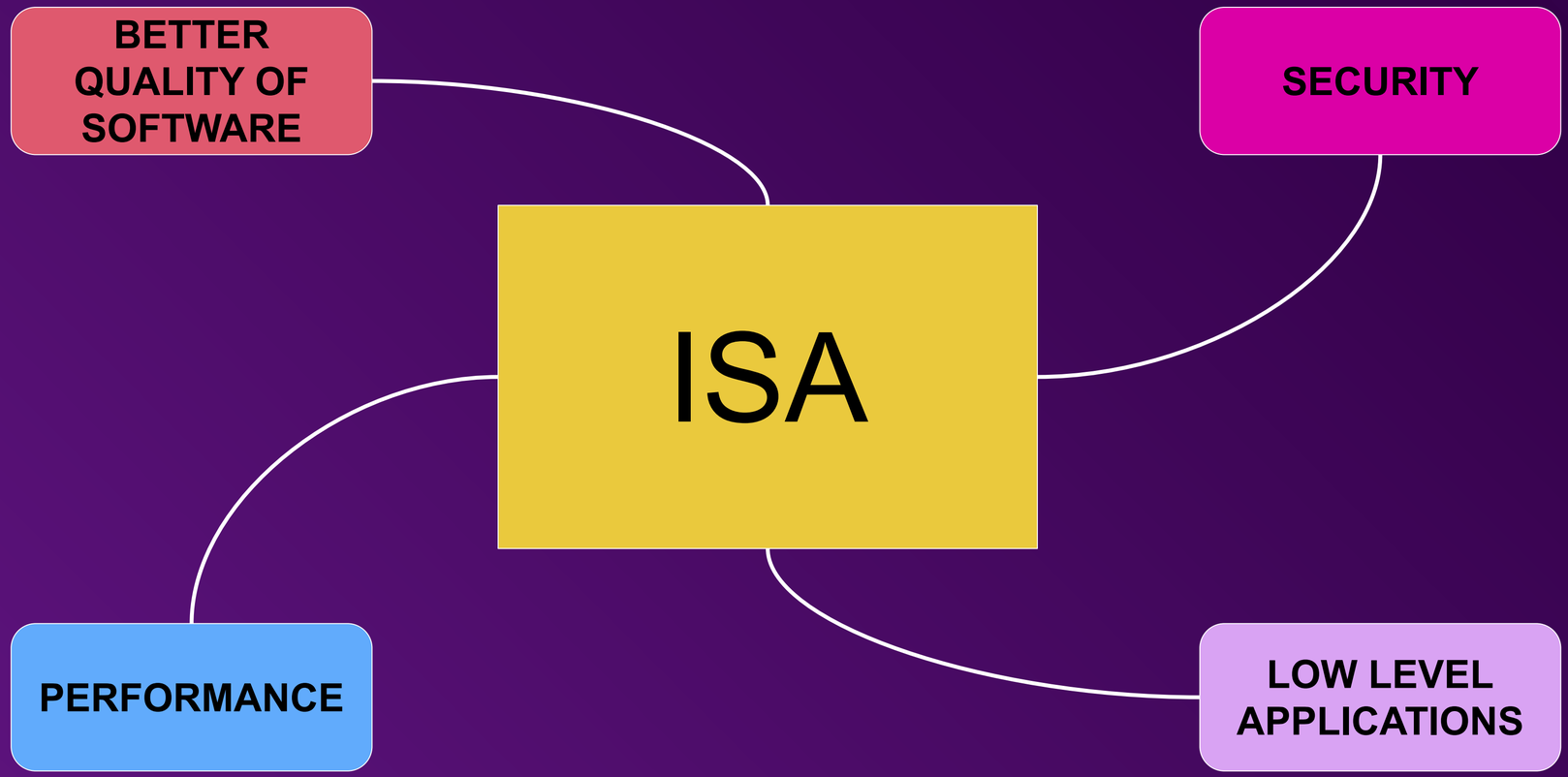
- Interface between **HW / SW**
- Defines how the CPU is controlled by the SW
- Includes instructions, IOs, etc
- Functionally independent of the HW
    - Intel's 8086 programs work on any subsequent family





Why is it important to master the ISA?

Instruction Set Architecture level



3 main types

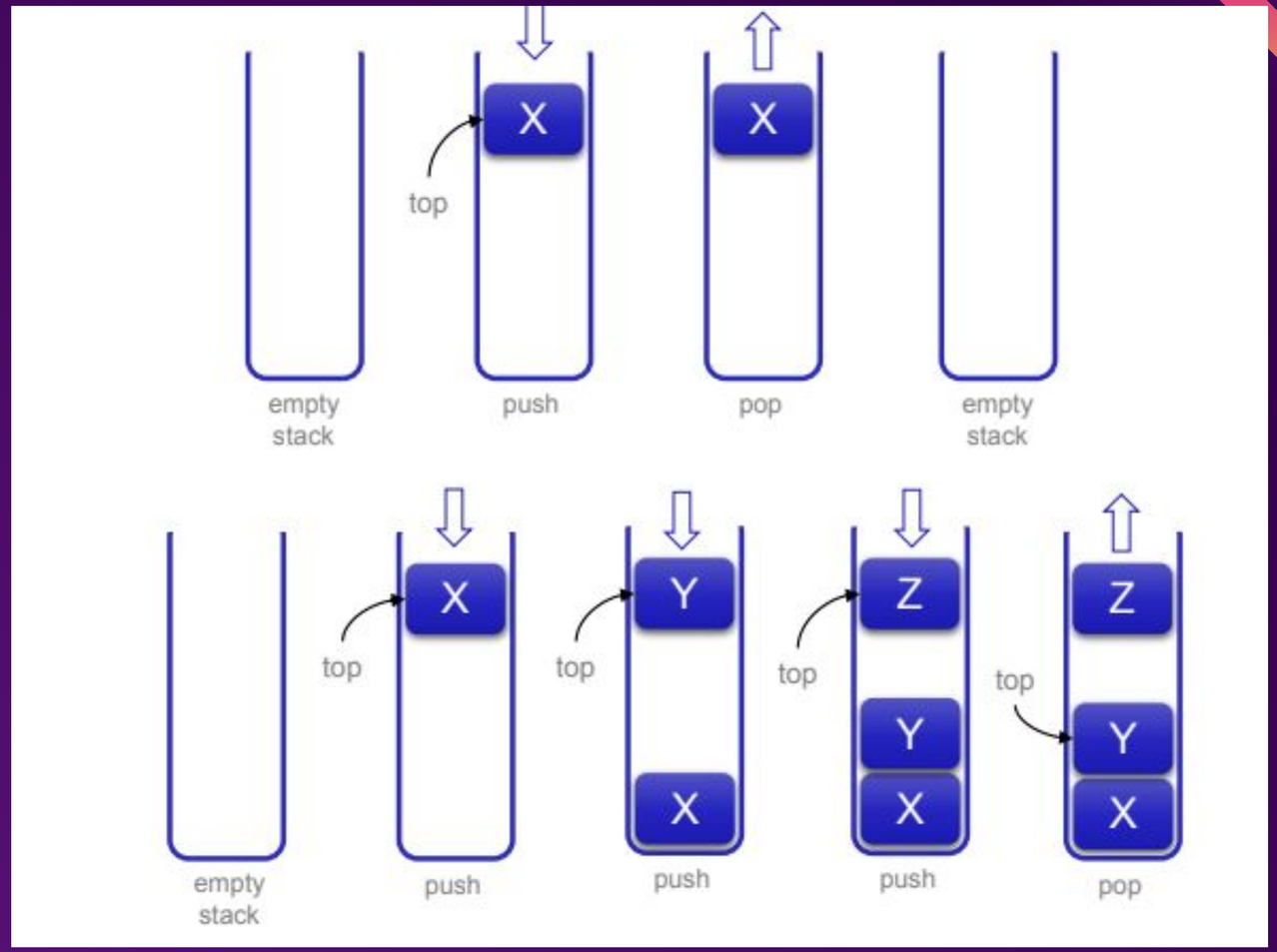
- Stack
- Accumulator
- General Purpose Registers (GPRs)

3 main types

- Stack
- Accumulator
- General Purpose Registers (GPRs)

3 main types

- Stack
- Two base operands take one parameter:
  - push and pop

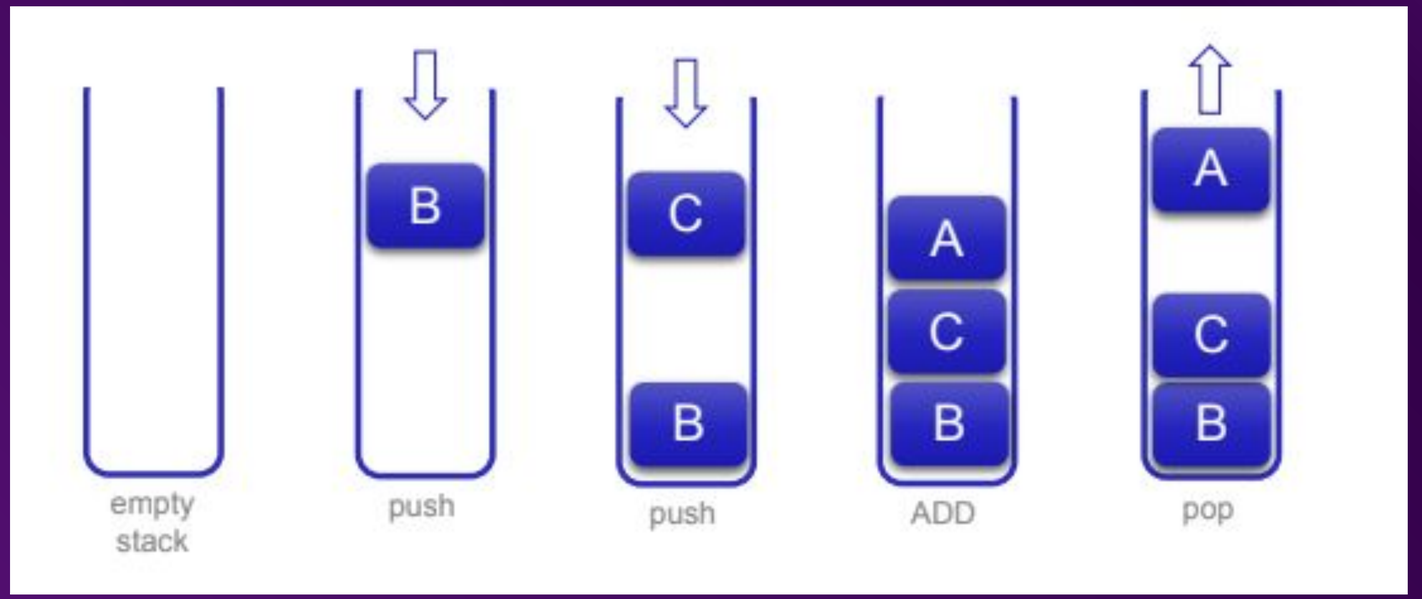


3 main types

- Stack
- Example
  - A = B+C

Simple model evaluation, short instructions

Efficient code generation jeopardized, stack becomes architecture bottleneck

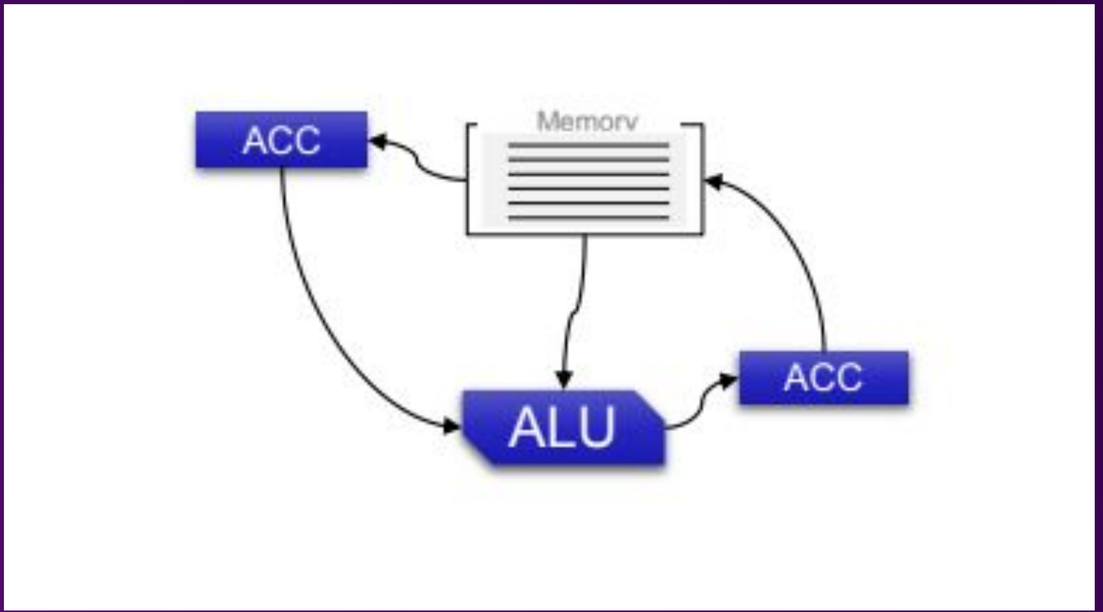


3 main types

- Stack
- Accumulator
- General Purpose Register (GPR)

3 main types

- Accumulator
  - One operand is the accumulator register, the other is in memory
  - Direct access
  - ALU writes back to accumulator



3 main types

- Accumulator
- Example
  - A = B+C

Short instructions

Central point at accumulator. Memory traffic is increased

1. LOAD address B from memory to ACC
2. Fetch value from address C in memory
3. ADD the two values
4. Save result back in ACC
5. Store ACC at address A in memory

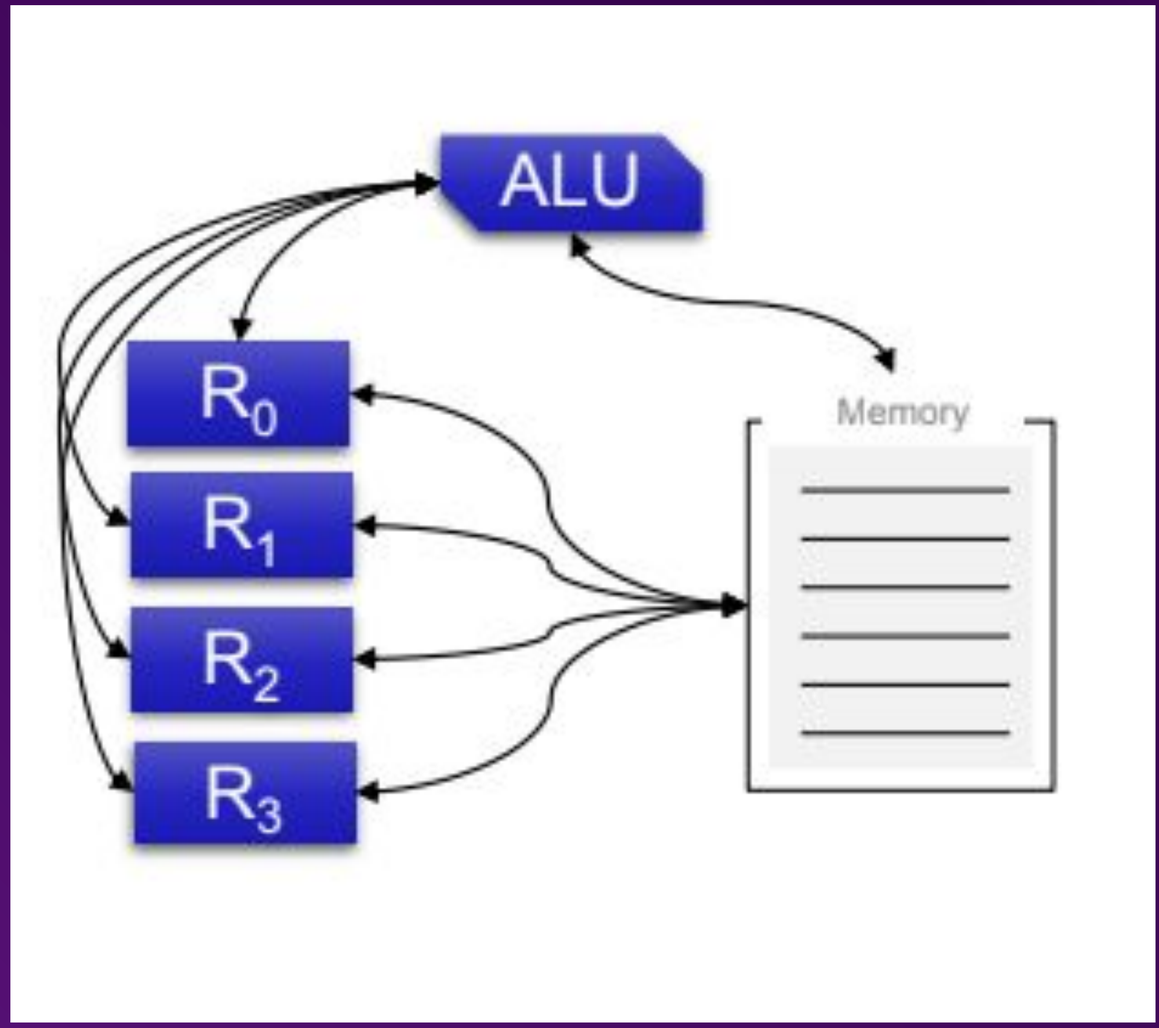
3 main types

- Stack
- Accumulator
- General Purpose Registers (GPRs)

3 main types

- GPRs

  General Purpose Registers



3 main types

- GPRs
- Example
  - A = B+C

1. LOAD R0, B
2. LOAD R1, C
3. ADD R2, R0, R1
4. STORE A, R2

Easy code generation, local storage

All operands are named what will generat longer instructions

Activity 2

- Find examples of processors of each type of ISA seem so far. What are their advantages? Disadvantages? What are the two ISA classification types? Explain each, with examples.

- Groups of 2 students

Computer Level Hierarchy

- Abstraction layers
- Divide and conquer approach
- Virtual representations
- Possibility to work entirely on each layer

Level 6	User	Executable Programs
Level 5	High Level Language	C++ , Java
Level 4	Assembly Language	Assembly Code
Level 3	System Software	Operating System
Level 2	Machine	Instruction Set Architecture
Level 1	Control	Microcode
Level 0	Digital Logic	Circuits , Gates

System Software Level

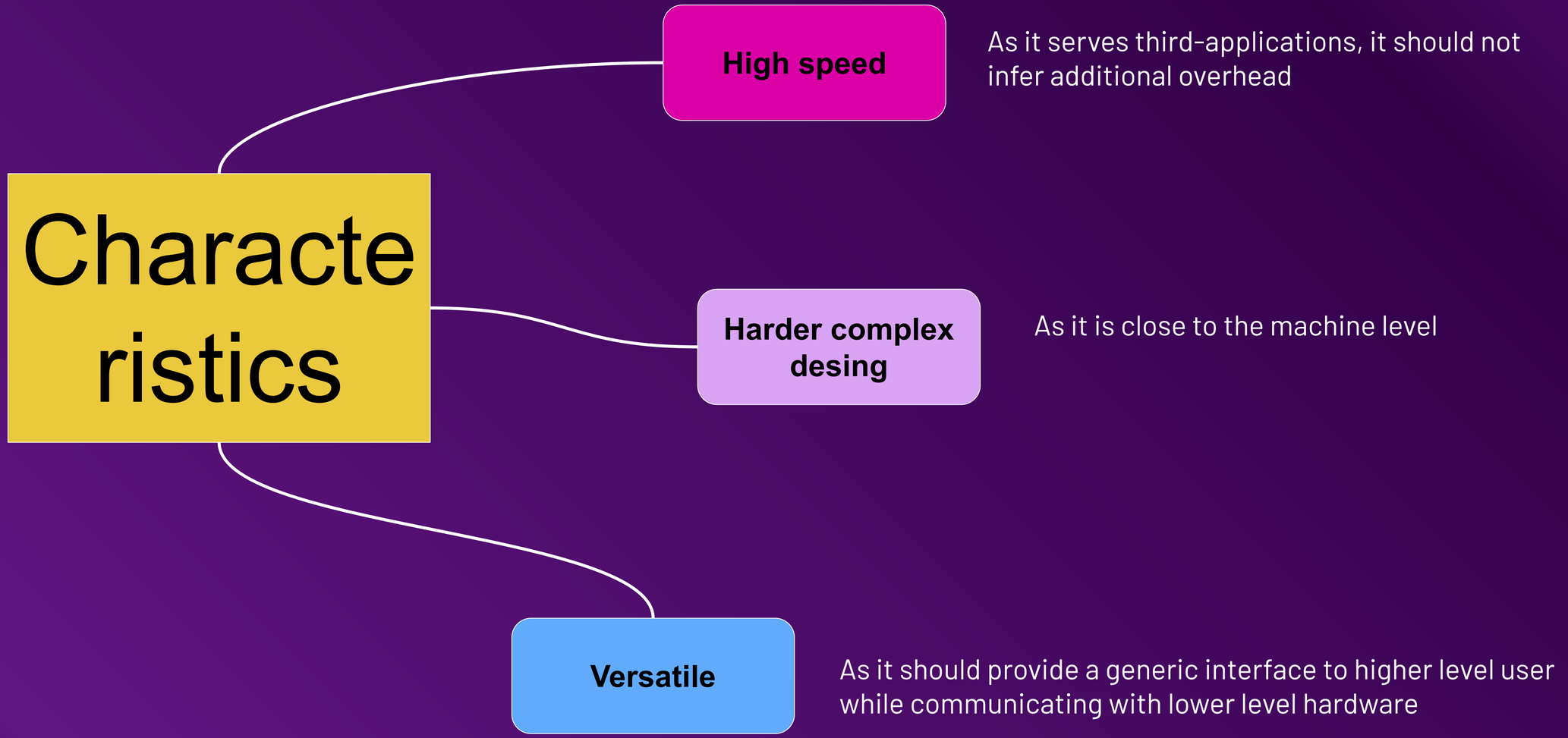
- Deals with the **abstraction and protection** of lower levels (machine level)
  - Multiprogramming
  - Memory handling
  - I/O
- Device drivers are usually written at this level
- Controls executing processes in the system
- Assembly language (lvl 4) instructions usually pass through this level unmodified

System Software Level

- Provides services/platform to another software
  - Different applications
  - Operating systems
  - Software Engines
  - Automation

- System software is usually general-purpose

System Software Level



Activity 3

Find examples of operating systems, old and new, and discuss about (possible) changes they have in their design. The services remain the same? What have changed?

- Groups of 2 students

Computer Level Hierarchy

- Abstraction layers
- Divide and conquer approach
- Virtual representations
- Possibility to work entirely on each layer

Level 6	User	Executable Programs
Level 5	High Level Language	C++ , Java
Level 4	Assembly Language	Assembly Code
Level 3	System Software	Operating System
Level 2	Machine	Instruction Set Architecture
Level 1	Control	Microcode
Level 0	Digital Logic	Circuits , Gates

Assembly Code Level

- Acts on assembly generated from **higher levels (lvl 5)** and in assembly written at this level directly
- **One** assembly instruction is translated to **one** machine instruction
- Assembly language is textual
  - programs are sequence of readable commands
- Many compilers are written in assembly

Assembly Code Level

Instruction	Class	Operation
AND	Computation	Logical product
OR	Computation	Logical sum
XOR	Computation	Logical diff
SLT	Computation	Set if less than
ADD	Computation	Add
SUB	Computation	Subtract
LDW / LDB	Load/Store	Load word/byte
STW / STB	Load/Store	Store word/byte
BEZ	Branch	Branch if zero
BNZ	Branch	Branch if not zero
LDI	Pseudo	Load immediate
HCF	Pseudo	Halt and catch fire

```
main

    ldi r2,ask
    ldi lr,retask
    bnz sp,print
retask
    ldi r2,name
    ldi lr,retget
    bnz sp,getname
retget
    ldi r2,str

    ldi     r2,str
    ldi lr,retstr
    bnz sp,print
retstr
    ldi     r2,name
    ldi lr,retstr2
    bnz sp,print
retstr2
    hcf
```

```
print
    ldb r3,r2
    stw r3,0xf000
    add r2,1
    bnz r3,print
    bnz sp,lr

getname
    ldw r3,0xf004
    stb r3,r2
    add r2,1
    bnz r3,getname
    bnz sp,lr

str "\nhello "
ask "enter your name: "
name    "                "
```

Viking ISA

https://github.com/sjohann81/viking

Computer Level Hierarchy

- Abstraction layers
- Divide and conquer approach
- Virtual representations
- Possibility to work entirely on each layer

Level 6	User	Executable Programs
Level 5	High Level Language	C++ , Java
Level 4	Assembly Language	Assembly Code
Level 3	System Software	Operating System
Level 2	Machine	Instruction Set Architecture
Level 1	Control	Microcode
Level 0	Digital Logic	Circuits , Gates

Higher Level Language

- Common use programming languages
  - C/C++
  - Java
  - Pascal
  - Python
- Provides a “more convenient” interface for applications
- In general, programming languages at this level are structured english

Higher Level Language

- The same high-level code can be reused for different targets
- Programmer focus on higher-level aspects of design
  - Data types
  - Structures
  - Algorithm
- Code can be translated to assembly code or interpreted
  - C, Ada, C#, ... translated
  - Sh, TCL, Perl, Python, ... interpreted

Computer Level Hierarchy

- Abstraction layers
- Divide and conquer approach
- Virtual representations
- Possibility to work entirely on each layer

Level 6	User	Executable Programs
Level 5	High Level Language	C++ , Java
Level 4	Assembly Language	Assembly Code
Level 3	System Software	Operating System
Level 2	Machine	Instruction Set Architecture
Level 1	Control	Microcode
Level 0	Digital Logic	Circuits , Gates

Higher Level Language

- Programs (executables) and users
- Most daily applications reside in this level
  - Word processors
  - Games
  - Browsers
- Lower levels are nearly invisible from this level

Computer Level Hierarchy

Level 6	User	Executable Programs
Level 5	High Level Language	C++ , Java
Level 4	Assembly Language	Assembly Code
Level 3	System Software	Operating System
Level 2	Machine	Instruction Set Architecture
Level 1	Control	Microcode
Level 0	Digital Logic	Circuits , Gates

Activity 4

Discuss and present the difference between application (user) software and system software.

- Groups of 2 students

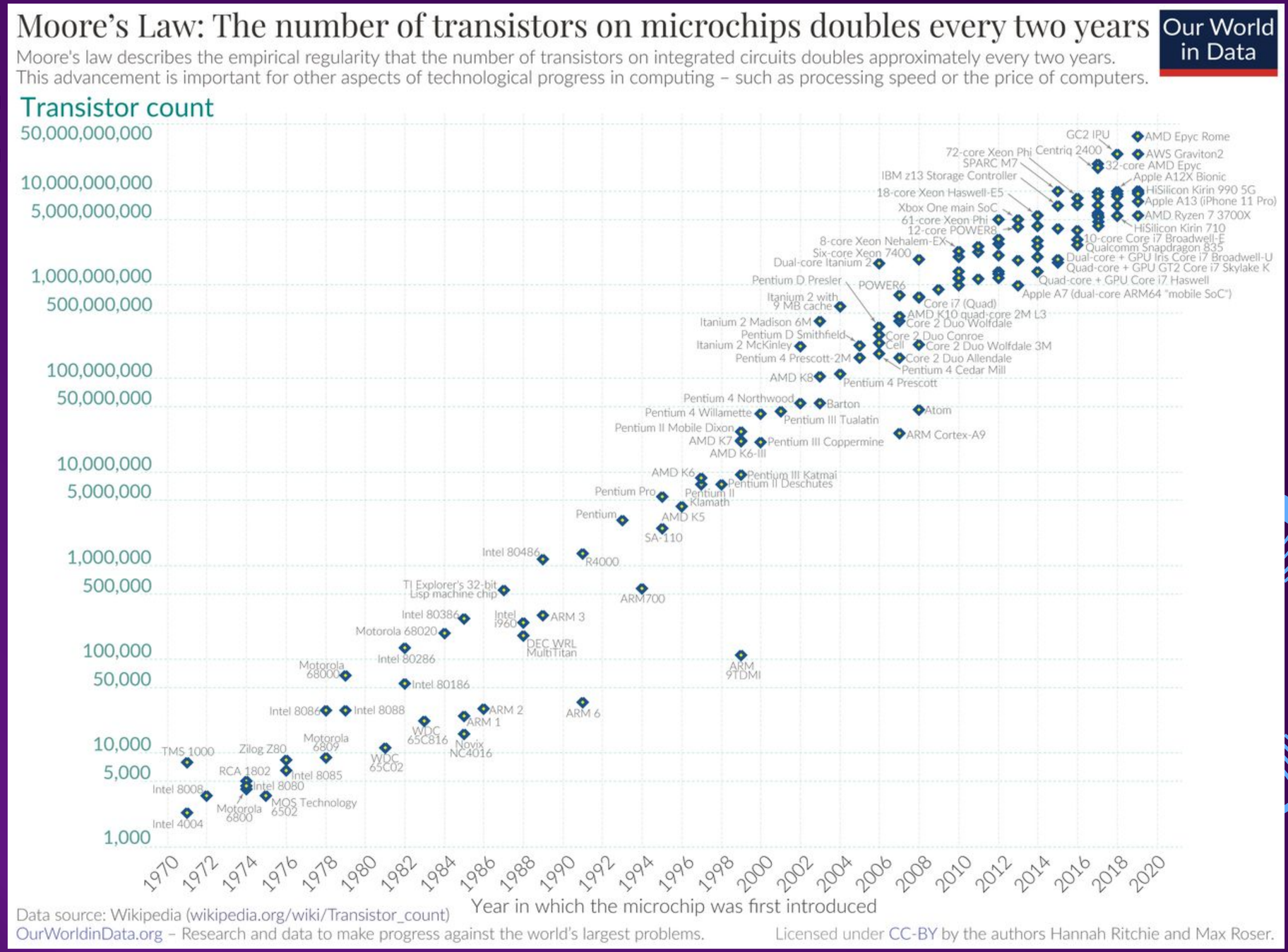
Moore’s Law

QUT

MOORE'S LAW

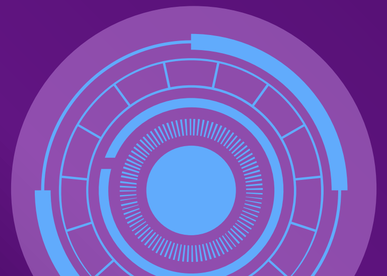
with

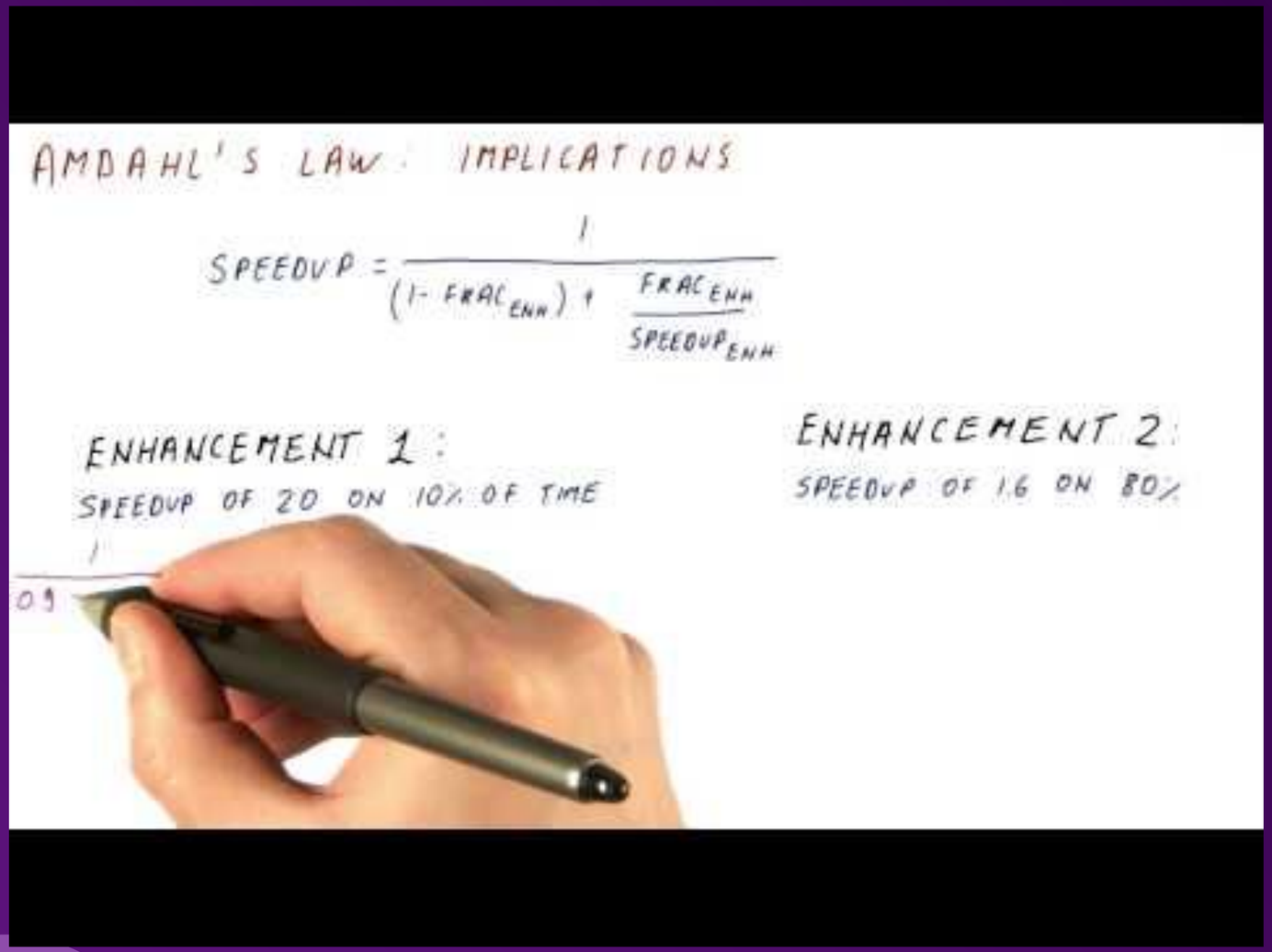
A/Prof. Alexander Dreiling

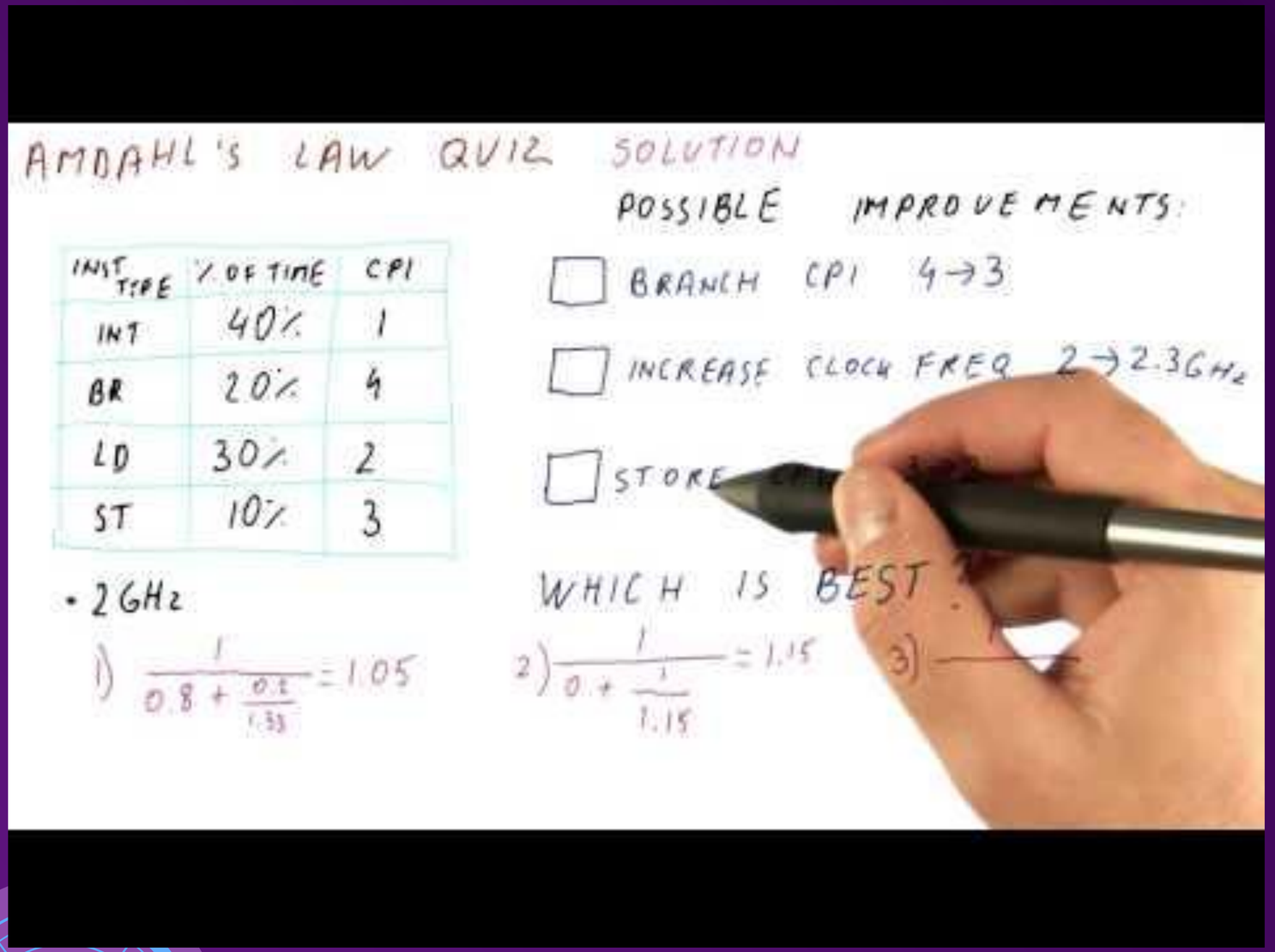

Moore's Law: The number of transistors on microchips doubles every two years
Our World in Data
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.
Transistor count
50,000,000,000
10,000,000,000
5,000,000,000
1,000,000,000
500,000,000
100,000,000
50,000,000
10,000,000
5,000,000
1,000,000
500,000
100,000
50,000
10,000
5,000
1,000
GC2 IPU
AMD Epyc Rome
72-core Xeon Phi
Centriq 2400
AWS Graviton2
SPARC M7
32-core AMD Epyc
IBM z13 Storage Controller
Apple A12X Bionic
18-core Xeon Haswell-E5
HiSilicon Kirin 990 5G
Xbox One main SoC
Apple A13 (iPhone 11 Pro)
61-core Xeon Phi
AMD Ryzen 7 3700X
12-core POWER8
HiSilicon Kirin 710
8-core Xeon Nehalem-EX
10-core Core i7 Broadwell-E
Six-core Xeon 7400
Qualcomm Snapdragon 835
Dual-core Itanium 2
Dual-core + GPU Iris Core i7 Broadwell-U
Quad-core + GPU GT2 Core i7 Skylake K
Pentium D Presler
POWER6
Quad-core + GPU Core i7 Haswell
Itanium 2 with 9 MB cache
Core i7 (Quad)
Apple A7 (dual-core ARM64 "mobile SoC")
AMD K10 quad-core 2M L3
Itanium 2 Madison 6M
Core 2 Duo Wolfdale
Pentium D Smithfield
Core 2 Duo Conroe
Itanium 2 McKinley
Cell
Core 2 Duo Wolfdale 3M
Pentium 4 Prescott-2M
Core 2 Duo Allendale
Pentium 4 Cedar Mill
AMD K8
Pentium 4 Prescott
Pentium 4 Northwood
Barton
Pentium 4 Willamette
Pentium III Tualatin
Atom
Pentium II Mobile Dixon
ARM Cortex-A9
AMD K7
Pentium III Coppermine
AMD K6-III
AMD K6
Pentium III Katmai
Pentium II Deschutes
Pentium Pro
Pentium II
Klamath
Pentium
AMD K5
SA-110
Intel 80486
R4000
TI Explorer's 32-bit Lisp machine chip
ARM700
Intel 80386
Intel i960
ARM 3
Motorola 68020
DEC WRL MultiTitan
Intel 80286
ARM 9TDMI
Motorola 68000
Intel 80186
Intel 8086
Intel 8088
ARM 2
ARM 6
ARM 1
WDC 65C816
Motorola 6809
Novix NC4016
TMS 1000
Zilog Z80
WDC 65C02
RCA 1802
Intel 8085
Intel 8008
Intel 8080
MOS Technology 6502
Motorola 6800
Intel 4004
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
Year in which the microchip was first introduced
Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
OurWorldinData.org – Research and data to make progress against the world's largest problems.
Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.


UPSCALED
IBM'S
2NM CHIPS

Amdahl's Law




AMDAHL'S LAW: IMPLICATIONS
SPEEDUP = 1 / (1 - FRAC_ENH) + FRAC_ENH / SPEEDUP_ENH
ENHANCEMENT 1:
SPEEDUP OF 20 ON 10% OF TIME
ENHANCEMENT 2:
SPEEDUP OF 1.6 ON 80%


AMDAHL'S LAW QUIZ SOLUTION
POSSIBLE IMPROVEMENTS:
BRANCH CPI 4→3
INCREASE CLOCK FREQ 2→2.3GHz
STORE
WHICH IS BEST?
INST TYPE | % OF TIME | CPI
INT | 40% | 1
BR | 20% | 4
LD | 30% | 2
ST | 10% | 3
• 2GHz
1) 1 / (0.8 + 0.2/1.33) = 1.05
2) 1 / (0.4 + 1/1.15) = 1.15
3)

Amdahl's Law

- Gives the possible **gain** from speeding up a given resource
- Speed-up is limited by the needed time for the unchanged portion of the program

$$SpeedUp = \frac{1}{(1 - p) + \frac{p}{s}}$$

p → improvement factor
s → acceleration

$$S = \frac{T_{sequential}}{T_{parallel}}$$

Amdahl's Law - Example 1

- Assuming we are improving the hardware of a web server. The new CPU is 10x faster than the current one. The current CPU is busy with computation 40% of the time, while for the other 60% it is waiting for IO operations. What is the gain obtained by replacing this CPU by the new one?

p= 40% => 0.4

s = 10

$$Gain = \frac{1}{(1-p) + \frac{p}{s}}$$

G = (1 / (1-0.4)+0.4/10)
G = (1 / (0.6+0.04))
G = 1.56

Amdahl's Law - Example 2

A common operation in graphic processing is the square root. Floating point implementations of such operation vary a lot. Assuming that the floating point instruction are used 50% of the time, while FPSQR (floating point square root) is responsible for 20% of the execution time, what would yield a better gain: improving the FPSQR hardware speeding-up its execution by a factor of 10 or improve ALL FP instructions by a factor of 1.6?

p= 50% => 0.5

s = 1.6

$$Gain = \frac{1}{(1-p) + \frac{p}{s}}$$

G = 1.23

Amdahl's Law - Example 2

A common operation in graphic processing is the square root. Floating point implementations of such operation vary a lot. Assuming that the floating point instruction are used 50% of the time, while FPSQR (floating point square root) is responsible for 20% of the execution time, what would yield a better gain: improving the FPSQR hardware speeding-up its execution by a factor of 10 or improve ALL FP instructions by a factor of 1.6?

p= 20% => 0.2

s = 10

$$Gain = \frac{1}{(1-p) + \frac{p}{s}}$$

G = 1.22

Activity 5

Discuss and present what other types of performance laws are used for computers? Give examples of their usage and why they are Important.

- Groups of 2 students

Marcela Gonçalves dos Santos
marcela.goncalvesdossantos@concordia.ca

Computer Architecture

Laws and Arithmetic Binary