V.1. Computer Architecture: Possibile Visions

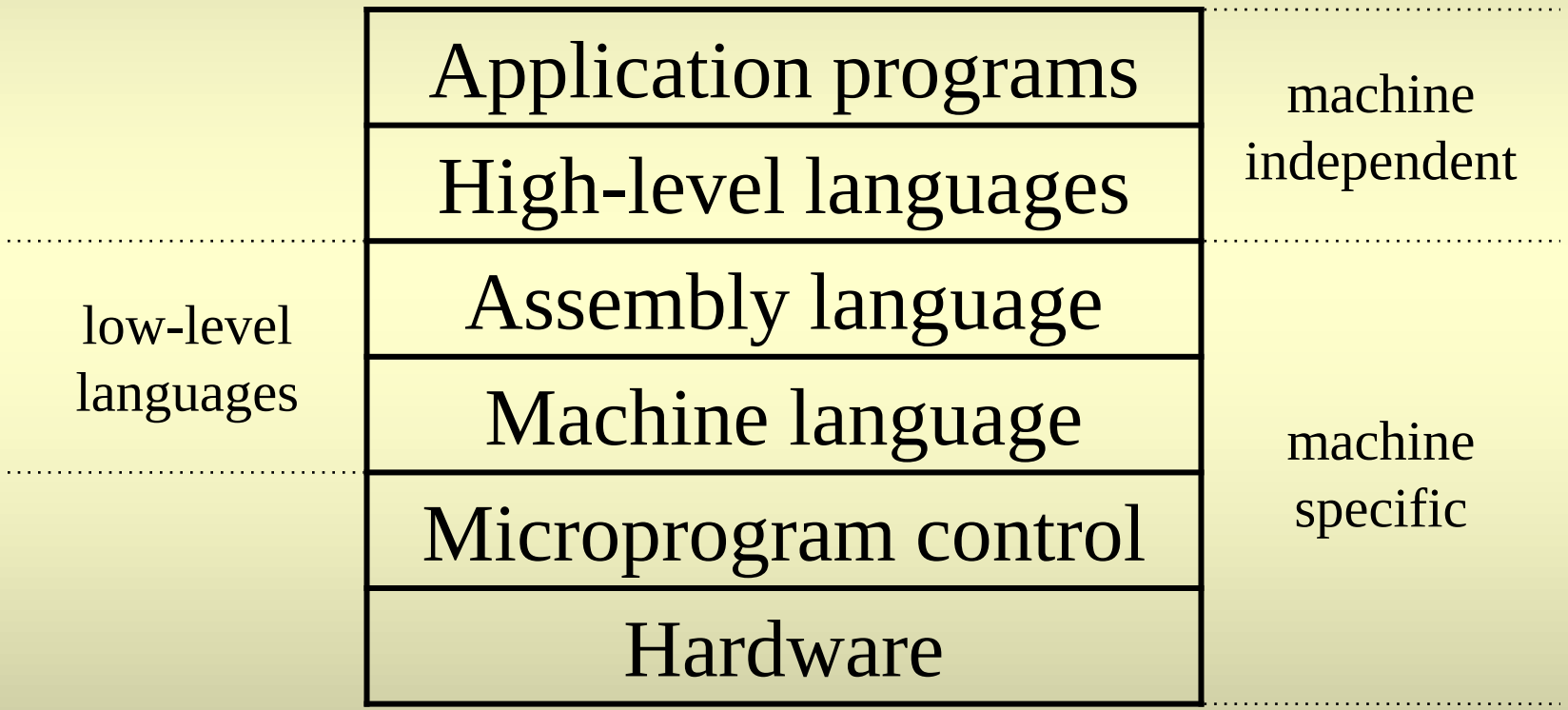
Programmer vs. User

- computer architecture - made of
  - instruction set architecture (ISA)
  - machine organization
- user's vision
  - hardware system
    on which runs
  - operating system
    on which run
  - software applications

Programmer's Vision (1)

- represented by the instruction set architecture
- materialized by various languages
- in increasing order of the abstraction level
  - machine language
  - assembly language
  - high level languages

Programmer's Vision (2)



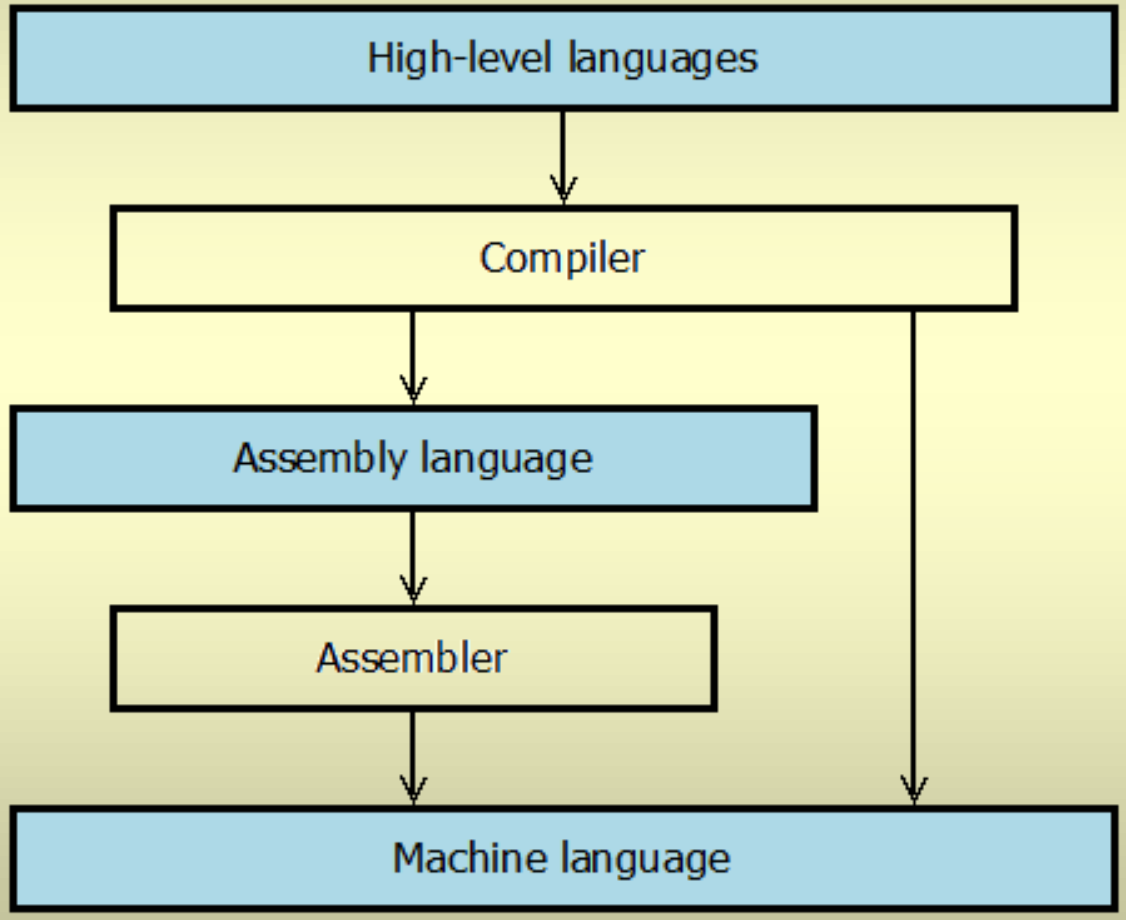
Low-level languages

- machine language
  - each processor has its own language
  - made of words over the binary alphabet {0, 1}
- assembly language
  - slightly higher level
  - instructions correspond to machine language
- examples
  - machine language: **0100 1010 1101 0010**
  - assembly language: **sub myvar,5**

Language Translation (1)

- the only language understood by the processor is machine language
  - to be executed, a program must be translated
- assembler
  - assembly language to machine language
- compiler
  - high-level language to machine language
  - directly or via assembly language

Language Translation (2)



High-level Languages - Advantages

- program development is faster
  - less instructions (and more comprehensive)
- program maintenance is easier
  - same reasons as above
- programs are portable
  - few machine-dependent details
    - but not absent - portability is not quite 100%
  - each language requires a dedicated compiler

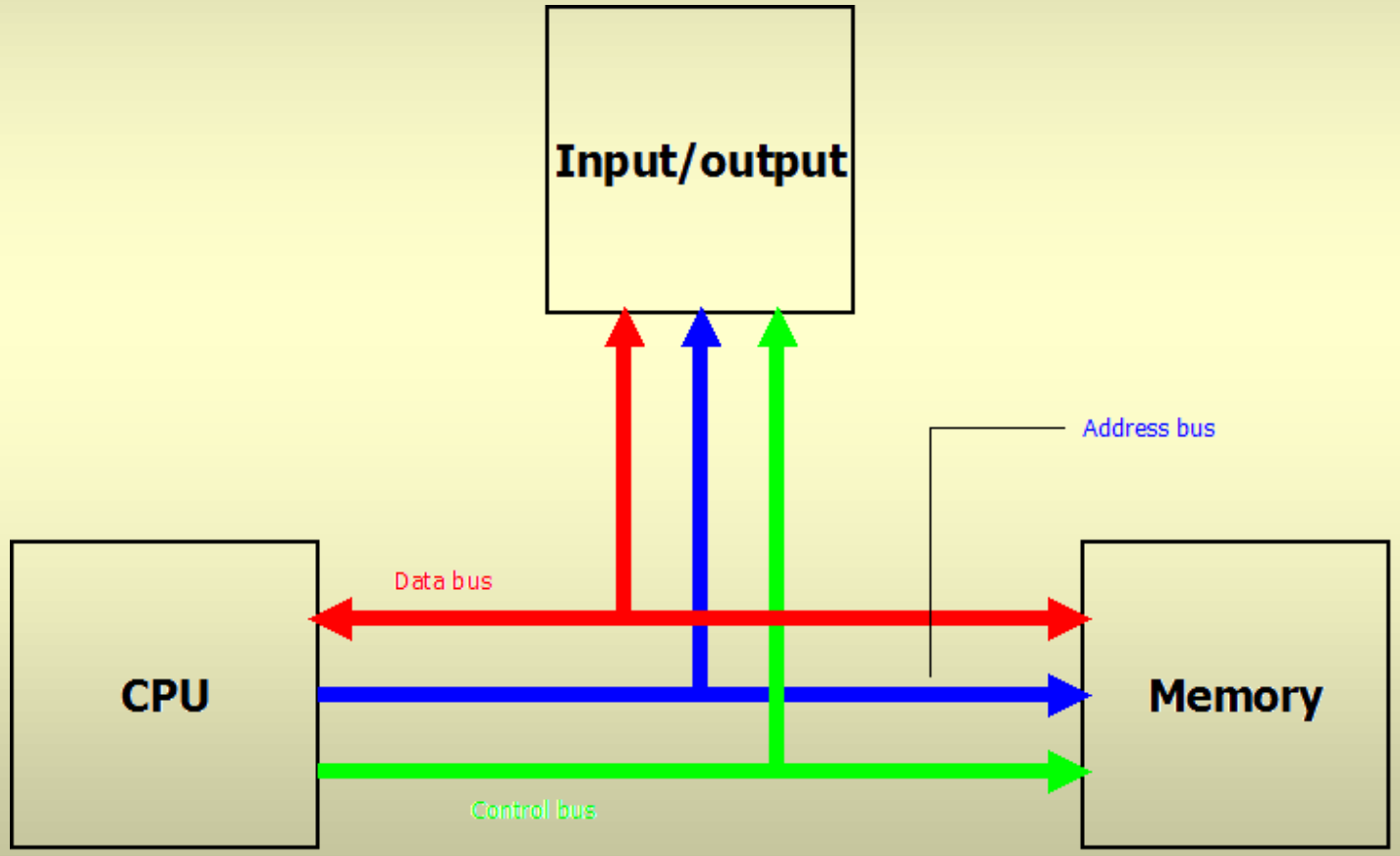
Assembly Language - Advantages

- efficiency
  - compact code - less memory needed
  - speed - faster execution
  - compilers also pursue efficiency
    - but the programmer can get better results
- access to the hardware resources
  - example: processors have bits that indicate carry and overflow after addition
    - not available from high-level languages

Architect vs. Implementer

- architect - high-level design
  - complex components, which are not detailed
  - the main components
    - processor
    - memory
    - input/output (I/O) devices
  - buses through which the components above are connected

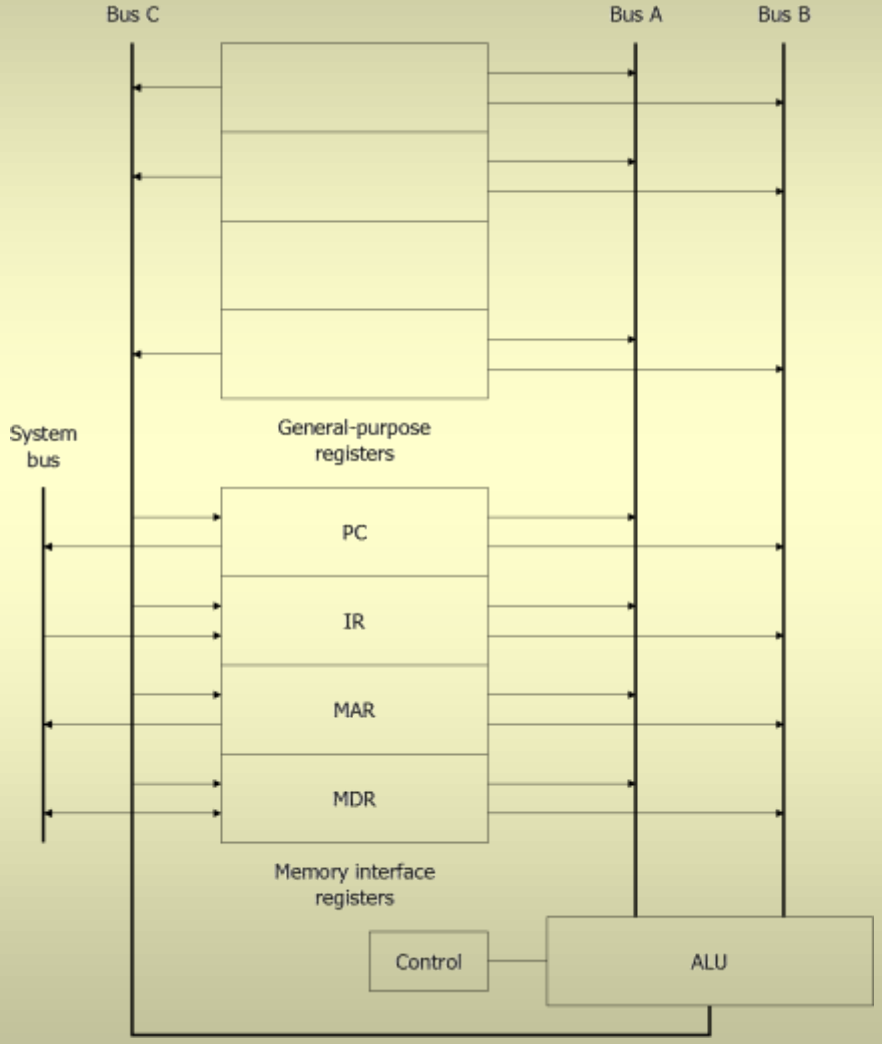
Architect's Vision



Implementer's Vision

- design in detail the high-level components used by the architect
  - down to the level of logic gates
- example: the processor
  - control unit
  - datapath: registers, arithmetic logic unit
- the same instruction set architecture can be implemented in different ways

Datapath


Bus C
Bus A
Bus B
General-purpose registers
System bus
PC
IR
MAR
MDR
Memory interface registers
Control
ALU

V.2. Computer Organization

Main Components

- central processing unit (CPU)
- memory
- peripheral devices (I/O = input/output)
- buses
  - data
  - address
  - control

Central Processing Unit

- also called processor
- executes the instructions indicated by the programmer
- performs data processing
- coordinates the activity of the other components

Memory

- information storage
  - data
  - instructions
- provides the information on request
- plays passive role
  - "responds" to external requests
  - the memory is never the initiator of a transfer

Peripheral Devices

- communication to the "outside world"
- various functions
  - acquire data
  - display data
  - printing
  - storage (persistent)
  - etc.

Buses

- connections between the CPU, memory, and peripheral devices
- three categories
  - data bus - data and instructions
  - address bus - memory and I/O addresses
  - control bus - signals used by the CPU to communicate and control the other circuits

V.3. The Memory

Types of Memory

- ROM (*Read-Only Memory*)
  - its contents can be read, but not modified
  - non-volatile (does not lose its contents upon power down)
- RAM (*Random Access Memory*)
  - its contents may be read and modified
  - volatile (loses its contents upon power down)

ROM - Technologies

- PROM (*Programmable ROM*) - its contents can be programmed by the user
- EPROM (*Erasable PROM*) - can be erased and reprogrammed more than once
  - UVEPROM (*Ultra-Violet EPROM*) - erased by exposure to UV radiation
  - EEPROM (*Electrical EPROM*) - erased through electrical impulses

RAM - Technologies

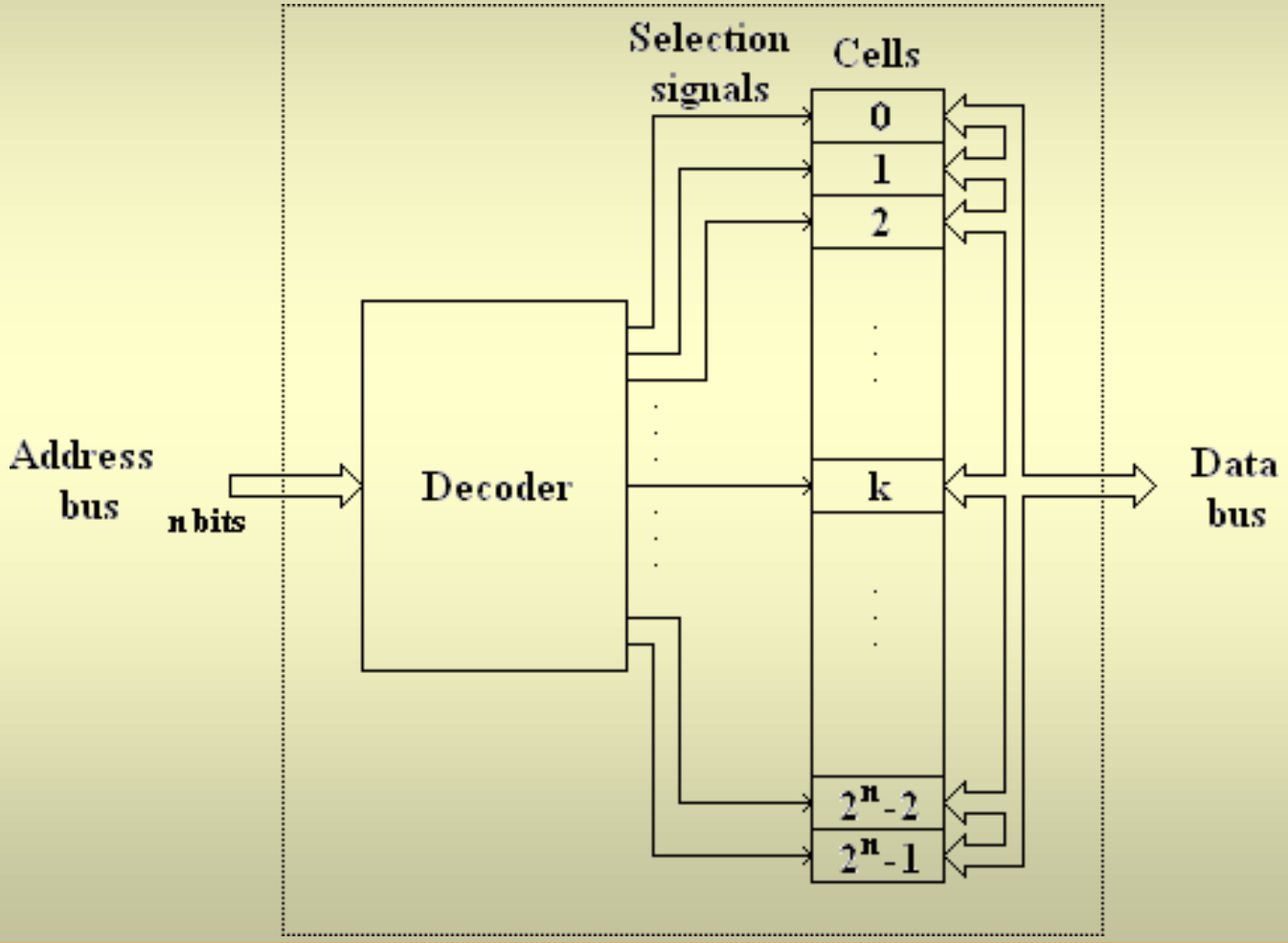
- SRAM (*Static RAM*)
  - high speed
  - high price
- DRAM (*Dynamic RAM*)
  - slower
  - high integration density → smaller
  - lower price

Memory Structure (1)

- single-dimensional cell array (locations)
- each cell is associated a unique number - the *address*
- decoding block - selects the location associated with the address
- the size of the memory circuit - given by the number of address bits

$$size = 2^{no\_address\_bits}$$

Memory Structure (2)



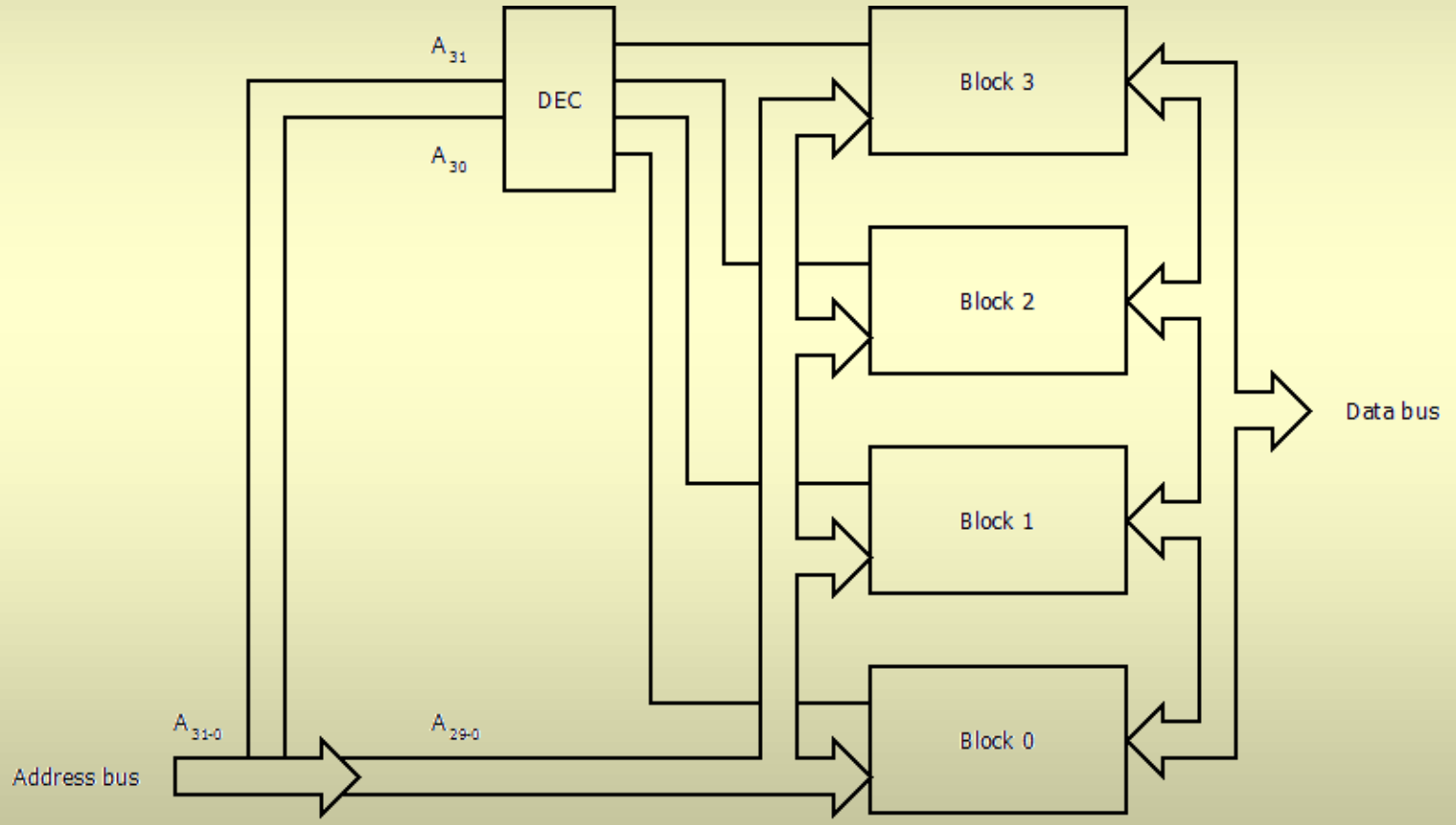
Memory Address Space

- limited by the width of the address bus
  - so it is part of the computer architecture
  - example: 32-bit processors - al most 4 GB (=2^{32})
- not possible to plug in the computer memory beyond the maximum allowed by the address bus
  - but less is possible

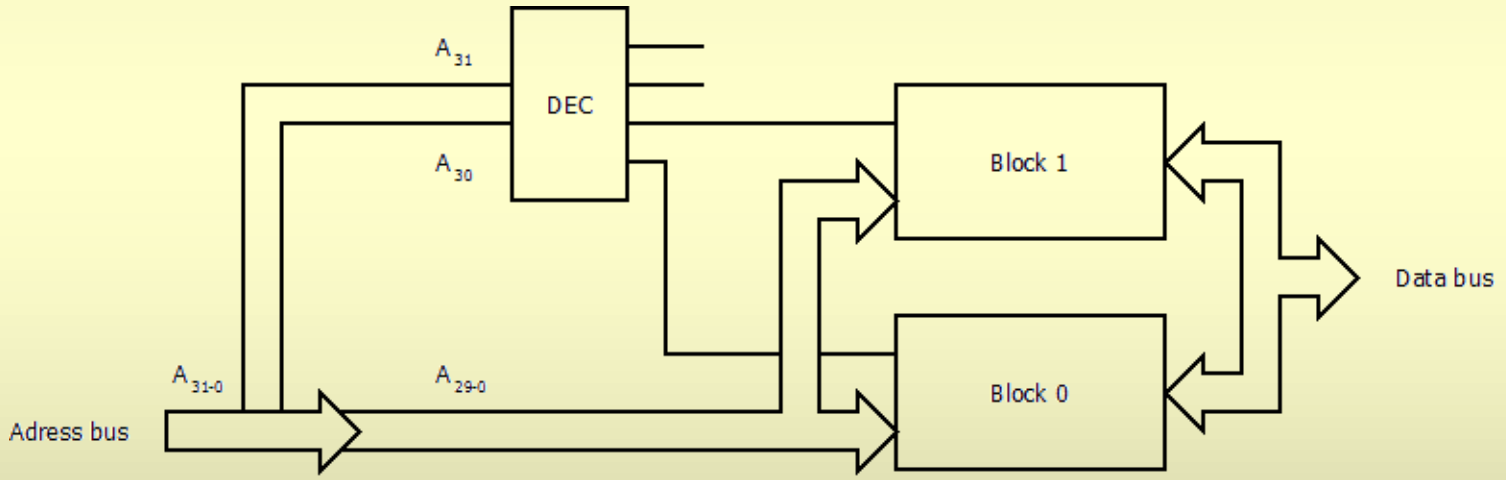
Address Decoding

- example: 32-bit processor
- suppose we have 4 memory circuits (block), 1 GB each
  - a total of 4 GB - the maximum possible
- for a given address, how is the right memory circuit selected?
- what if we have only 2 circuits, 1 GB each?

Full Decoding



Partial Decoding



Types of Addresses

- absolute address
  - the number associated to the location
  - numbering starts at 0
- relative address
  - position with respect to a reference byte
  - example: the index of an element in an array
- symbolic address
  - alpha-numeric identifier attached to an address
  - example: the name of a variable

Byte Order (1)

- a variable may occupy multiple bytes in memory
  - their addresses are consecutive
- example: consider an *unsigned int* variable
  - it occupies 4 bytes
  - suppose their addresses are from 150 to 153
  - which byte is stored at the lowest address? which is stored at the highest address?

Byte Order (2)

- the decision is made when the processing unit (CPU) is designed
  - the CPU reads and writes the memory locations
- variants
  - *big endian* - the most significant byte is stored at the highest address
  - *little endian* - the least significant byte is stored at the highest address

Example

unsigned int x = 0xB67A49E3; // $B67A49E3_{(16)}$

big endian

address	value
	. . .
153	$B6_{(16)} = 10110110_{(2)}$
152	$7A_{(16)} = 01111010_{(2)}$
151	$49_{(16)} = 01001001_{(2)}$
150	$E3_{(16)} = 11100011_{(2)}$
	. . .

little endian

address	value
	. . .
153	$E3_{(16)} = 11100011_{(2)}$
152	$49_{(16)} = 01001001_{(2)}$
151	$7A_{(16)} = 01111010_{(2)}$
150	$B6_{(16)} = 10110110_{(2)}$
	. . .

V.3.1. Cache Memory

The Problem

- processor is faster than memory
  - is forced to wait until it gets the data and instructions from memory
- the performance of the processor is not fully exploited
- causes
  - technological development
  - economics

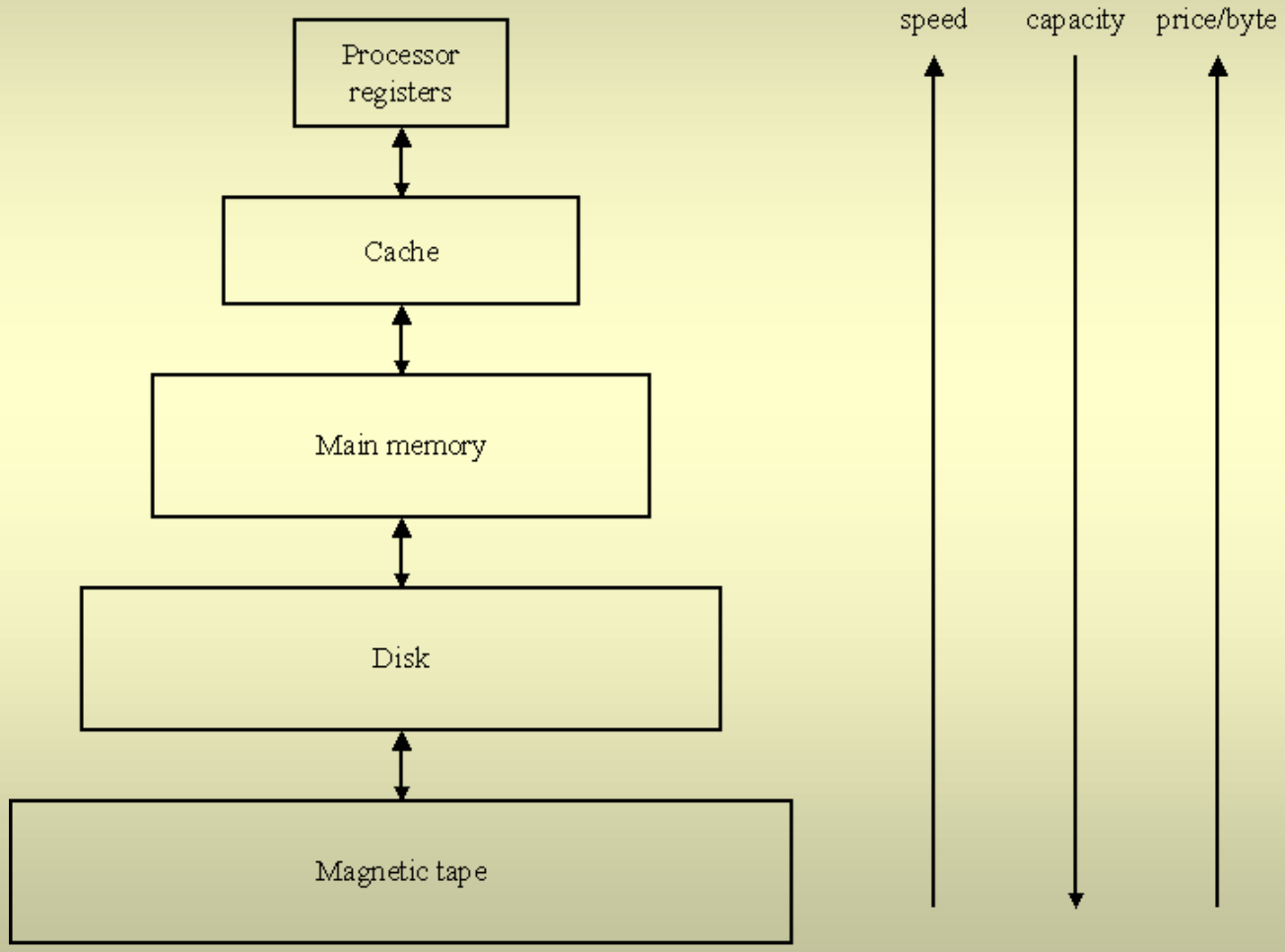
The Solution

- locality laws - empirically determined
- hold true for most applications
- types of locality
  - spatial
  - temporal

The Memory Hierarchy (1)

- different storage levels
  - with different characteristics
- contradictory requirements
  - size - as large as possible
  - access speed - as high as possible
- these requirements cannot be met at the same time by the same kind of storage

The Memory Hierarchy (2)



How Does It Work?

- all information is stored on the lowest level
- on higher levels, subsets of this information are brought
  - the higher the level, the smaller the subsets
- problem - how to maximize overall speed?
- rephrase: which subsets should be brought to the higher levels?
  - in order to get higher speed

Locality Laws - Consequences

- at a certain moment, a program is only a (small) part of its memory locations
  - these parts should be kept on the higher levels
  - which are smaller, but faster
- but which locations are they?
  - most likely, the locations that have been accessed most recently
- special case - the cache memory

Cache Memory

- small and fast circuit
  - between the processor and the main memory
- stores the main memory locations that have been accessed most recently
- the processor tries to access a location
  - first look in the cache
  - if the location is not found in the cache - look in the main memory

Characteristics

Speed

- very high - same level as the processor
  - technology (SRAM)
  - small size → simpler decoding block

Price

- reasonable
  - due to the small size