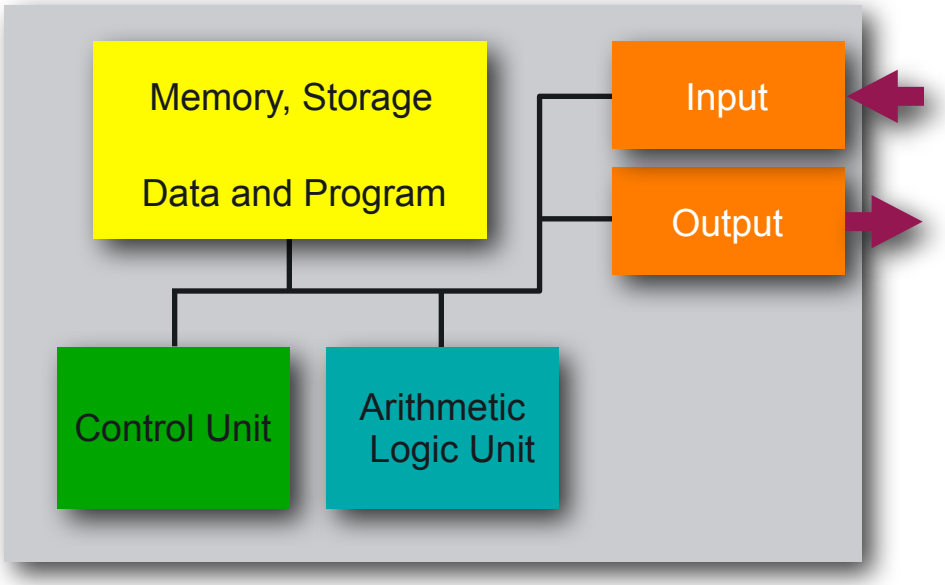
Items of Computer Architecture

- microprocessor architecture
  → is used for the overall representation (i.e. abstracting from details) of the internal structure of microprocessors
- principle of organization and programming model
  → presentable in form of a block configuration
- programming model
  → commonly, a stand-alone characteristic for each microprocessor exists
- architectural features
  → characteristic of particular features determines the classification of a microprocessor
- architectural classes
  → result from historical and technological reasons

Block Diagram of a von Neumann's Computer

main parts:

- central processing unit (CPU) including
  - control unit
  - arithmetic logic unit, ALU
- memory
- input /output unit (I/O unit)
- bus system interconnecting components

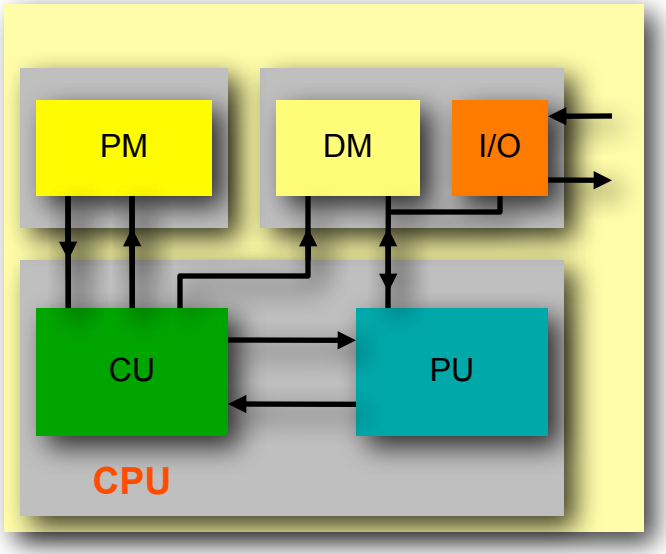


Development of Architectural Variants

- structure of computer is independent from the problem that has to be solved
  → for each new problem a separate program must be loaded
- program and data are located in the memory
  - memory cells have regular word length, separately addressable by absolute addresses
  - only the program context determines, if memory cell content is an element of program or data
- during development process two different memory structures have evolved and thereby 2 architectures:
  - Harvard architecture
  - Princeton architecture

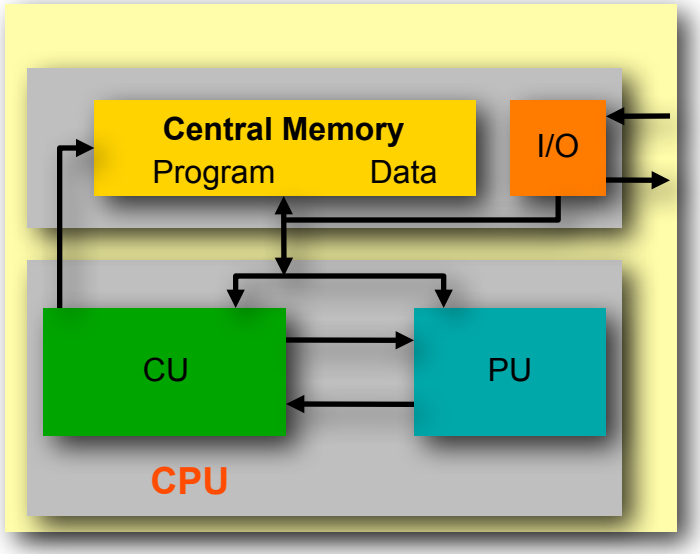
Harvard/ Princeton Architecture

Harvard Architecture



- separate memories for program and data
- parallel access to data and program due to separate busses

Princeton Architecture



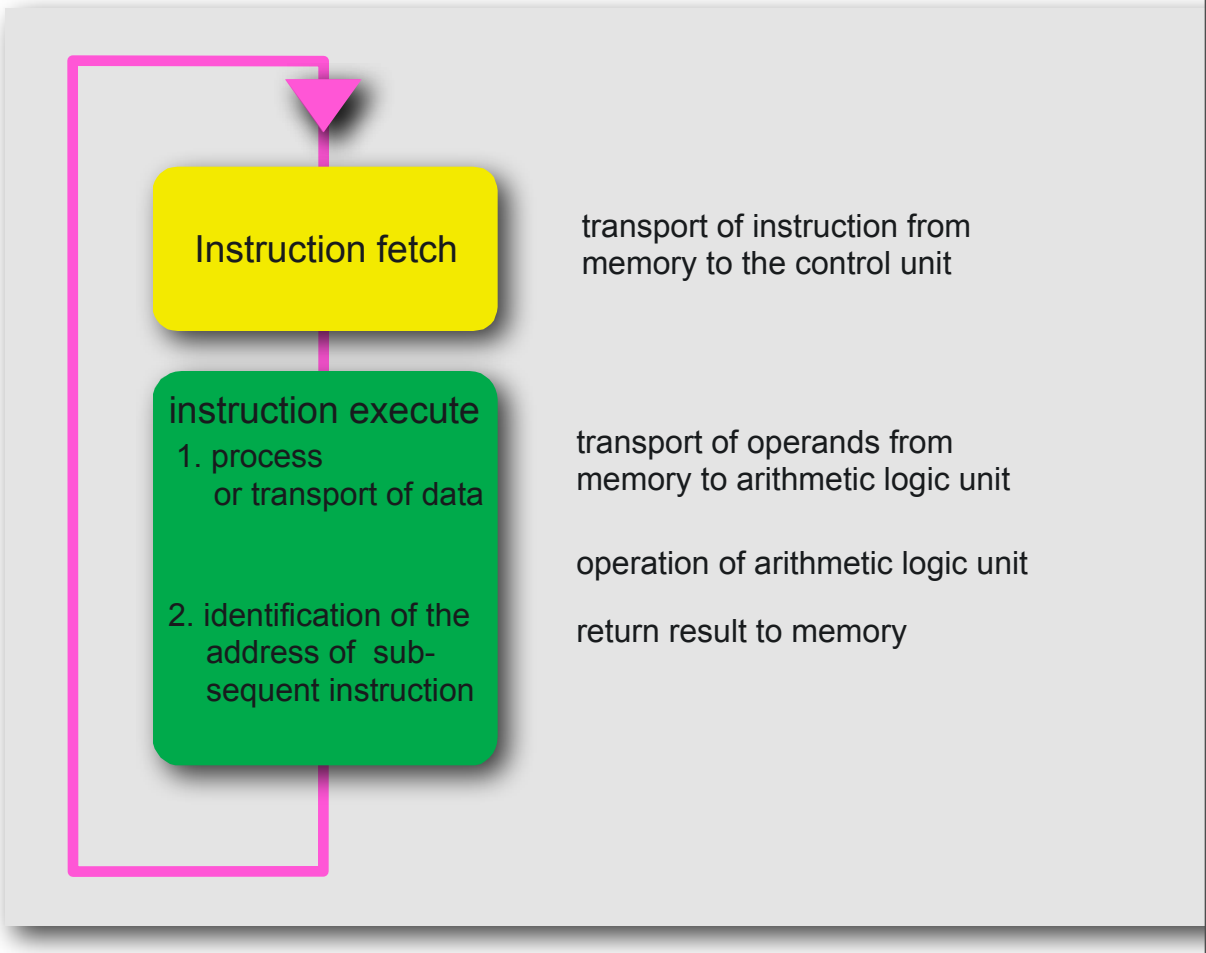
- best approach to the original v. N. principle with common memory („central memory") for data and program
- simplification in structure, but smaller communication performance

Operation Principle of v. N. Computer (1)

- independent from memory structure, equal operation principle
- structure of CPU including control unit and arithmetic logic unit directly corresponds with the sequential processing of instructions in (simplistic) two phases, the operation principle of v. N. computer:
  - phase 1
    - instruction is loaded from memory into control unit → a counter in the CPU („program counter“) is incremented by program flow
  - phase 2:
    - instruction processing
    - required data are loaded into the arithmetic logic unit and processed in compliance with the instruction code (opcode), finally storing
    - subsequently return to phase 1
- sequential operation principle

Operation Principle of v. N. Computer (2)

control unit = finite state machine, which continuously processes instruction cycles, consisting of 2 phases



Operation Principle of v. N. Computer (1)

- von Neumann‘s computer architecture is characterized by:
  - one control unit for management
  - one arithmetic logic unit for executing operations
  - one storage unit for storing programs and data
  - one I/O unit for activating peripheral units
  - one common bus for transfer of data and instructions (Princeton)
- instruction processing cycle is partitioned into non-overlapping sub-phases → restriction of computing power → „von Neumann‘s bottleneck“
- with refinement, evolutionary variation of the von Neumann‘s architecture or radical turning away from this there have been partially obtained considerable progresses in performance and flexibility of computers

Features

- computers are described on the basis of instruction and data streams
- instruction stream: machine instructions are transferred from main memory into the control unit and therefrom conveyed as decoded control information to the arithmetic logic unit
- data stream: data which are transferred as operands for arithmetic and logic information or as their results between main memory and arithmetic logic unit
- classification criterion: simple or multiple occurring of instruction and data streams → detailed information about computer structure is not supplied
- quantitative statements about computer structure, computer performance and topological aspects are not possible
- it is differentiated between four categories of classification of computers only

Categories

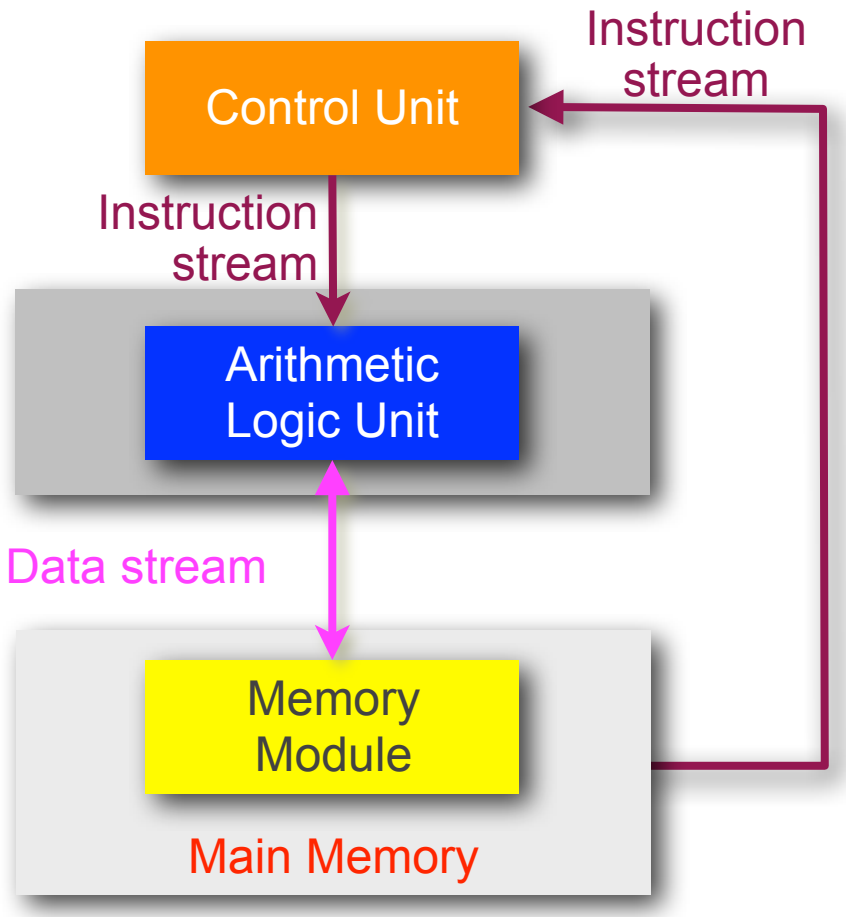
Data stream \ Instruction stream	Single	Multiple
Single	**SISD** **classical v. Neumann's computer, f.e. PCs …**	**MISD** **systolic arrays**
Multiple	**SIMD** **vector computer, array computer**	**MIMD** **multiprocessor-systems, f.e. solid parallel systems**

M.J. Flynn: *Very High-Speed Computing Systems*, Proc. IEEE, Vol.54, 1966, pp.1901-1909

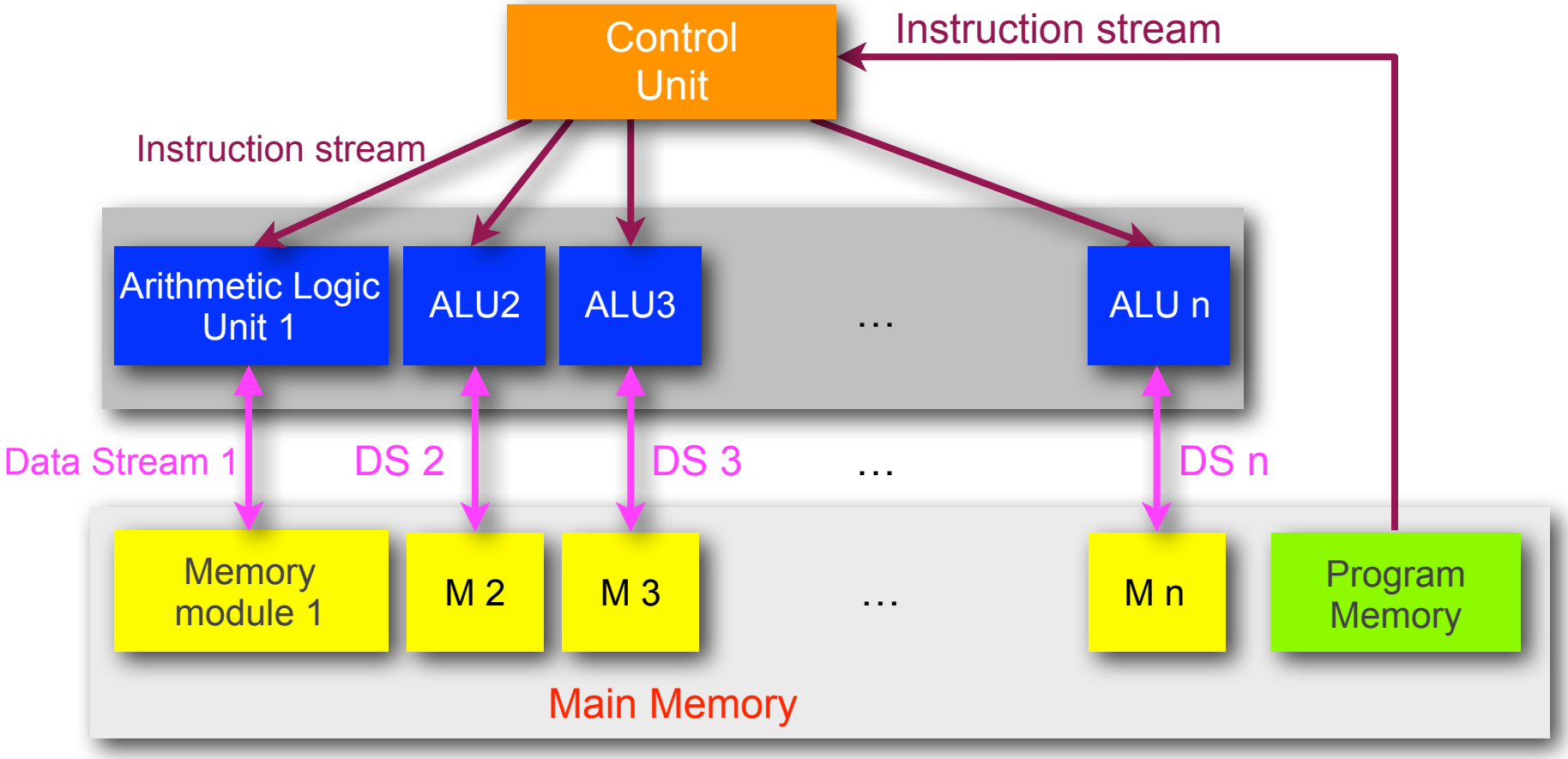
see also list of high performance supercomputers world wide: http://www.top500.org

SISD-Computer

- executing a program, the control unit generates an instruction stream, whose instructions are executed sequentially by the arithmetic logic unit
- required operands and results are fetched from main memory by a bidirectional data stream in the arithmetic logic unit respectively they are written back from arithmetic logic unit into the main memory
- Flynn‘s classification only covers a rough structure and no further differentiation
- unforeseeable, if phases of instruction processing cycle overlap or not



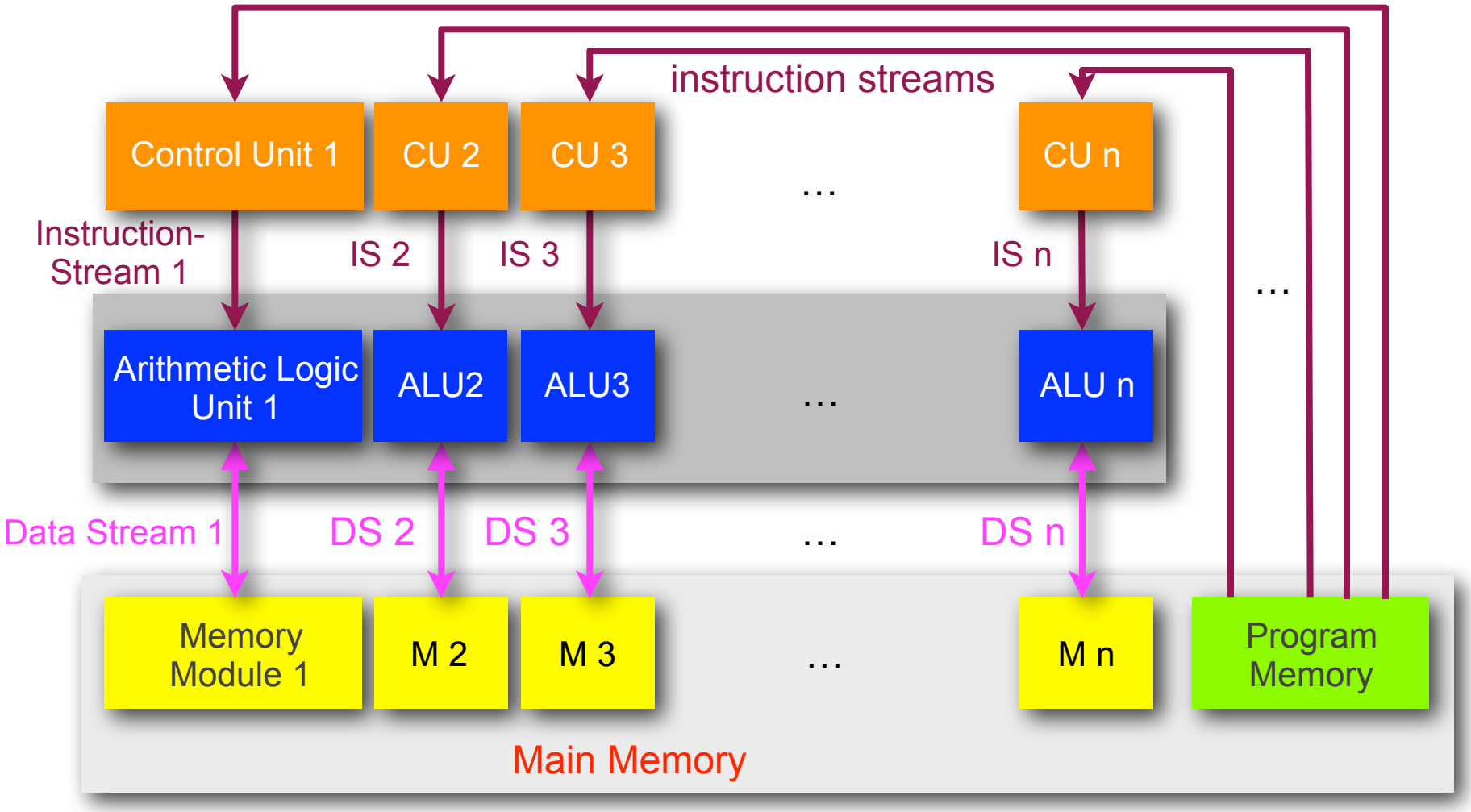
SIMD-Computers (1)



SIMD-Computers (2)

- have only one control unit, too
- instruction decoded from the control unit simultaneously can be applied to multiple operands in multiple arithmetic logic units (processing units, processing elements) - also called *instruction broadcasting*
- each processing unit has a separate bidirectional data access path to the main memory
- main memory is often segmented into independent single units
- SIMD-computers extend the von Neumann‘s structure with multiplication of the number of arithmetic units

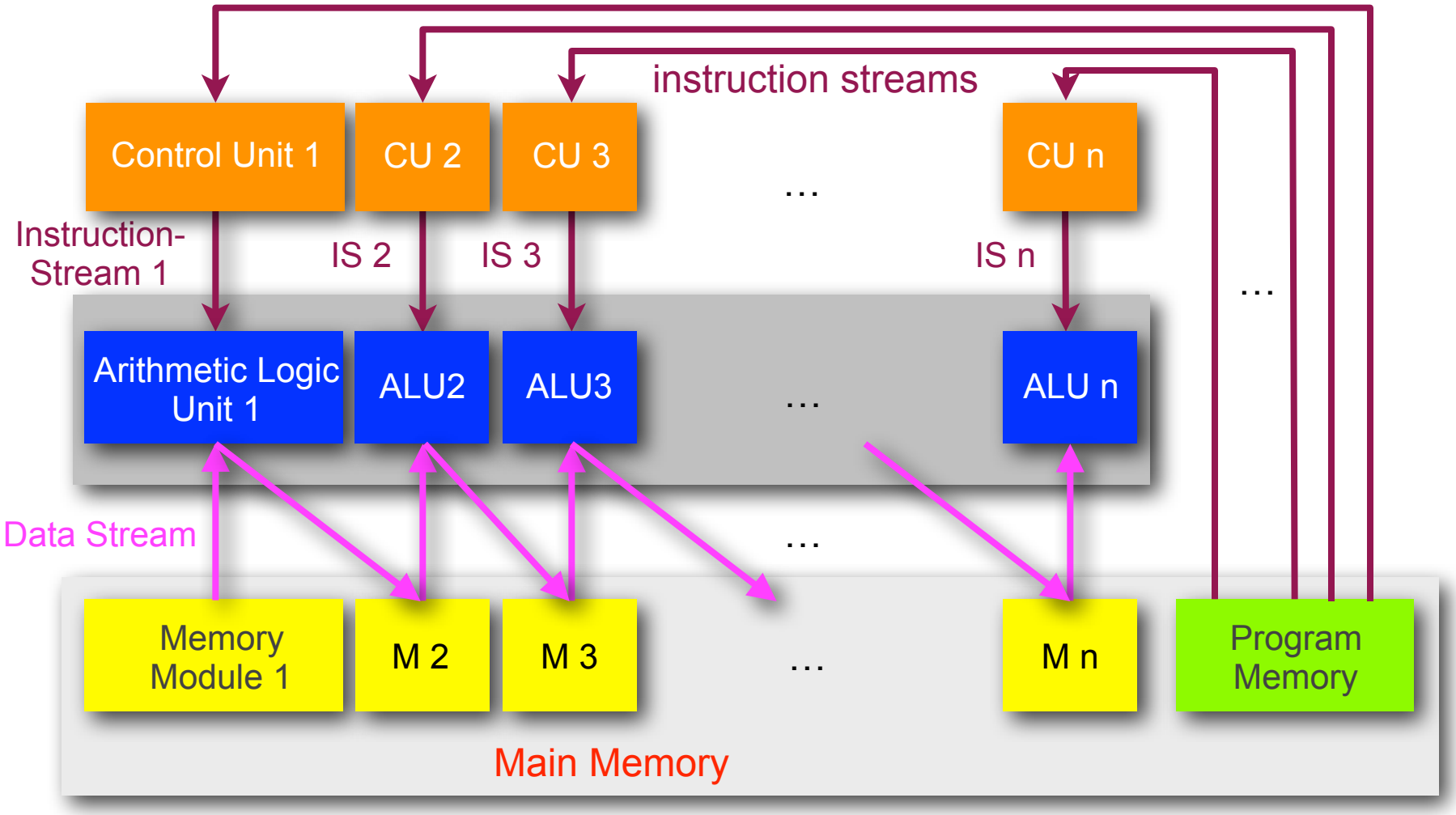
MIMD-Computer (1)



MIMD-Computer (2)

- computers which contain more than one CPU are named *multiprocessor systems*
- MIMD-systems consist of various complete processors, there exist most diverse architectural variations (processor: one or multiple arithmetic logic units can be assigned to a control unit)
- components of MIMD systems are coupled with diverse interconnect topologies
- multiple instruction and data streams from a common physical main memory or from independent memory modules are responsible for distributing information

MISD-Computer (1)



MISD-Computer (2)

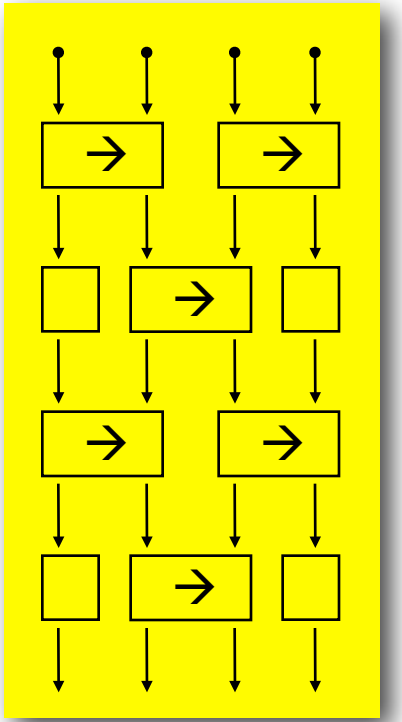
- MISD: only a few computer and there is hardly one, that has been commercially successful or that had any impact on computational science, respectively
- exception: systolic arrays
  - the description for a pipe-network of DPUs (Data Path Units)
  - mostly in a matrix arrangement, streams of data are passed through the system activated by the clock, which is the only global communication between them
  - term „systolic“ compares the data streams across the array with the circulation of the blood → heart is the clock generator
  - a normal systolic array needs no instructions because the operations in the DPUs are triggered automatically when data reach the actual DPU-input (“transport triggered”)
  - today used for processing one complex data stream (MPEG and so on) with several similar DPUs , f.e. as embedded systems in FPGAs

Example of a Systolic Array (1)

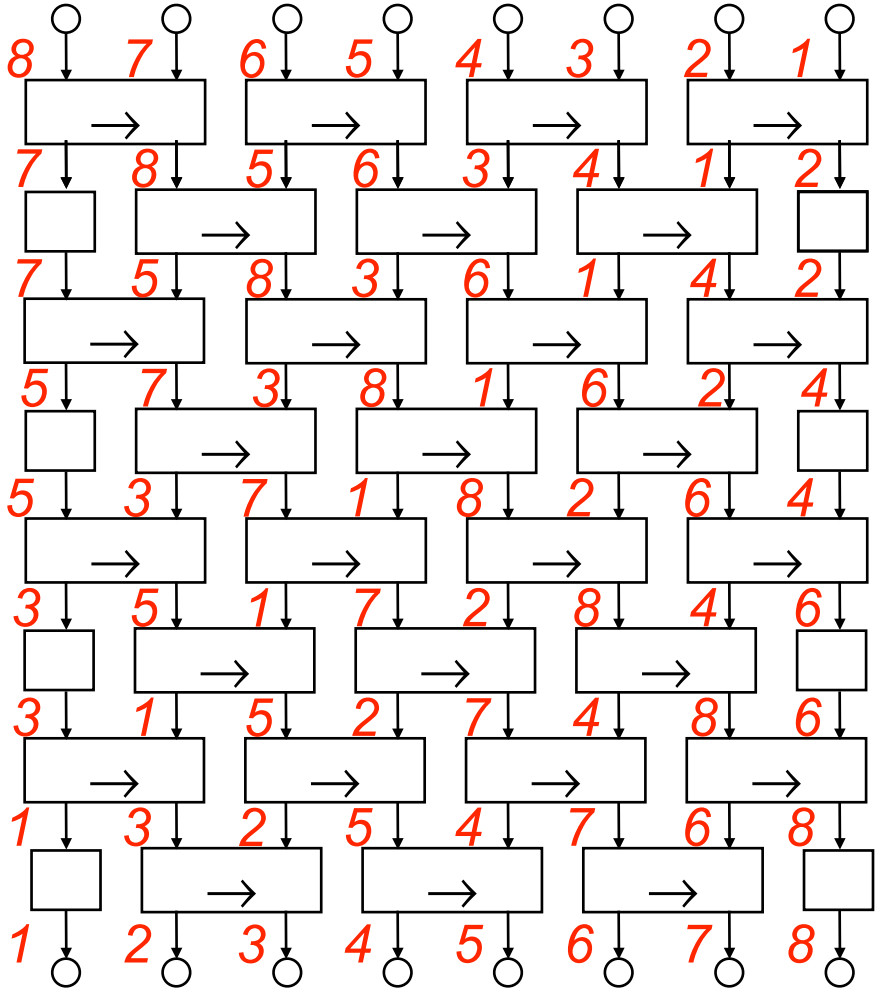
A systolic array is a special-purpose parallel device, made out of a few simple cell types which are regularly and locally connected.
[H.T. Kung]

example:

- parallel input
- elementary cells („CondSwap", „Nop")
- regular structure



Example of a systolic array (2)



... with algorithm
odd-even transposition sort (OETS), n=8

data have been „pumped" synchronously through one- or two-dimensional arrays and thereby doing calculations on these data

processors execute predetermined operations.

K+A-2_F 18

Features

- significantly more complex than Flynn‘s classification system
- objective: improvement of means of expression, describing the parallel processing in computer structures
- applicable for specification of essential quantitative characteristics of computer involving parallelism and pipelining
- 3 levels of computer structure are characterized
  - control unit level (processor level, program level)
  - arithmetic logic unit level (instruction execute level, ALU level)
  - word level (level of basic processing units)
- on this levels exist different types of parallelism and pipelining which can be determined with ECS quantitatively

Definitions (1)

- parallelism
  → exists, when – referred to a specified abstraction level – more than one action can be executed in a certain time
- serialism
  → in contrast exists, when – on a certain degree of abstraction – the defined actions are not executable simultaneously i.e. when only one action is executable
- concurrency
  → on one structural level and at one point of time can be executed multiple – in its kind comparable/identical – tasks, because inside the structural level exist multiple resources, applicable for the execution

Definitions (2)

- pipelining
  - tasks are divided inside a structural level of a computer into an equal or various number of simpler sub-steps executed one after the other, whereas a separate computer sub-unit is responsible for the execution of each sub-step
  - thereby all sub-units can execute sub-steps from different tasks simultaneously
  - after the result of a sub-step is available it is passed to the next sub-unit for further processing if applicable
  - sub-units also can be skipped when they are not required for solving a task
  - the final result of task exists after all sub-units, that were required for the task, have been passed

ECS Triplet (1)

- for classification of a computer respectively, description of a computer structure → each triplet element represents the degree of parallelism of a computer view level

  $t_{computer} := (l_{computer}, r_{computer}, w_{computer})$

- each processor exists of
  - l, being the number of specialized control units, which have for their part a certain number of
  - r, being the number of arithmetic units with a certain number of
  - w, covering the number of elementary pipeline stages.
    (in the literature l is also named as k, r is named as d too)

- thereby define the numbers l, r, w the extent of concurrency of computer on the actual view level.

- example:

  $t_{computer} := (1, 1, 32)$ classical computer,1 CU, 1 ALU, 32 bit word length

ECS Triplet (2)

- extension of ECS triplets (for sub-classification of pipelining)
  - *l'* degree of macro pipelining, number of sub-controllers (control units), that process the same data stream with different instructions
  - *r'* degree of instruction pipelining, max. number of sequential instructions, which can be executed simultaneously
  - *w'* degree of phase pipelining, number of phases, an instruction is subdivided temporally
  - $t_{computer} := (l_{computer} * l'_{computer}, r_{computer} * r'_{computer}, w_{computer} * w'_{computer})$
- mostly, multiple ECS-triplets are required for description of real computers

ECS Triplets (2)

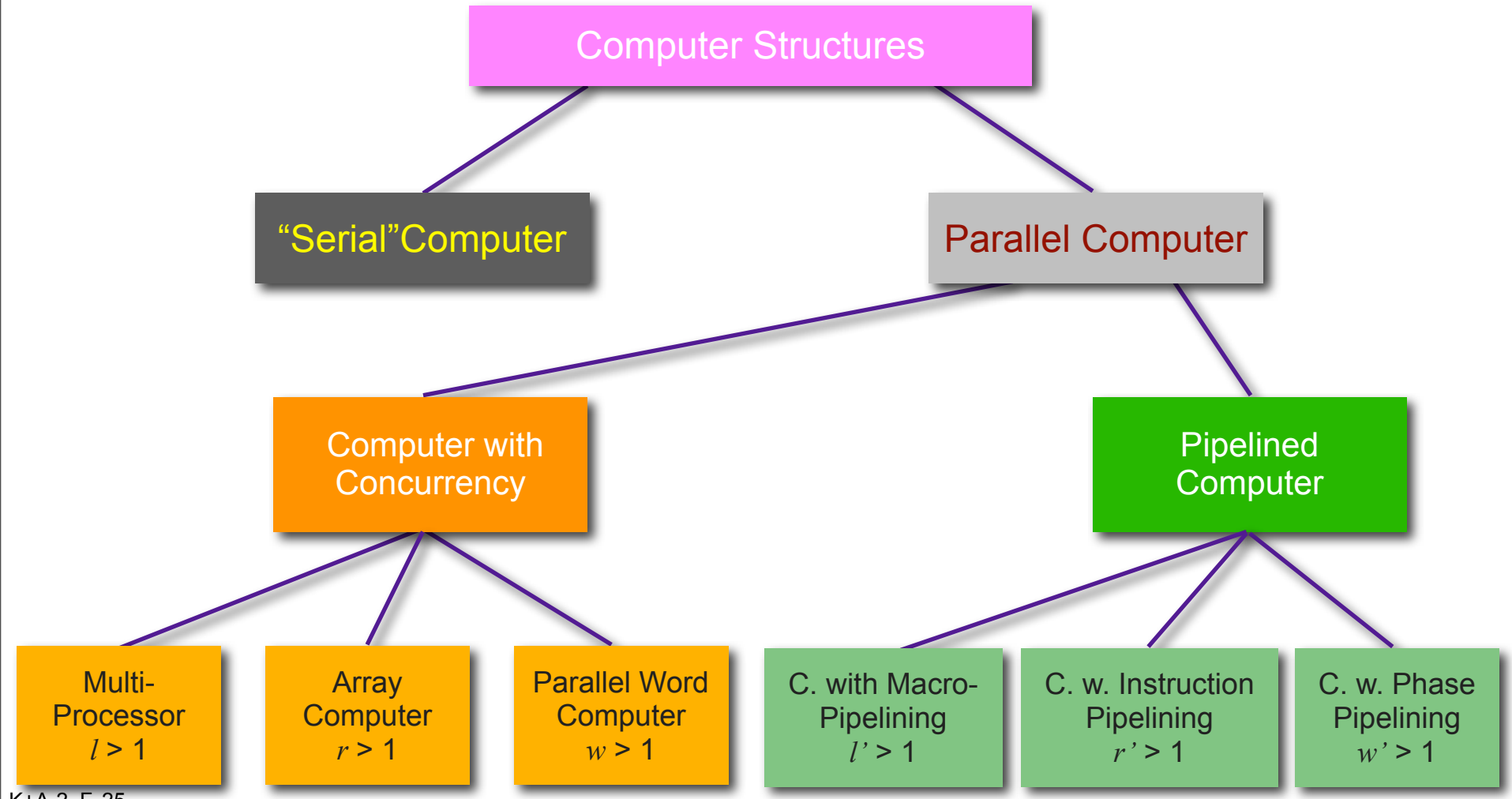
- and operator/concatenation operator "*" describes the true parallelism with no interaction of particular hardware
- or operator "+" is being applied, if structures can be used mutually
- xor operator "v" is used, if structures exclude each other
- "~" operator represents variable elements
- examples:

  $t_{Athlon} := (1, 1*9, 32) \rightarrow$ 1 CU, 9 shared ALUs, 32 Bit word length

  $t_{i486} := (1, 1, 32*5) \rightarrow$ 1 CU, 1 ALU, 32 bit word length, 5 instruction phases

  $t_{Pentium} := (1, *2, 32*5) + (0, *1, 80*8) \rightarrow$ 1 CU, 2 integer-ALU (U-pipe, V-pipe), 32 bit word length, 5 instruction phases + 1 floating point ALU, 80-bit-wide, 8 phases

ECS-Classification of Processors



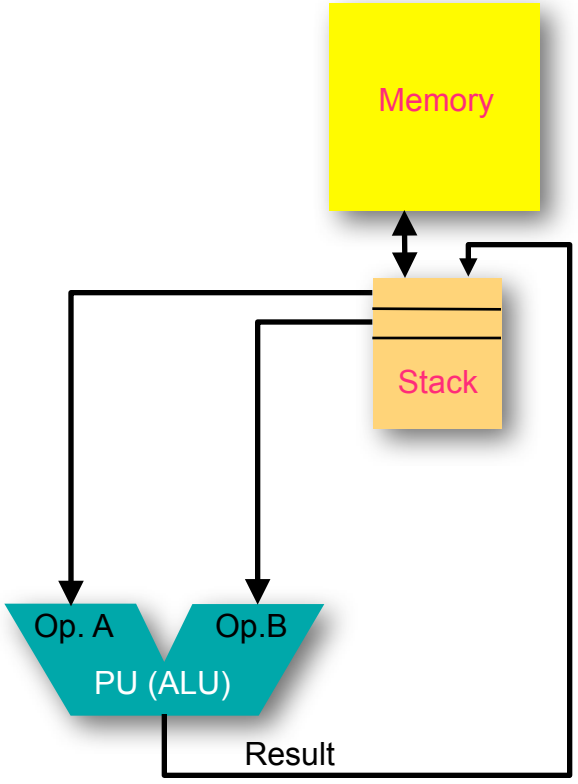
K+A-2_F 25

Classification for the Internal Memory Model (1)

specifies the access range of a processor to the operands during operation is executed, following architectural classes exist:

1. stack architecture

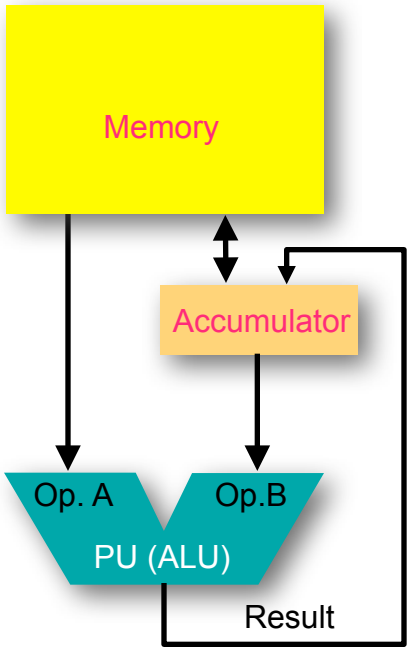
- processor accesses on a stack (implemented in hardware) wherein source and destination operands have been stored
- advantages:
  - accesses may be done implicitly (in a defined order, LIFO)
  - definition of operand addresses not required → shorter instruction word
- disadvantages:
  - transport operations required (data transfer to and from stack)
  - if required, also operations for data resorting in the stack



Classification for the Internal Memory Model (2)

2. accumulator architecture

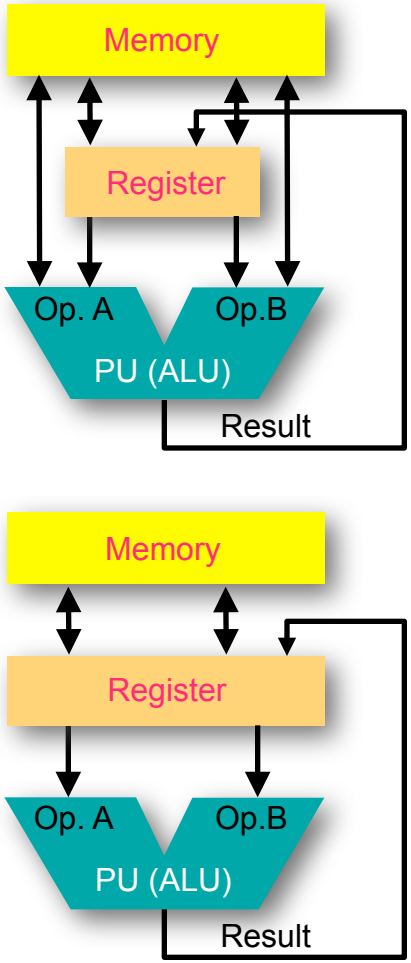
- a special register (accumulator) exists,
  it can be used as source or destination of an operation
- advantages:
  - simple architecture
  - each operation needs only 1 address because the accumulator always is used as a operand
- disadvantages:
  - accumulator emerges as bottleneck of system, because it is used often
  - this requires that its content often has to be buffered



Classification for the Internal Memory Model (3)

3. (General-Purpose-) Register Architecture

- 2 alternatives:
  - register memory architectures: register and memory address are specified explicitly
    - advantage: data can be used from the memory directly
    - disadvantage: different access times of register and memory
  - load-store-architectures: operations are executed between registers only, between internal processor register and external memory only data exchange possible
    - advantage: fast operations
    - disadvantage: additional transport operations register/memory



Classification to the internal storage/memory model (3)

type	advantages	disadvantages
stack (1, 2 and 3 address processor)	no explicit declaration of operands required, therefore minimal op-code length for arithmetic instructions	extensive exchange operations from and to stack required, i. e. growing code length, if required
accumulator (1 address processor)	particularly implicit declaration of operands possible; simplest architecture, stack is possibly abdicable	central register is used very intensely, therefore a possible bottleneck with need for intermediate storing
register-memory (2 and 3 address processor)	data can be used without load access for operations	load accesses in memory require essentially more time, therefore delays compared with register access
register-register (load-store)	quickest possible operations by register operands	explicit load accesses are required, code length increases

Classification for the Number of Address Informations (1)

- instruction word for an processing operation must contain the following data (transport operation = special case of “operation”):
  - operation code
  - address operand 1
  - address operand 2
  - address result
  - address subsequent instruction
  - address alternative subsequent instruction

→ “N address machine” → N = 5

- disadvantage: very long instruction word

→shortening by reducing details of address information

→omitting address informations of subsequent instruction or alternative subsequent instruction → “3 address machine”

Classification for the Number of Address Informations (2)

- further reducing of operand addresses possible? yes.

	op. 1	op. 2	result
3 address machine	ADR1	ADR2	ADR3
2 address maschine	ADR1	ADR2	ADR1
1 address machine	fixed	ADR2	fixed
0 address machine	fixed	fixed	fixed

- instruction word indeed is more compact, although the transport expense increases with further reduction (data memory↔register).