Architectural Basics of Computers

Prof. Dr.-Ing. Christian Paetz

Agenda

- What is a computer
- Virtual Machines
- Basic Structure of a Computer
- Memory
- ALU
- N address machines
- Internal registers
- addressing

Term ‚Computer'

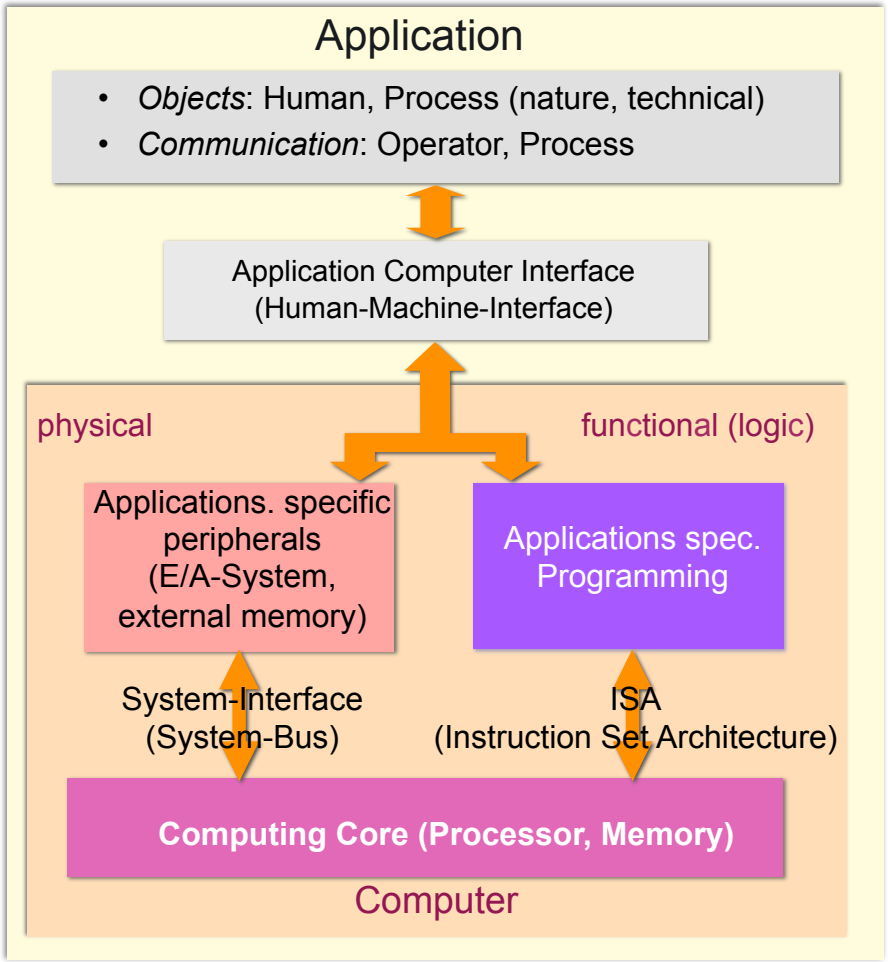
Computer, *computare*:
derived from ‚computing', dealing with numbers

now:

A computer is a general-purpose device that can be programmed to carry out a set of arithmetic or logical operations automatically. Since a sequence of operations can be readily changed, the computer can solve more than one kind of problem.

- to process
- to store
- to exchange with environment

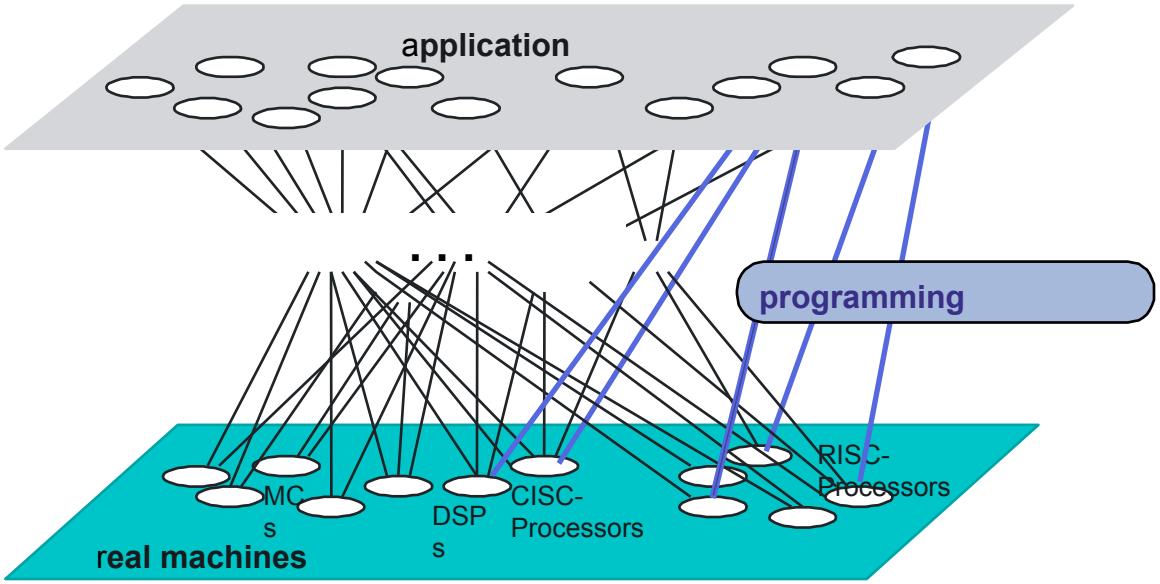
Principle of Application and Computer



- Computer must interact with humans and/or processes
- Interfaces needed (physical/devices and functional/logical)
- Devices – required peripherals
- Application – requires control program (Software)
- Devices are connected to computing core using System Bus
- Software is connected to computing core using instructions

Application versus real machines

- There is a plurality of different computers (real machines)
- Problem: Adaptation (= Communication) of different applications to different real machines



For the different real machines with different architectures and different instruction sets every application needs to be developed(programmed) again and again.

It is very useful to find a tool to minimize this effort.

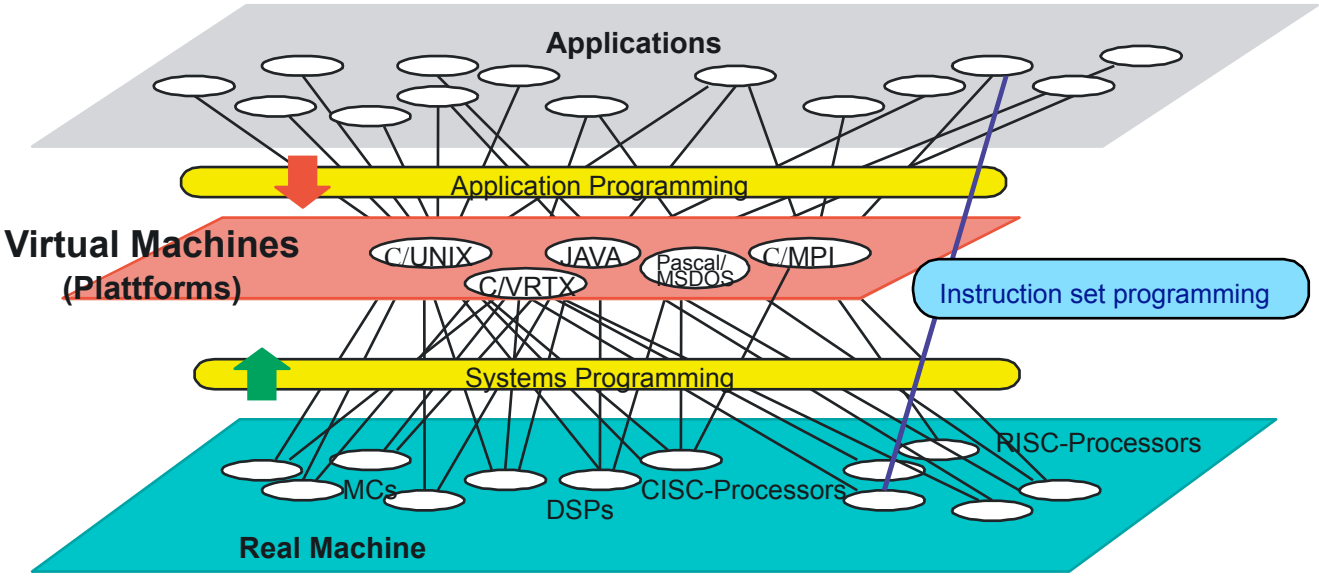
→ The virtual machine

If it´s there	and	you can see it	it´s *REAL*.
If it´s there	and	you *can't* see it	it´s *TRANSPARENT*.
If it´s *not* there	and	you can see it	it´s *VIRTUAL*.
If it´s *not* there	and	you *can't* see it	it´s *GONE*.

Roy Wilks, 1983

Between the application and the real machine there is a new apparent layer **..... The Virtual Machine**

This may be higher programming languages to simplify the description of application problems (algorithms) independent of the final target real machine
(C, Pascal, FORTRAN, ... + SW-frameworks)



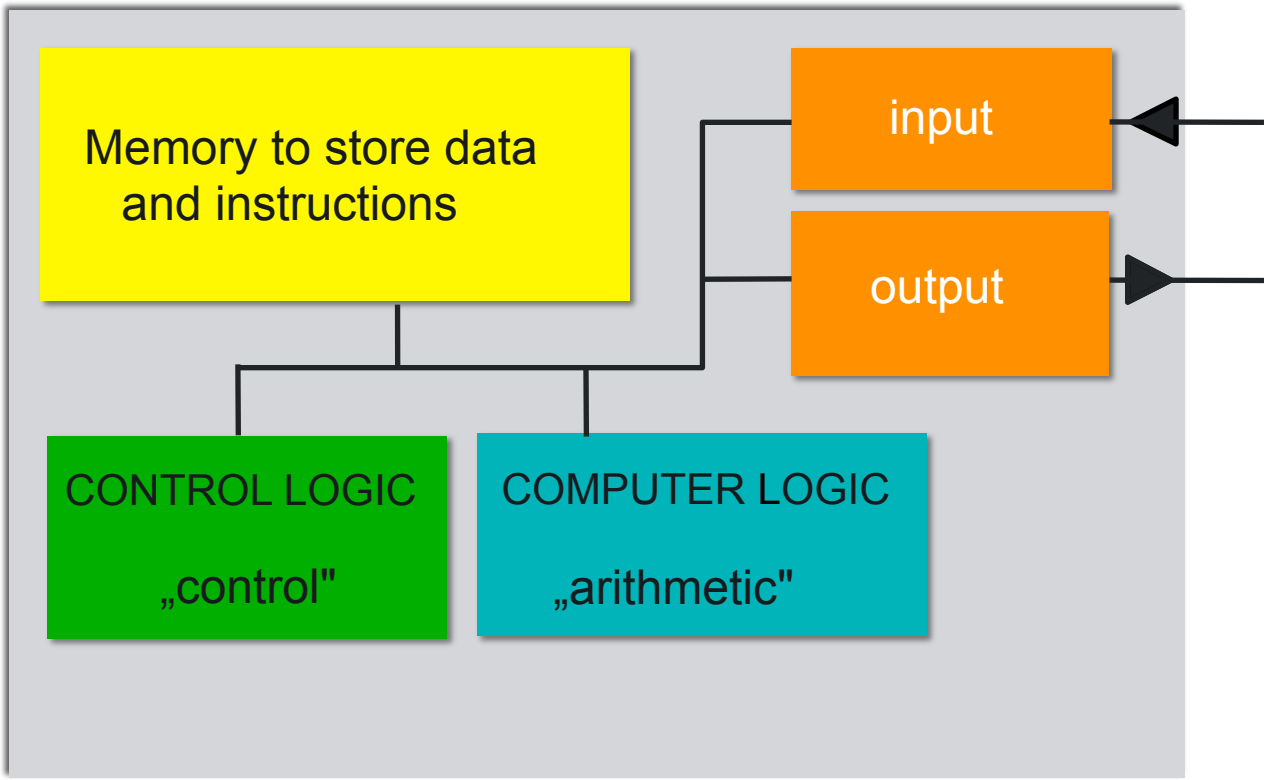
From the application to the real machine

- The step from the application to the virtual machine is called ‚**application programming**‘ (e.g. C or Java Programming).
- The adaption to a specific processor (real machine) is done using so called **compilers** (C to IS).
- The step from the instruction set to the virtual machine is called ‚**systems programming**‘.
- Combining application and system programs is sometimes referred to as ‚**embedded programming**‘ if the real machine is a microcontroller type of real machine.

Basic Structure of a (our) machine (computer)

- The structure of the machine is independent of the type and structure of the problem to be solved
- Memory is designed in a homogeneous way, structured into cells of same size with sequential address
- Data and Programming Code = linear sequence of instruction
  - Instructions usually do not contain real value but refer to adresses of memory containing the value
- Linear sequence can be excaped using Jump or branch operations
- Machine works on binary code

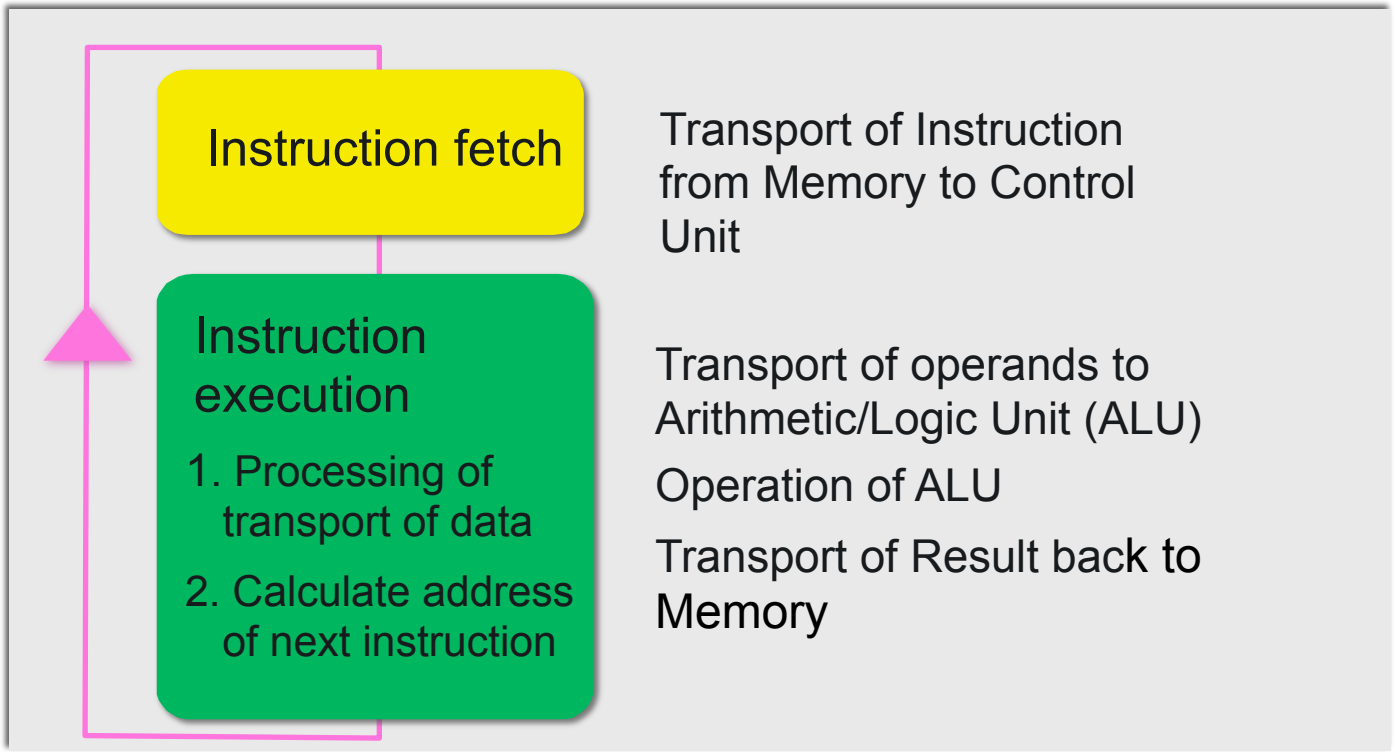
Basic Structure of Multi Purpose Machine



All five basic assumptions are found here!

Basic Principle

Control Unit = Automat, continously executing instructions in two steps



Most contemporary machines use this principle

- **Memory** (= ‚Container')
  - Memory Cell with two possible status values ‚0' and ‚1' (, Container is empty or full')
  - One 0/1 = Yes/No-Information = 1 Bit
  - Combination of N Cells closed to the ALU is called ‚register'
  - One set of N*N memory cells is called memory array or short ‚memory'
- **Busses** – connections of a certain size (number of bits transported in parallel) (parallel vs. serial bus**)**
- **Units to process data** (Arithmetic/Logic Unit = ALU) also need a certain size

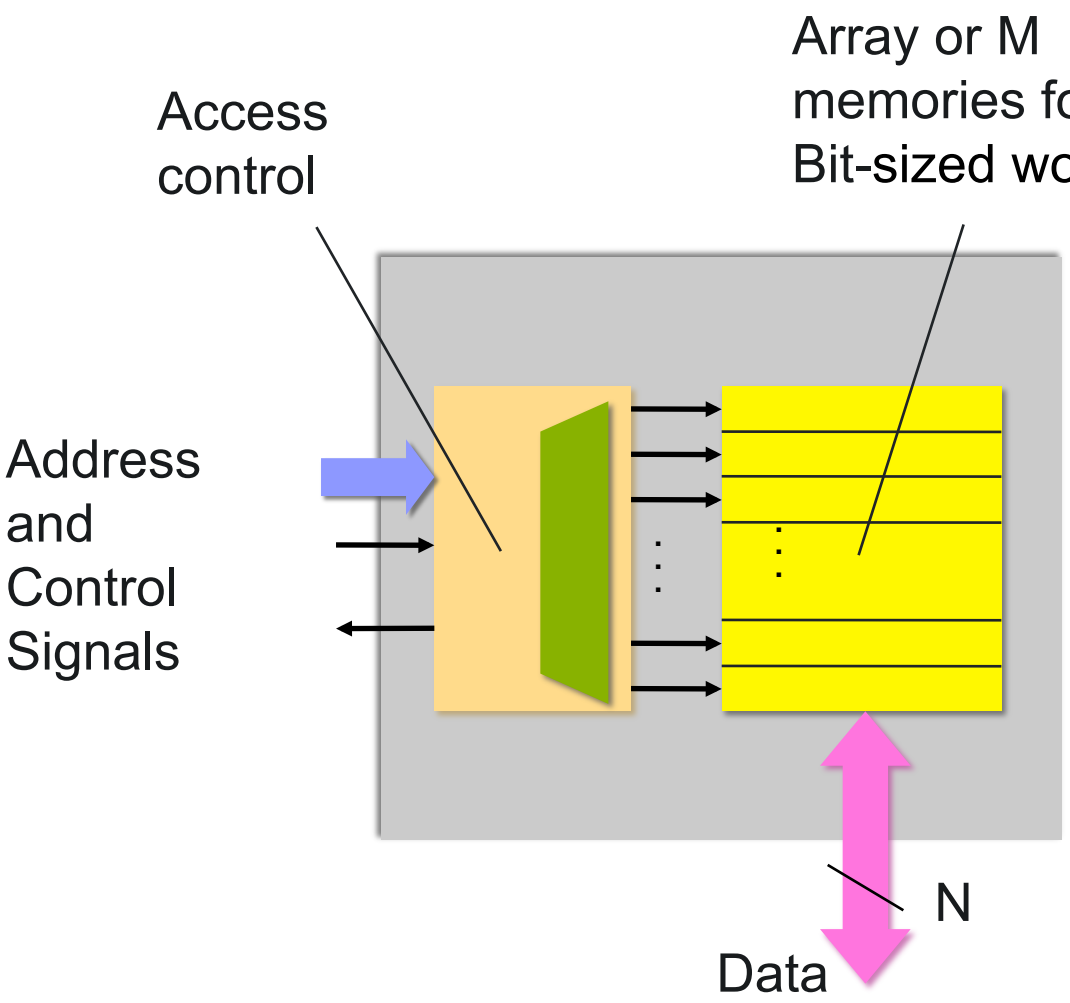
- Machines can be differentiated referring to the size of their ALU and internal bus systems (e.g. 64 Bit machine)

Memory

Memory usually has the following parameter:

- Capacity
- Access Width (size of the data word)
- Access Time
- Access Rate (‚Memory Bandwidth‘) = 1/Cycle Time
- Access Organisation
  - RAM – Random Access Memory
  - CAM – Content Addressed Memory
  - LIFO – Last In First Out
  - FIFO – First In First Out

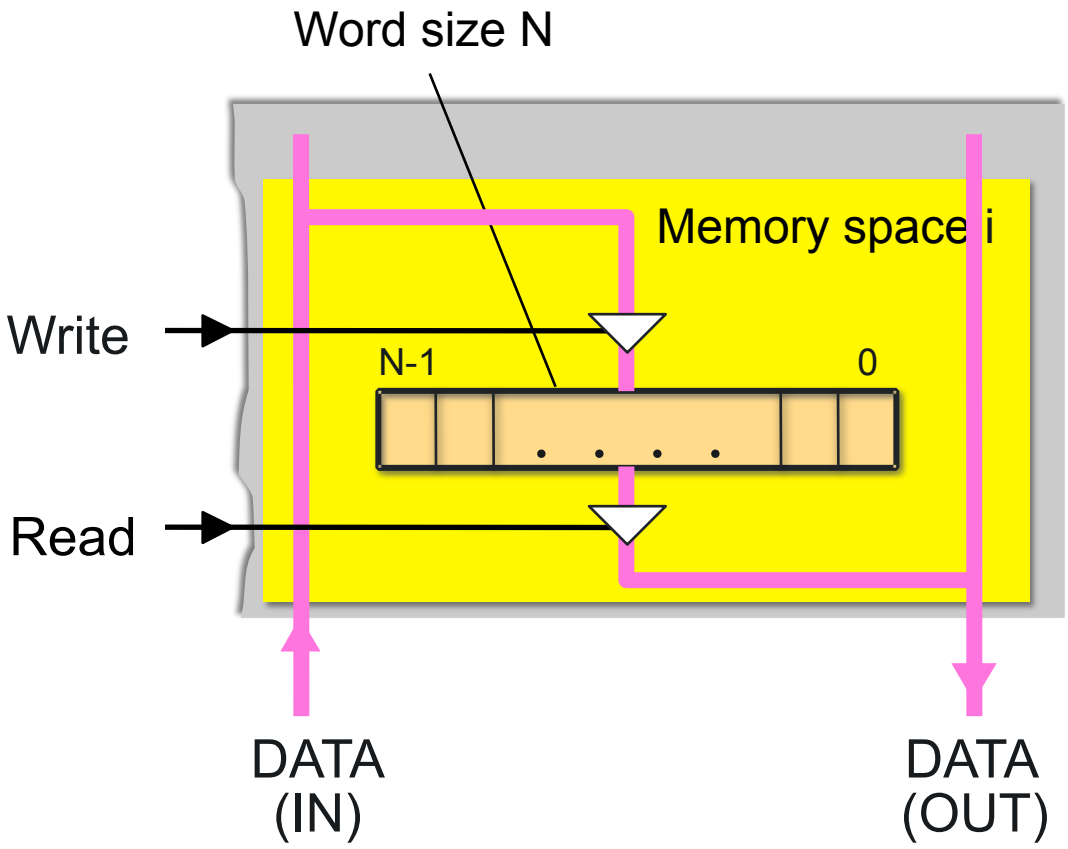
Memory RAM



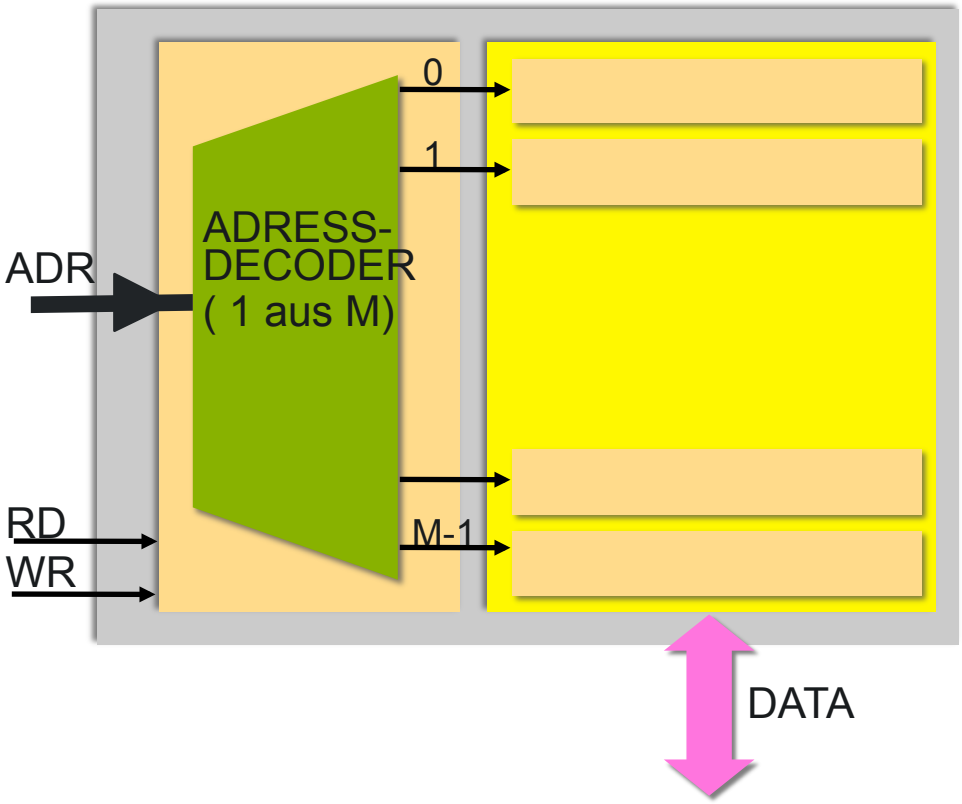
- Memory Array for M words of N size
- Read/Write
- Random access to any data word by direct association of any memory word to a given address
- Access control by control signals to allow read/write (gates) -> ‚Gates‘

RAM: Memory Space

- Access control using control signals to allow Read/Write etc



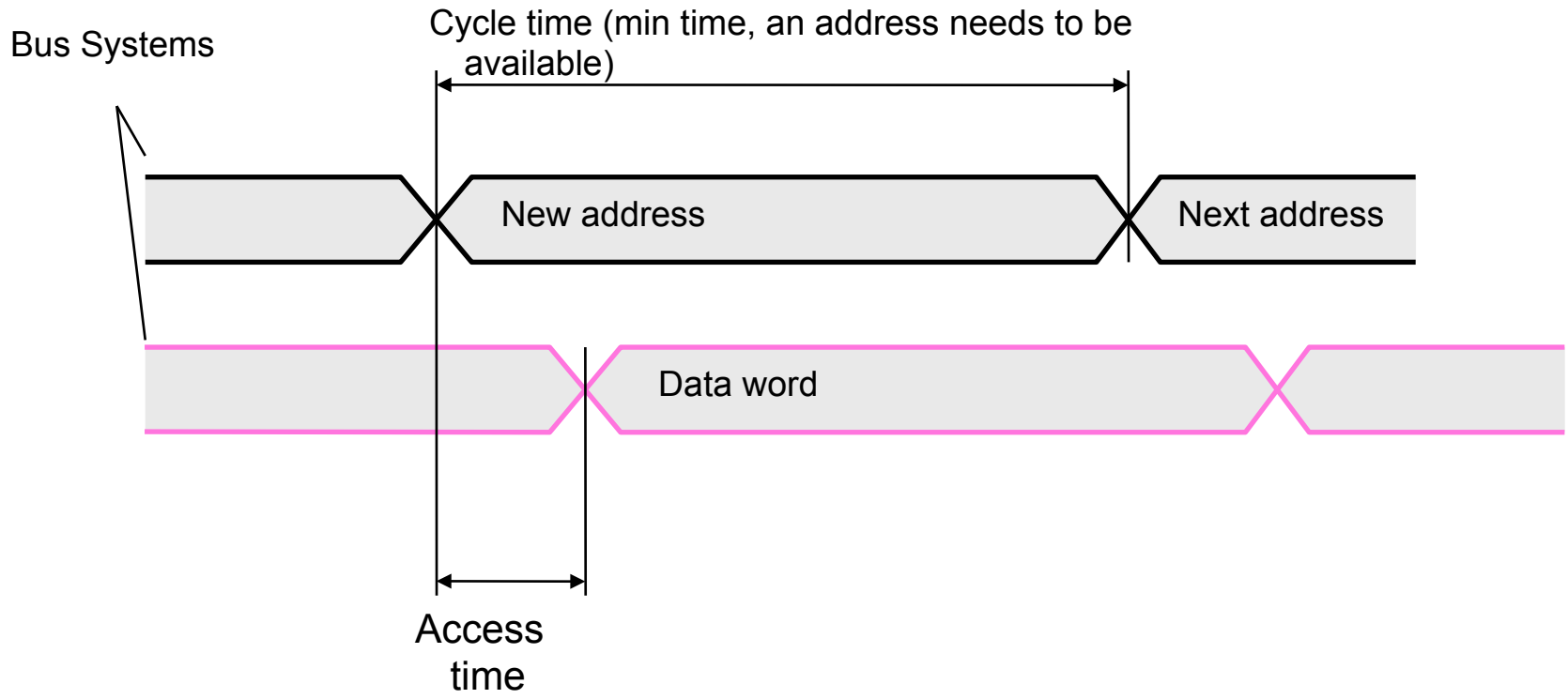
RAM: Address decoder



- Address decoder generates memory space selection lines by decoding the submitted address

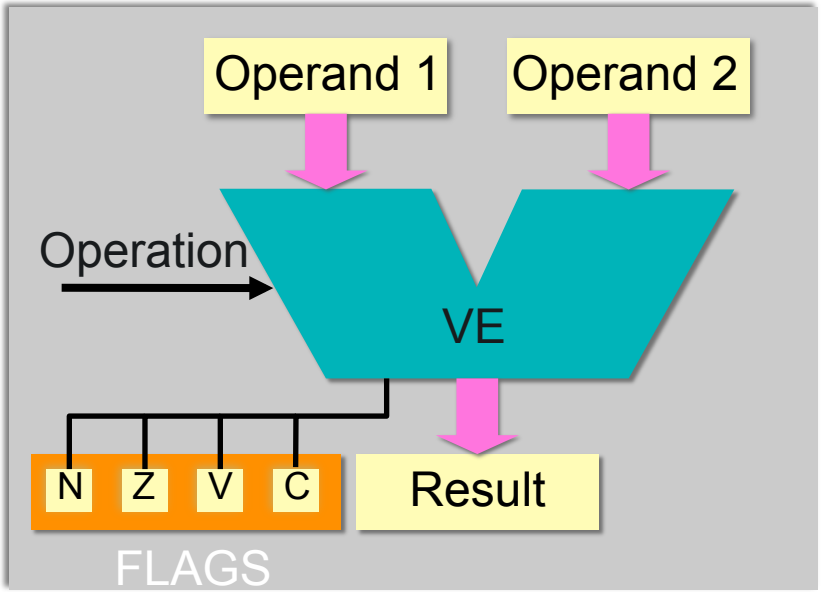
RAM: Access time, Cycle time

- important parameter of a memory
- has great impact on performance of memory and machine



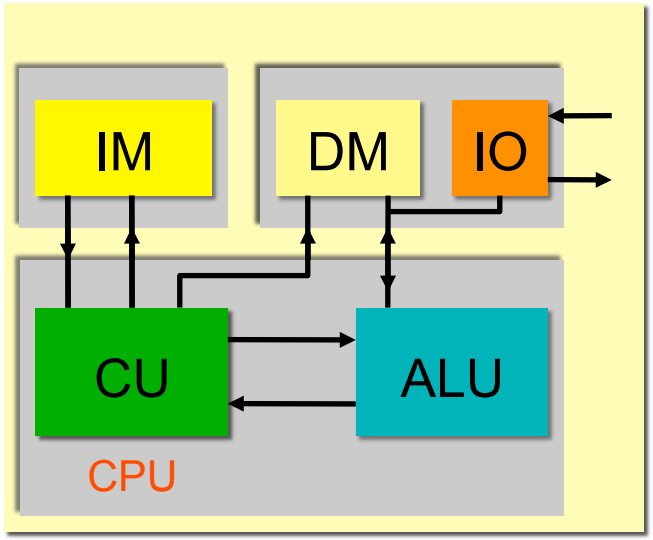
Processing Units

- Processing in information technology?
  - Manipulation of a binary word (a memory cell) according to a given rule
  - Connection of two (or more) binary words (Operands) to one (or more) binary words (result) according to a certain rule (Operation)



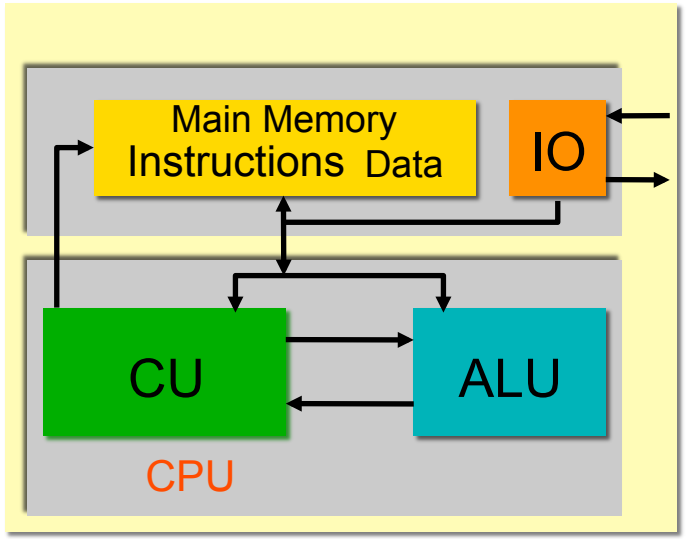
Basic Architectures

Harvard-Architecture



- Seperate memory for data and instructions
- Parallel access to instructions and data
- Separate busses

Princeton-Architecture



- Common memory for data and instructions(„main memory“)
- Simplification but less powerful

Central Processing Unit (CPU)

Central Processing Unit

= functional unit , consists as a minimum of

- one arithmetic/ logic unit (*compute unit*) and
- one control unit (*without instruction memory*)

The central processing unit makes sure that:

- The operands in the memory in I/O port are well addressed.
- The operation is specified.
- The result is awaited.
- The result is transported back to memory or to I/O port.
- The next instruction is determined (may depend on the result of previous operation result).

To perform this operation the following data needs to be encoded in every instruction

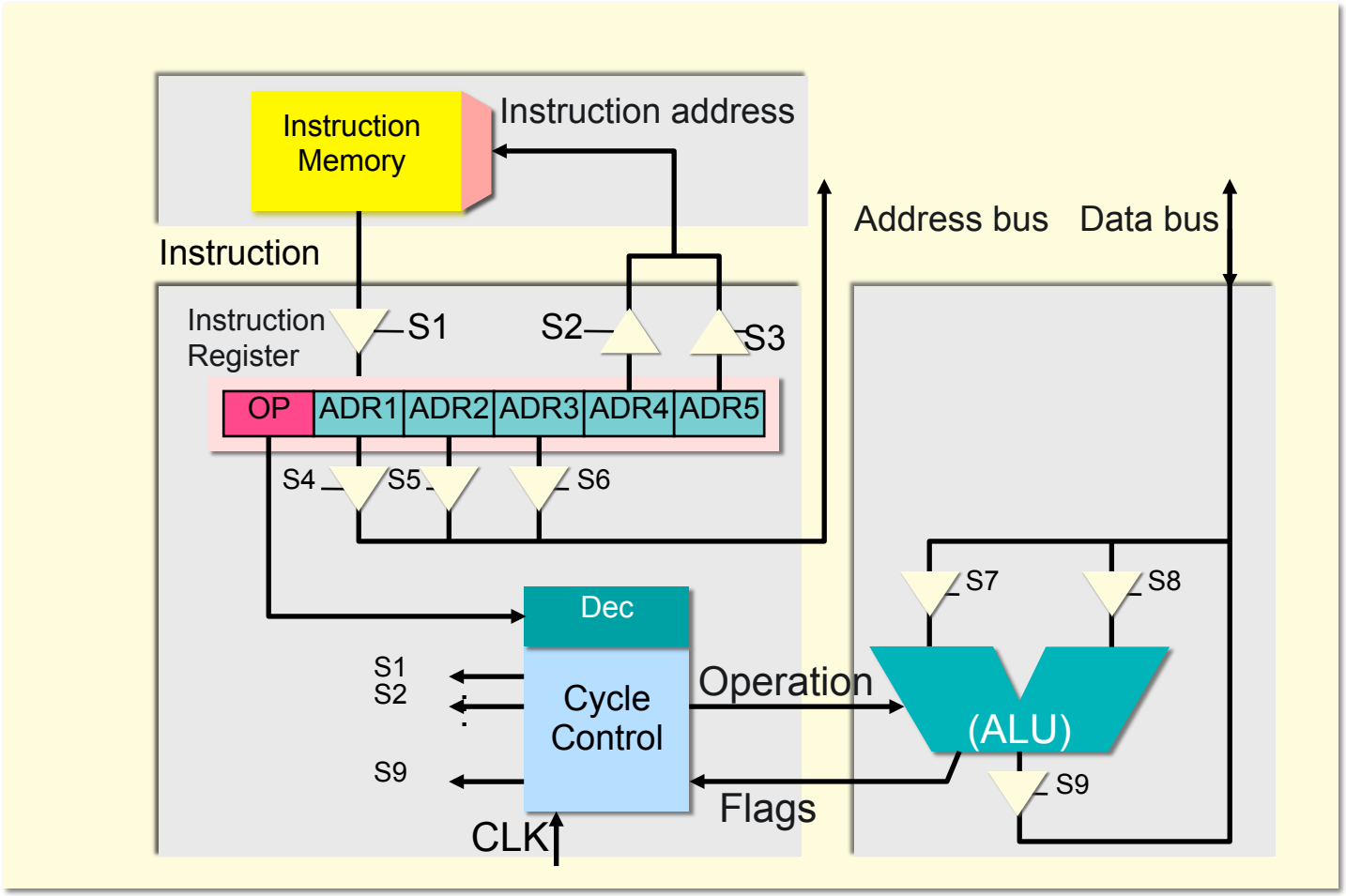
- If instruction is a processing instruction

Operation	ADR Operand 1	ADR Operand 2	ADR Result	ADR of next Instruction	ADR of altern. Instruction

- If instruction is transport instruction: Special case of data manipulation instruction, (no manipulation)

→This structure is known as „N address concept“, here 5 address concept

5-Address-Machine



Simplification of CPU Concept

Instruction Decoder generated signals for Gates S1...S9

- Disadvantage: Instruction word is very long

  -> shortening of instruction word by eliminating addresses

a. Remove ‚Next Instruction' address

Since most of the instructions are inline and jumps to other memory locations are infrequent, it is more efficient to add the following structural enhancements:

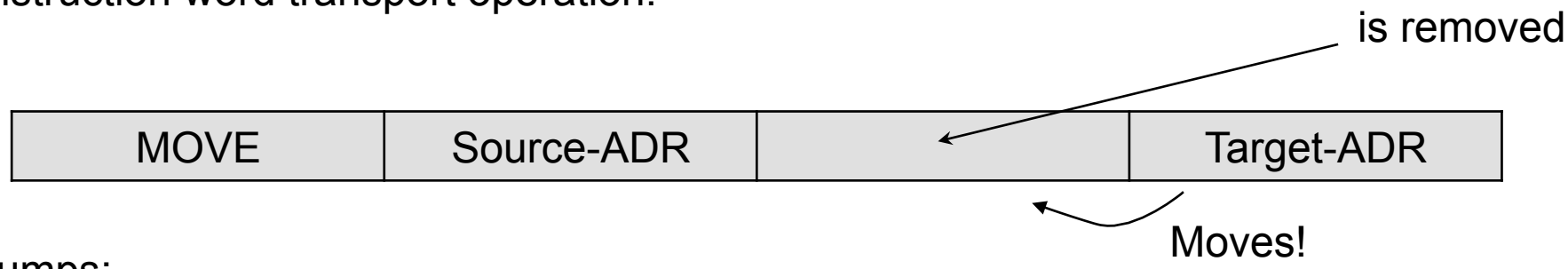
1. An instruction counter (register), containing the address of the next instruction, to be incremented with every instruction
2. Extention of instruction set with „JUMP" operation to block linear execution: conditional or unconditional jump (essentially load IR with new value)

   -> 3-address-machine

Instruction word processing operation:

Operation	**ADR Operand 1**	**ADR Operand 2**	**ADR Result**

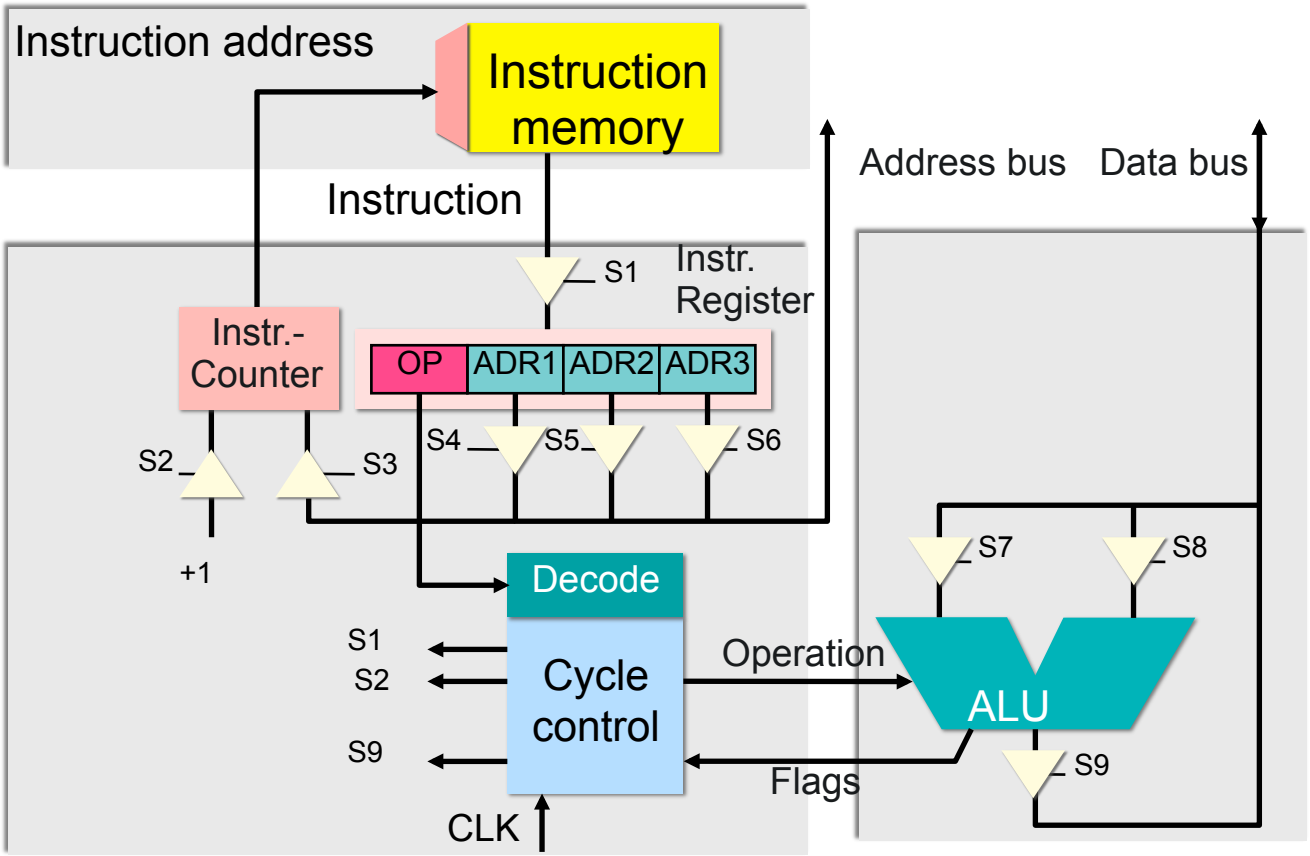
Instruction word transport operation:



Jumps:

JUMP/ COND	ADR Next instruction

3-Address-Machine



Further Reduction of Addresses

b. Reduction of operands addresses

Reduction by dual use of memory spaces resp. agreement on fixed address , disadvantage: transport effort increases

	Operand 1	Operand 2	Result	
3-address-machine	ADR 1	ADR 2	ADR 3	
2-address-machine	ADR 1	ADR 2	ADR 1	
1-address-machine	Fix	ADR 1	Fix	Transport
0-address-machine	Fix	Fix	Fix	Even more transport

Instruction Set of 2 Address Machine

- today most CPUs are 2, 1, 0 –address-machines
  - Instruction word processing

Operation	ADR Operand 1/ Result	ADR Operand 2

  - Instruction word transport

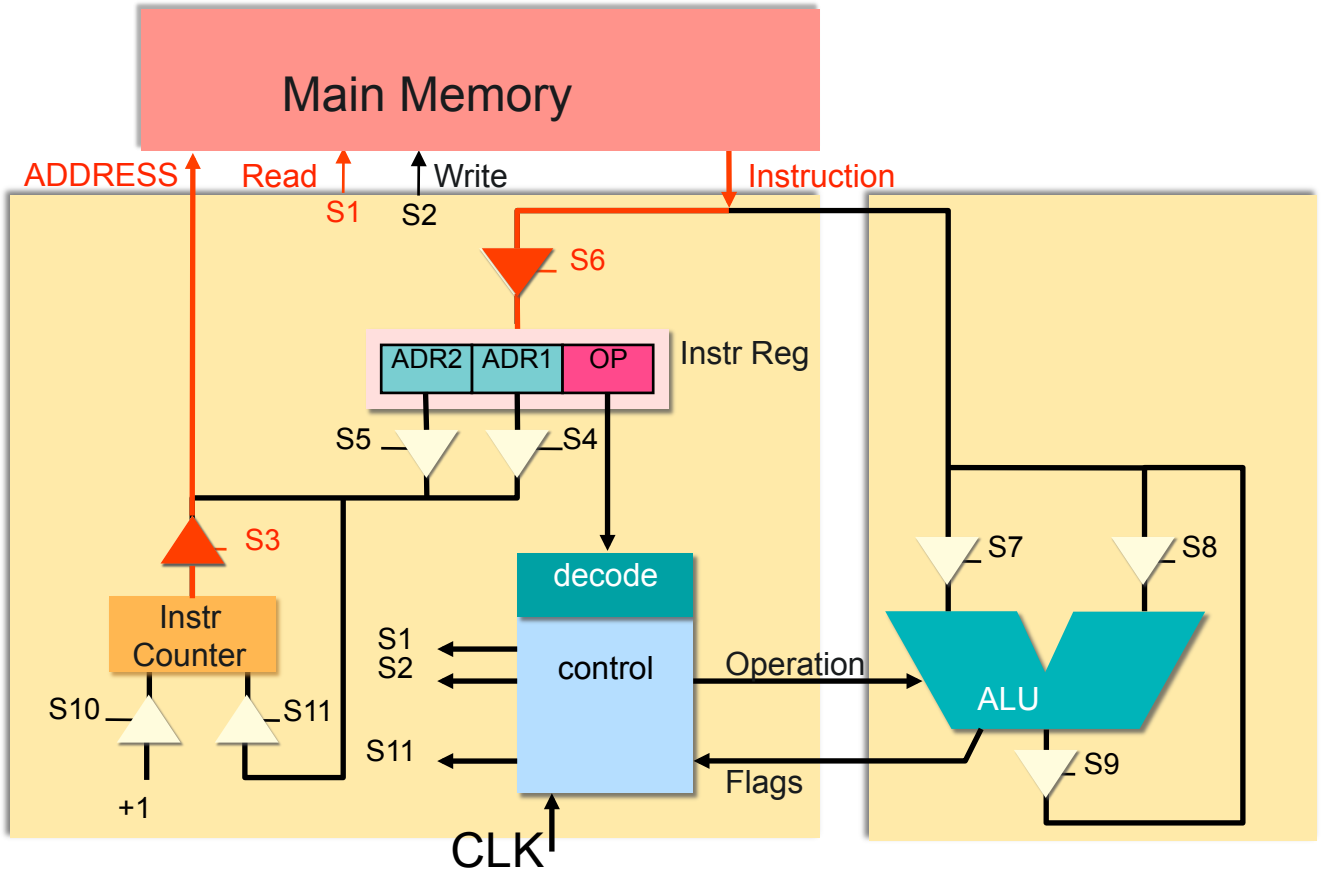
MOVE	Source-ADR	Target-ADR

  - Program Control

Jump/ Conditional	ADR next instruction

2-Address-Machine

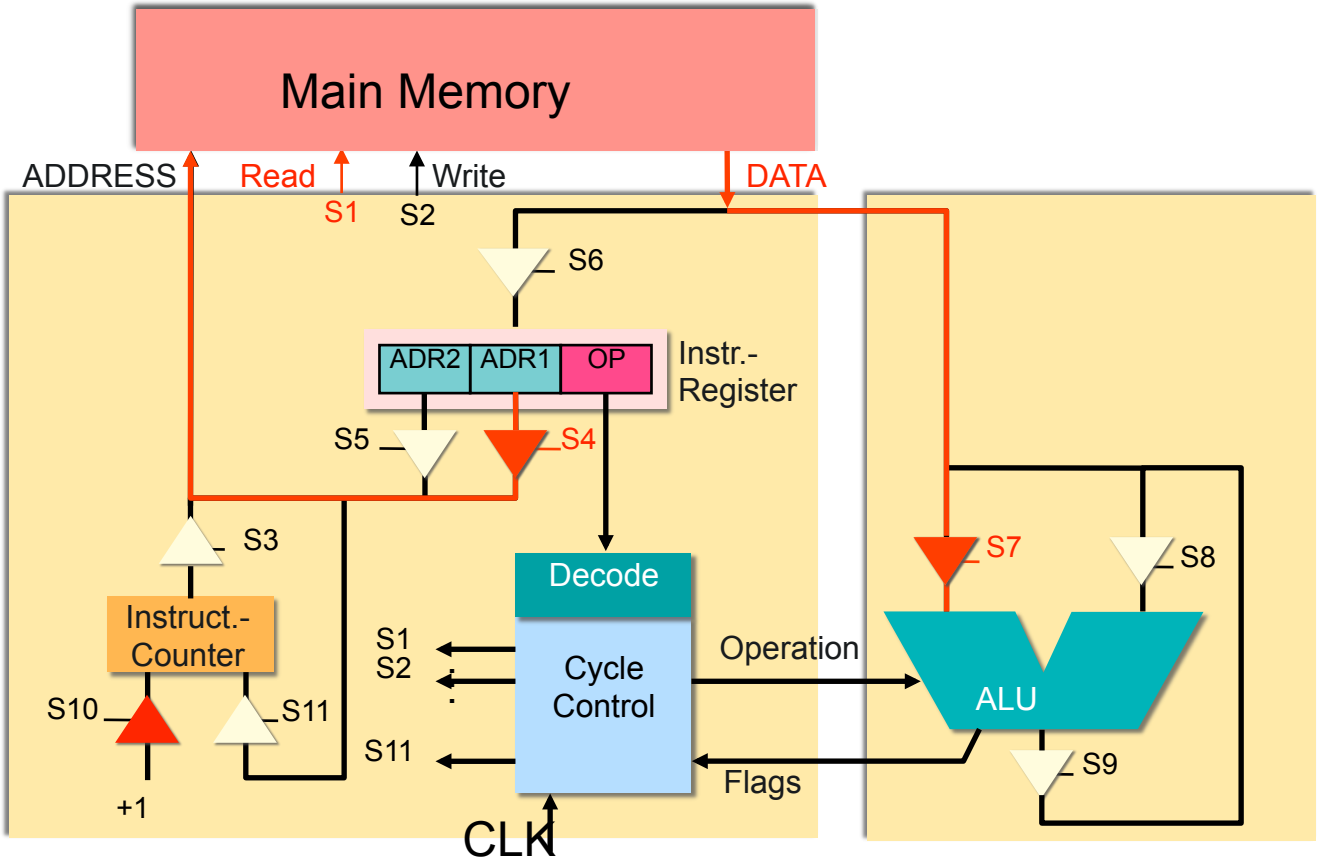
Step 1: Operation Fetch



- S3 open – value of instruction counter is address for memory
- S1 Read – Memory Content (Instruction) is read
- S6 open – Instruction is stored in instruction register

2-Address-Machine (Princeton-Architecture)

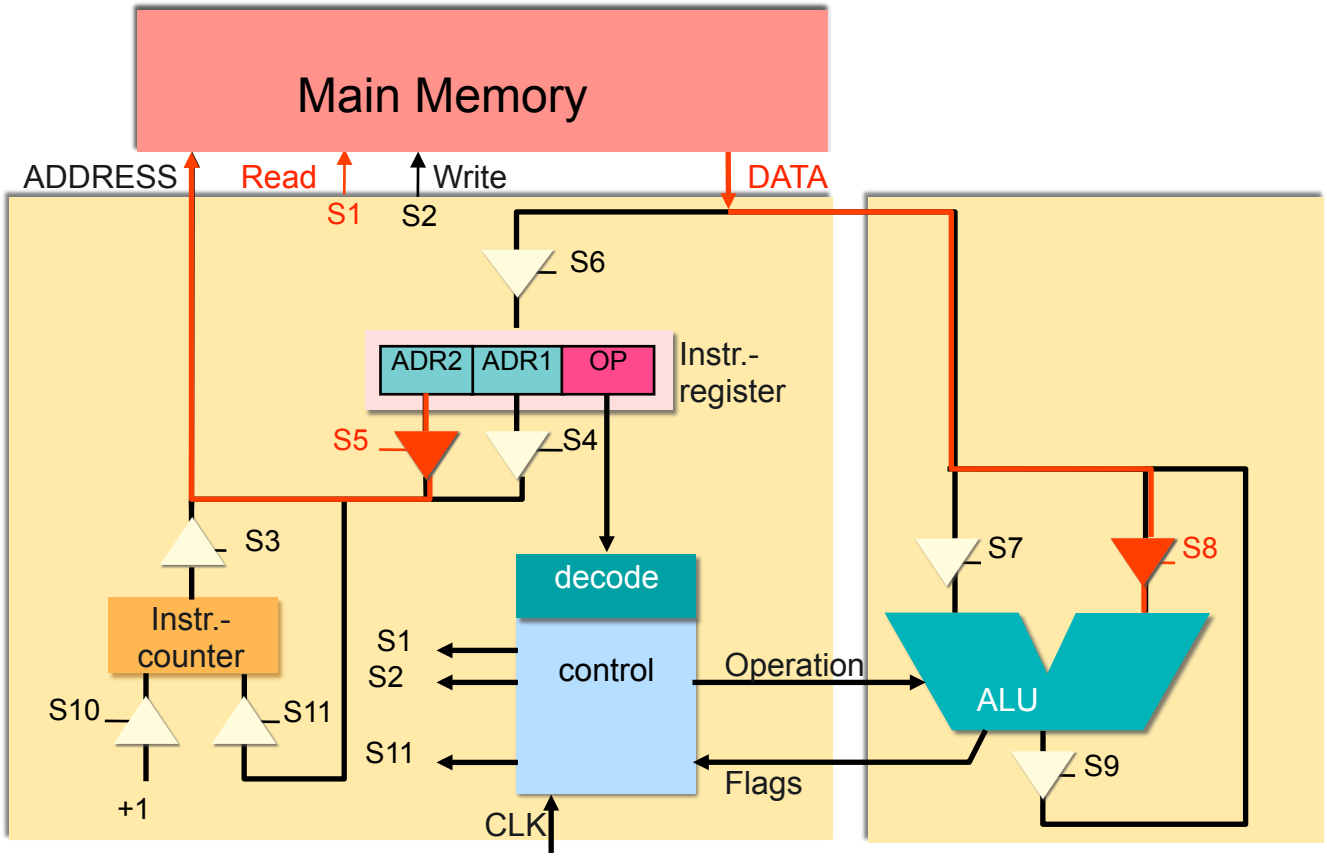
Step 2: Analyse Instruction („Process“) – Operand 1



- S4 open – ADR1 (Operand1) address to memory
- S1 read – memory is read
- S7 open – write into ALU
- S10 open – Counter+1

2-Address-Machine (Princeton-Architecture)

Step 3: Analyse Instruction („Processing“) – Operand 2



- S5 open –

ADR2 on memory

- S1 read –

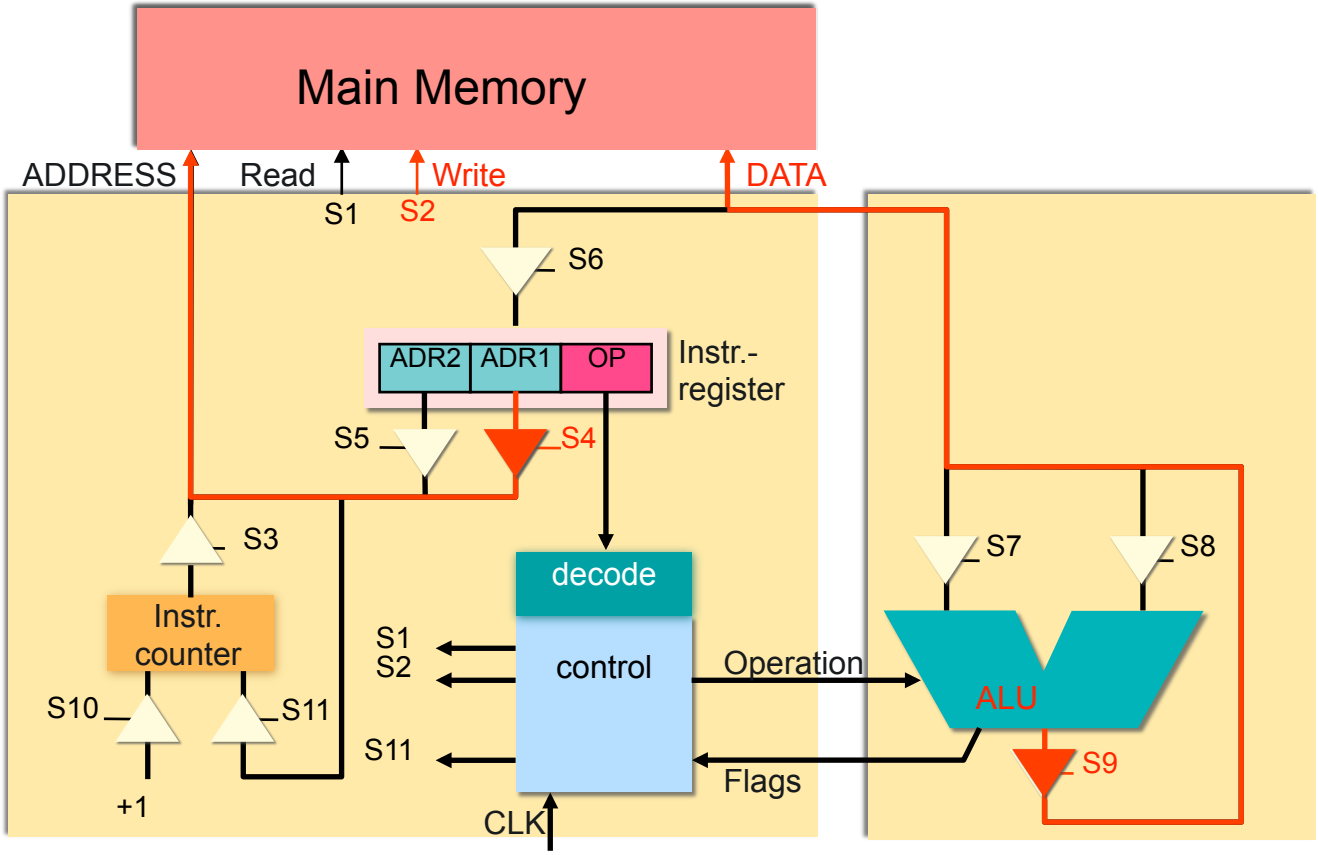
Memory is read

- S8 open –
Op is written into ALU

→Processing of Op1 and Op2 in ALU

2-Address-Machine (Princeton-Architecture)

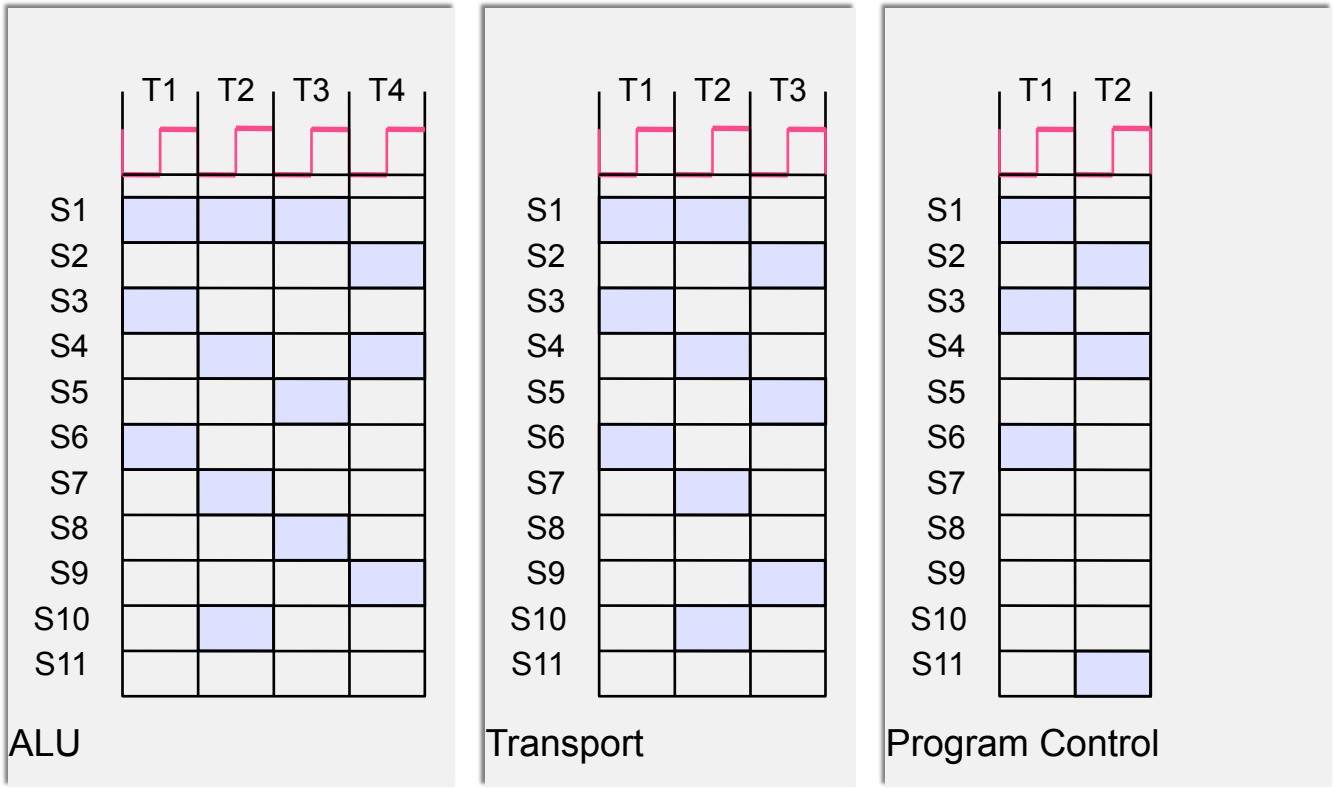
Step 4: Saving the results



- S9 open – result is sent to memory
- S4 open – ADR1 (old) supplied for result
- S2 write – result is written into memory

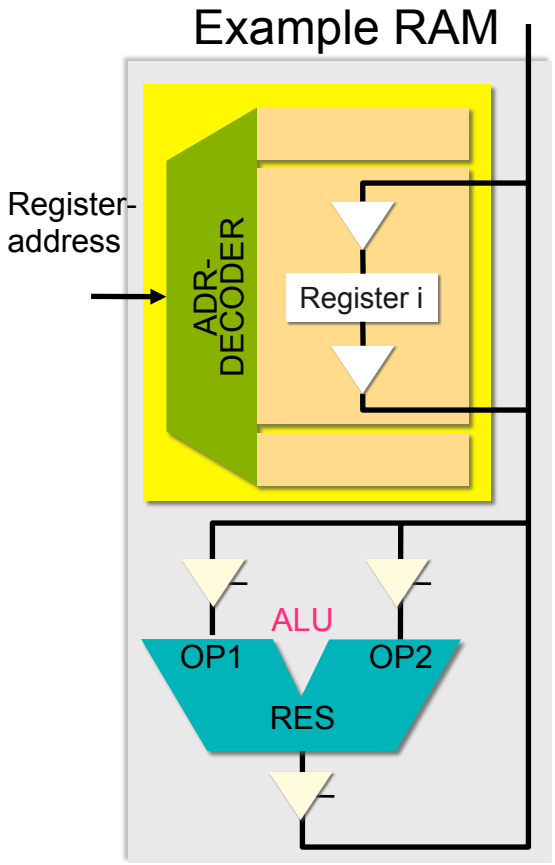
2-Address-Machine (Princeton-Architecture)

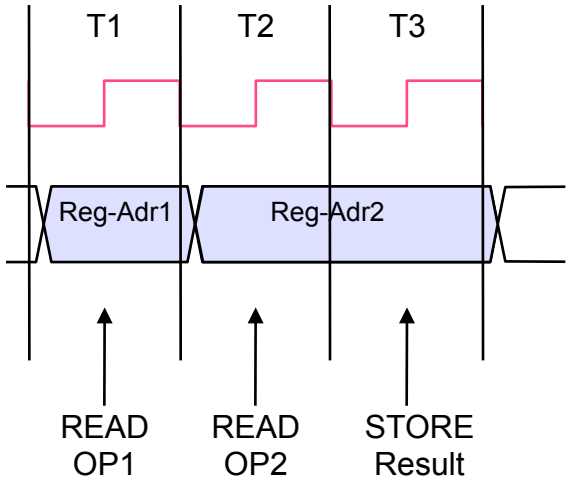
Signals



Data Register

- within the CPU there is a small (universal) memory to reduce access to main memory
- 2 Options:
  - RAM
  - LIFO





Addressing

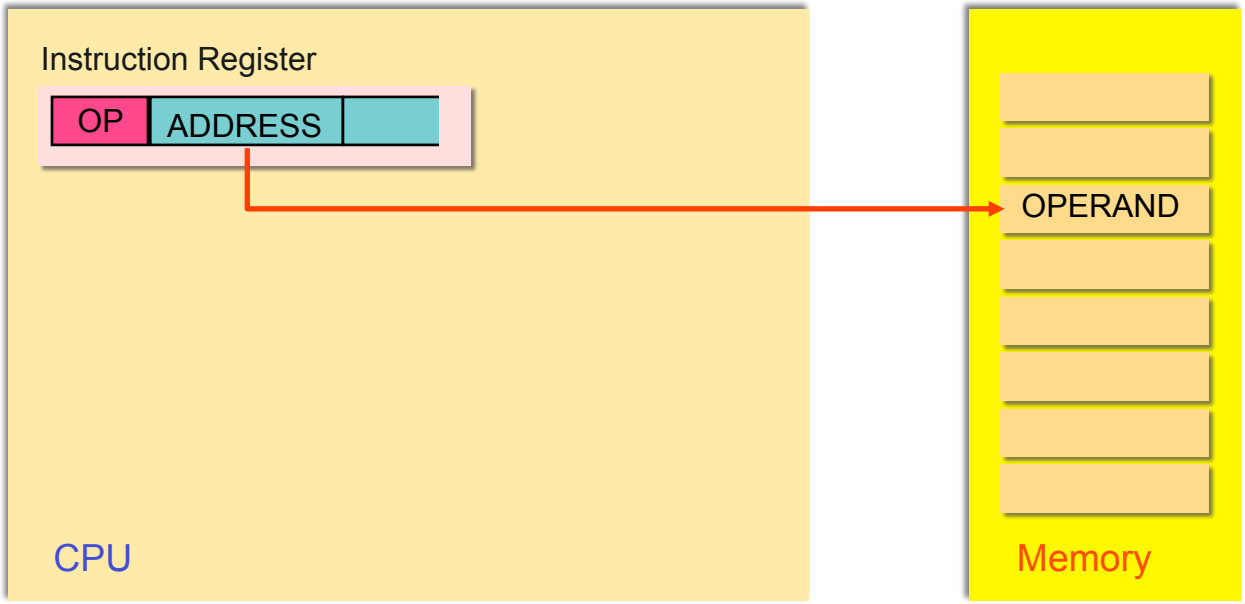
2 Versions:

1. Processor knows all combinations of addresses and registers in instruction register = symmetrical addressing

2. Reduction to a minimum of what is needed = LOAD/STORE-concept

Processing operations are limited to register access – simple organization, similar processing time for all ALU operations but more instructions needed.

Direct Addressing



Instruction contains direct address of operand resp. next instruction.

Addresses must be constants – big disadvantage.

Addressing

Improvement: addresses must be variable

- Address = „Variable“
- can be stored -> address register
- can be computed -> address computing

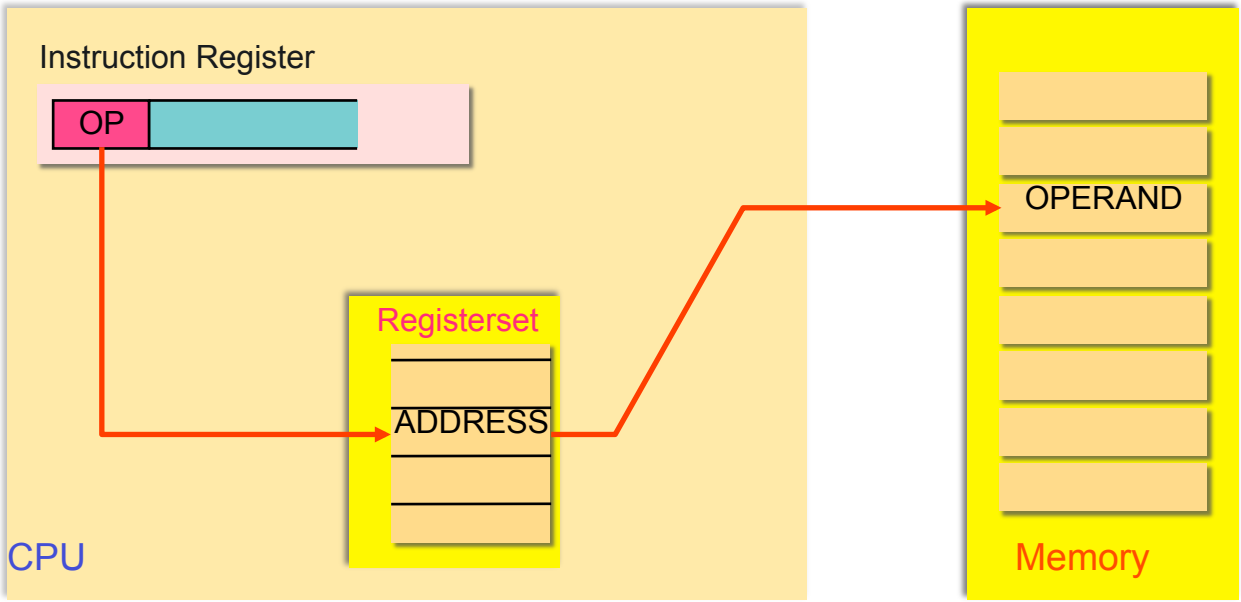
structural extensions needed:

- instruction must contain information regarding type of addressing (how to determine)
- memory to store addresses
- address-processing

2 Options:

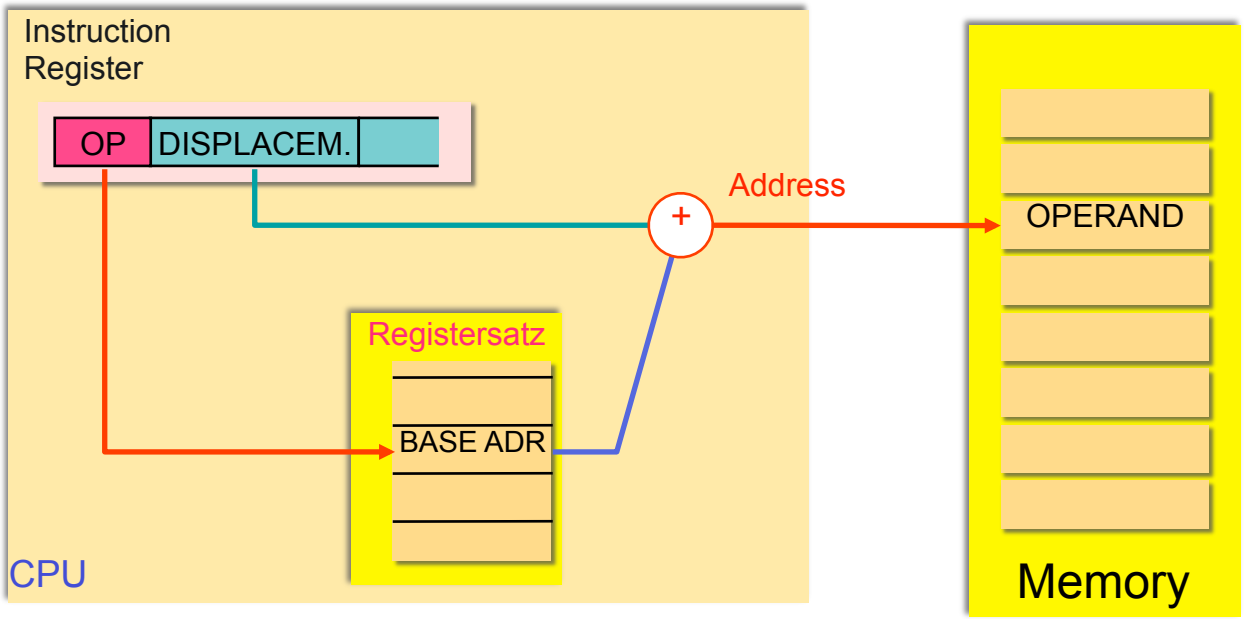
- separate address registers
- use the universal register set

Indirect Addressing



Instruction contains (explicit or implicit) reference to a register containing the address of the operand

Addressing: Indexed Addressing



Instruction contains a number (displacement) and a reference to a Register (index register).
The effective address is the sum of the content of the index register and the displacement.

Summary

- What is a computer
- Virtual Machines
- Basic Structure of a Computer
- Memory
- ALU
- X address machines
- Internal registers
- addressing