دانشکده مهندسی و فناوری کامپیوتر

درس معماری کامپیوتر

الهام چشمی خانی

Parts (text & figures) of this lecture are adopted from:

- "Computer System Architecture", MIT handouts, 2017
- "Computer Architecture" handouts, by Prof. Behrooz Parhami
- "Computer Architecture" handouts, by Prof. Onur Mutlu
- "Computer Architecture" handouts, by Dr. Hamed Farbeh

Instruction Set Architecture

طراحی سلسله مراتبی

- **نیاز به طراحی سیستم بهینه برای دستیابی به اهداف موردنظر**

✓ طراحی سلسله مراتبی

طراحی سلسله مراتبی

- **روند حرکت فناوری**
  - ✓ سیستم‌های نهفته، سرورها، محاسبات ابری، هوش مصنوعی، حوزه سلامت

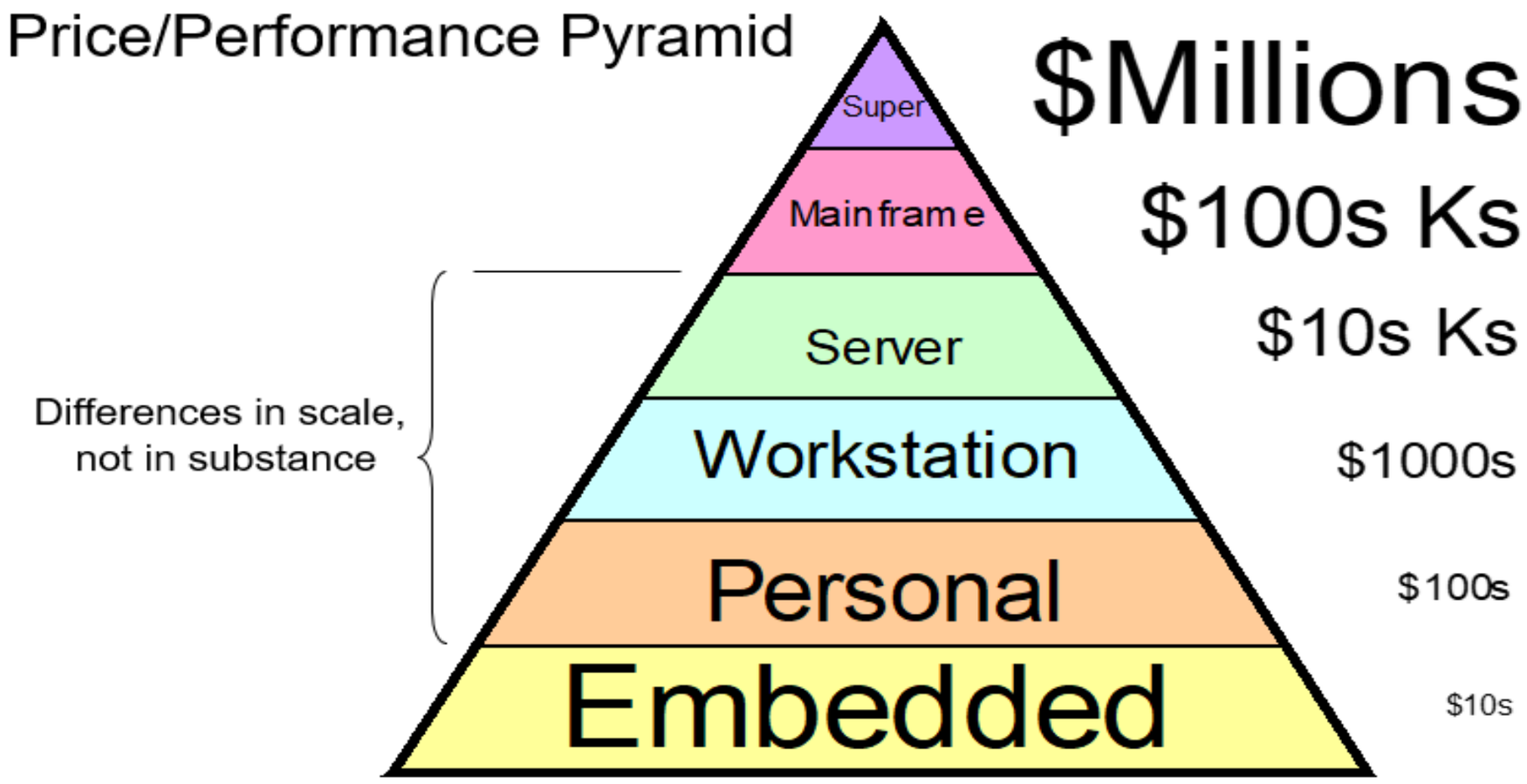


Figure 3.4 Classifying computers by computational power and price range.

طراحی سلسله مراتبی

Systems Prog.

- How does an assembly program end up executing as digital logic?
- What happens in-between?
- How is a computer designed using logic gates and wires to satisfy specific goals?

Digital Design

"C" as a model of computation

Programmer's view of how a computer system works

Architect/microarchitect's view: How to design a computer that meets system design goals. Choices critically affect both the SW programmer and the HW designer

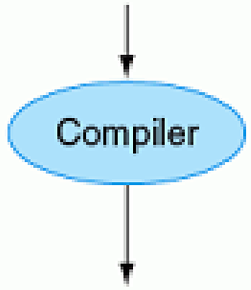
HW designer's view of how a computer system works

Digital logic as a model of computation

عملیات در کامپیوتر

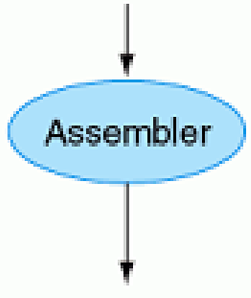
High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```



Assembly language program (for MIPS)

```
swap:
muli $2, $5,4
add  $2, $4,$2
lw   $15, 0($2)
lw   $16, 4($2)
sw   $16, 0($2)
sw   $15, 4($2)
jr   $31
```



Binary machine language program (for MIPS)

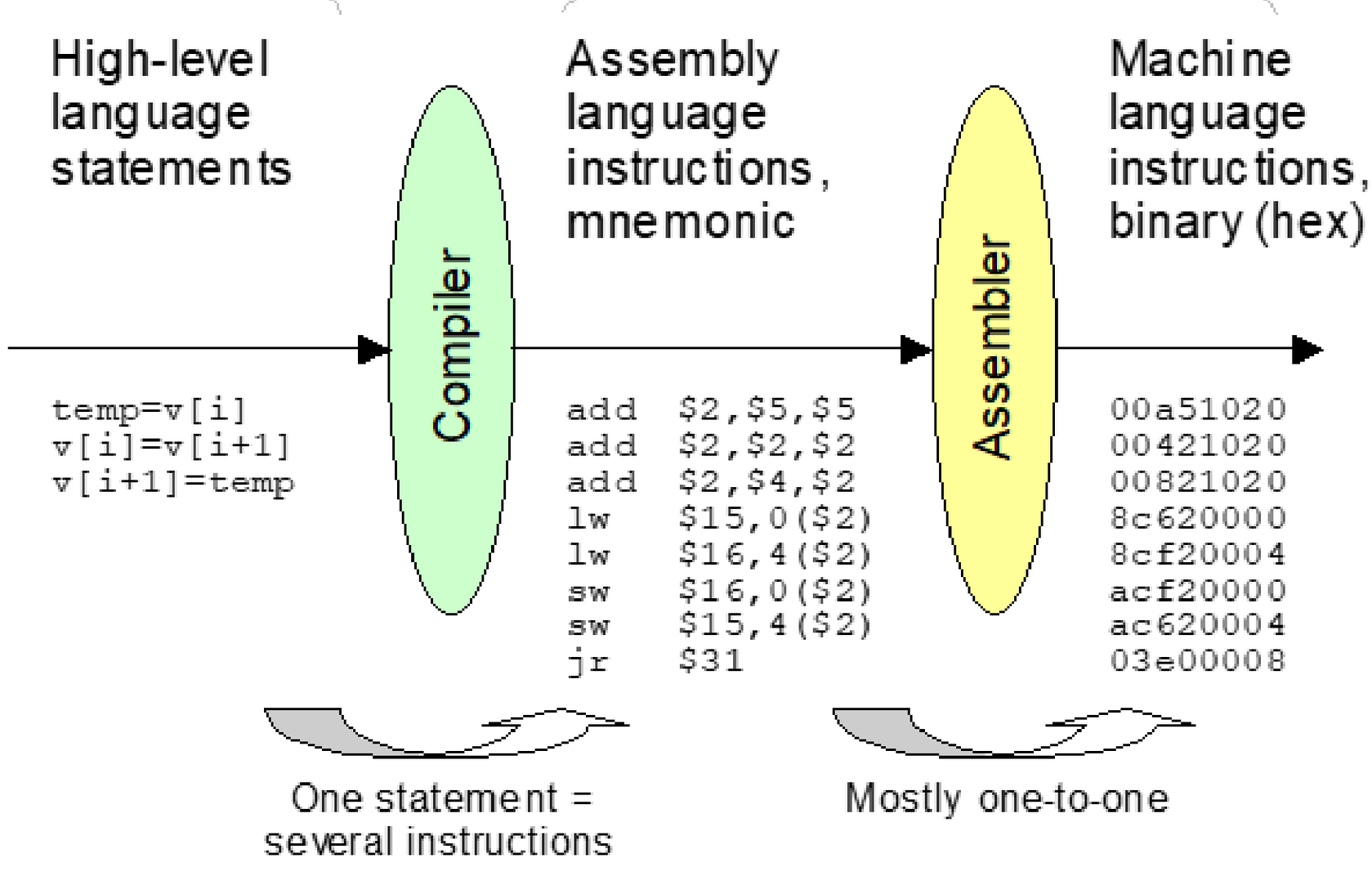
```
00000000101000010000000000011000
00000000000011000000110000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

عملیات در کامپیوتر

High- vs Low-Level Programming

More abstract, machine-independent;
easier to write, read, debug, or maintain

More concrete, machine-specific, error-prone;
harder to write, read, debug, or maintain



طراحی کامپیوتر

- The user of a computer can control the process by means of a program.

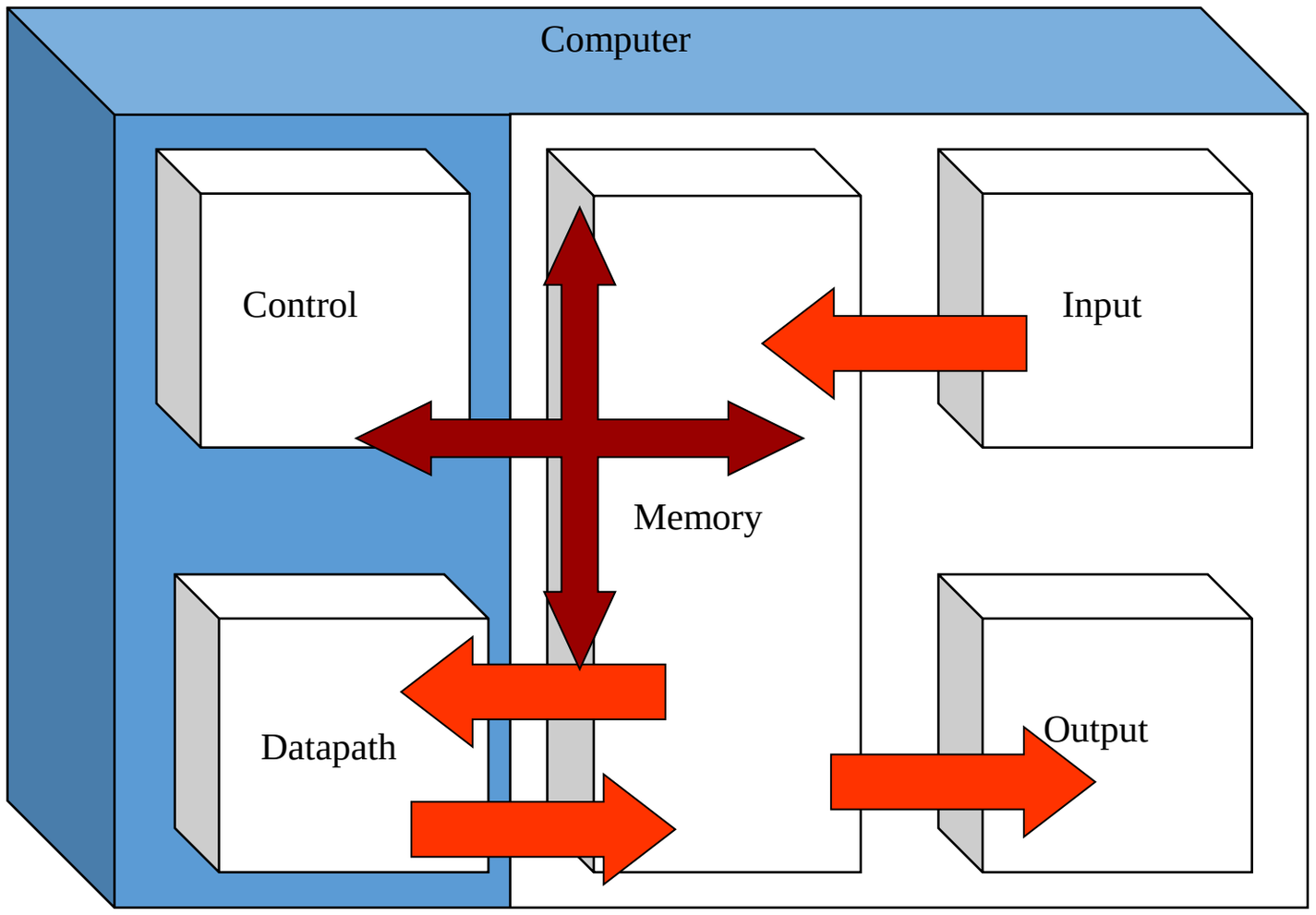
- A program is a set of instructions that specify the **operations**, **operands**, and the **sequence** by which processing has to occur.

- A computer instruction is a binary code that specifies a sequence of microoperations for the computer.

- Instruction codes together with data are stored in memory.

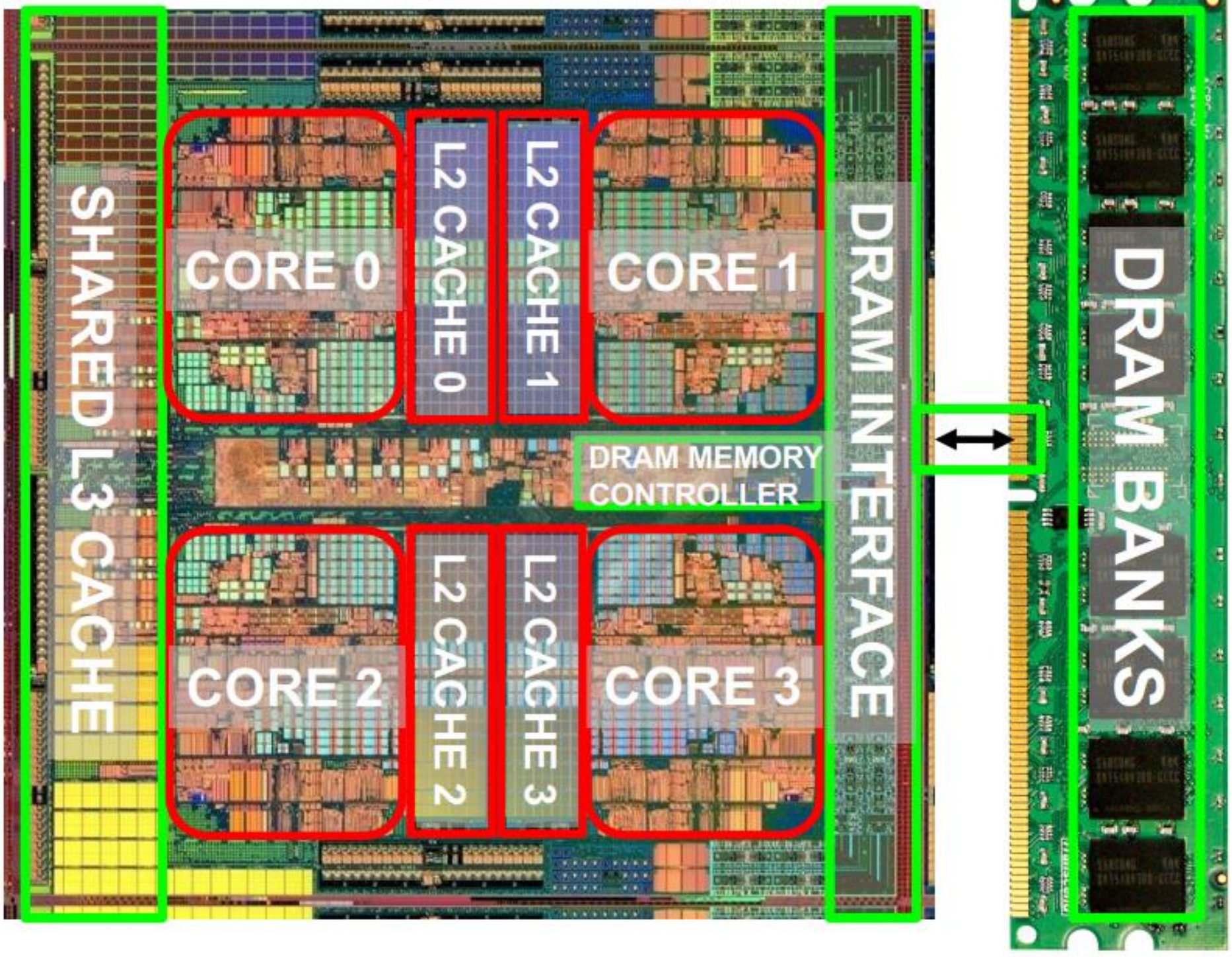
اجزای کامپیوتر

- ورودی
- خروجی
- حافظه
- واحد کنترل
- واحد مسیرداده



اجزای کامپیوتر

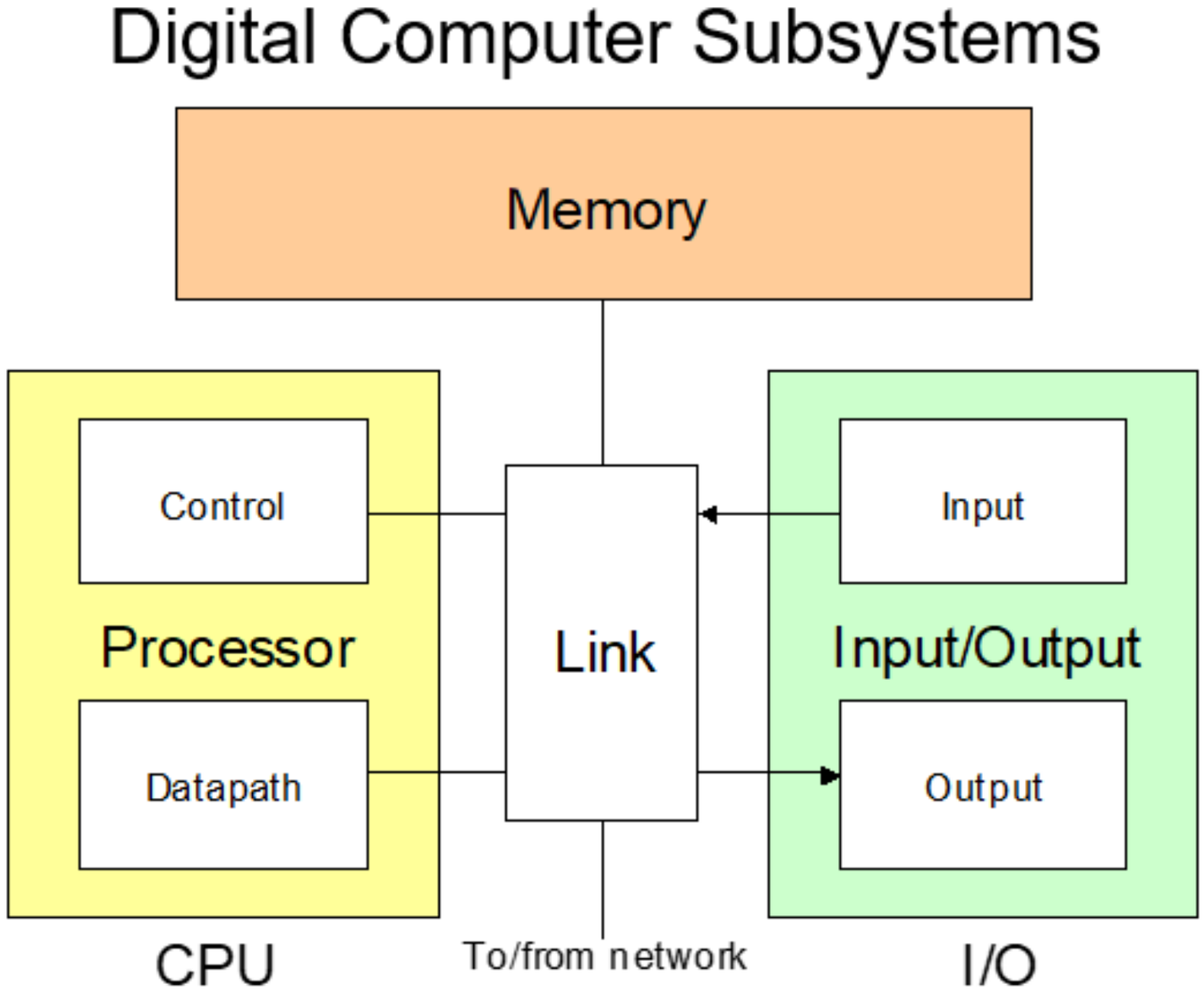
Multi-Core Chip



*Die photo credit: AMD Barcelona

اجزای کامپیوتر

- The internal organization of a digital system is defined by the sequence of operations it performs on data stored in its memory (registers).



Operations vs Microoperations

- An operation is a part of an instruction stored in the memory. It is a binary code that tell the computer to perform a specific operation
- To perform the operation, a sequence of control signals are issued to initiate a set of microoperations
- A sequence of microoperations are issued for the hardware implementation of the operation
- Operation is called Macrooperations because it specifies a set of microoperations

Recap

- Computer
  - A general-purpose device that can be programmed to carry out a set of arithmetic or logical operations (wiki)
  - A system that can execute bunch of instructions

- Goal: designing an efficient digital system, which is a computer → but, extremely simple!

- The first step is to determine the ISA

- Let’s learn what is ISA

Instruction Set Architecture

- Machine “words” and its “vocabulary”
  - To command the computer, you need to speak its language and the instructions are the words of a computer’s language and the instruction set is basically its vocabulary
  - Unless you know the vocabulary and you have a very good vocabulary, you cannot gain good benefits out of the machine
  - The only way that you can talk to your machine is through the ISA. This gives you an idea of the interface between the hardware and software

Instruction Set Architecture

- ISA is the portion of the machine which is visible to either the assembly language programmer or a compiler writer or an application programmer
- The instruction set architecture is the specification of what the computer can do and the machine has to be fabricated in such a way that it will execute whatever has been specified in your ISA
- ISA Specifies
  - Instructions
  - Data types
  - Registers
  - Memory access and addressing modes
  - Input/output
  - Interrupt

Instruction Set Architecture

- ISA is the starting point for designing a new machine

- uArch
  - The internal structure of the processor
  - The way a given ISA is implemented

- ISA
  - Can be implemented with different uArch
  - Different goals (performance, power, cost, …)

- Computer architecture
  - ISA + uArch

ISA Specifies

- Instructions
  - Instruction length

A hardware for a variable number of operands is more complicated than hardware for a fixed number. This situation illustrates the first of three underlying principles of hardware design:
Design Principle 1: Simplicity favors regularity
[Patterson]

- Unlike programs in high-level languages, the operands of arithmetic instructions are restricted;
- They must be from a limited number of special locations built directly in hardware called registers.
- Registers are primitives used in hardware design that are also visible to the programmer when the computer is completed, so you can think of registers as the bricks of computer construction.

- Instruction Format

ISA Specifies

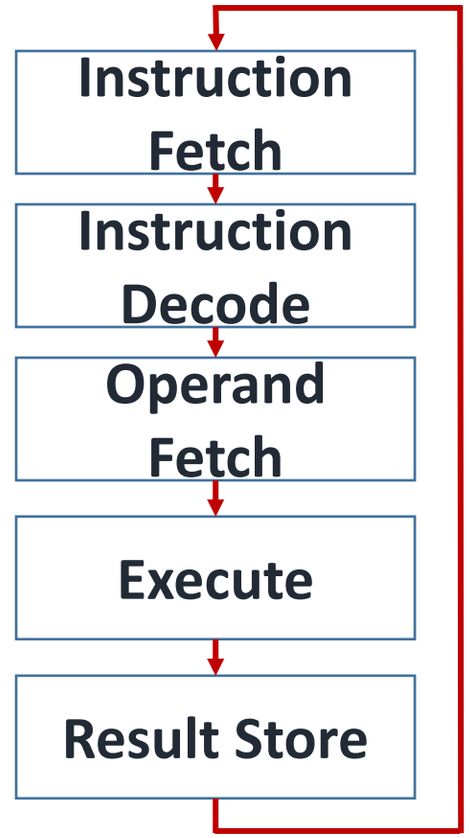
- Data types
  - Integer, single precision float, double precision float, character
  - Byte, word, double word, quad word, etc
- Registers
  - User-accessible Registers
  - Data registers (integer, floating point)
  - Address registers
  - General purpose registers
  - Control and status registers
  - …

ISA Specifies

- Memory Access and Addressing
  - Memory access
  - Only load and store
  - Otherwise
- Addressing modes
  - Immediate
  - Register
  - Direct
  - Memory Indirect
  - Register indirect

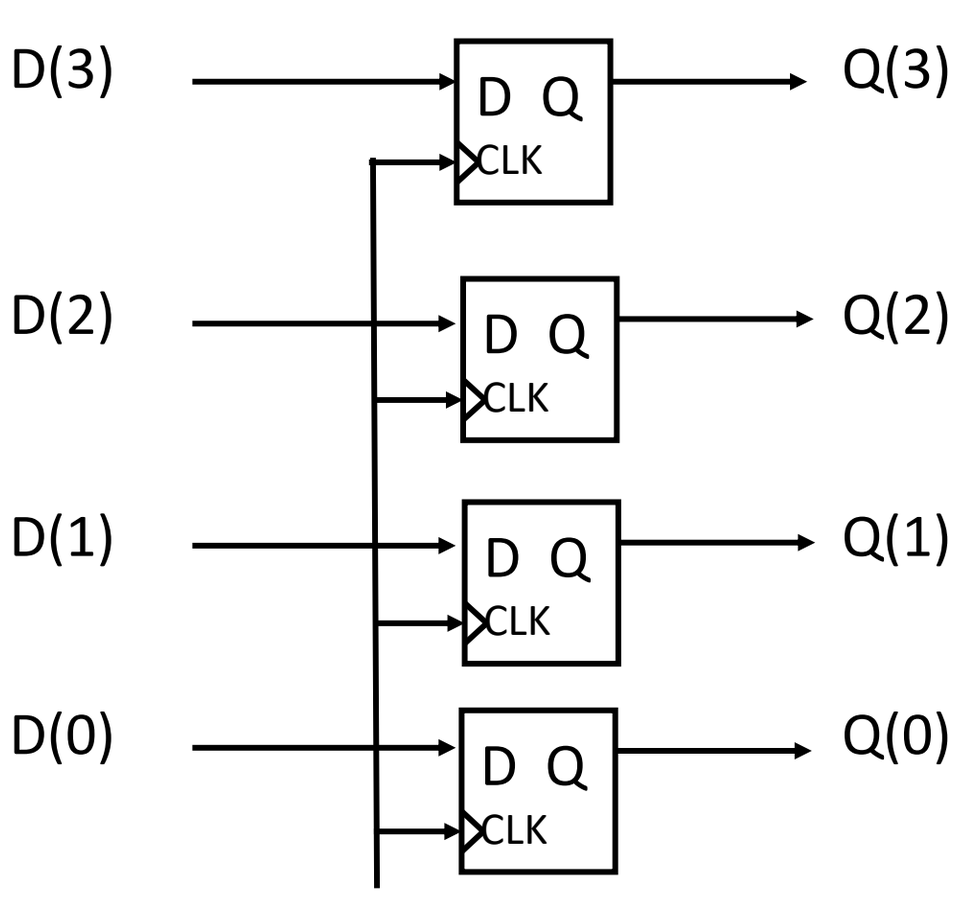
Von-Neuman Model

- Stored program computer
  - instructions and data stored in same memory
- Sequential instruction processing
  - Instructions fetched one by one from memory
  - Program counter incremented in each step



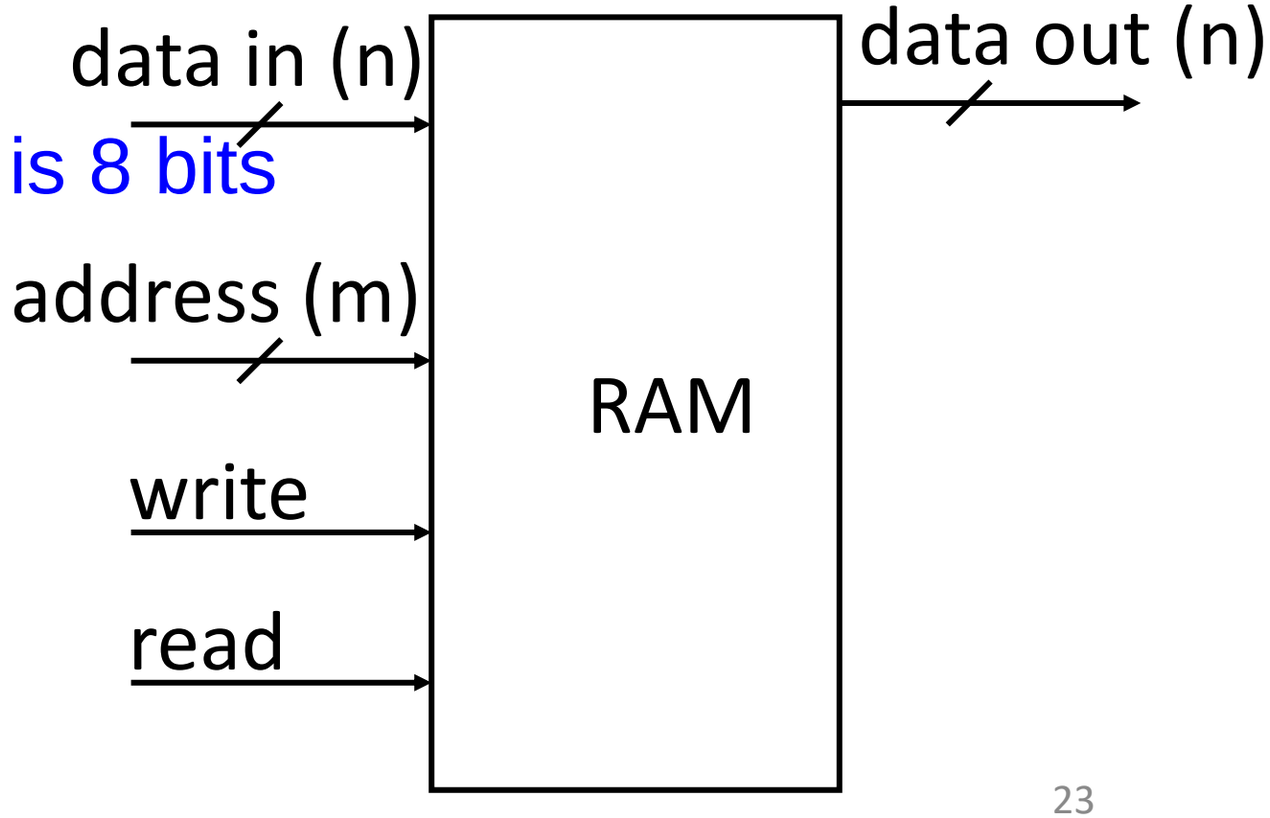
Register Definition

- Register
  - In typical nomenclature, a register is a name for a collection of flip-flops used to hold data (i.e. std_logic_vector)
  - Registers are the fastest place to hold data in a computer, so we want to use them as much as possible.

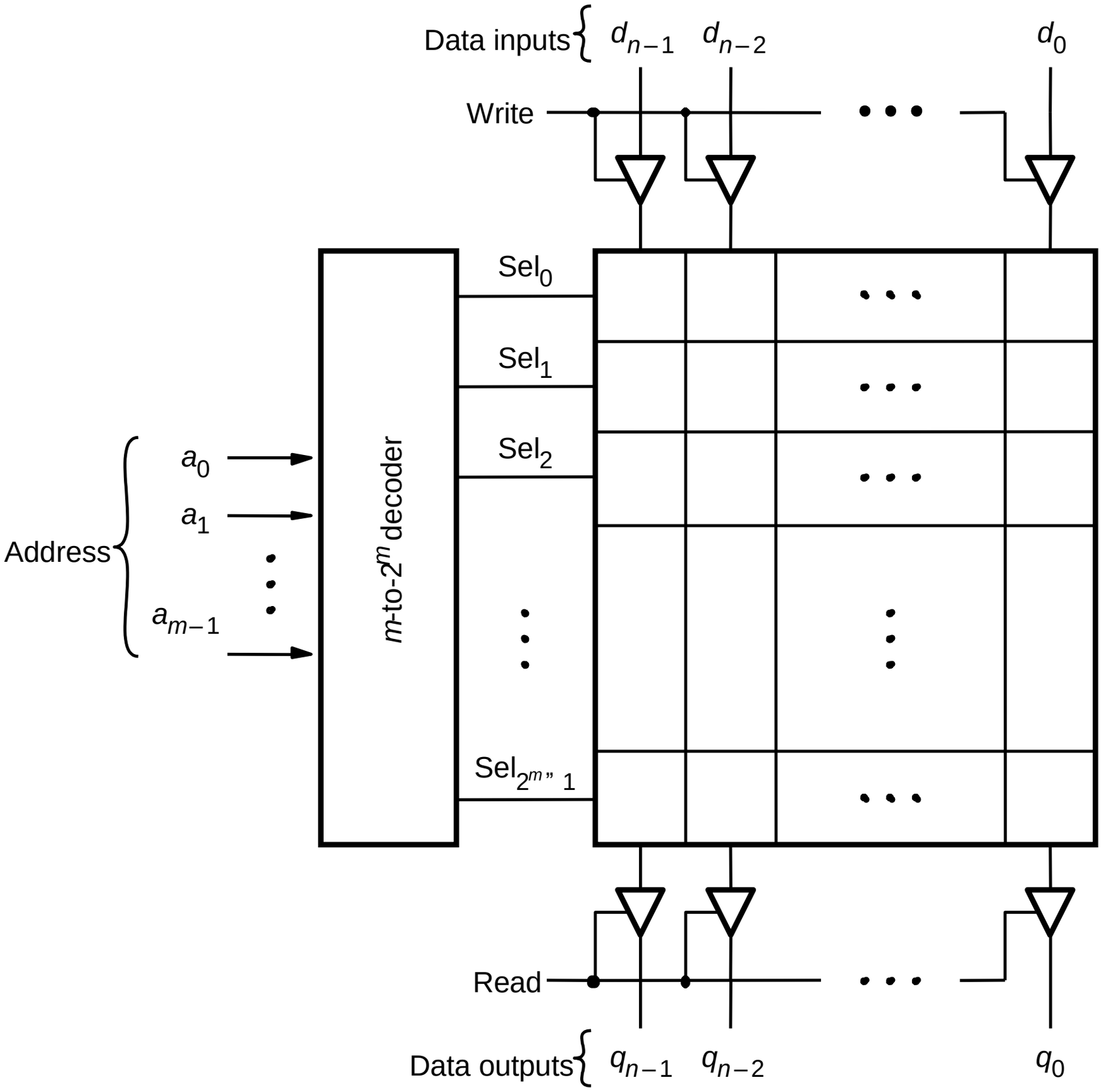


Memories (Random Access Memory)

- More efficient than registers for storing large amounts of data
- Can read and write to RAM
- Addressable memory
- Can be synchronous (with clock) or asynchronous (no clock)
- SRAM dimensions are:
  - (number of words) x (bits per word) SRAM
- Address is m bits, data is n bits
  - 2^m x n-bit RAM
- Example: address is 5 bits, data is 8 bits
  - 32 x 8-bit RAM
- Write
  - Data_in and address are stable
  - Assert write signal (then de-assert)
- Read
  - Address is stable
  - Assert read signal
  - Data_out is valid

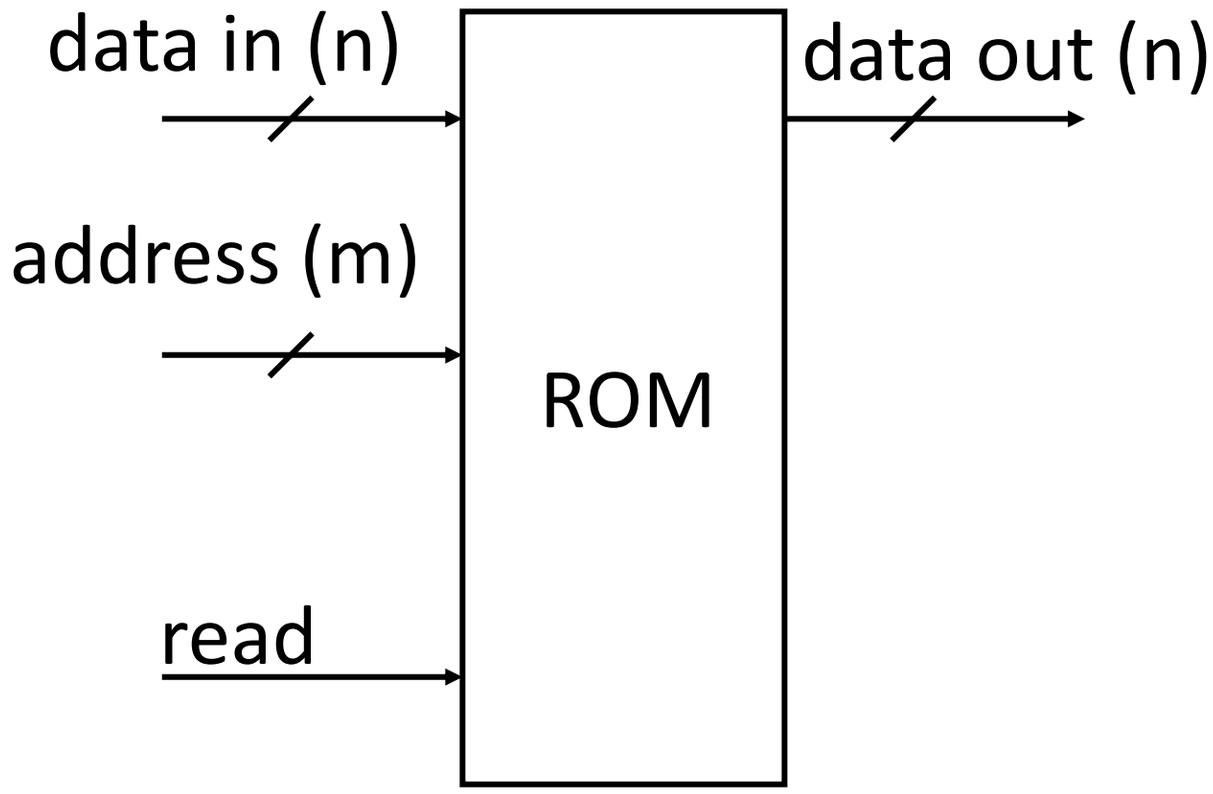


Memories (Random Access Memory)



Read Only Memory (ROM)

- Similar to RAM except read only
- Addressable memory
- Can be synchronous (with clock) or asynchronous (no clock)



Read Only Memory (ROM)