university of groningen

Computer Architecture 2023-24 (WBCS010-05)

Lecture 5: The von Neumann Model and The LC-3

Reza Hassanpour
r.zare.hassanpour@rug.nl

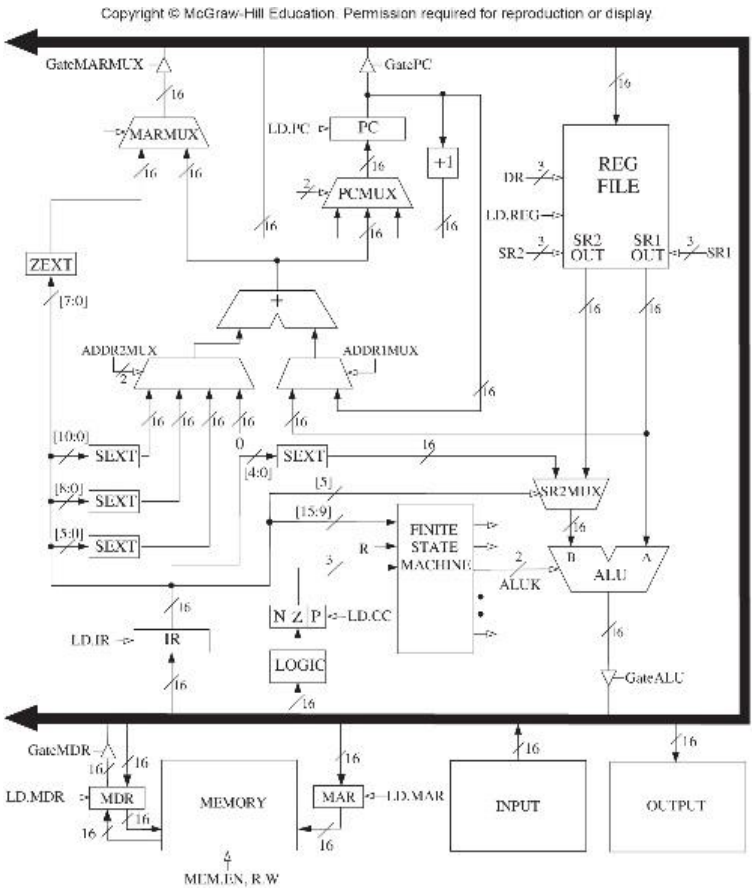
Introduction

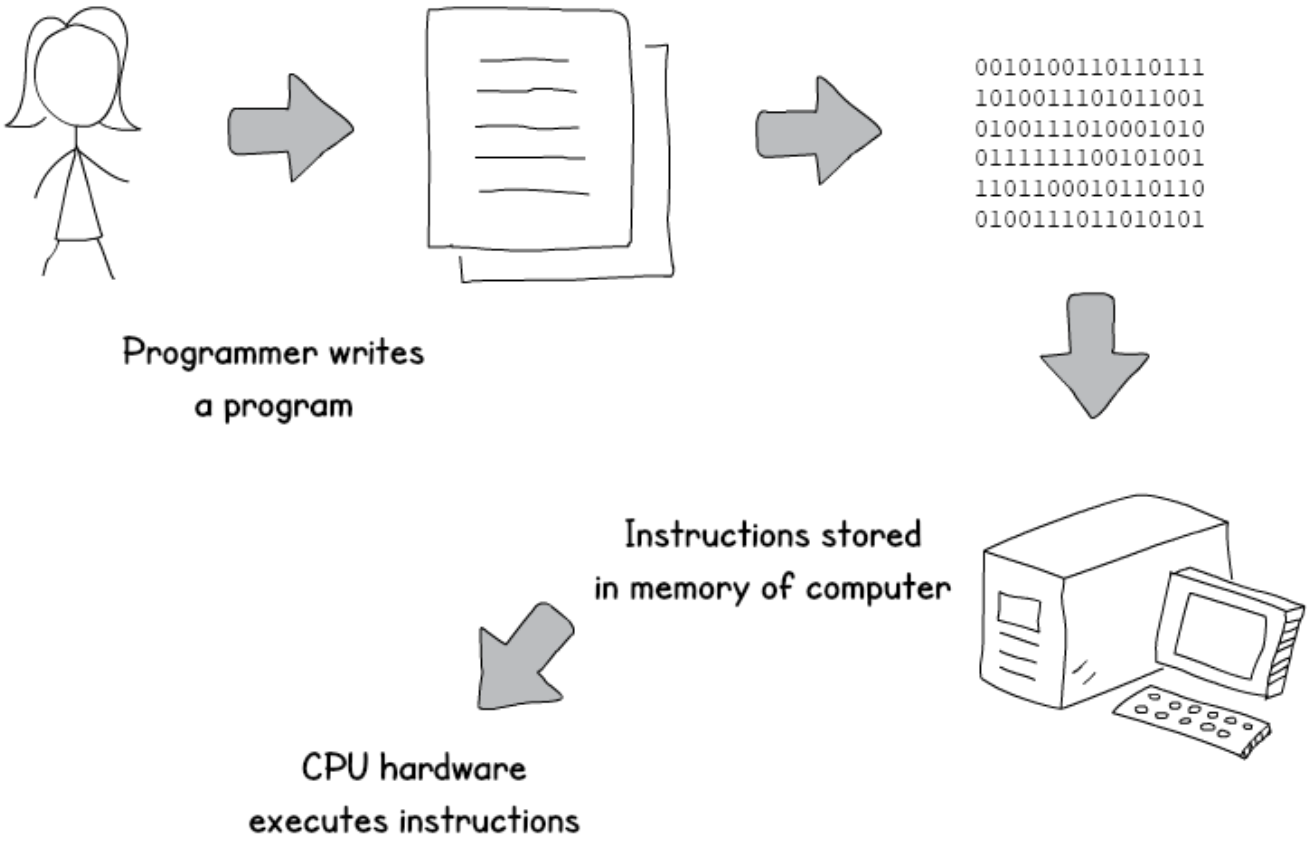
- Digital Computers:
  - We use the term digital computer to refer to a device that performs a sequence of computational steps on data items that have discrete values.

- Combinational Logic:
  - Digital logic that implement Boolean logic and produce specified outputs from certain inputs.
- Sequential Logic:
  - Digital logic in which the output depends upon present as well as the state of the circuit.

university of groningen

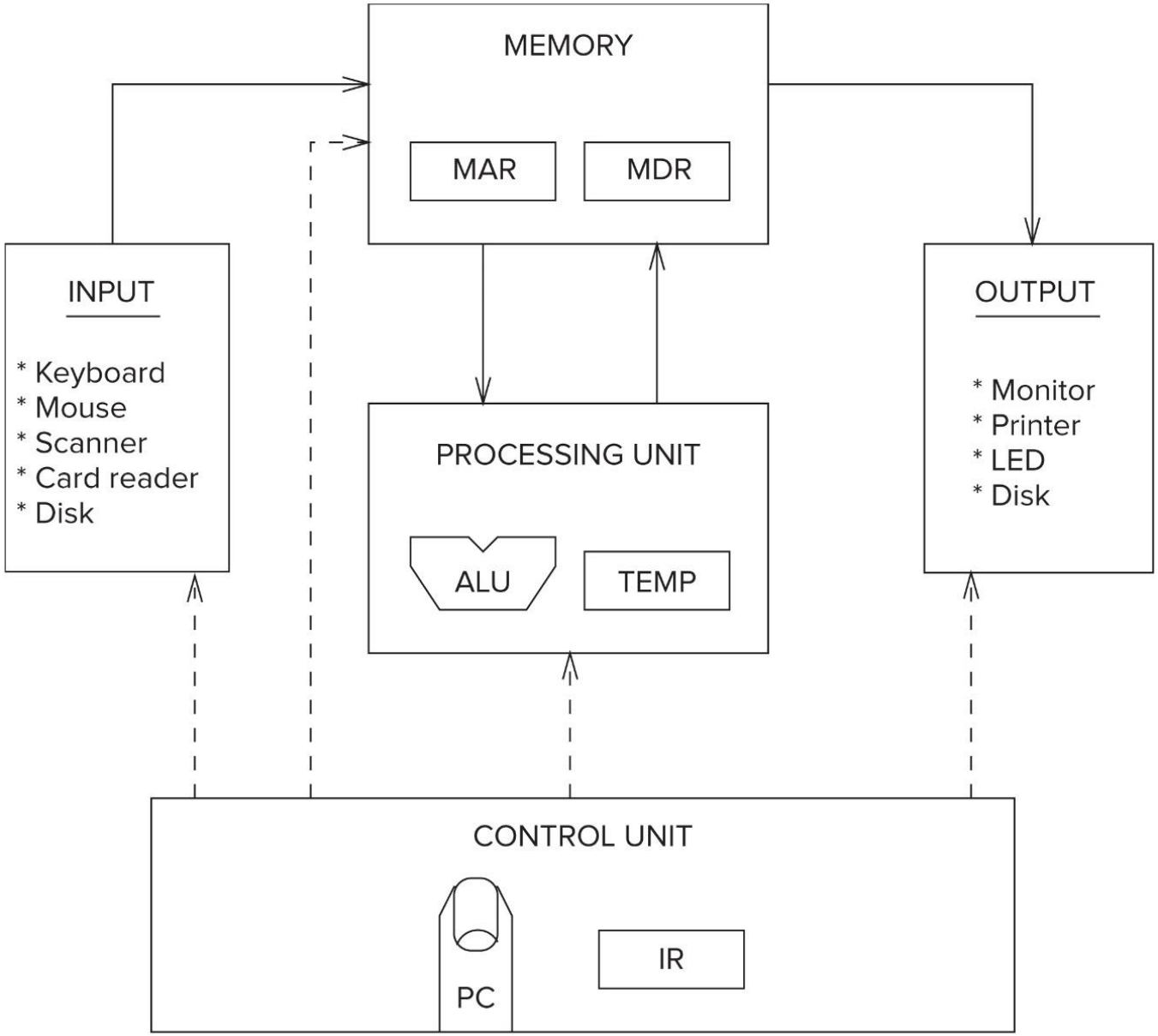
Solving a Problem using a Computer





university of groningen

Computer Organization: von Neumann Model 1

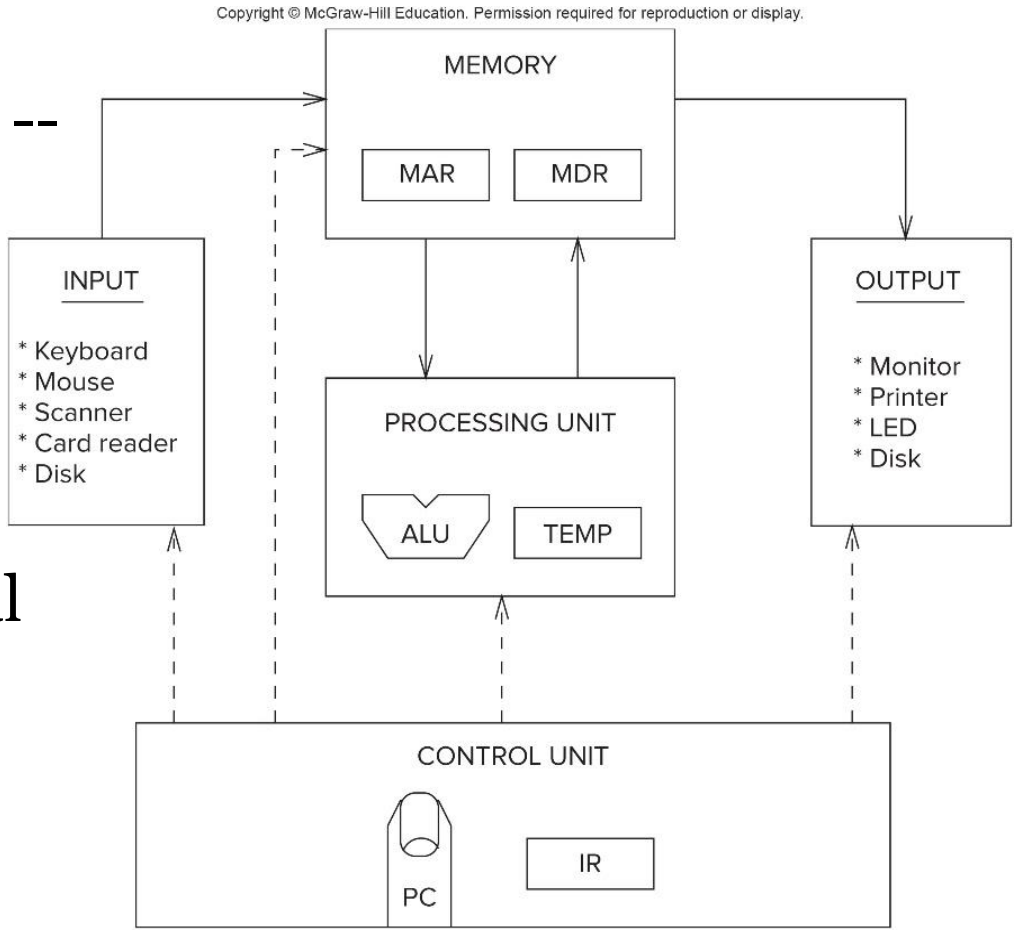


Fundamental model of a computer proposed by John von Neumann in 1946

university of groningen

Computer Organization: von Neumann Model 2

- Memory: stores data and instructions during program execution
- Processing Unit: transforms data -- logic circuits, temporary storage
- Control Unit: orchestrates fetching of instructions (from memory) and execution of instructions (in processing unit)
- Input: receives data from external environment
- Output: sends data to external environment

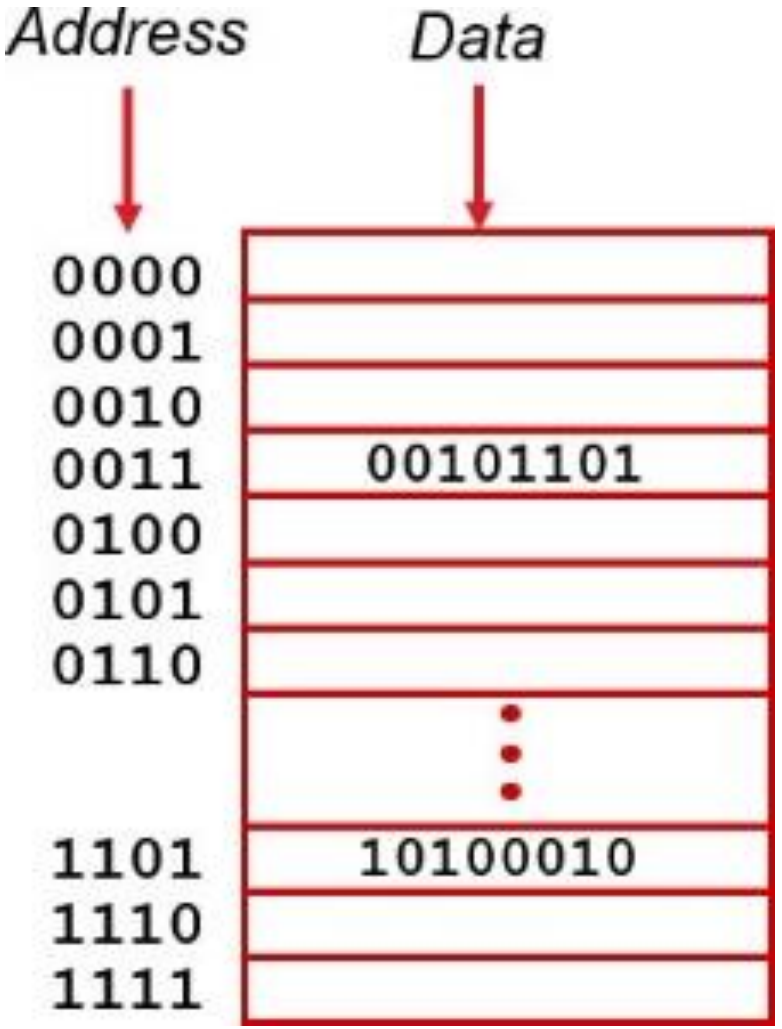


Processing Unit + Control Unit = CPU: Central Processing Unit

Memory

- Address space = number of locations 2^n
- 2^n locations means n-bit address

- **Addressability** = number of bits in each location
- Basic operations
- **LOAD** data from memory to CPU
- **STORE** data from CPU to memory



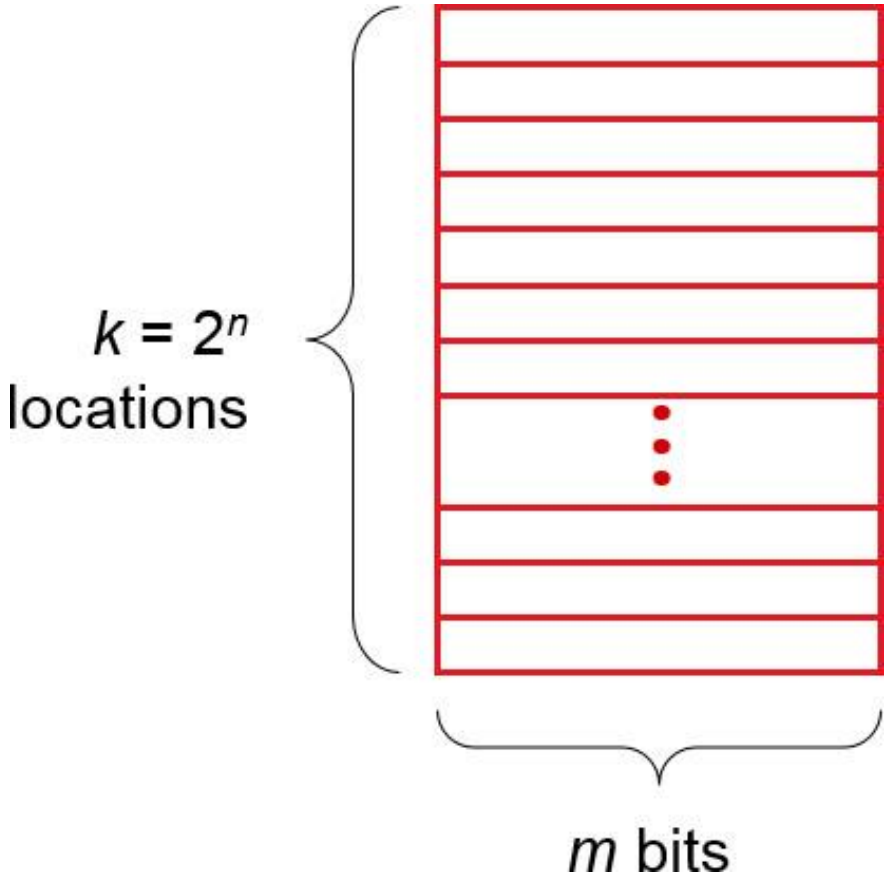
university of groningen

The Concept of Memory 1

- Now that we can store a bit, we can build a memory: a $k \times m$ array of stored bits

- **Address Space:**
- number of locations (usually a power of 2)

- **Addressability:**
- number of bits per location (for example, byte-addressable)



The Concept of Memory 2

- We have an array of $k = 2^n$ memory locatios.
- Each location stores an m-bit value. The value m is known as the addressability or the width of the memory.
- Each location has a unique identifier, known as an address. Addresses range from 0 to $k - 1$. The number of addresses (k) is known as the address space of the memory.
- WE (write enable) indicates whether we want to retrieve the stored value (WE = 0) or store a new value (WE = 1).

Memory Operation	Inputs	Output
READ	Address (n bit) WE = 0	Data (m bits) stored at specified address
WRITE	Address (n bits) New Data (m bits) WE = 1	New Data (m bits)

The Concept of Memory 3

- Characteristics of memory systems:
  - Location,
  - Capacity,
  - Unit of transfer,
  - Access method,
  - Physical characteristics.

Memory Location

- Processor memory: The registers that are included within the processor and termed as processor memory.
- Internal memory: It is often termed as main memory and resides within the computer.
- External memory: It consists of peripheral storage devices such as disk and magnetic tape that are accessible to processor via I/O controllers.

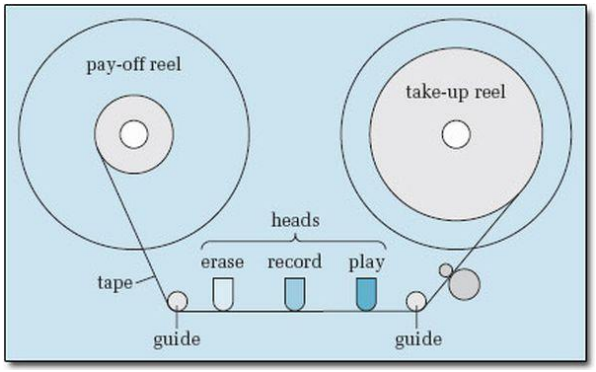
Memory Capacity and Unit of Transfer

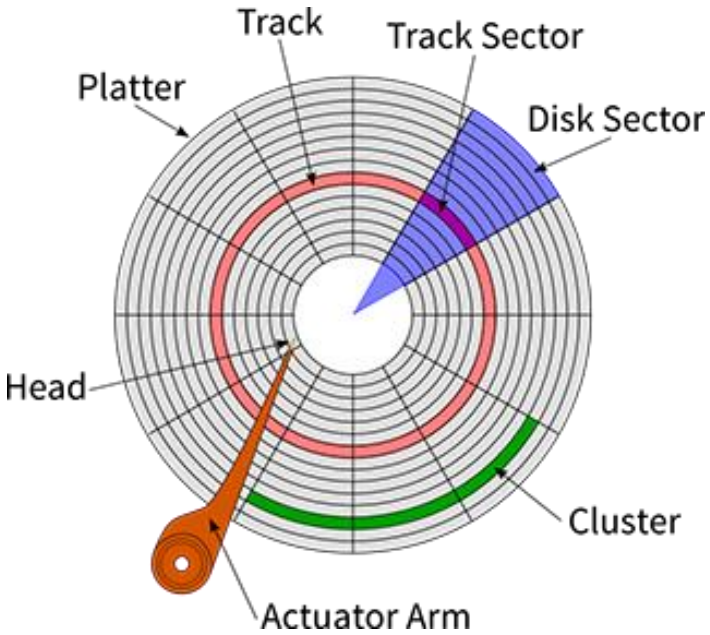
- Capacity is expressed in terms of the number of words or bytes.
- Word Size: Common word lengths are 8, 16, 32 bits etc.
- Unit of Transfer: The unit of transfer is equal to the number of data lines into and out of the memory module.
- The lines into or out of the memory are called memory bus

university of groningen

Access Type (1)

- Sequential access:
  - Start from the beginning and read through a specific linear sequence.
  - This means access time of data unit depends on position of records (unit of data) and previous location. e.g. tape
- Direct Access:
  - Individual blocks of records have unique address based on location. May include a sequential search. e.g. disk





Access Type (2)

- Random access:
  - Any location can be selected out randomly and directly addressed and accessed. e.g. RAM
- Associative access:
  - Random access type of memory that enables one to look for a specified match, and to do this for all words simultaneously. e.g. cache

Physical Characteristics

- Semiconductor: RAM, ROM
- Magnetic: Disk & Tape
- Optical: CD & DVD

- Data persistency:
  - Decay
  - Volatility
  - Erasable

Random Access Memory Types

Two basic kinds of RAM (Random Access Memory)

Static RAM (SRAM)

- Fast, maintains data as long as power applied (typically 6 transistors)

Dynamic RAM (DRAM)

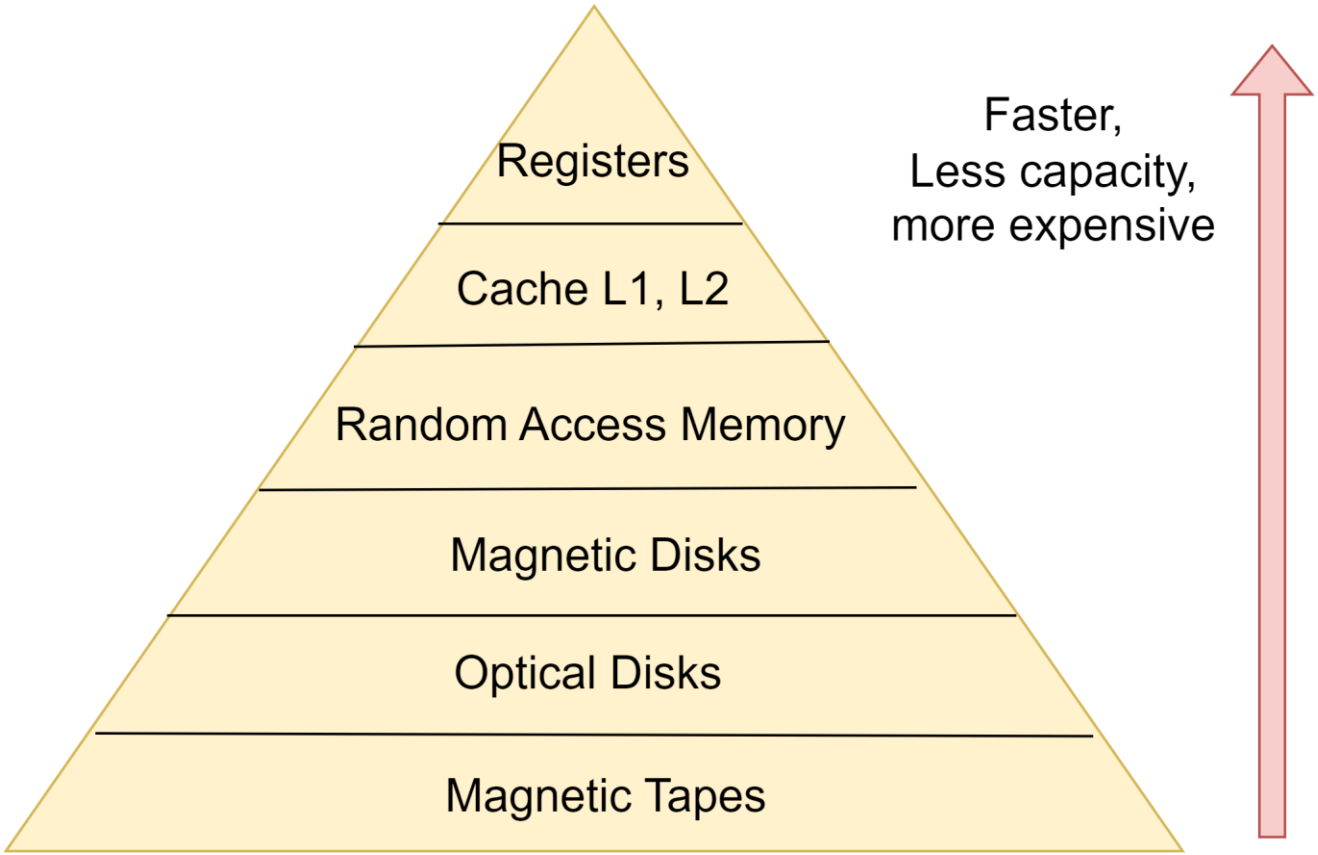
- Slower but denser (single transistor), bit storage decays in ms – must be periodically refreshed

But the logical structure is very similar

- Address decoder / word select line / word write enable

university of groningen

Memory Hierarchy



BUS

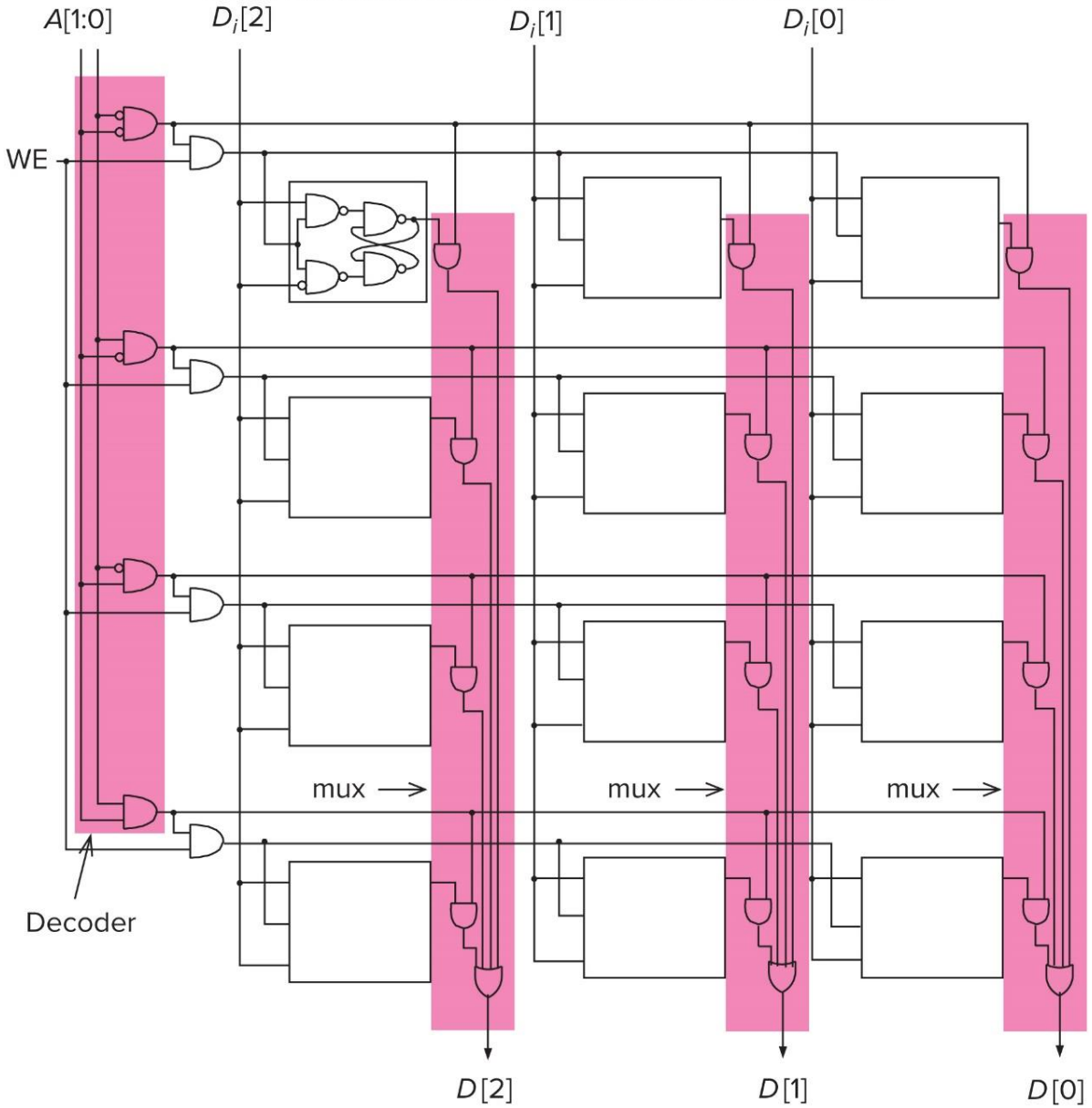
- A set of wires that enables communication between multiple devices.
- Only one device can use the bus at a time, or the message is garbled.
- Each wire of a bus can contain a single binary digit at any moment. However, a sequence of binary digits may be transferred using a wire over time.
- A bus typically have many parallel wires, and a computer system may contain a number of different buses

Bus Structure

- Bus lines can be classified into one of four groups
  - Data lines
  - Address Lines
  - Control Lines
  - Power
- Data Lines pass data from/to the device (memory)
  - Number of data lines represents width of the memory
- Address lines show the location of source or destination.
  - Width of address bus specifies maximum memory capacity
- Control lines define the operation (read/write/bus request/etc.)

university of groningen

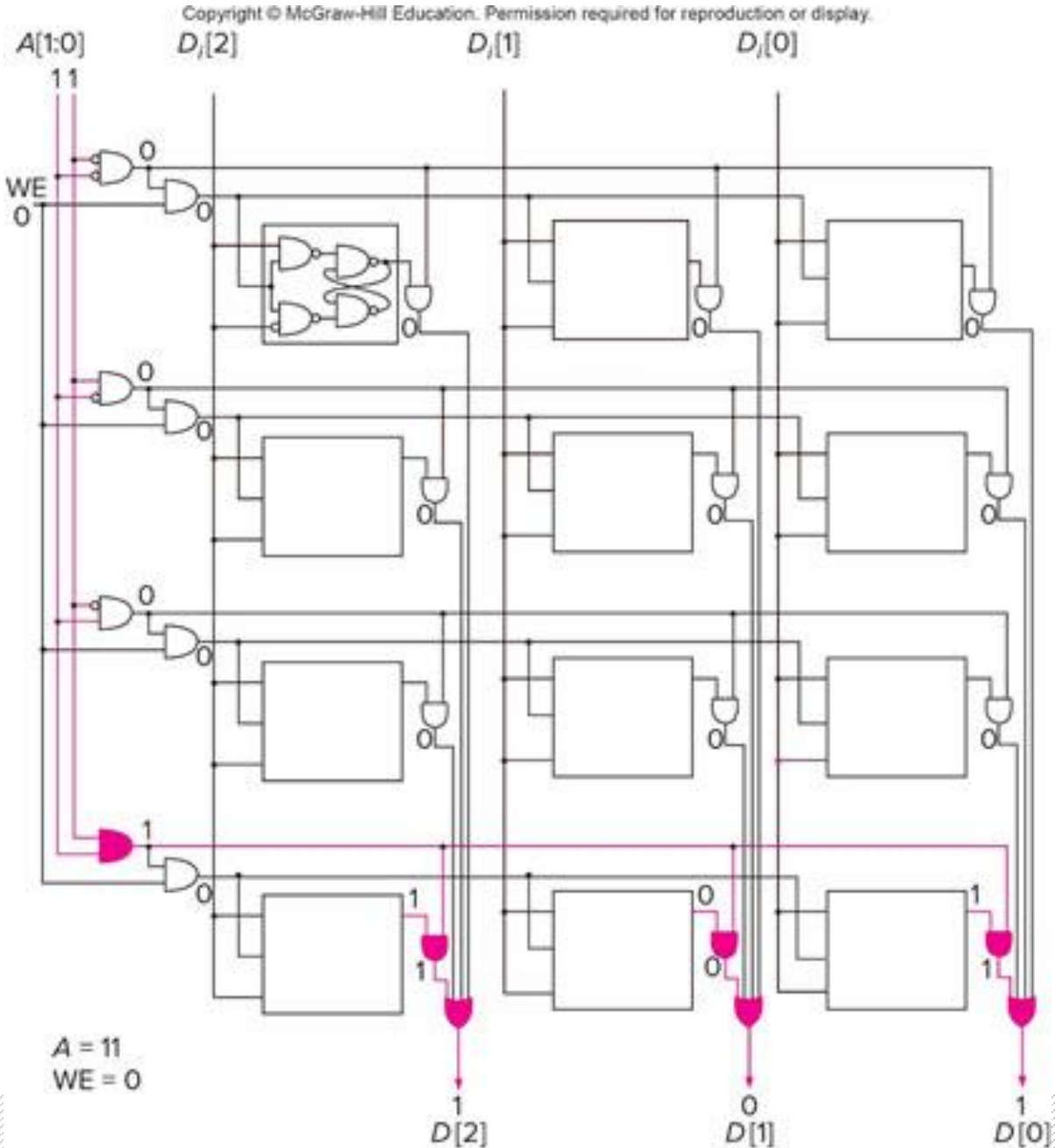
A Memory Circuit



- Bits are stored in an array of D-latches
- An m-bit value in each row
- Address is **decoded** to select a row
- **Mux** is used to output selected bit in each column
- To write, data input at the top and WE changes the latches in selected row

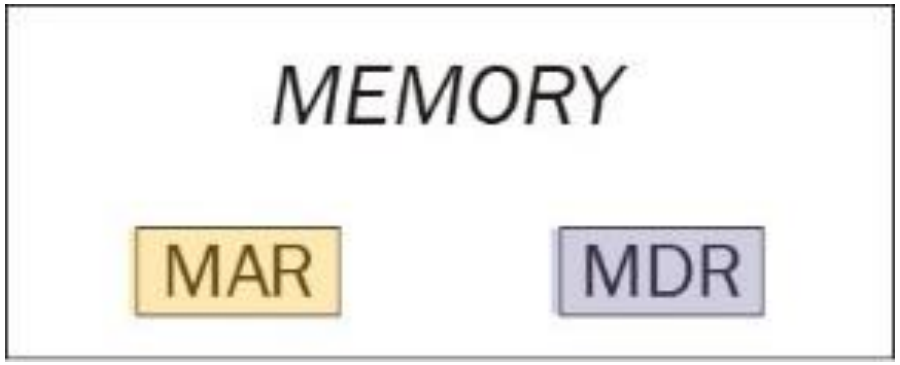
university of groningen

Example: Reading Address 3 of a 4 × 3 memory



Interface to Memory

- Registers used to move data into and out of memory
- MAR = Memory Address Register
- MDR = Memory Data Register



- To LOAD data from memory:

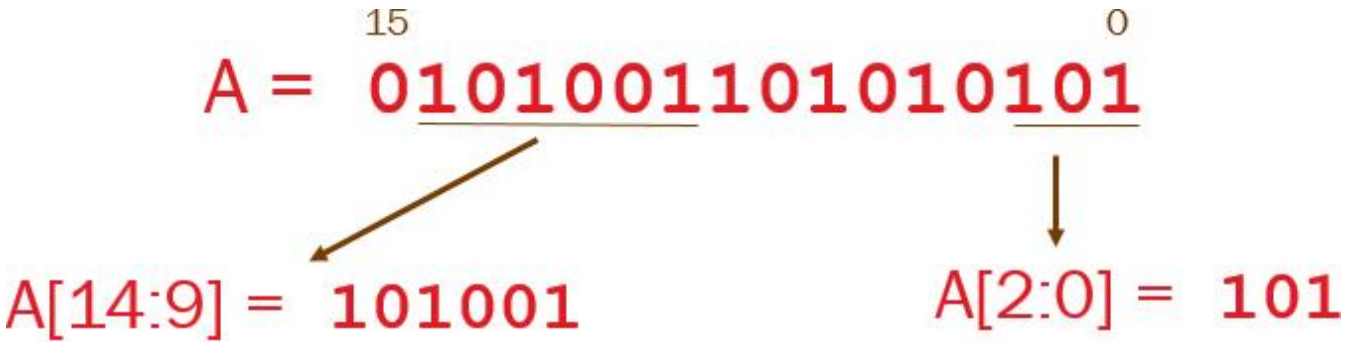
1. Write the address into the MAR
2. Send a "read" signal to memory
3. Read data from MDR

- To STORE data to memory:

1. Write the address into the MAR
2. Write the data into MDR
3. Send a "write" signal to memory

Notation: Multi-bit values

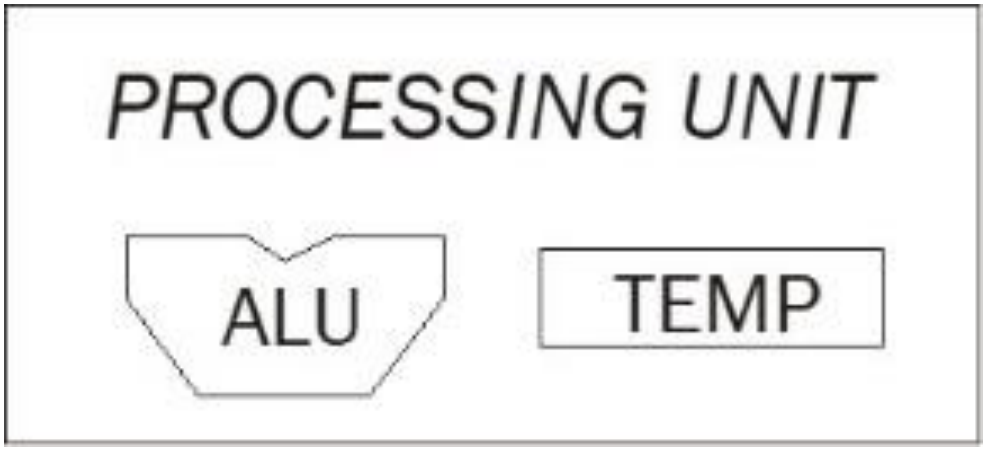
- **Convention:** Bits are numbered from right to left
- Bit 0 is the least significant (all the way to the right)
- Bit n-1 is the most significant (all the way to the left)
- Use brackets to denote a range of bits:
  A[left:right]



Processing Unit

› **Functional Units**

- Perform data transformations
- ALU = Arithmetic / Logic Unit
  add, subtract, AND, ...



› **Registers**

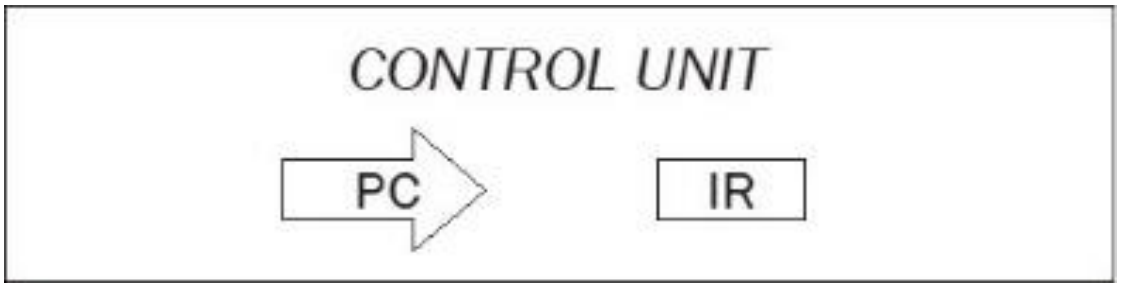
- Temporary storage of data

› Word Length

- ALU operations are usually performed on words
- Typical word lengths:
  64 bits (Intel Core), 32-bit (Intel Atom), **16 bits (LC-3)**

Control Unit

- Orchestrates the rest of the computer to carry out program instructions

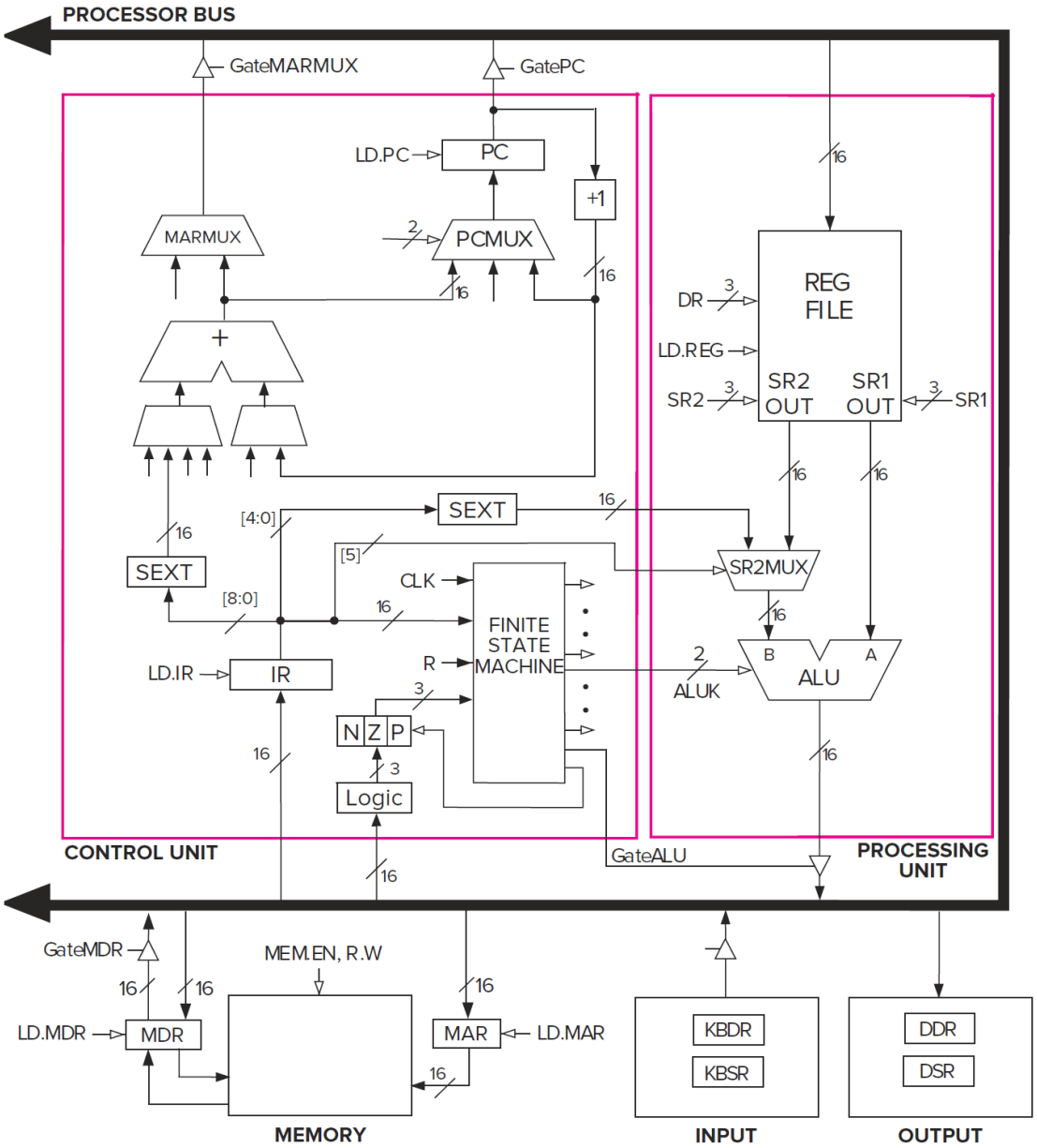


- Instruction Register (IR) holds the current instruction
- Program Counter (PC) holds the memory address of the next instruction to be executed. It is known as a "pointer" because it points to a memory location.
- Control unit is a state machine that does the following:

1. Read (fetch) the next instruction from memory (Address is in the PC)
2. Sends control signals to other parts of processor to move / transform data according to the instruction
3. Go to step 1, repeat forever…

university of groningen

Example Processor: LC-3



- Memory = 2^{16} locations, 16 bits in each location
- **ALU:** performs ADD, AND, NOT
  word length is 16 bits
- Instruction length is 16 bits

Instruction

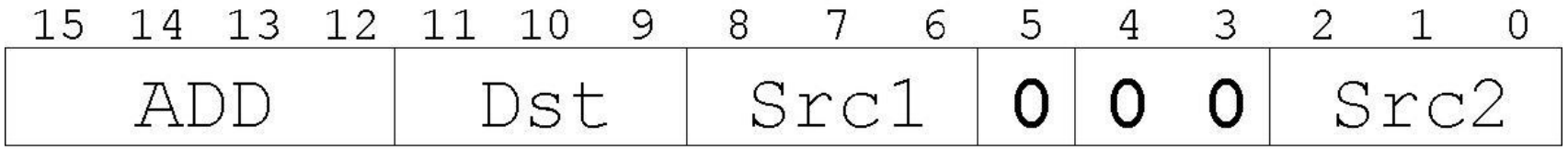
- A computer **instruction** is a binary value that specifies one operation to be carried out by the hardware. It's binary, so it looks just like the data in Chapter 2 -- a collection of bits. The components of the instruction are *encoded* as various bit fields of the binary value.
- Instruction is the **fundamental unit of work**. Once an instruction begins, it will complete its job.

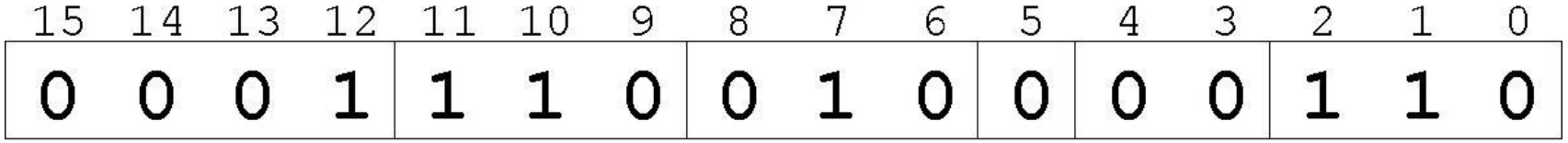
- Two components of an instruction:
- **opcode** specifies the operation to be performed (for example: ADD)
- **operands** specify the data to be used during the operation

- The set of instructions and their formats is called the **Instruction Set Architecture (ISA)** of a computer

LC-3 Instruction

- LC-3 has a four-bit opcode, bits [15:12] -- up to 16 different operations
- LC-3 has **eight** CPU registers. Registers are used as operands for instructions -- each operand requires 3 bits to specify which register
- **Example:** ADD instruction, opcode = 0001





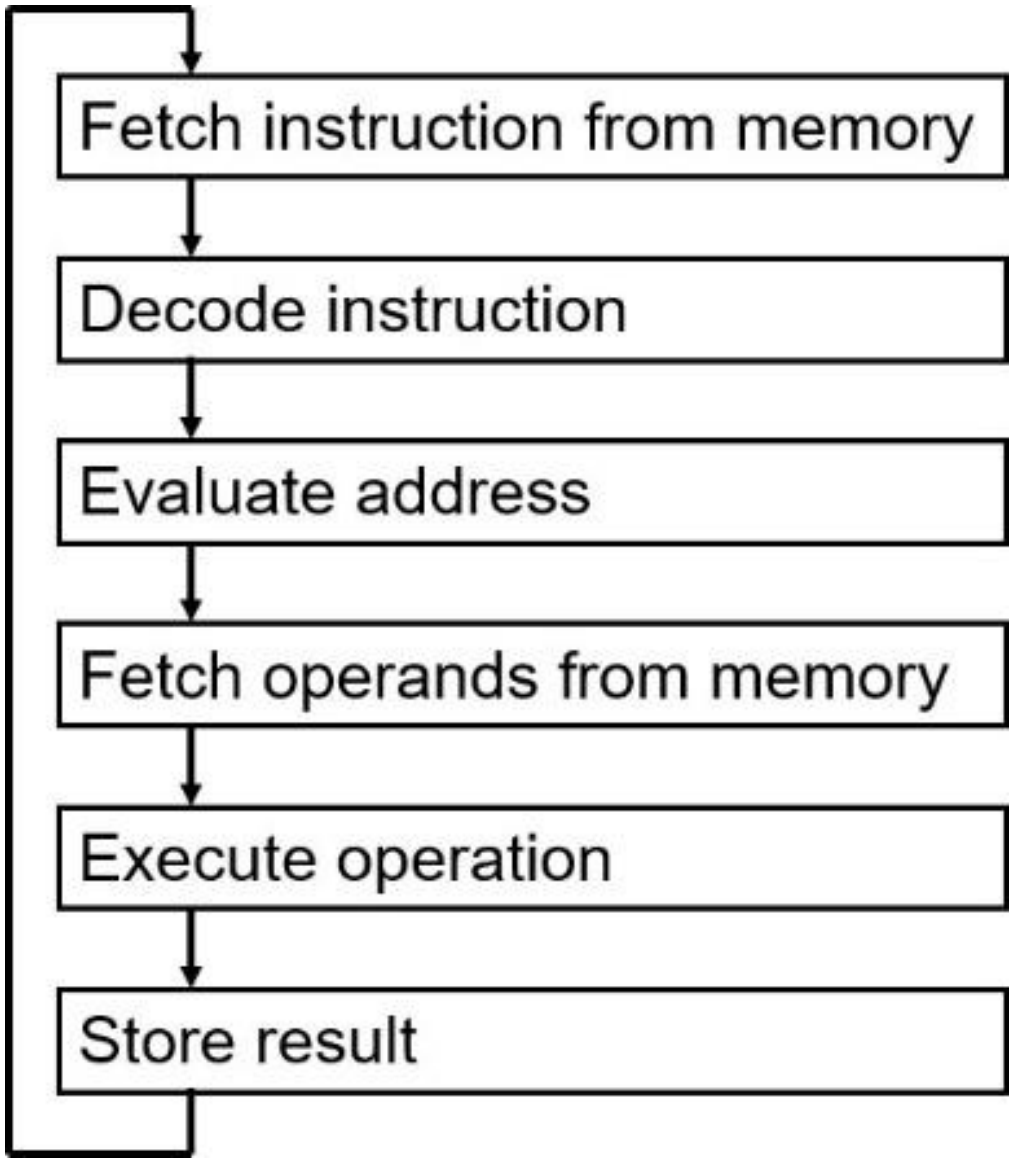
"Add the contents of R2 (010) to the contents of R6 (110), and store the result in R6 (110)."

Types of Instructions

- Instructions are classified into different types, depending on what sort of operation is specified

- **Operate**
- Will perform some sort of data transformation
- Examples: add, AND, NOT, multiply, shift, …
- **Data Movement**
- Move data from memory to a register in the CPU, or
- Move data from a register to a memory location
- **Control**
- Determine the next instruction to be executed

Processing an Instruction



- The diagram shows the **instruction cycle** -- the set of steps that must be performed to execute each instruction
- The Control Unit is responsible for performing this sequence of actions
- Each step may require one or more clock cycles

Instruction Processing: Fetch

› **Fetch**

- PC contains the address of the instruction to be fetched
- The control unit writes the PC value into the MAR to read the instruction stored at that location
- It also **increments the PC** to point to the next consecutive memory location. (It is assumed that the next instruction is stored in the next location in memory. This could be changed when the instruction is executed -- see "control instructions" later in this chapter)
- When the instruction is retrieved from memory (via the MDR), it is copied into the instruction register (IR)

Instruction Processing: Decode

- **Decode**
- Once the instruction is in the IR, the control unit "looks at" the opcode to determine what operation should be performed
- *One possibility*: a 4-bit decoder sets a particular control signal to 1, corresponding to the desired operation
- The opcode also determines how the other instruction bits should be interpreted

- *For example*: for ADD, bits [8:6] specify a register to be used as the first operand of the addition

Instruction Processing: Executing the Instruction

- **Evaluate Address**
- For memory instructions, determine which address to load/store
- **Fetch Operands**
- Get data values from registers and/or memory
- **Execute**
- Perform the necessary operations (for example: add)
- **Store Result**
- Write the result of the computation to a register or to memory

- Not every step is required for every instruction. For example, if there is no memory access, then the Evaluate Address step is not performed

Changing the Sequence of Instructions

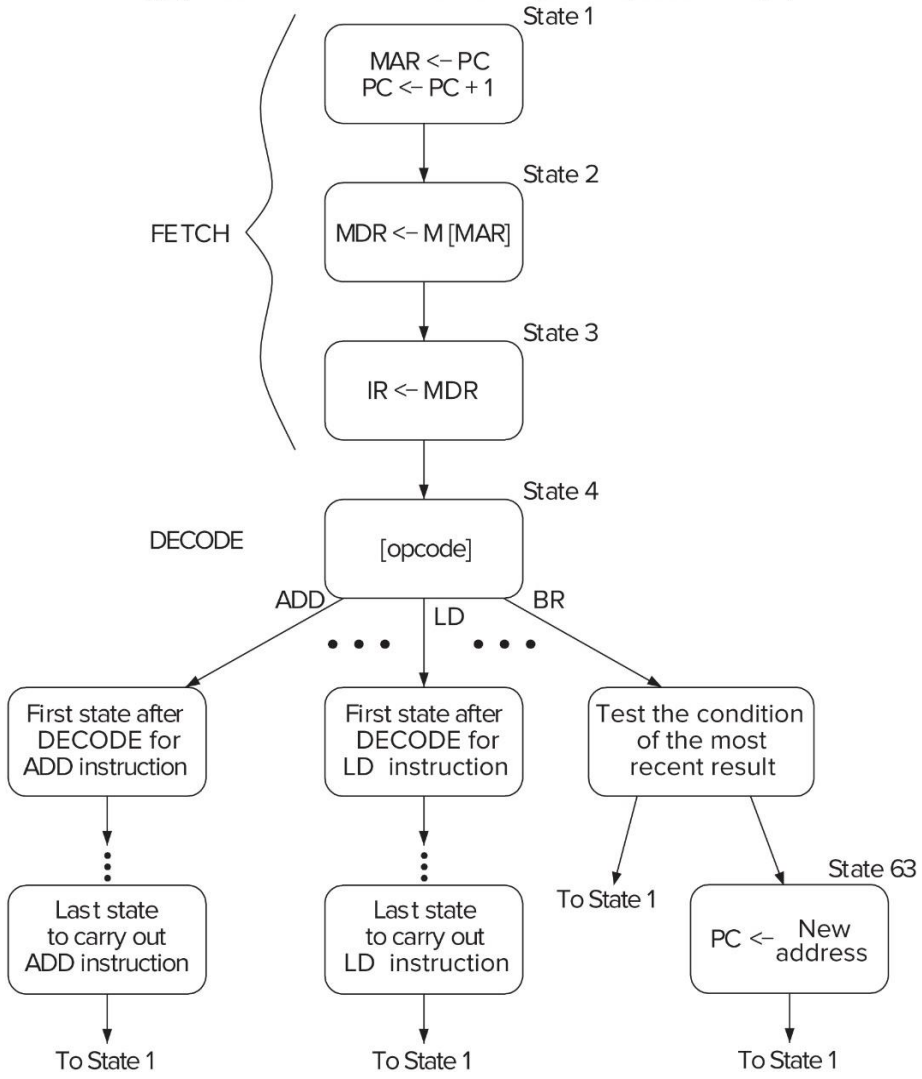
- In the Fetch step, we incremented the PC to point to the next instruction
- Sometimes, we want to execute a different instruction, not the next one in memory. This is the job of a **control** instruction

- A control instruction may change the PC value. In other words, instead of pointing to the next place in memory, a new address can be calculated and stored in the PC register
- This change may be **conditional** – for example: if the last value calculated was zero, write a new value to the PC. These are typically called **branch instructions**.
- The change may be **unconditional** -- it always happens. These instructions are usually called **jump instructions**.
- LC-3 control instructions will be explained in Chapter 5

university of groningen

Control of the Instruction Cycle

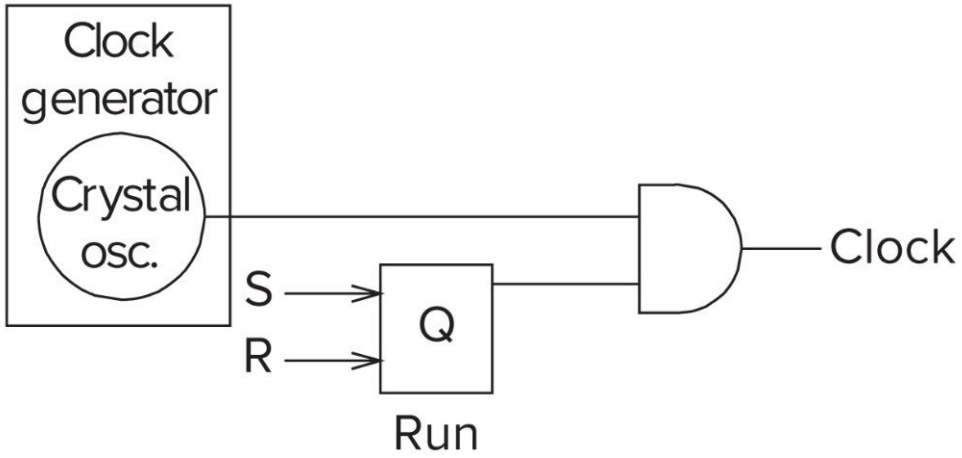
- The control unit is a state machine. A partial state diagram is shown here
- The instruction cycle starts with State 1, which initiates the Fetch phase. This is done for every instruction
- After the Decode phase (State 4), the next state will depend on the specific opcode



Appendix C has a complete description of the LC-3 state machine.

Stopping Execution

- The control unit's state machines keeps running, as long as the clock signal keeps going up and down
- The program can halt execution by turning off the clock signal



- In the circuit above, we "gate" the clock signal using a bit stored in the latch. If the control unit sets that bit to zero, the clock will stop.
- In some machines, this is done by a HALT instruction. In LC-3 (and others), the latch is part of a special register controlled by the operating system.

university of groningen

Questions?