Lecture 2

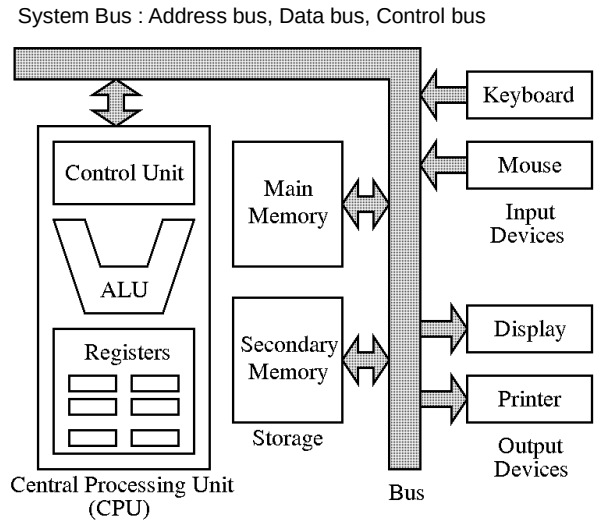
Computer Structure,
Memory Systems

Topics

- Computer Structure
- Memory Systems

Computer Organization

- A computer is composed of a microprocessor (CPU), memory, and Input/Output units.
- Microprocessor executes instructions of programs in memory.
- All components are connected via system bus.



3

Memory

- Memory is a sub-system that stores **instructions** and **data** in binary format (0's and 1's).
- Types of memory:
  - **Registers** are inside the microprocessor.
  - **Read Only Memory (ROM)**
    - Used to store programs permanently that does not change (Firmware programs : built in factory).
  - **Random Access Memory (RAM)** (Read/Write Memory)
    - Used to store programs and data temporarily.

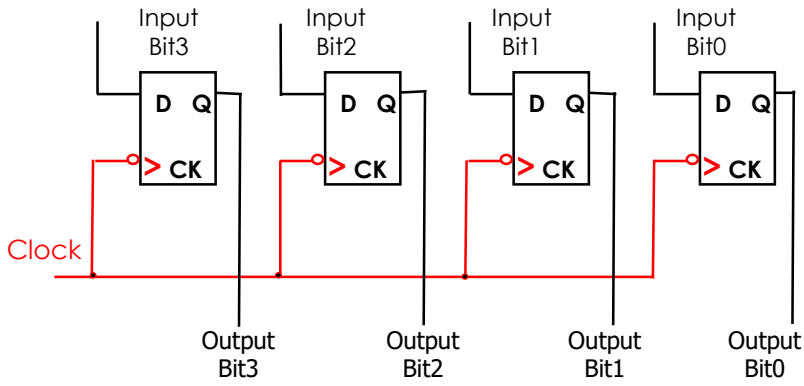
4

Registers

- A register is a collection of **flip-flops** that are clocked (synchronized) as a unit.
- A flip-flop is a 1-bit memory device.
- The following example is a 4-bit register.

Truth Table

D Input	Q(t+1) Output
0	0
1	1



5

Registers

- **Registers** are very fast special memory units within the CPU.
- They hold the data numbers operated.
- Different CPUs have different set of registers.

- General-purpose registers are used to store data during CPU execution.
  - **Accumulator** (ACC) register contains temporary arithmetic and logic results.

- Special-purpose registers:
  - **Memory Address Register** (MAR) contains memory address of data when transferred between CPU and memory.
  - **Memory Data Register** (MDR) contains the data when transferred between CPU and memory.

6

Arithmetic and Logic Unit (ALU)

- **Arithmetic and Logic Unit** (ALU) is a circuit that performs arithmetic and logical operations such as Add, Subtract, Complement, Shift, etc.
- ALU receives data from main memory or registers, performs a computation, and if necessary, writes result back to main memory or registers.

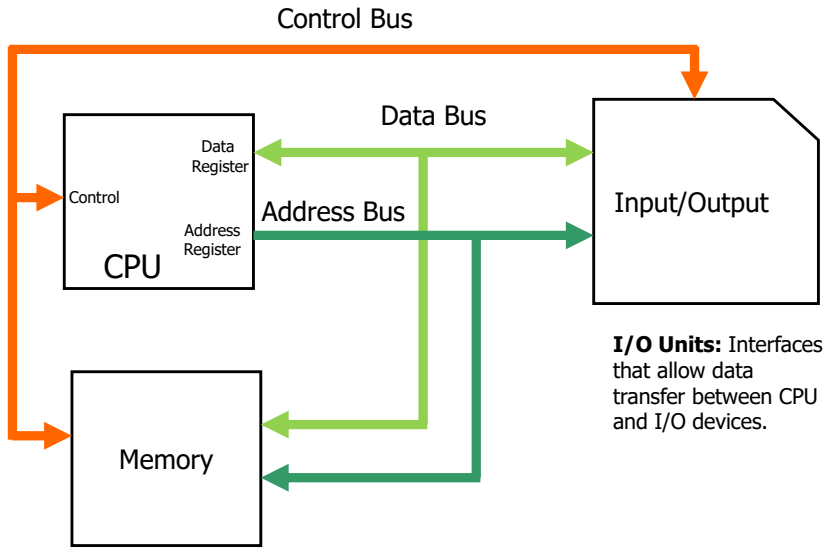
7

Control Unit

- **Control unit** is a circuit that sends **control signals** to ALU, memory and other parts of system.
  - Control unit is **instruction controlled.**
  - It can recognize several hundred different instruction codes.
- **Control unit** manages computer operation within and outside of CPU.
  - Execution cycles
  - System bus
  - System status
  - Interrupt handling

8

Basic Computer Connections

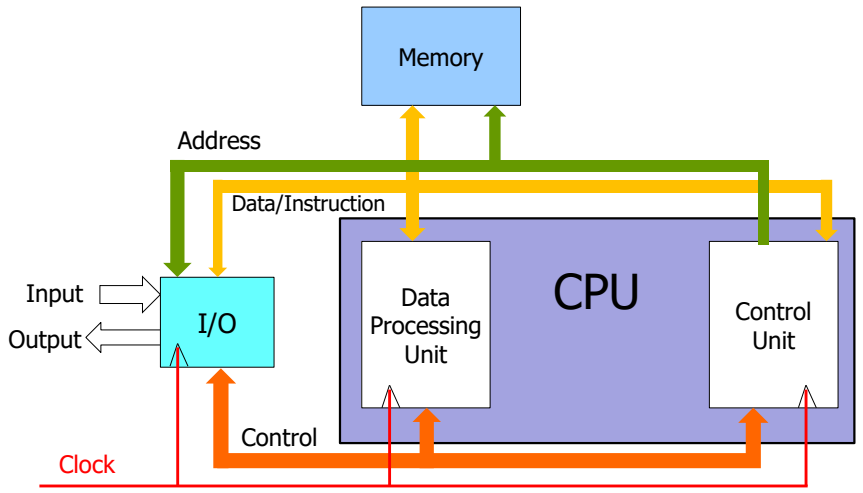


I/O Units: Interfaces that allow data transfer between CPU and I/O devices.

9

Von Neumann Architecture

Data and Instructions are stored in the same memory.

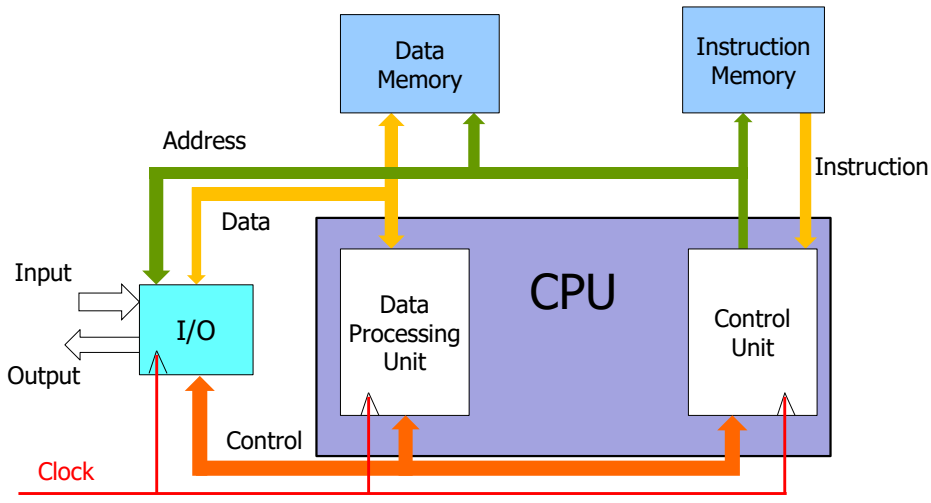


Data Processing Unit :
Contains ALU and registers.

10

Harvard Architecture

Data and Instructions are stored in two separate memories.



11

Operation of Computer

- The instructions in memory are executed by computer sequentially one at a time.
- **Example:** Two decimal numbers (20 and 30) will be stored in memory addresses $3000 and $3001. ($ symbol is used for hexadecimal notation.)
- Sum of two numbers will be calculated and stored on address $3002.

Pseudo Instructions

1. Store 20 to address $3000
2. Store 30 to address $3001
3. Load ACC (accumulator) register from address $3000
4. Add content of address $3001 to ACC
5. Store content of ACC to address $3002
6. Stop

Assembly Instructions

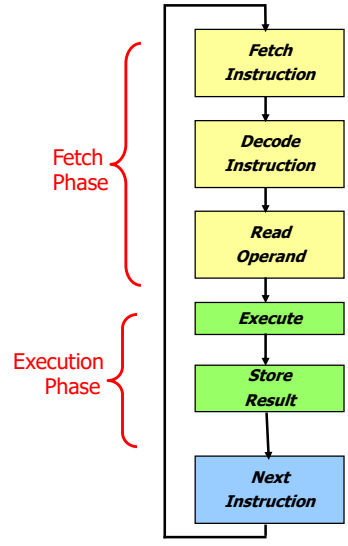
```
START
   STA 20, $3000
   STA 30, $3001
   LDA A, <$3000>
   ADD A, <$3001>
   STA A, $3002
   INT
```

Memory Address	Memory Content (Data)
$3000	20
$3001	30
$3002	50

12

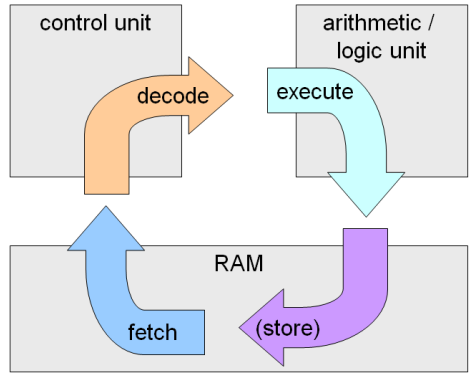
Instruction Cycle

- An instruction cycle consists of two main phases.
- **Fetch Phase** (Fetch & Decode)
  - Brings instruction from memory, then decodes it.
  - Also reads operand from memory if necessary.
- **Execution Phase** (Execute & Store)
  - Performs operation that the instruction requires
  - Storing the result to memory is optional



13

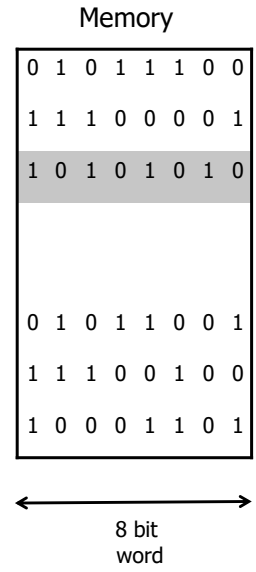
CPU and RAM usage during an Instruction Cycle



14

Instruction Format

- Instructions are stored in the **memory**.
- They are executed in a **sequence**.
- The computer fetches the next instruction and decodes it.
- Each CPU has different set of instructions.



15

Instruction Templates

- **0 Address Instruction Template**

Opcode	Register	Immediate data

Example: LDA A, 5
A : Accumulator register
5 : Immediate data (operand)

- **1 Address Instruction Template**

Opcode	Register	Operand address

Example: LDA A, <$3000>
A : Accumulator register
$3000 : Memory address of data (operand)
The <> symbols are used for memory address.

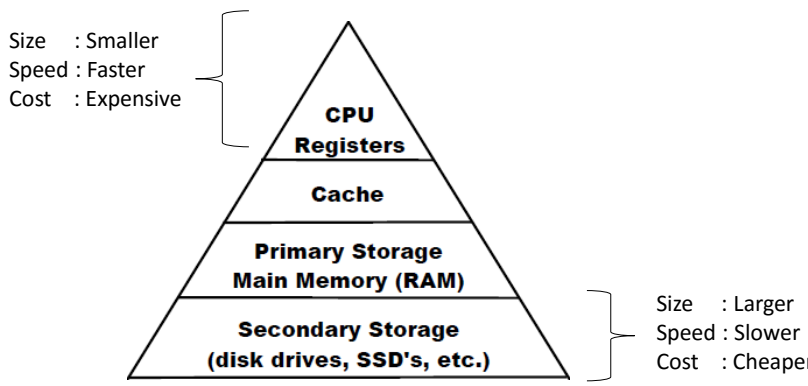
16

Topics

- Computer Structure
- Memory Systems

17

Comparison of Memory Types



18

Comparision of Memory Types

Memory Unit	Example Size	Typical Speed
Registers	16, 64-bit registers	~1 nanoseconds
Cache Memory	4 - 8 Megabytes	~5-60 nanoseconds
Primary Storage (i.e., main memory)	2 – 32 Gigabytes	~100-150 nanoseconds
Secondary Storage (i.e., disk, SSD's, etc.)	500 Gigabytes – 4 Terabytes	~3-15 milliseconds

SIZES
1 KB = 10^3 bytes
1 MB = 10^6 bytes
1 GB = 10^9 bytes
1 TB = 10^{12} bytes

SPEEDS
1 second
= 10^3 milli sec
= 10^6 micro sec
= 10^9 nano sec
= 10^{12} pico sec

19

ROM and RAM Types

- Memory holds instruction code numbers and data numbers.

- **ROM (Read Only Memory)** : Contents are not erased when power is off.
- **RAM (Random Access Memory)** : Contents are erased when power is off.

  - **Non-volatile memory types**
    - Read only memory (ROM)
    - Programmable read-only memory (PROM)
    - Erasable programmable ROM (EPROM)
    - Electrically erasable programmable ROM (EEPROM)
    - Flash memory

  - **Volatile memory types** (Vaporizing)
    - Dynamic random-access memory (DRAM)
    - Static random-access memory (SRAM)

20

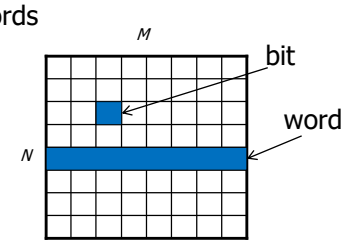
Usage of RAM

- **DRAM:** Dynamic Random Access Memory
  - Refreshed periodically
  - Used in Main memory
- **SRAM:** Static Random Access Memory
  - Read/write is faster
  - Used in Cache memory for high-speed

21

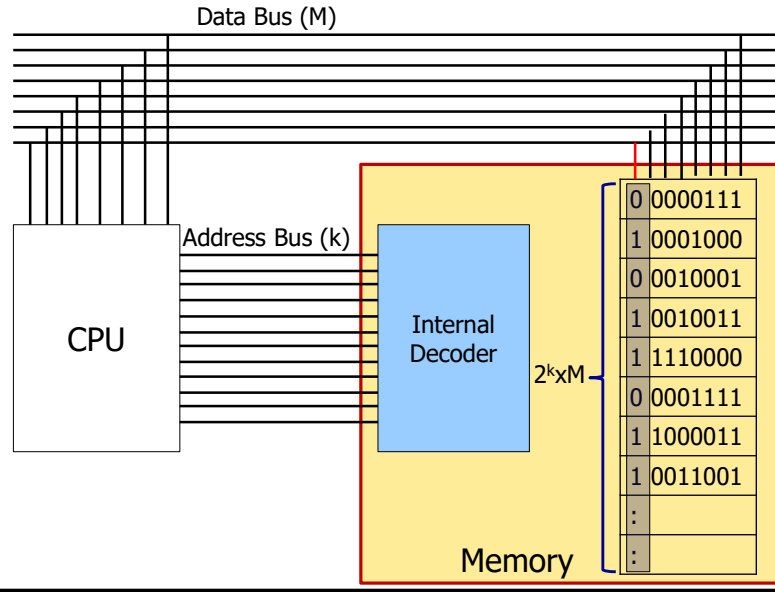
Memory Units

- Each entry in a memory unit is called a **word**.
- Each word is composed of M bits (Data bus width).
- CPU can access memory at the word level, not at the bit level.
- Size (N) of a memory is the number of words
  $N = 2^k$
  k: Address bus width
- Memory is a matrix of NxM bits
  - N: number of rows (words)
  - M: number of columns (bits)
- Word : 1 byte (8 bits)
- Double word : 2 bytes (16 bits)



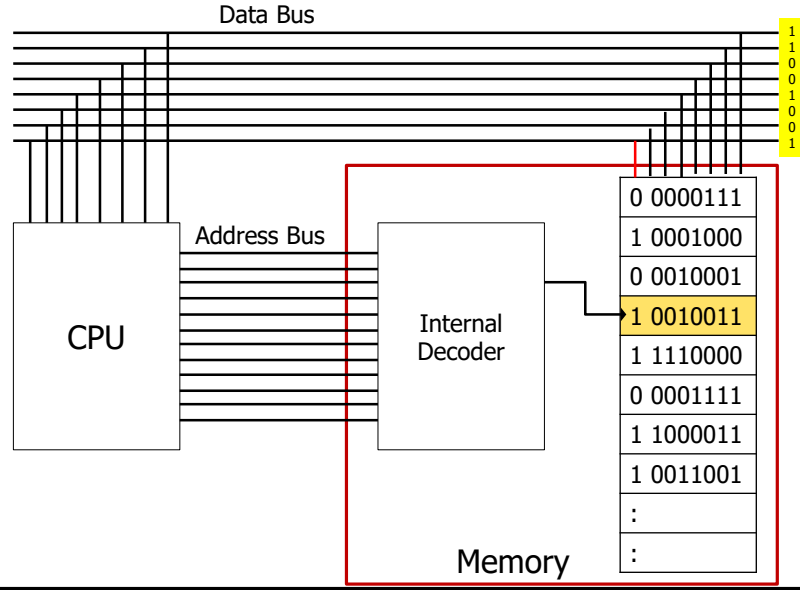
22

CPU and Memory Connection



23

Example: Accessing a memory location



24

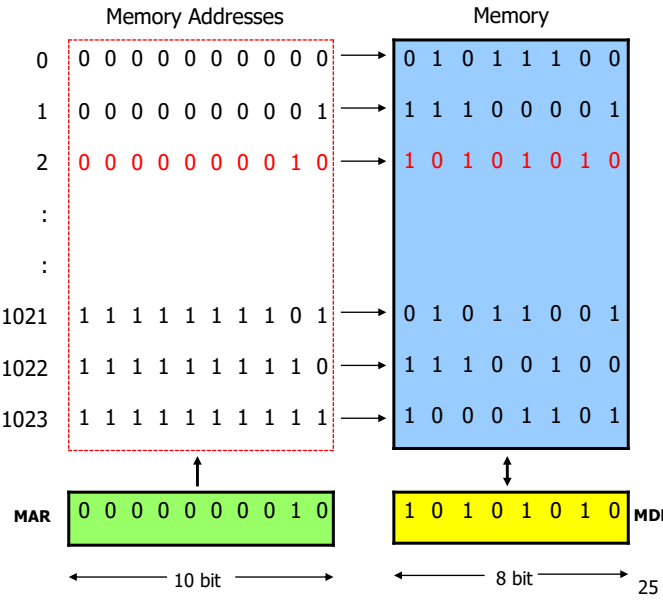
Example: MAR and MDR registers

MAR : Memory Address Register

MDR: Memory Data Register

Data bus = 8 bit
Address bus = 10 bit

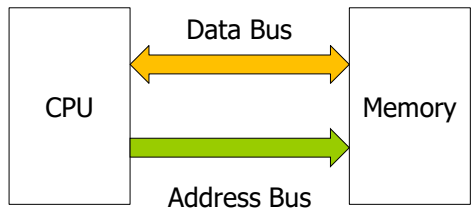
2^{10}=1024 (1 KB) address locations.



25

Memory Access

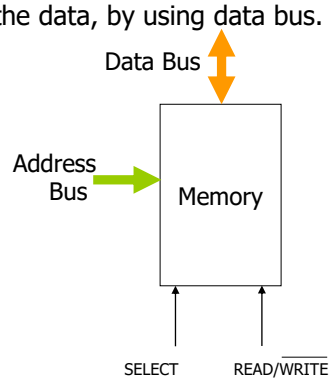
- Each memory location has a unique address.
- Operand address from an instruction is copied to the **MAR,** which finds the operand location in memory.
- CPU determines if an instruction is a store or retrieval operation.
- Transfer takes place between the **MDR** and memory.
- MDR is a two-way register.



26

Memory Access

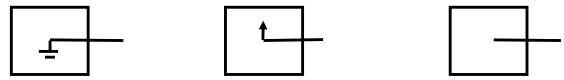
- Microprocessor accesses (for Read or Write) information in memory as follows :
  1. Select the memory chip, by using part of address bus. (Among multiple memory chips.)
  2. Identify memory location within the chip, by using the rest of the address bus.
  3. Access the data, by using data bus.



27

Tri-State Buffer

- Tri-state buffer is an important circuit element that is used in memory access.
- Buffer is a logic circuit that has three states:
  - Logic 0
  - Logic 1
  - High impedance (Isolated)
- When circuit is in high impedance mode, it is disconnected from the output completely.

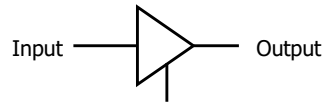


Output is Low (Logic 0) | Output is High (Logic 1) | High Impedance (Isolated)

28

Tri-State Buffer

- Tri-State buffers are used to control the direction of data flow in memory access.
- The following is the circuit symbol of a Tri-State buffer.
- Circuit has two inputs and one output.
  - First input is the normal "**data**" input for circuit.
  - Second input is an "**enable**" signal.
    - If enable is set high, output follows the input.
    - If enable is set low, output is disconnected.

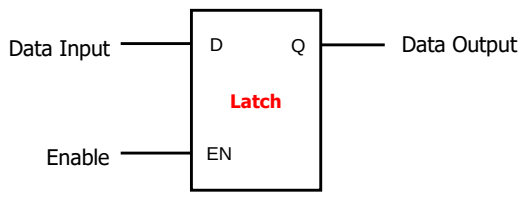


Enable

29

Basic Memory Element (D-latch)

- The basic memory element is a Data Latch.
- D-latch is a 1-bit memory device, similar to flip-flop.
- Latch has three connections.
  - Input
  - Output
  - Enable



Block diagram

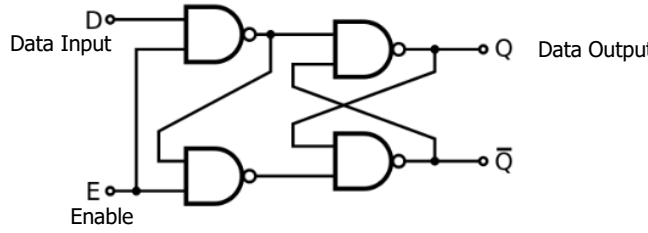
Truth Table

D Input	Q(t+1) Output
0	0
1	1

30

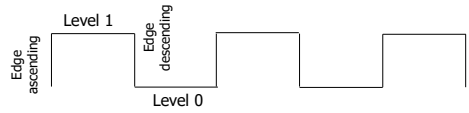
D-Latch as Logic Gates

- The following diagram shows a D-latch implementation, based on NAND logic gates.



Differences between Latch and Flip-Flop

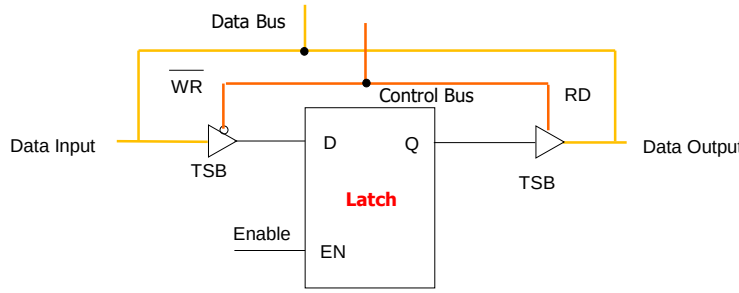
LATCH	level-triggered	asynchronous	not-clocked
FLIP-FLOP	edge-triggered	synchronous	clocked



31

Connections of Basic Memory Element (D-latch) and Tri-state Buffers

- Data is always present on input (D).
- Output (Q) is always set to the content of D-latch.
- Tri-state buffers (TSB) are added both at the Input (Write Signal) and Output (Read Signal) of latch.



32

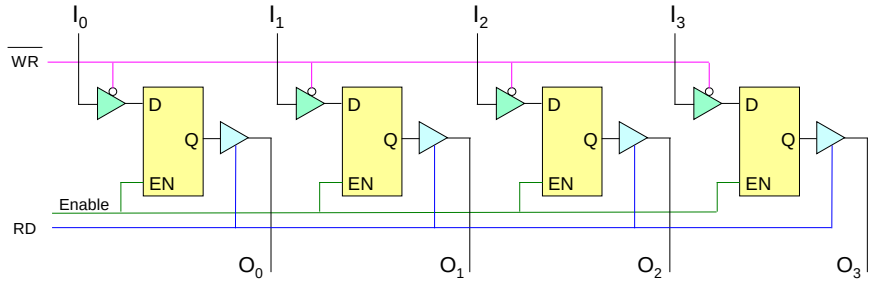
Write and Read Control Signals

- The $\overline{WR}$ (write) signal controls the input tri-state buffer.
  - WR is considered as an input signal, according to the latch.
  - The bar over WR means that this is an active low signal.
    - If WR is 0, then input data reaches latch input.
    - If WR is 1, then input wire is disconnected.
- The RD (read) signal controls the output tri-state buffer.
  - RD is considered as an output signal, according to the latch.
  - RD is an active high signal.
    - If RD is 1, then data reaches latch output.
    - If RD is 0, then output wire is disconnected.

33

Example: 4-bit Memory Register

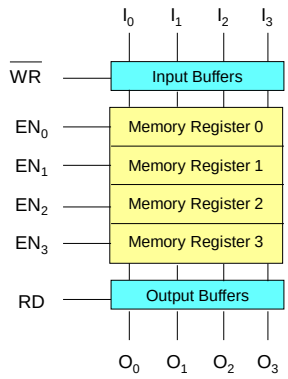
- 4 latches are connected together in parallel.



34

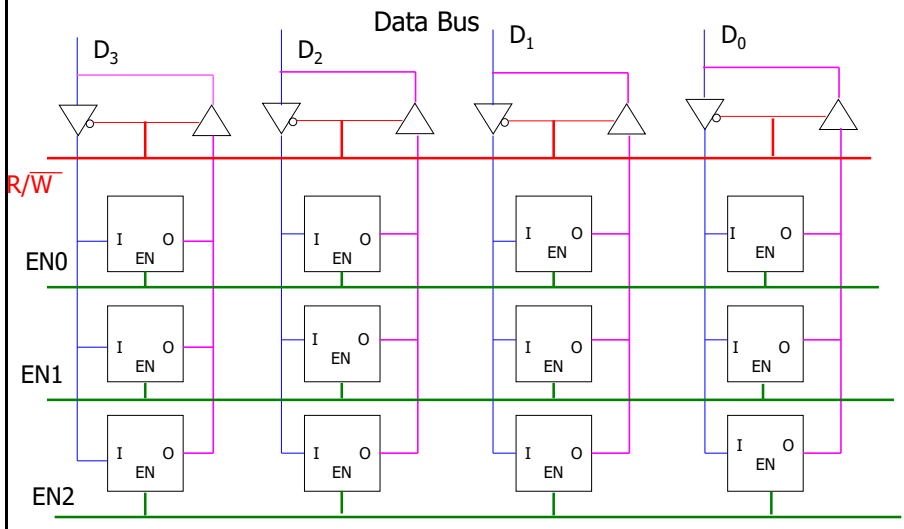
Memory Addressing

- Using the RD and WR control signals, we can determine the **direction of data flow** either **into** or **out** of memory.
- Using the Enable input signals, we enable individual memory registers.
  - Only one Enable can be active at a time.



35

Example: 3x4Bit Memory with latches

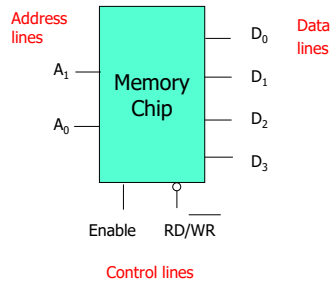


Total number of bits (latches) in memory = 3 x 4Bit = 12 Bits

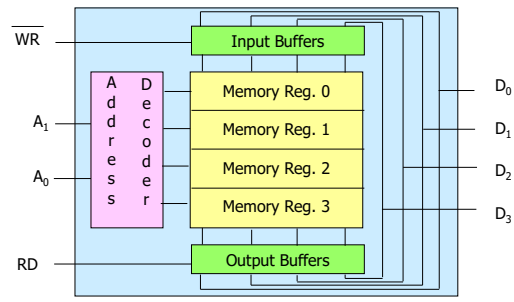
36

Design of a Memory Chip

Block diagram of a memory chip (4x4Bit).



Inside of the memory chip.

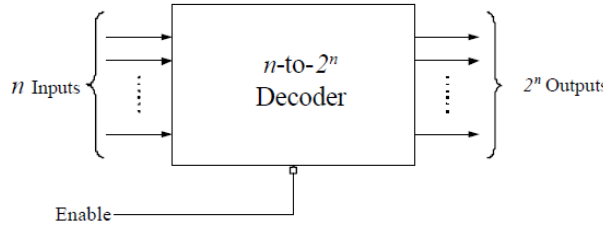


4x4Bit Memory (Total 16 bits)

37

Memory Component : Decoder

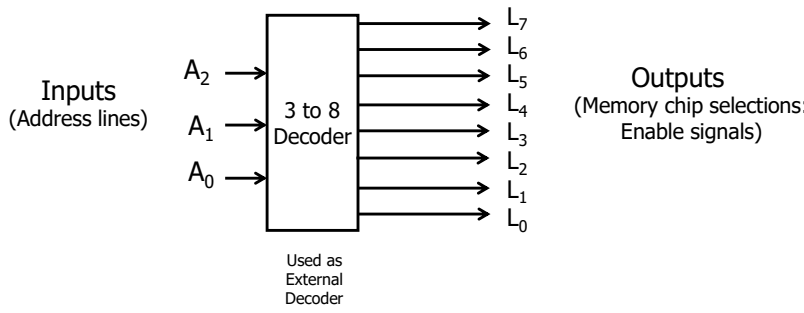
- Decoder is a circuit (usually as a chip) that converts binary information from n-coded inputs to maximum of 2^n outputs.
- Only one output is 1, others are 0.
- **Address Decoders** are used for **location (row) selection** within a memory chip. (Internal Decoder)
- They are also used to for **memory chip selection** among multiple memory chips. (External Decoder)



38

Example : 3-to-8 Decoder

- There are 3 input lines.
- There are 8 output lines. (2^3)
- Depending on inputs, only one output line will be 1, other output lines will be 0.



39

Truth Table for 3-to-8 Decoder

- Only one output line is selected (as 1) at a time.

INPUTS			OUTPUTS							
A_2	A_1	A_0	L_7	L_6	L_5	L_4	L_3	L_2	L_1	L_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

40