جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

قسم الامن السيبراني
Department of Cyber Security

Subject:

Computer Architecture

Class:

third

Lecturer:

Asst. raed alshmary

Lecture: (1)

Introduction

1. Introduction

Computer architecture is the organization of the components which make up a computer system and
the meaning of the operations which guide its function. It defines what is seen on the machine interface,
which is targeted by programming languages and their compilers.
Computer architecture can be defined as a set of rules and methods that describe the functionality,
management and implementation of computers. To be precise, it is nothing but rules by which a system performs
and operates.

Computer Architecture can be divided into mainly three categories, which are as follows

1– **Instruction set Architecture or ISA** − Whenever an instruction is given to processor, its role is to read
and act accordingly. It allocates memory to instructions and also acts upon memory address mode (Direct
Addressing mode or Indirect Addressing mode).
2 – **Micro Architecture** − It describes how a particular processor will handle and implement instructions
from ISA.
3 – **System design** − It includes the other entire hardware component within the system such as virtualization, multiprocessing.

Role of computer Architecture:

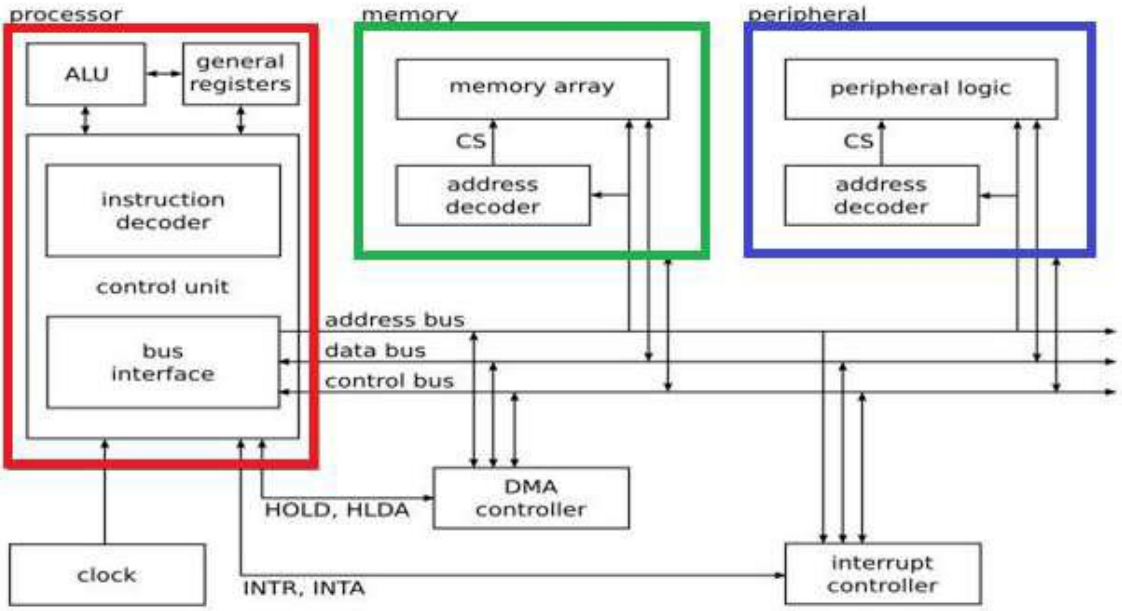
The main role of Computer Architecture is to balance the performance, efficiency, cost and reliability of a
computer system.
For Example − Instruction set architecture (ISA) acts as a bridge between computer's software and hardware.
It works as a programmer's view of a machine.
Computers can only understand binary language (i.e., 0, 1) and users understand high level language (i.e., if
else, while, conditions, etc). So to communicate between user and computer, Instruction set Architecture plays
a major role here, translating high level language to binary language.

Structure Computer Architecture:

Example structure of Computer Architecture as given below. Generally, computer architecture consists of
the following −

- Processor
- Memory
- Peripherals

All the above parts are connected with the help of system bus, which consists of **address bus**, **data bus** and **control bus.**
The diagram given below depicts the computer architecture −



1.1. RISC and CISC Architecture

Means classification of instructions set to categories to increase the performance of
Computer (Classification Instruction Set According Languages program)

1. Reduced Instruction Set Computer (RISC)

An important aspect of computer architecture is the design of the instruction set for the processor. The instruction set chosen for a particular computer determines the way that machine language programs are constructed. A computer with a large number of instructions is classified as a complex instruction set computer, abbreviated CISC. In the early 1980s, a
Reduced Instruction Set Computer or RISC Architecture
The fundamental goal of RISC is to make hardware simpler by employing an instruction set that consists of only a few basic steps used for evaluating, loading, and storing operations. A load command loads
data but a store command stores data.

Characteristics of RISC:

1. It has simpler instructions and thus simple instruction decoding.
2. More general-purpose registers.
3. The instruction takes one clock cycle in order to get executed.
4. The instruction comes under the size of a single word.
5. Pipeline can be easily achieved.
6. Few data types.
7. Simpler addressing modes.

2. Complex Instruction Set Computer or Architecture (CISC)

The fundamental goal of CISC is that a single instruction will handle all evaluating, loading, and storing operations, similar to how a multiplication command will handle evaluating, loading, and storing data,
which is why it's complicated.

Characteristics of CISC:

1. Instructions are complex, and thus it has complex instruction decoding.
2. The instructions may take more than one clock cycle in order to get executed.
3. The instruction is larger than one-word size.
4. Lesser general-purpose registers since the operations get performed only in the memory.
5. More data types.
6. Complex addressing modes.

Both CISC and RISC approaches primarily try to increase the performance of a CPU. Here is how both of

these work:
1. CISC: This kind of approach tries to minimize the total number of instructions per program, and it does so at the
cost of increasing the total number of cycles per instruction.

2. RISC: It reduces the cycles per instruction and does so at the cost of the total number of instructions per program.

BYJU'S The Learning App

$$CPU\ Time = \frac{Seconds}{Program} = \frac{Instructions}{Program} \times \frac{Cycles}{Instructions} \times \frac{Seconds}{Cycle}$$

When programming was done in assembly language earlier, there was a desire to make the instructions perform more
tasks. It is because assembly programming was arduous and error-prone and led to the evolution of CISC architecture. But as the dependency of high-level language on assembly language
decreased, RISC architecture prevailed

Example
Suppose we need to add two different 8-bit numbers:

1. CISC approach: There would be a single instruction or command for this, such as ADD, that would perform the task.

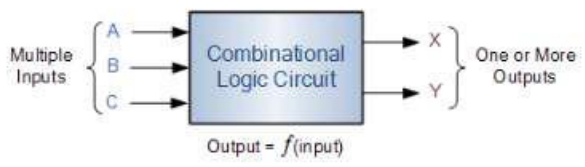
2. RISC approach: In this case, the programmer would write the very first load command in order to load
data in the registers. Then it would use a suitable operator and store the obtained result in the location that is desired.
The add operation here is divided into parts, namely, **operate, load, and store**. Due to this, RISC programs are much longer, and they require more memory to get stored, even though they require fewer
transistors because the commands are less complex.

combinational circuit is defining In digital electronics, a combinational circuit is **a circuit in which the output depends on the present**
combination of inputs. Combinational circuits are made up of **logic gates**. The output of each logic gate is
determined by its logic function.

difference between combinational and sequential circuit

1. **combinational circuit** is time-independent. The output it generates does not depend on any of its
previous inputs.
2. **sequential circuits** are the ones that depend on clock cycles. They depend entirely on the past as well as the present inputs for generating output.

Parameters	Combinational Circuit	Sequential Circuit
Meaning and Definition	It is a type of circuit that generates an output by relying on the input it receives at that instant, and it stays independent of time.	It is a type of circuit in which the output does not only rely on the current input. It also relies on the previous ones.
Feedback	A Combinational Circuit requires no feedback for generating the next output. It is because its output has no dependency on the time instance.	The output of a Sequential Circuit, on the other hand, relies on both- the previous feedback and the current input. So, the output generated from the previous inputs gets transferred in the form of feedback. The circuit uses it (along with inputs) for generating the next output.
Performance	We require the input of only the current state for a Combinational Circuit. Thus, it performs much faster and better in comparison with the Sequential Circuit.	In the case of a Sequential Circuit, the performance is very slow and also comparatively lower. Its dependency on the previous inputs makes the process much more complex.
Complexity	It is very less complex in comparison. It is because it basically lacks implementation of feedback.	This type of circuit is always more complex in its nature and functionality. It is because it implements the feedback, depends on previous inputs and also on clocks.
Elementary Blocks	Logic gates form the building/ elementary blocks of a Combinational Circuit.	Flip-flops form the building/ elementary blocks of a Sequential Circuit.
Operation	One can use these types of circuits for both- Boolean as well as Arithmetic operations.	You can mainly make use of these types of circuits for storing data.

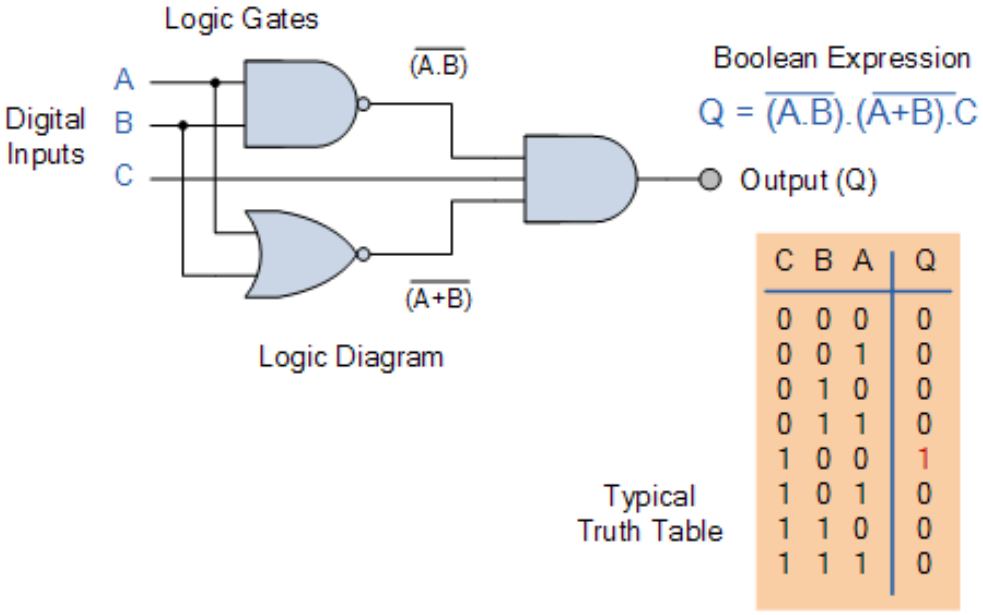


Combinational Logic Circuits
Combinational Logic Circuits are memoryless digital logic circuits whose output at any instant in time depends only on the combination of its inputs.

Unlike Sequential Logic Circuits whose outputs are dependent on both their present inputs and their previous output state giving them some form of ***Memory.***

The outputs of **Combinational Logic Circuits** are only determined by the logical function of their current input state, logic "0" or logic "1", at any given instant in time.

For example, you have Three digital inputs (A, B, C) and output (Q), Design Combinational Logic Circuits (your
Answer should appear (Logical diagram, Boolean Expression and Typical truth table)?
Solution



C	B	A	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

that the differences between Combinational circuitry and State circuitry solution

1. **Combinational circuitry** behaves like a simple function. The output of combinational circuitry depends only on the current values of its input.
2. **State circuitry** behaves more like an object method. The output of state circuitry does not just depend on its inputs — it also depends on the past history of its inputs.

that the Difference Between Half Adder and Full Adder

There is a primary difference between half adder and full adder. Half adder only adds the current inputs as 1-bit
numbers and does not focus on the previous inputs. On the other hand, Full Adder can easily carry the current
inputs as well as the output from the previous additions.

What is a Half Adder?

It is a combinational logic circuit. You can design it by connecting one AND gate and one EX-OR gate.
A half-adder circuit consists of two input terminals- namely A and B. Both of these add two input digits (one bit numbers) and generate the output in the form of a **carry and a sum**. Thus, there are two output terminals.
The output that one obtains from the EX-OR gate is the sum of both the one-bit numbers. The output
obtained from the AND gate is called the carry. But you cannot forward the carry that you obtain in one addition into another addition. It is because of the absence of any logic gate to process it. Thus, it's called the Half Adder circuit.
We can write the equation of output for both the gates in the form of a logical operation that the logic gates perform. Here, we write the carry equation in the form of AND operation and the sum equation in the form of EX-OR operation.

Logical Expression of Half Adder

Sum (S) = $A \oplus B$

Carry (C) = $A \bullet B$

Truth Table

Here is a truth table representing the possible outputs obtained from the possible inputs in a Half Adder:

Input		Output	
A	B	CARRY	SUM
0	0	0	0
1	1	1	0
0	1	0	1
1	0	0	1

What is Full Adder?

A full adder is a circuit that has two AND gates, two EX-OR gates, and one OR gate. The full adder adds three binary digits.
Among all the three, one is the carry that we obtain from the previous addition as C-IN, and the two are inputs A and B. It designates the input carry as the C-OUT and the normal output as S (or SUM).
Just like the Half Adder, the Full Ladder is a combinational type of logic circuit- meaning, it has no storage element. But it has additional logic gates. Thus, it adds the previous carry to generate the complete output. Thus, it is called the Full Adder.
One can also designate a Full Adder using one OR gate and two Half Adders. The OR gate here generates a carry that it obtains after the addition. We obtain the sum of these digits in the form of output from the second Half Adder.
The equation for the output that you can obtain by the EX-OR gate is the sum of all the binary digits. Here, the output that you obtain from the AND gate is the carry that you obtain by addition. This equation is in the form of a logical operation.

Logical Expression of Full Adder

$$\text{CARRY-OUT} = AB + BC_{in} \oplus AC_{in}$$

$$\text{SUM} = (A \oplus B) \oplus C_{in}$$

Truth Table

A truth table represents the possible outputs obtained from the possible inputs. A truth table for the Full Adder is as follows:

Input			Output	
A	B	C	SUM	CARRY OUT
0	0	0	0	0
1	1	1	1	1
0	1	1	0	1
1	0	1	0	1
0	0	1	1	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1

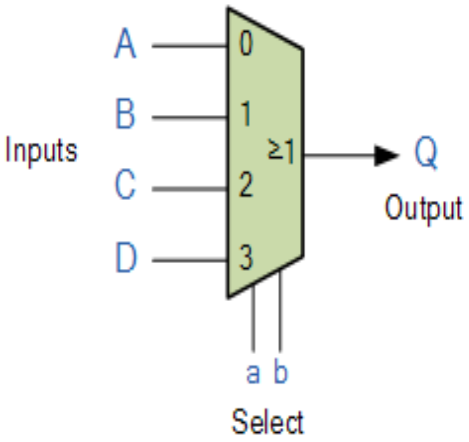
Q:\What are the main difference between Half Adder and Full Adder? Solution

Parameter	Half Adder	Full Adder
Basics	The Half Adder is a type of combinational logic circuit that adds two of the 1-bit binary digits. It generates carry and sum of both the inputs.	The Full Adder is also a type of combinational logic that adds three of the 1-bit binary digits for performing an addition operation. It generates a sum of all three inputs along with a carry value.
Adding the Previous Carry	The Half Adder does not add the carry obtained from the previous addition to the next one.	The Full Adder, along with its current inputs A and B, also adds the previous carry.
Hardware Architecture	A Half Adder consists of only one AND gate and EX-OR gate.	A Full Adder consists of one OR gate and two EX-OR and AND gates.
Total Inputs	There are two inputs in a Half Adder- A and B.	There are a total of three inputs in a Full Adder- A. B. C-in.
Usage	The Half Adder is good for digital measuring devices, computers, calculators, and many more.	The Full Adder comes into play in various digital processors, the addition of multiple bits, and many more.
Logical Expression	Here is the logical expression of Half Adder: $C = A * B$ $S = A \oplus B$	Here is the logical expression of Full Adder: $Cout = (AB) + CinA \oplus CinB$ $S = A \oplus B \oplus Cin$

What Means by multiplexer?

Multiplexing is the generic term used to describe the operation of sending one or more analogue or digital signals over a common transmission line at **different times** or **speeds** and as such, the device we use to do just that is called the multiplexer.

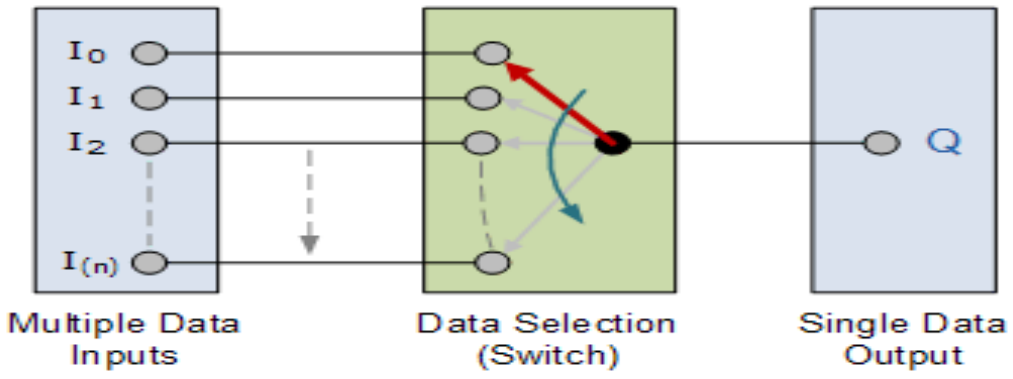
The *multiplexer*, shortened to "MUX" or "MPX", is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal. Multiplexers operate like very fast acting multiple position rotary switches connecting or controlling multiple input lines called "channels" one at a time to the output.



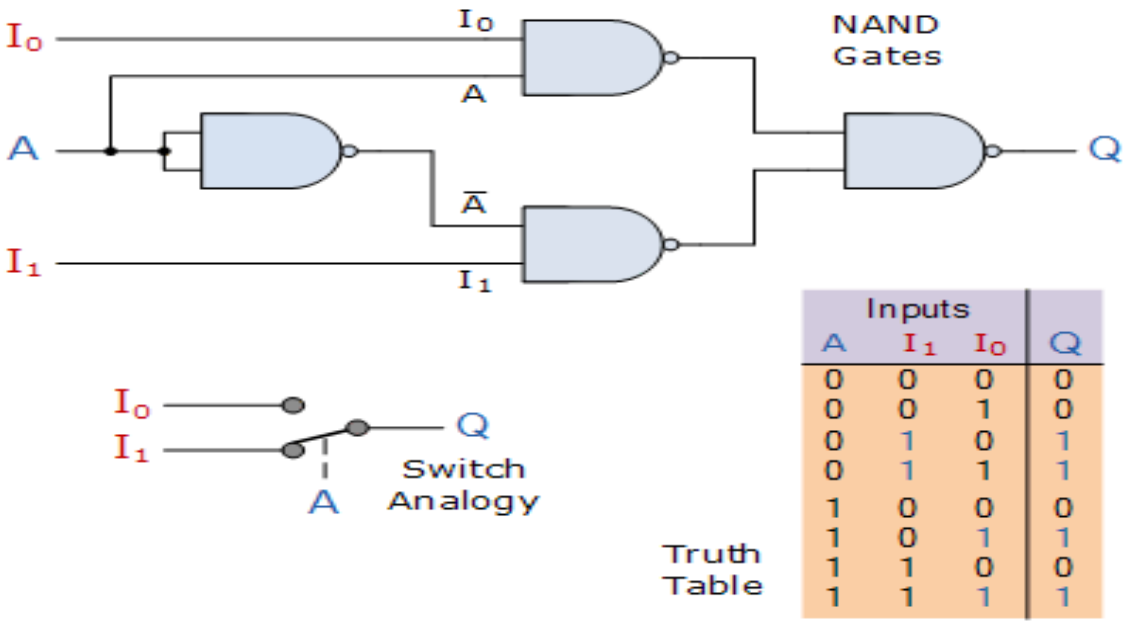
The Multiplexer

The multiplexer is a combinational logic circuit designed to switch one of several input lines to a single common output line

A. Basic multiplexing Switch



B. 2- multiplexing Switch



Truth Table

Inputs			
A	I_1	I_0	Q
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

The data distributor, known more commonly as the demultiplexer or "Demux" for short, is the exact opposite of the Multiplexer we saw in the previous tutorial.

The demultiplexer takes one single input data line and then switches it to any one of a number of individual output lines one at a time. The **demultiplexer** converts a serial data signal at the input to a parallel data at its output lines as shown below.