VLSI SYSTEMS AND ARCHITECTURE
2021-22
Lecture 5
ALU Design and Microprocessor Design

By Dr. Sanjay Vidhyadharan

CISC vs. RISC

Features	CISC	RISC
Semantic Gap	Low	High
Code Size	Small	Large, but RAMs are cheap!
Cost	High	Low
Speed	Fast only if the compiler generates appropriate code	Slower, but the problem is overcome using more registers and pipelining.

CISC vs. RISC

CISC Processor

CISC Code:

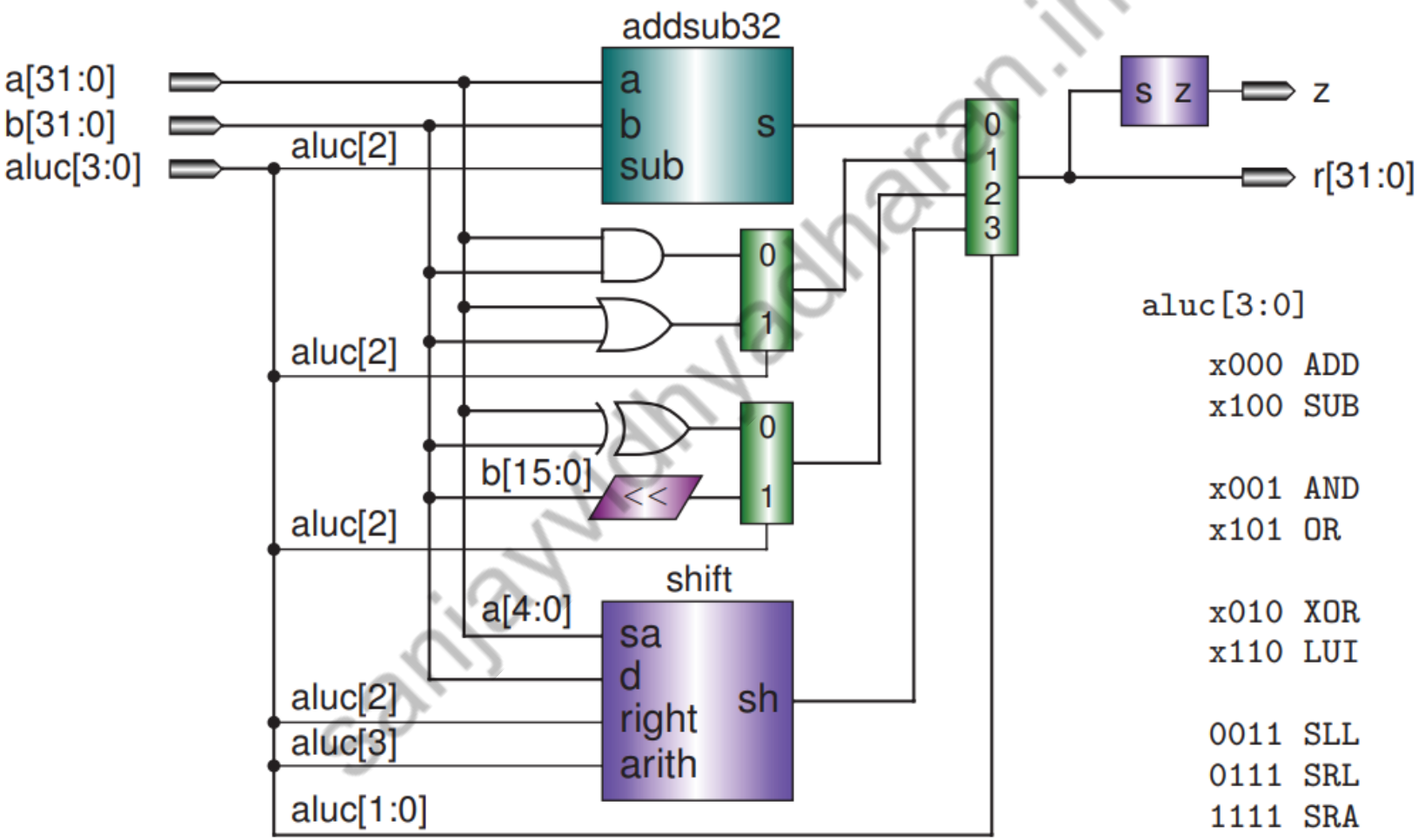
```
sub   [A], [A], [B]
add   [B], [B],1
```

RISC Code:

```
lw  R1, [A]
lw  R2, [B]
sub  R3, R1, R2
add  R4, R2, 1
sw  [A], R3
sw  [B], R4
```

A, B are memory locations and R registers of RISC
sw store word

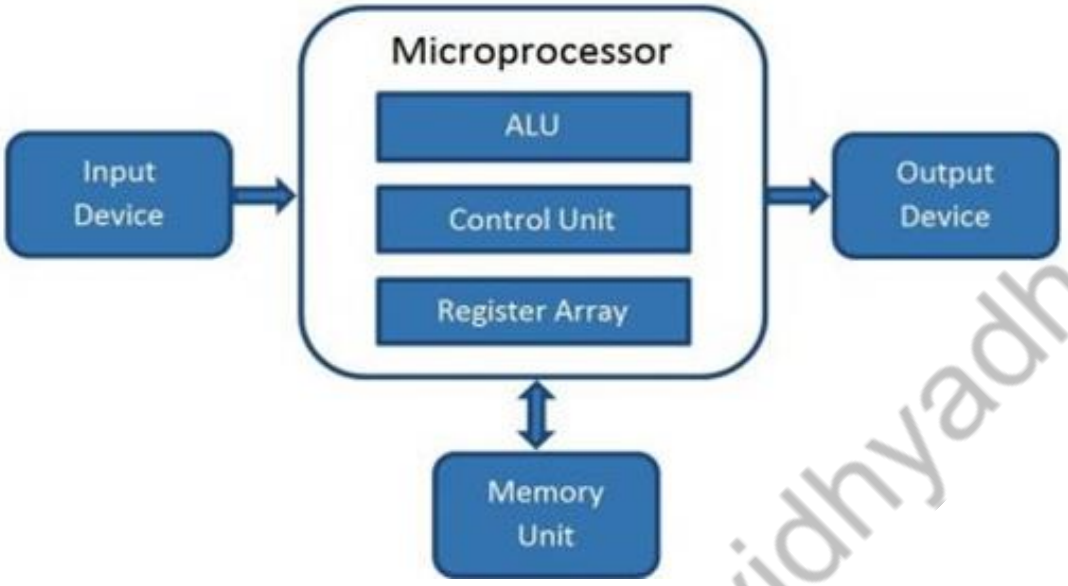
ALU Design

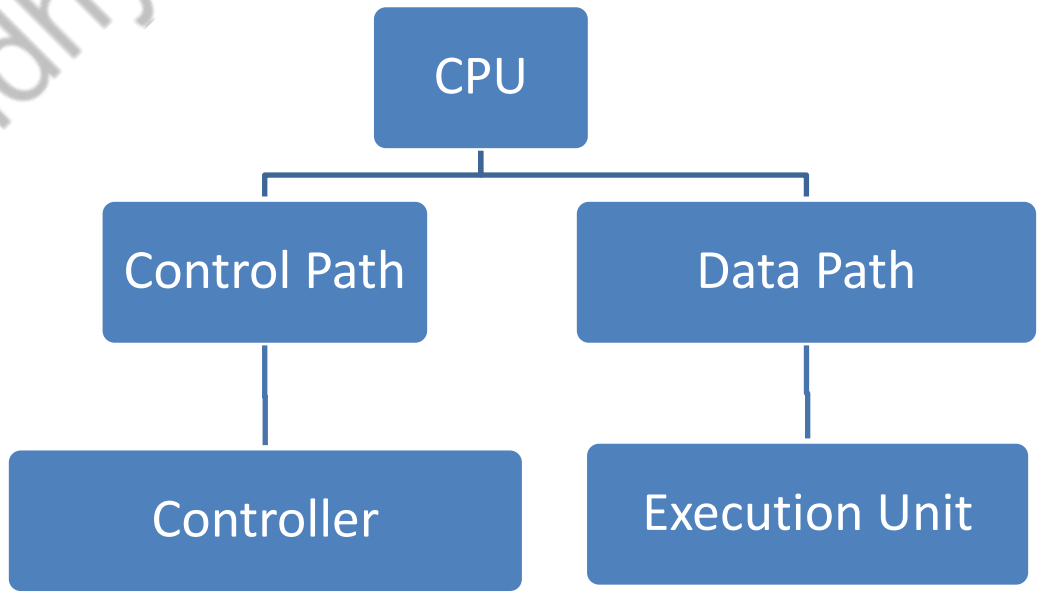


Microprocessor Design

- Given an Instruction Set, how do we begin to visualize and create the hardware architecture of the microprocessor ?
- How do we optimize the architecture for certain objectives (say speed or cost or a mix of them) ?
- These are some of the issues we would like to address first in the context of a typical CISC instruction set, and later, in the context of a Reduced Instruction Set Computer (RISC).

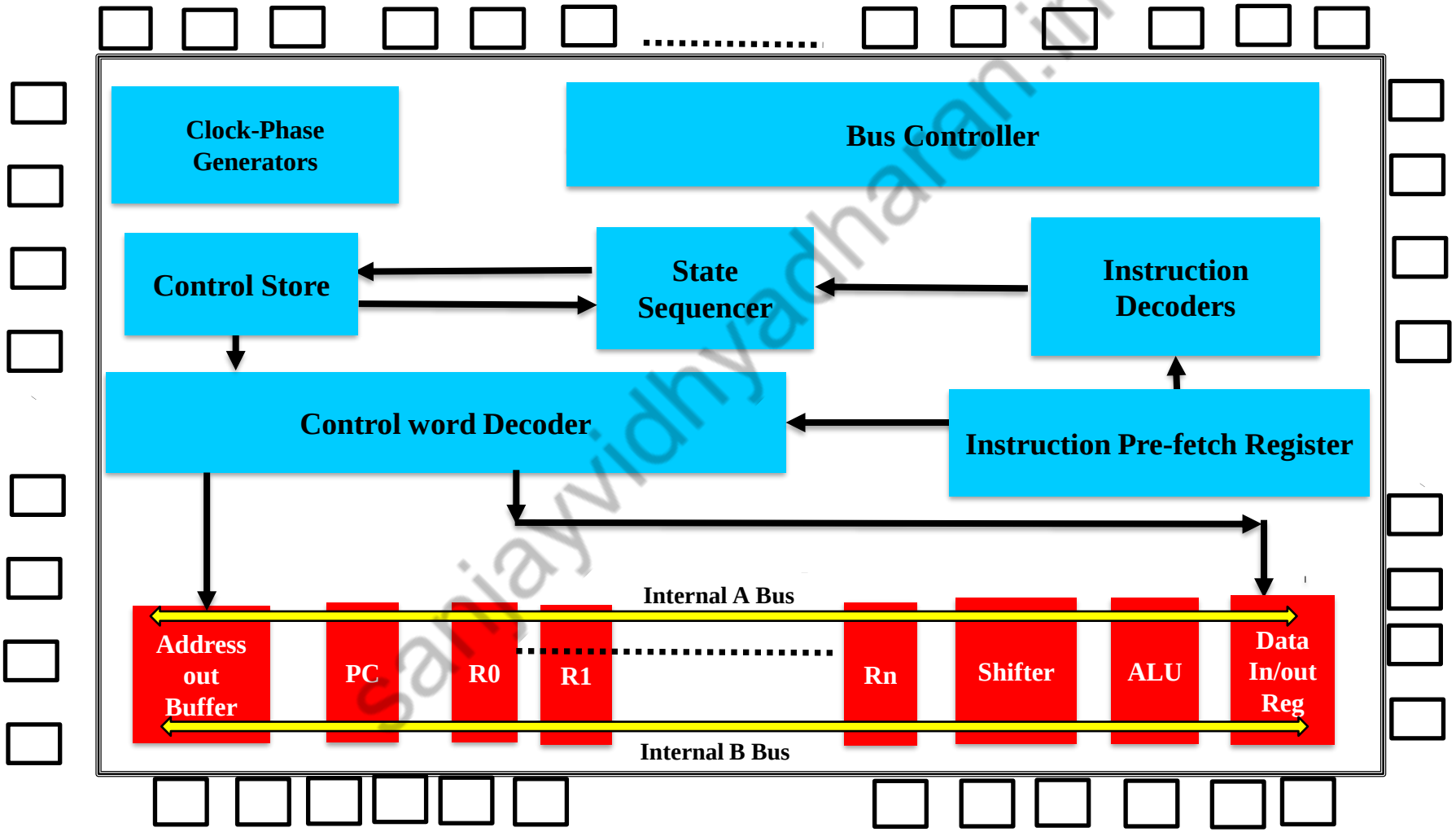
Microprocessor Design





Microprocessor Design

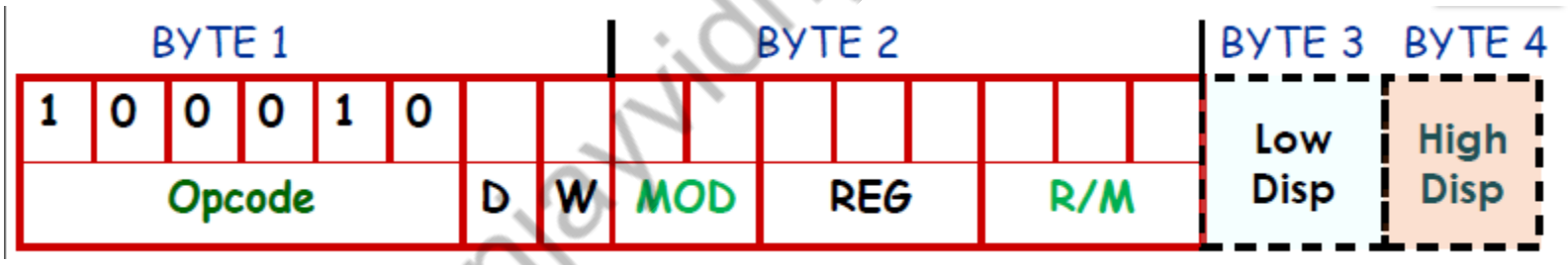
Pads for Bus Control, Clock, Reset, Interrupts, Testing and Power Supply



Example of Instruction Decoding

Inst.	[31:26]	[25:21]	[20:16]	[15:11]	[10:6]	[5:0]	Meaning
add	000000	rs	rt	rd	00000	100000	Register add

8086: Write the equivalent machine language code: **SUB BX, [DI]**

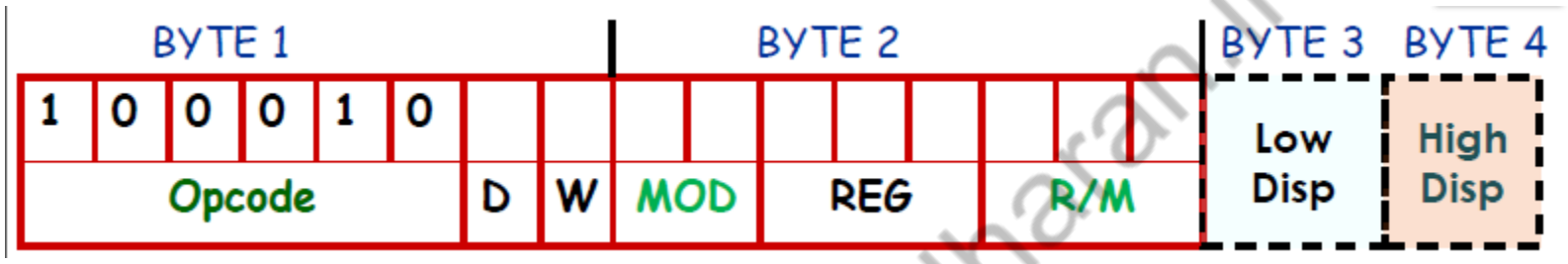


001010 1 1 00 011 101

= 2 B 1 D (H).

Example of Instruction Decoding

8086: Write the equivalent machine language code: **SUB BX, [DI]**



Steps for executing instruction

1. **Fetch the first instruction half-word**
2. **Find ADD control sequence**
3. **Fetch the remaining instruction half-word**
4. **Calculate the operand address**
5. **Fetch the operand**
6. **Add**
7. **Store the result**

	cycle 1	cycle 2	cycle 3	cycle 4	cycle 5	cycle 6
Instruction 1	Fetch	Decode	Execute			
Instruction 2		Fetch	Decode	Execute		
Instruction 3			Fetch	Decode	Execute	
Instruction 4				Fetch	Decode	Execute

Flowchart Method

Flowchart Objectives

- Hardware flow charts capture the activities inside a processor on a control step by control step basis.
- Acts as a design representation at the register transfer level as well as a tool for optimization of the design
- Coverts Written Description of Problem to RTL Implementable Language

E.g. $RX \rightarrow A \rightarrow ALU,$

"put the contents of register RX on the A bus to the ALU."

Flowchart Method

Flowchart Objectives

- Limit controller size to some fraction of chip area
- Make CPU as fast as possible
- Complete the project as early as possible
- Make the flowcharts easy to translate into hardware

Flowchart Method

The inputs required for the flow chart are

1. Instruction set summary

   (i) Instruction formats

   (ii) Operations (ADD, AND, SUB, and so on)

   (iii) Addressing modes (Base Plus Displacement, Register Indirect, Indexed, and so on)

   (iv) Registers (as seen by the programmer)

2. Execution unit specification

   (i) Programmer 's register set

   (ii) Additional registers (such as the instruction register, program counter, and temporary registers)

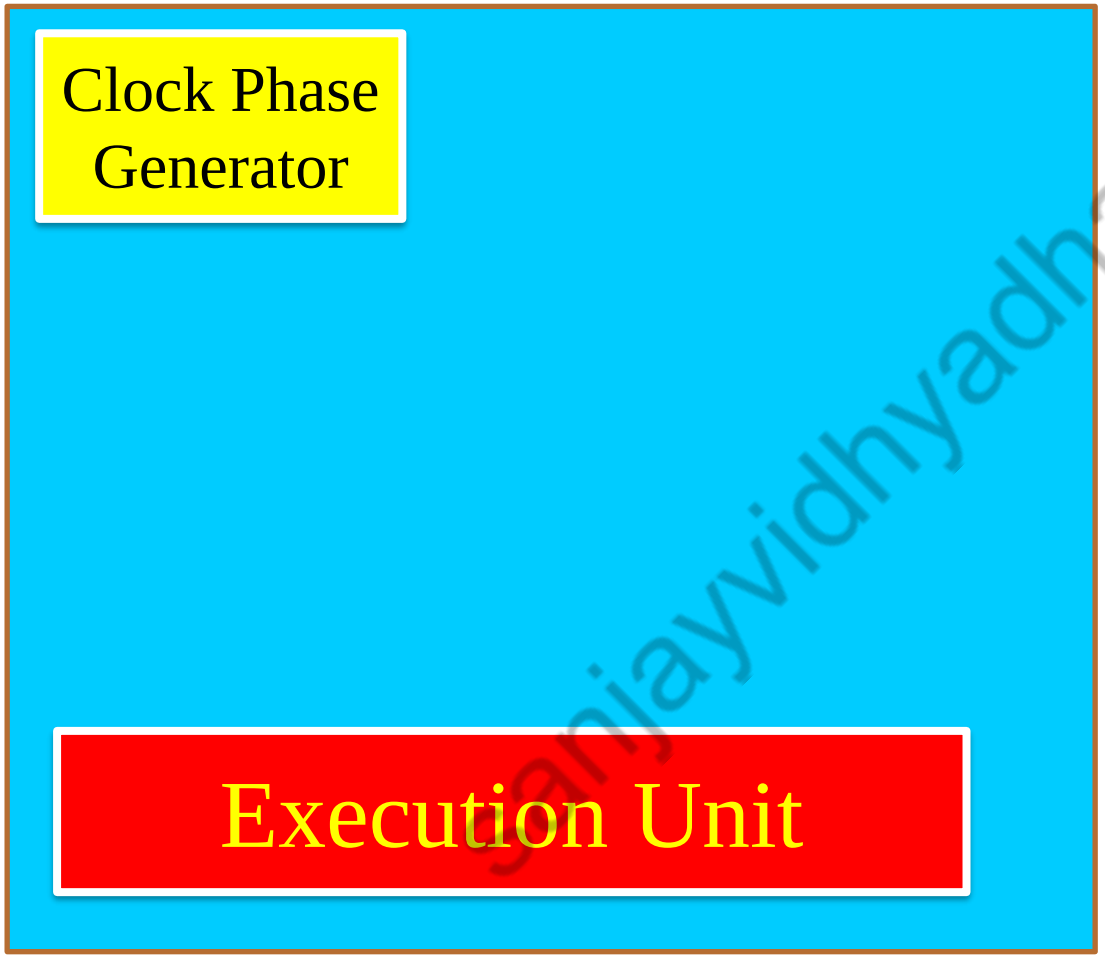
   (ii) ALU and any special function units (such as a shifter)

   (iv) Internal data paths

   (v) Rules of operation

Flowchart Method

Step 1



The architecture specification is the only input.

Begin with a guess for the execution unit.

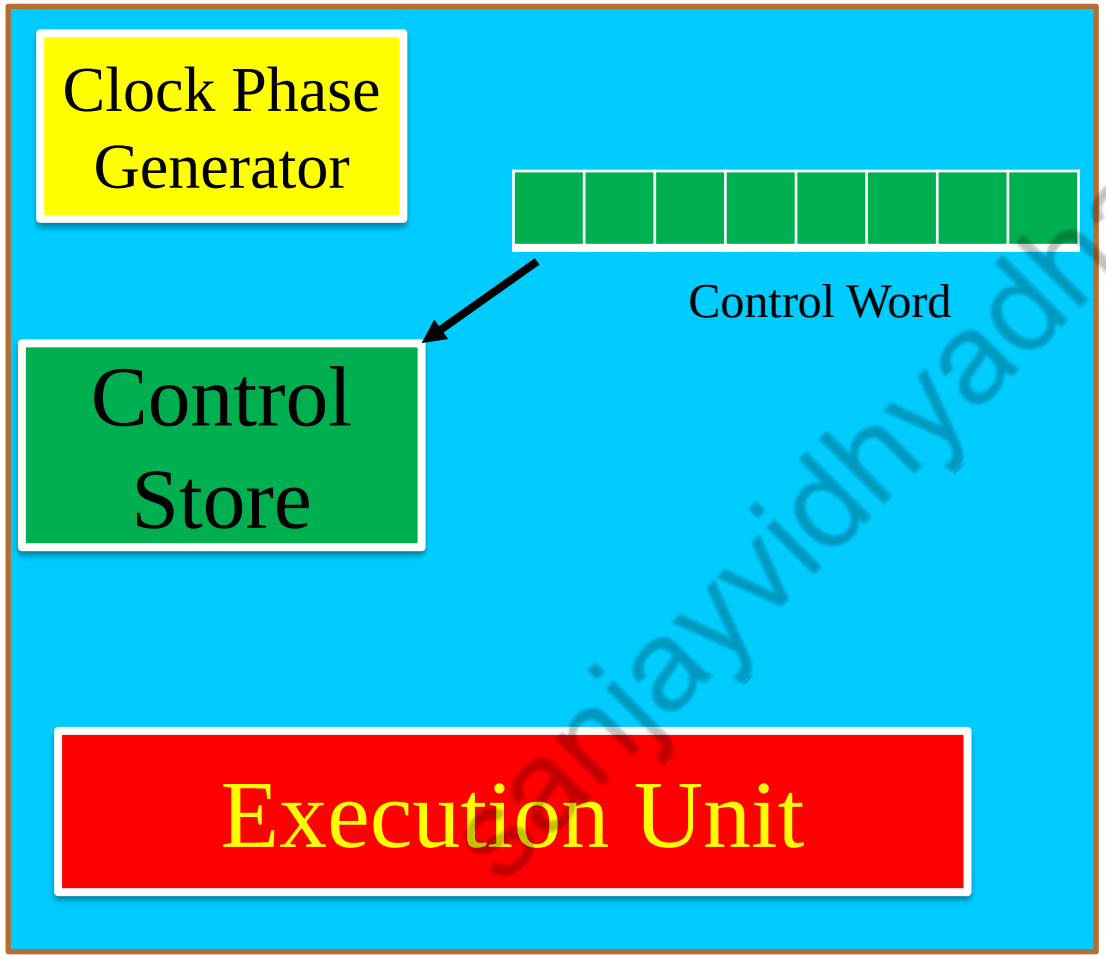
Do flowcharts for the instructions.

This modifies and refines the execution unit and develops the control store and control strategy.

The final execution unit is derived output

Flowchart Method

Step 2



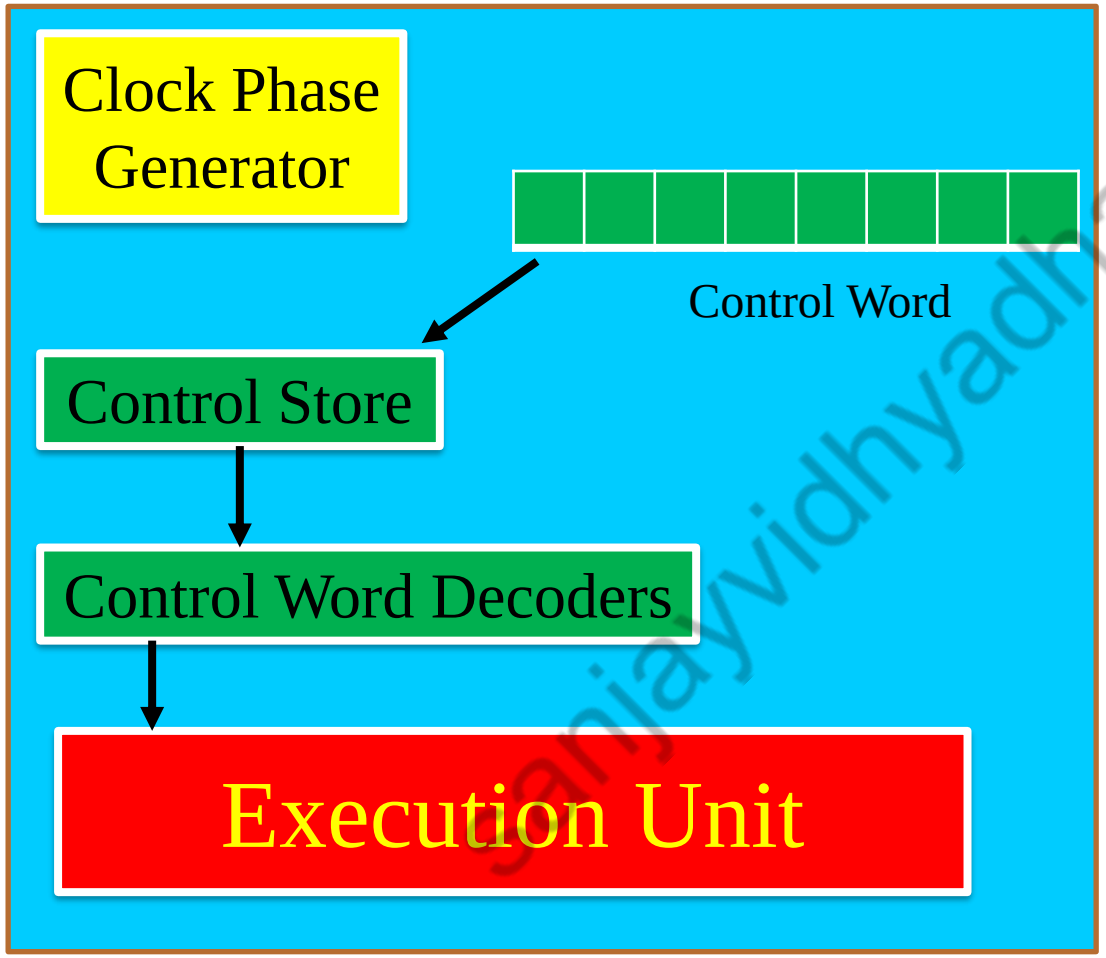
Once the flowcharts are fairly complete, derive the control word format using the flowchart states.

When the flowcharts are complete, so is the execution unit.

Control word format is derived output

Flowchart Method

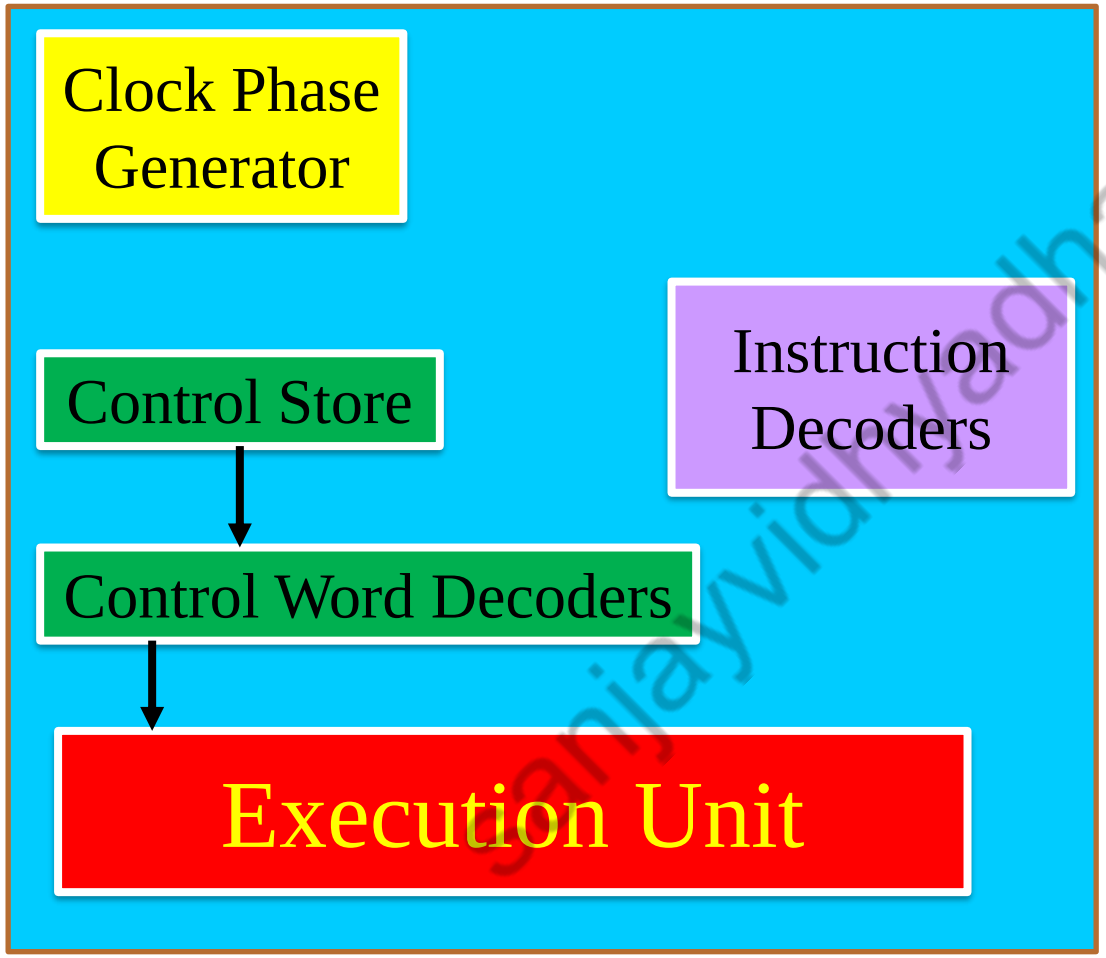
Step 3



After defining the control word format, you assign bit patterns to the control fields in a way that minimizes control word decoders between the control store and the execution unit.

Flowchart Method

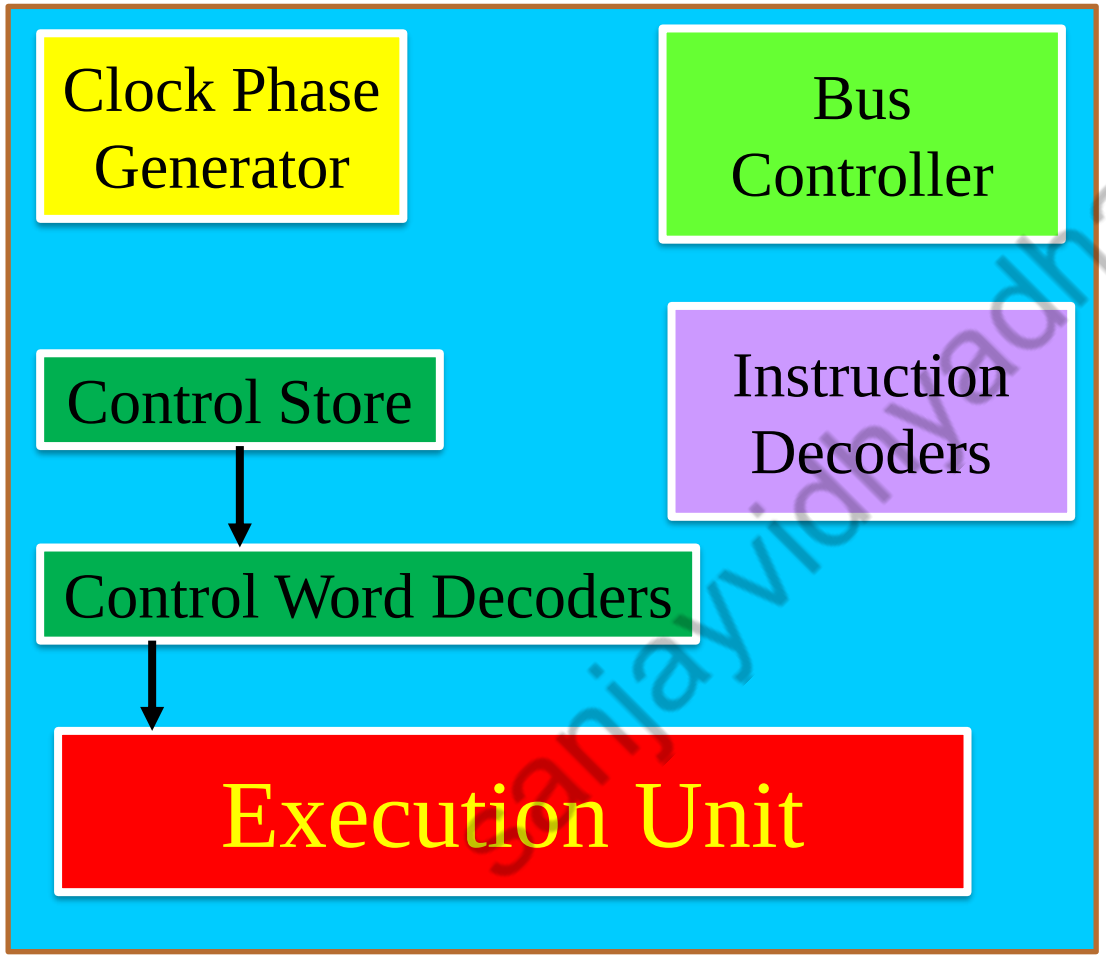
Step 4



Instruction decoders are defined by the flowcharts and the architecture specification.

Flowchart Method

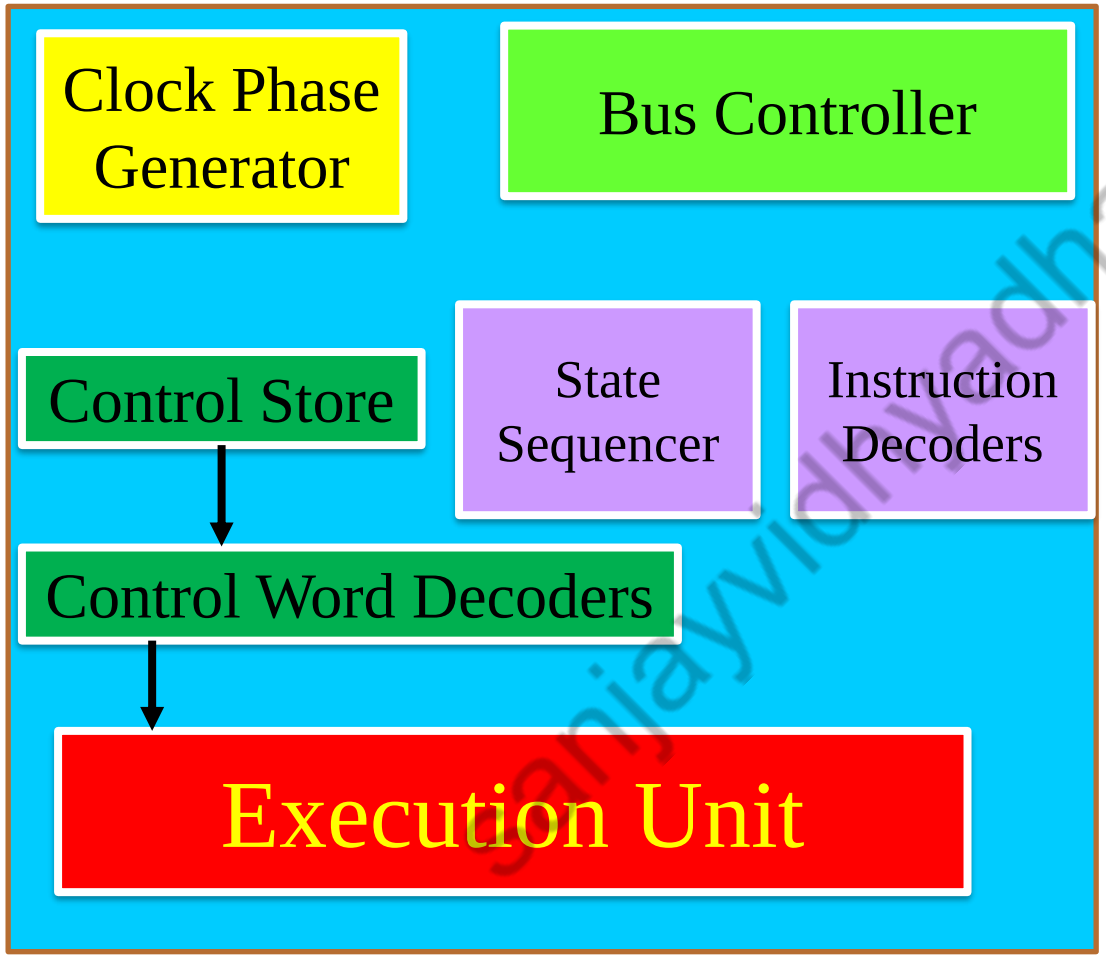
Step 5



Completed flowcharts, control word format, and the initial bus specification define the bus controller.

Flowchart Method

Step 5



Last is the logic of the state sequencer, the part of the chip that says what to do next. ("Where's the next control word?")

Once everything around it is defined, you build exactly what you need! (The state sequencer is derived output.).

Min Instruction Format

Instruction Format

First Word

OP	RX	MODE	RY
Operation Code	First Operand Reg	Second Operand Address Mode	Second Operand Reg

Second Word

Displacement

Optional, depending on second operand address mode

Programmer's Register Set

R0
R1
R2
R3
--

RN

Min Instruction Set

Instruction Set

- **ADD**
- **AND**
- **BZ** – Branch if Zero bit is set (Register Indirect Only)
- **LOAD**–Second operand is source and Rx is destination.
- **POP**–Post increment with register indirect only.
- **PUSH**–Pre decrement with register indirect only.
- **STORE**
- **SUB**
- **TEST**

Min Instruction Set

Address Modes

First Word

OP	RX	MODE	RY
Operation Code	First Operand Reg	Second Operand Address Mode	Second Operand Reg

Second Word

Displacement

1. AR Register direct. The result is stored in RY. For two operand instructions, RY also is an operand source
2. AL Register indirect. RY holds an operand address.
3. AB-Base (RY) plus displacement second instruction word) is an operand address.

Thank you