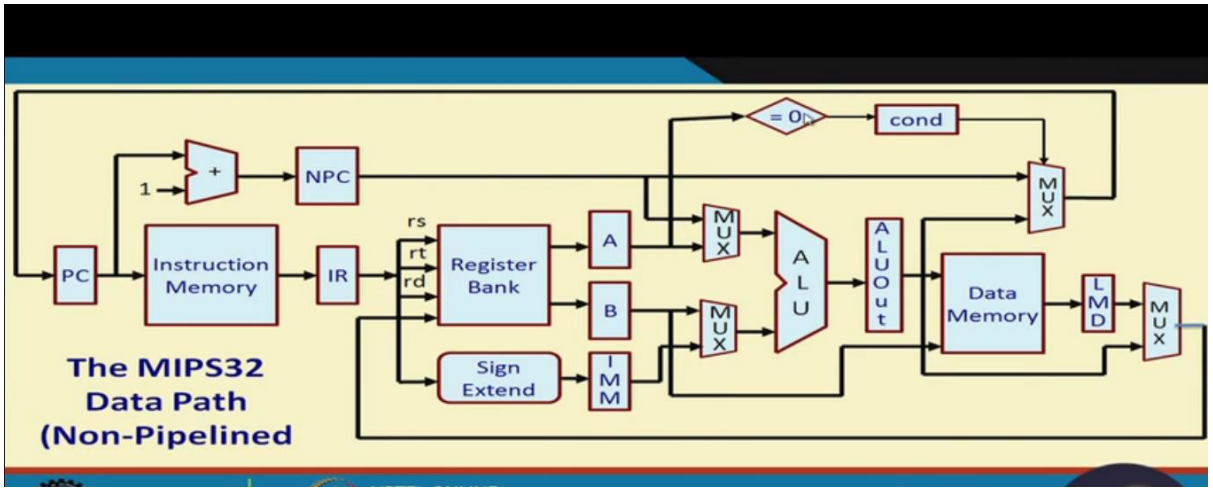
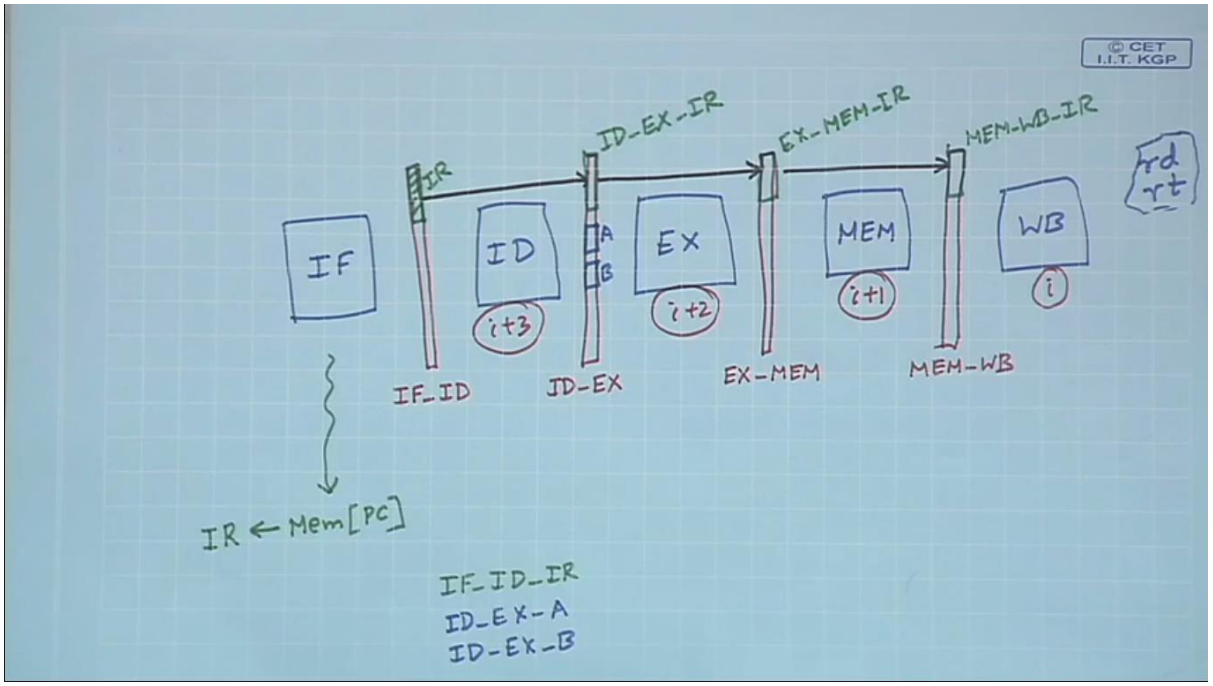
Design of High performance MIPS-32 Pipeline

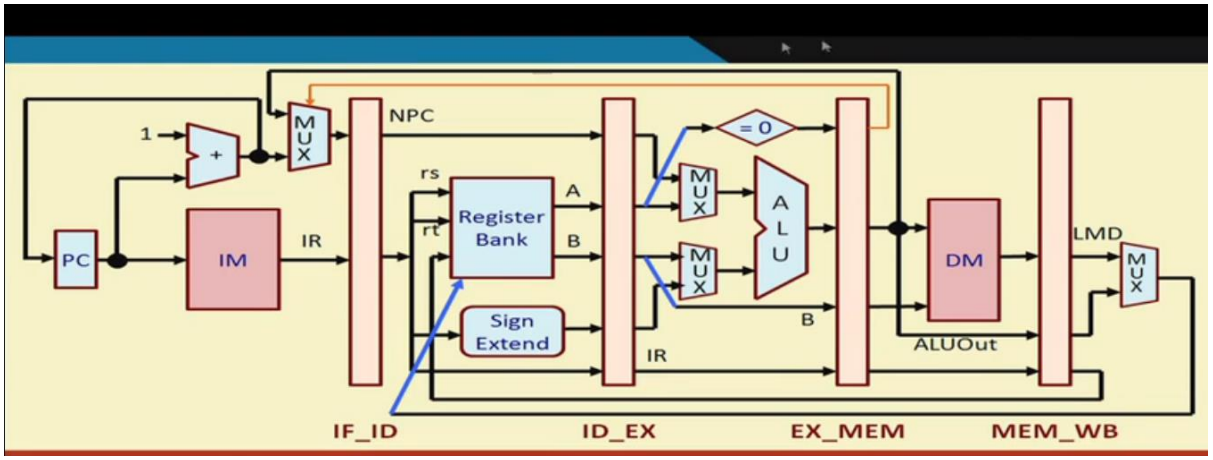
The objectives of this module are to discuss the basics of pipelining and discuss the implementation of the MIPS pipeline.

Pipelining is a particularly effective way of organizing parallel activity in a computer system. The basic idea is very simple. It is frequently encountered in manufacturing plants, where pipelining is commonly known as an assembly line operation. By laying the production process out in an assembly line, products at various stages can be worked on simultaneously. You must have noticed that in an automobile assembly line, you will find that one car's chassis will be fitted when some other car's door is getting fixed and some other car's body is getting painted. All these are independent activities, taking place in parallel. This process is also referred to as pipelining, because, as in a pipeline, new inputs are accepted at one end and previously accepted inputs appear as outputs at the other end. As yet another real world example, Consider the case of doing a laundry. Assume that Ann, Brian, Cathy and Daveeach have one load of clothes to wash, dry, and fold and that the washer takes 30 minutes, dryer takes 40 minutes and the folder takes 20 minutes. Sequential laundry takes 6 hours for 4 loads. On the other hand, if they learned pipelining, how long would the laundry take? It takes only 3.5 hours for 4 loads! For four loads, you get a Speedup = 6/3.5 = 1.7. If you work the washing machine non-stop, you get a Speedup = 110n/40n + 70 ≈ 3 = number of stages.

To apply the concept of instruction execution in pipeline, it is required to break the instruction execution into different tasks. Each task will be executed in different processing elements of the CPU. As we know that there are two distinct phases of instruction execution: one is instruction fetch and the other one is instruction execution. Therefore, the processor executes a program by fetching and executing instructions, one after another. The cycle time τ of an instruction pipeline is the time needed to advance a set of instructions one stage through the pipeline. The cycle time can be determined as


The MIPS32
Data Path
(Non-Pipelined
PC
Instruction Memory
IR
Register Bank
rs
rt
rd
Sign Extend
A
B
IMM
NPC
1
+
= 0
cond
MUX
ALU
ALUOut
Data Memory
LMD


© CET
I.I.T. KGP
IF
ID
EX
MEM
WB
IR
ID-EX-IR
EX-MEM-IR
MEM-WB-IR
A
B
i+3
i+2
i+1
i
rd
rt
IF_ID
ID-EX
EX-MEM
MEM-WB
IR ← Mem[PC]
IF_ID_IR
ID_EX-A
ID-EX-B


PC
IM
IR
1
+
MUX
NPC
rs
rt
Register Bank
A
B
Sign Extend
IMM
= 0
ALU
DM
ALUOut
LMD
IF_ID
ID_EX
EX_MEM
MEM_WB

The basic features of pipelining are:

• Pipelining does not help latency of single task, it only helps throughput of entire workload
• Pipeline rate is limited by the slowest pipeline stage
• Multiple tasks operate simultaneously
• It exploits parallelism among instructions in a sequential instruction stream
• Unbalanced lengths of pipe stages reduces speedup
• Time to “fill” pipeline and time to “drain” it reduces speedup
• Ideally the speedup is equal to the number of stages and the CPI is 1

Let us consider the MIPS pipeline with five stages, with one step per stage:

• IF: Instruction fetch from memory
• ID: Instruction decode & register read
• EX: Execute operation or calculate address
• MEM: Access memory operand
• WB: Write result back to register