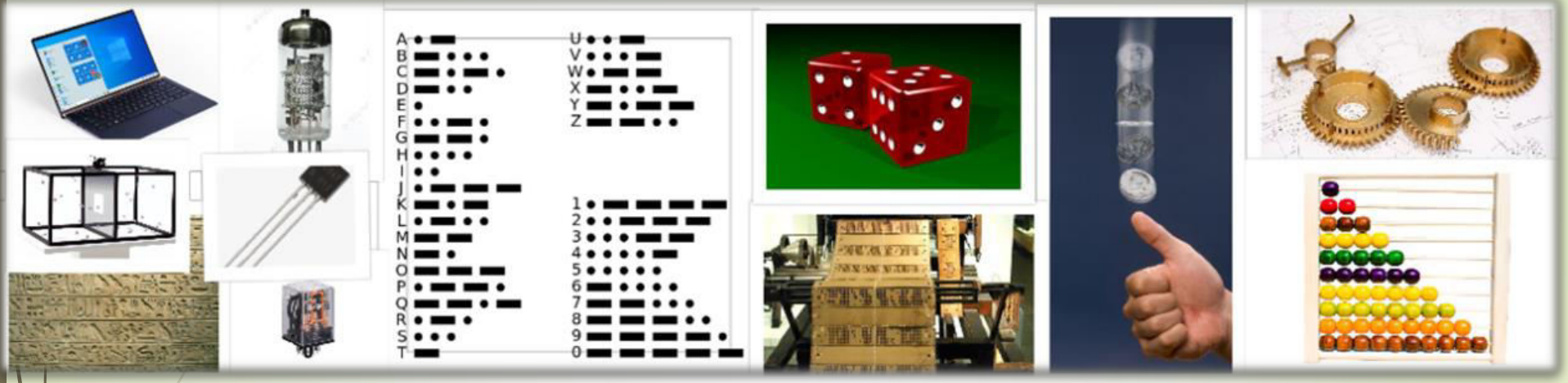


Computer Architecture (CS F342)

Design and Analysis of Instructions

Pipelined-based Design Fundamentals

Reason: we have to take the cycle completion time to be = critical stage (generally which is the memory stage)

It does increase but not 5 times

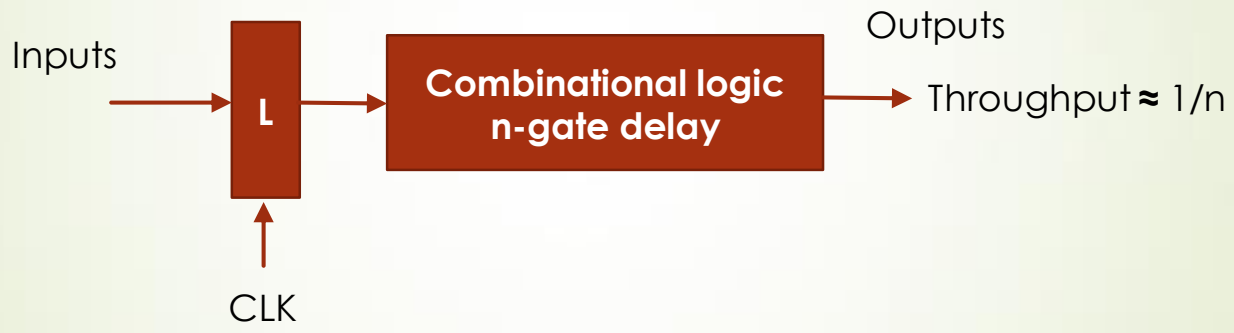
1

Problems of Multicycle Processor

- The fundamental problem
  - Split the slowest instruction, lw, 5-steps
  - Processor clock cycle time does not improve 5-times
- The steps take unequal length of time
- 5-non-architectural register and a additional multiplexer

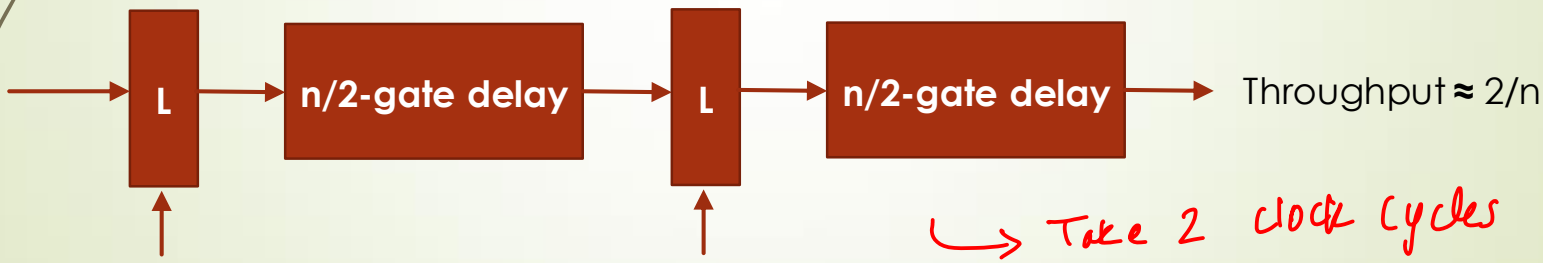
How to improve the processor performance

- Measure throughput (output/unit-time)
- For example



How to improve the processor performance

- For example



→ Take 2 clock cycles

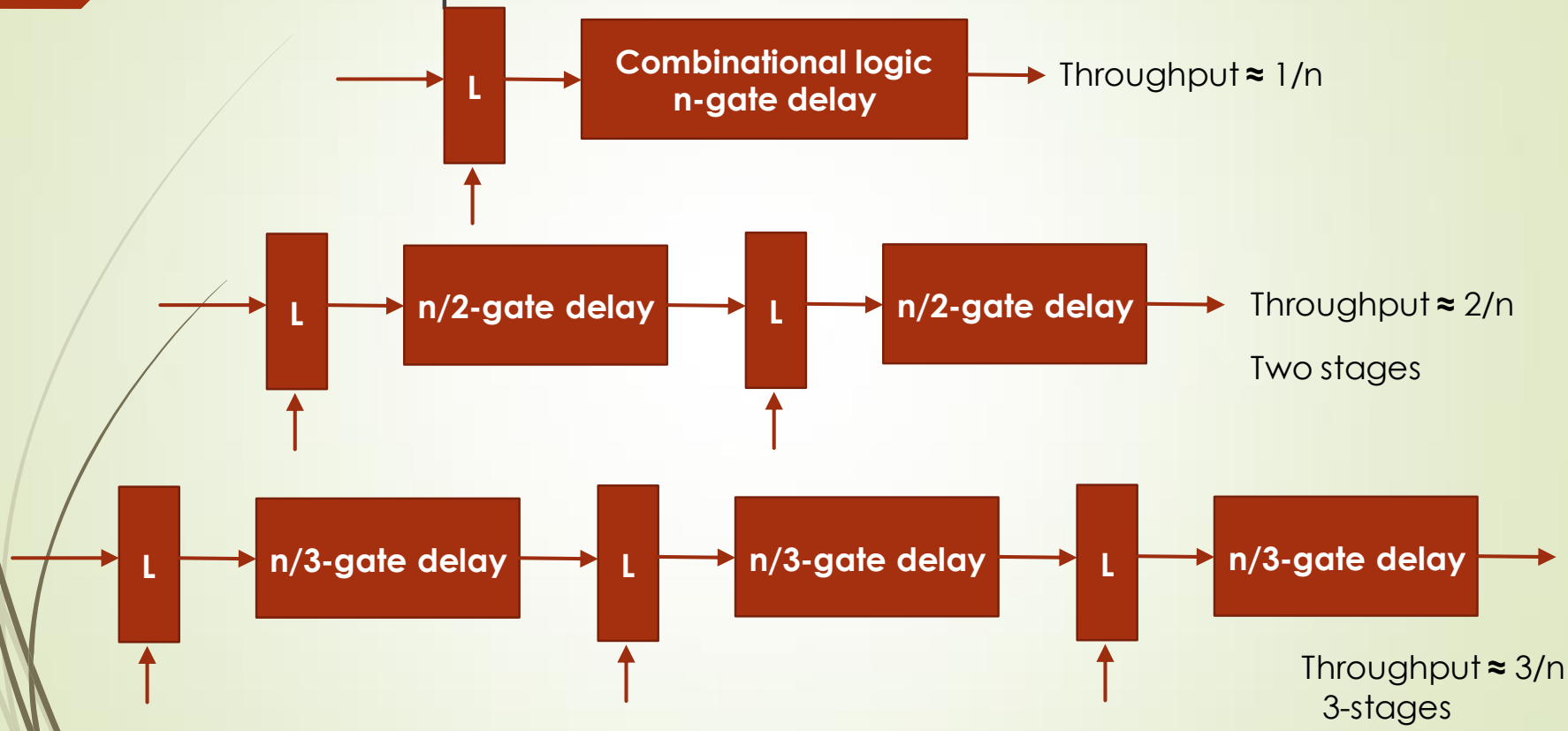
In 1st cycle, Ins 1 goes to second latch.

In 2nd " , Ins2 goes to " " and Ins1 goes to destn.

Clock Time Period set here = $\frac{n}{2}$ gate delay

How to improve the processor performance

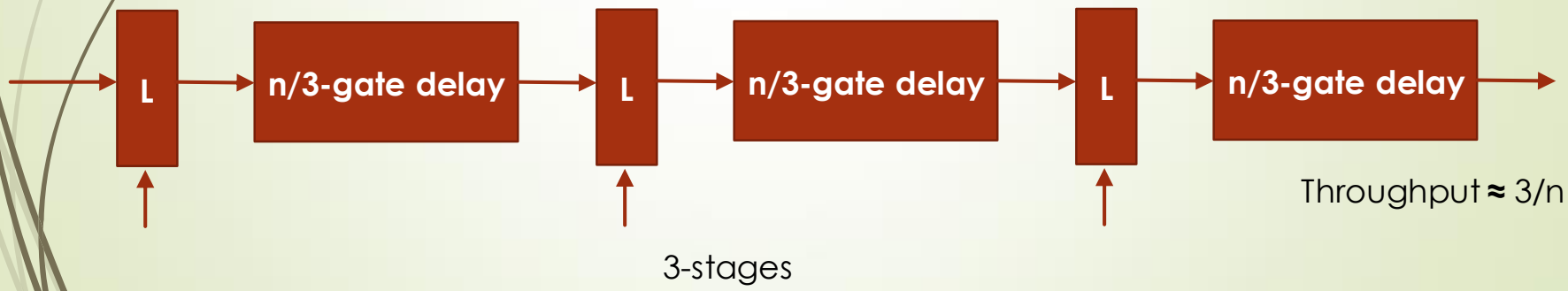
- For example



- Increasing such pieces would increase the no. of instructions processed per unit time
- However, latch cost and delay would be there too.

Pipelined-based Design Methodology

- k-fold increase in throughput
  - Increase in performance
    - Partitioning the logics
    - Adding new buffer
  - Inputs are overlapped in execution



Limitations of Pipelined-based Methodology

- Assumed inter-stage buffers does not introduce additional delay
- Increase in performance as the stages increase
- What if the stages increase to infinite

Limitations of Pipelined-based Methodology

- What if the stages increase to infinite
- Constrains
  - Clocking
  - Physical limitation on partitioning the logics
- Cost

Minimum clock period in Pipeline-based Systems

- Pipelined-based design
  - Combinational logic (F)
  - Latch (L)
- Max. propagation delay in F: T_M
- Min. propagation delay in F: T_m
- Proper latching delay: T_L

Minimum clock period in Pipeline-based Systems

- Consider the 2-scenarios
- Case-1:
  - Inputs x_1 applied at the stage at time T_1
  - Outputs of F must be valid at $T_1 + T_M$
  - Latching at L of the outputs must be valid until: $T_1 + T_M + T_L$

Minimum clock period in Pipeline-based Systems

- Case-2:
  - Inputs x_2 applied at the stage at time T_2
  - Effect of the outputs can be found at least at $T_2 + T_m$
  - Condition of 2-nd set of signals does not overrun the 1-st set: $T_2 + T_m > T_1 + T_M + T_L$
- Clock period (T): $T_2 - T_1 > T_M - T_m + T_L$
- Max. clocking rate cannot exceed 1/T

Earliest time at which the second instr^n reaches the latch must be more than the time it takes for the latch to be free of the first instr^n.

Minimum clock period in Pipeline-based Systems

- Clock period has two parts
  - T_M-T_m
  - T_L
- T_M-T_m ≈ 0
- T_L:
  - feedback loop and stabilizing of the signal
  - worst-case clock skew

This can effectively be made 0. But T_L would always be there.

Tradeoff between Cost and Performance

Total gate count is same => cost also same

- Cost of non-pipelined design: G
  - Gate count
- Cost of adding a latch: L
- Cost of k-stages pipelined design (C): G + k * L
- Cost of pipeline design increases linearly w.r.t depth of pipeline

Tradeoff between Cost and Performance

- Consider the latency in the non-pipeline design: T
- Performance or throughput: 1/T
- Throughput of pipelined design (P): 1/(T/k + S)
- The additional delay S because of latches
- P is a non-linear function of k

Time taken to process 1 instr^n in pipeline architecture.

Tradeoff between Cost and Performance

- Cost/performance ratio

$$\frac{C}{P} = \frac{G + k * L}{\frac{1}{(\frac{T}{k} + S)}}$$

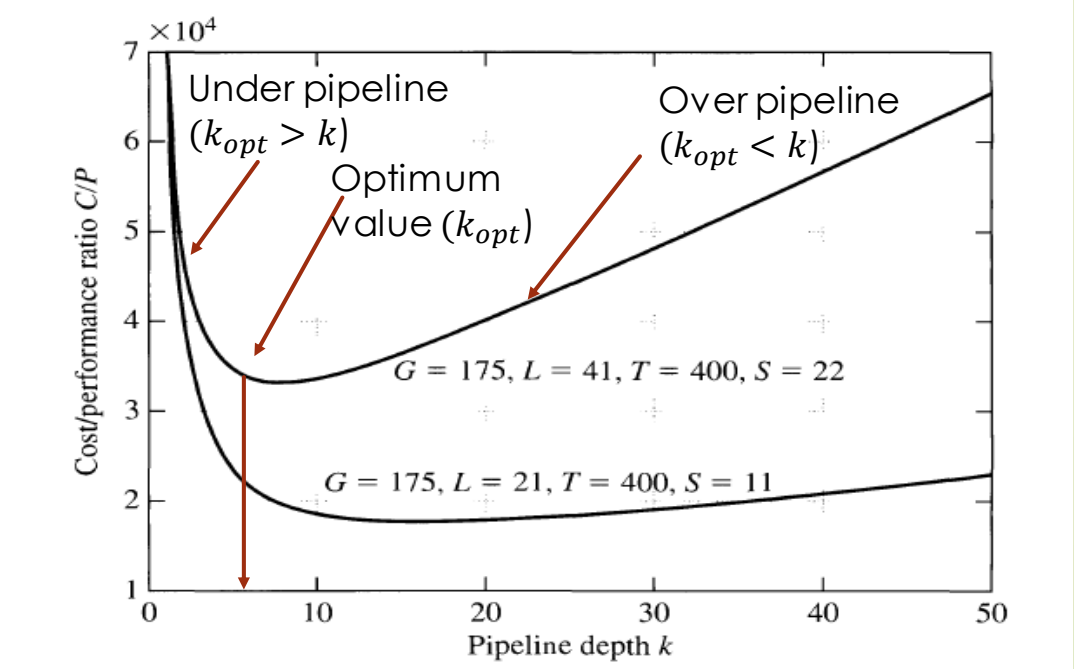
$$= LT + GS + LSk + \frac{GT}{k}$$

- Find minimum cost/performance ratio

Tradeoff between Cost and Performance

- Find minimum cost/performance ratio
- First derivative (w.r.t k)

$$k_{opt} = \sqrt{\frac{GT}{LS}}$$

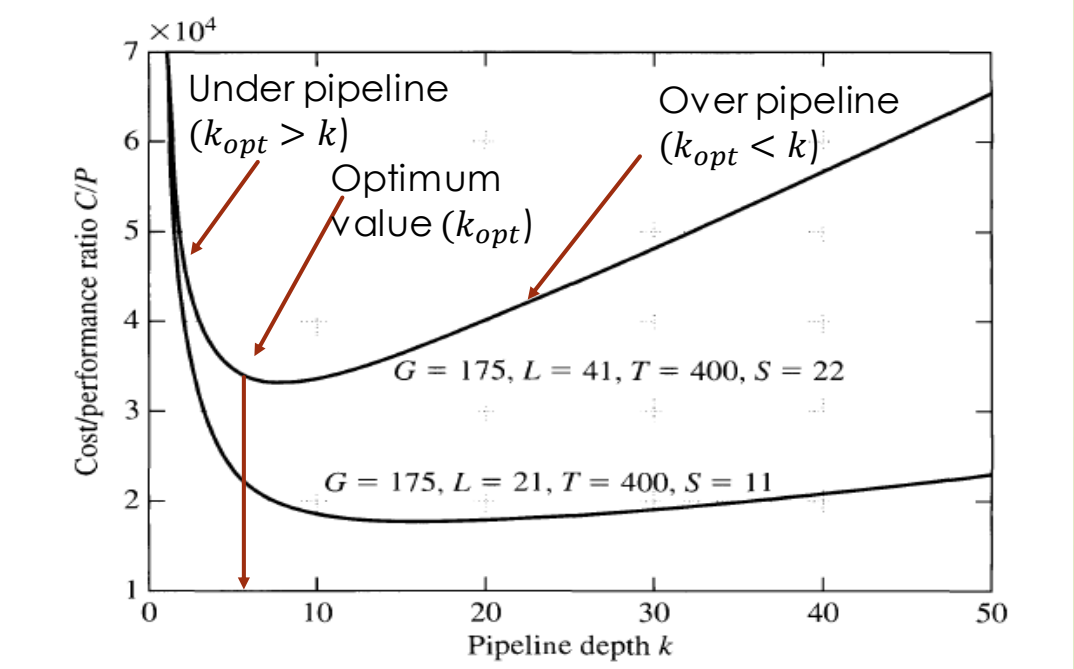


Tradeoff between Cost and Performance

- Find minimum cost/performance ratio
- First derivative (w.r.t k)

$$k_{opt} = \sqrt{\frac{GT}{LS}}$$

- **No consideration on dynamic behavior or runtime**



Pipeline Idealism

- Motivation: k-stages pipeline increases k-fold increase in throughput
- In reality this is difficult to achieve
- Are there hidden assumptions?

Pipeline Idealism

- Are there hidden assumptions?
- Yes, 3-assumptions, called <u>pipeline idealism</u>
  - Uniform sub-computations → As if division of components is possible
  - Identical computations → As if each unit takes same delay
  - Independent computations → As if the components are not interconnected (like || mein hota then ?)

Pipeline Idealism: Uniform sub-computations

- The computation can be evenly partitioned into uniform-latency sub-computations
  - No (minimize) internal fragmentation
  - No (minimize) additional delay by inter-stage buffer & clocking

Pipeline Idealism: Uniform sub-computations

- The computation can be evenly partitioned into uniform-latency sub-computations
  - No (minimize) internal fragmentation
  - No (minimize) additional delay by inter-stage buffer & clocking

Pipeline Idealism: Identical computations

- The same computation is to be performed repeatedly for all instructions (or for all input data set)
  - No (minimize) external fragmentation
  - All pipeline stages are always be utilized

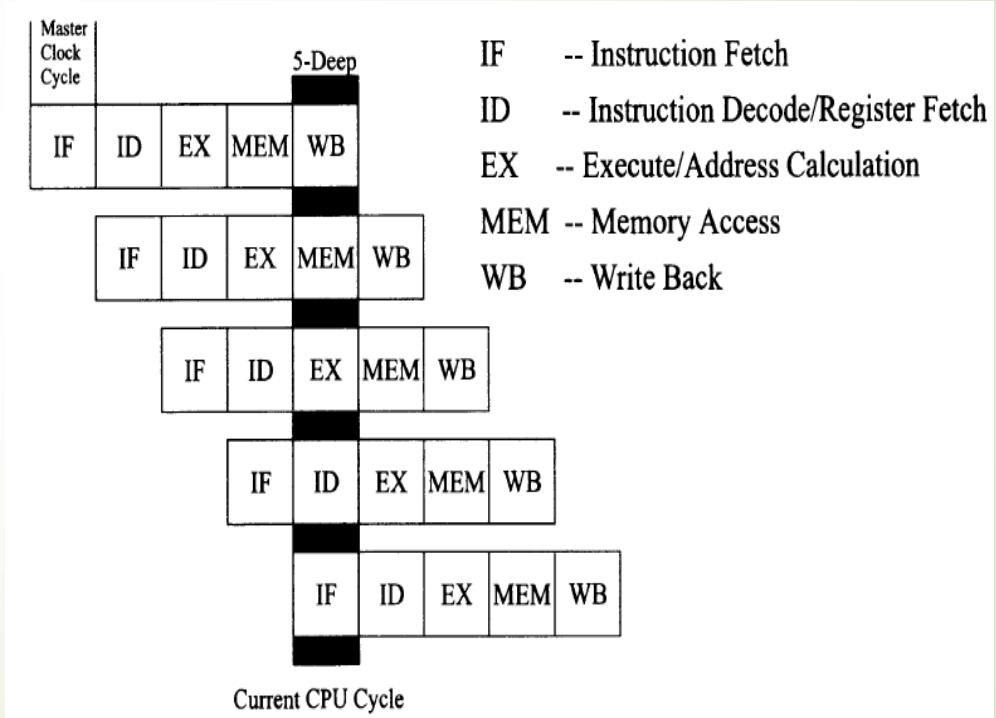
Pipeline Idealism: Independent Computations

- No data or control dependencies between any pair of computations
- Pipeline operates in streaming mode

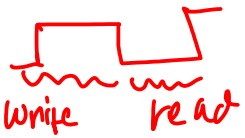
Actual Implementation

Instruction pipeline or pipelined processor

- An Implementation technique
  - Exploits parallelism among the instructions
  - Overlapping the execution
- Instruction cycle
  - A logical concept
- Machine cycle
  - A physical concept
- Fill time
- Drain time



Note: WB stage can coincide with ID → Reading and writing in Reg file both.

So, to avoid this,  ← 1 clock cycle, divide into two parts.

The Pipeline architecture is kind of mix of both multicycle and single cycle. The architecture is same as single cycle. However the single cycle is broken down into 5 cycles for processing each stage.

Summary

- Problems of multicycle design methodology
- Pipelined-based design methodology
- Limitations and optimum value for depth
- Pipeline idealism
- Instruction pipeline technique