Computer Architecture

CSEN 3104

Lecture 6

Dr. Debranjan Sarkar

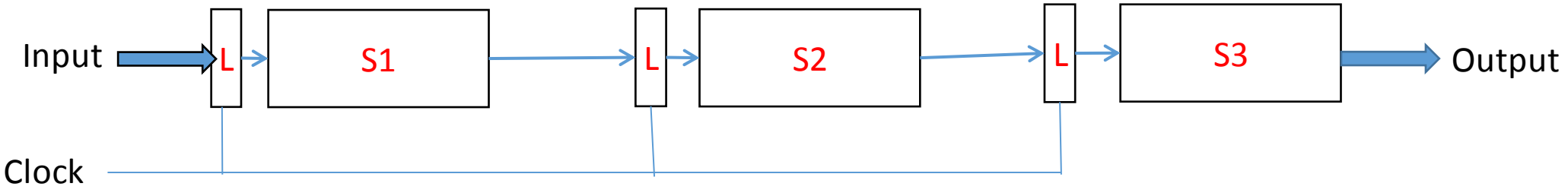
Basics of Pipelining

Basics of pipelining

- Modern CPUs employ a variety of speedup techniques viz.
  - cache memory
  - Pipelining etc..
- Pipelining allows the processing of several instructions to be partially overlapped (parallelism)
- Pipelining increases the overall instruction throughput.
- All the common steps involved in instruction processing by the CPU can be pipelined:
  - Instruction Fetching (IF)
  - Instruction decoding (ID)
  - Operand loading (OL)
  - Execution (EX)
  - Operand Storing (OS)

Principles of Pipelining

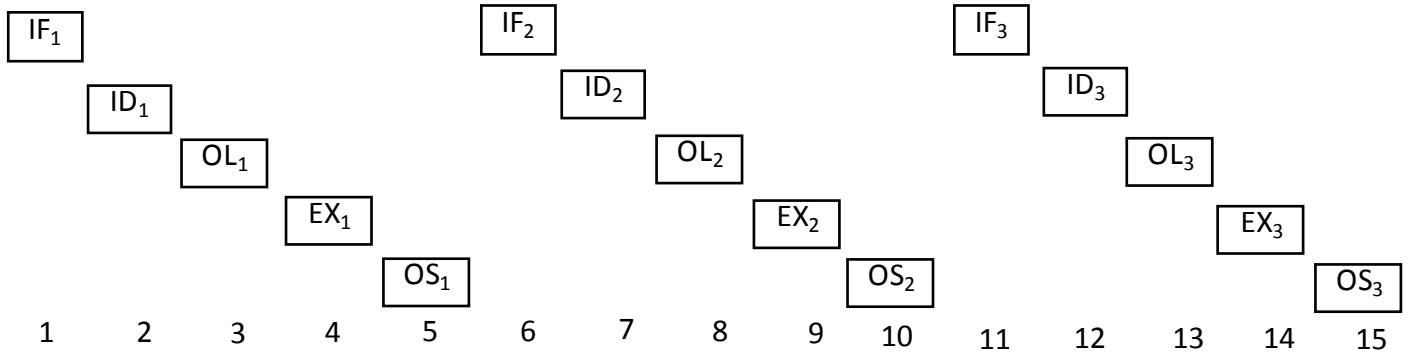
- Decomposes a sequential task into subtasks
- Each subtask is executed in a special dedicated stage
- These stages are connected with one another to form a pipe like structure.
- Instructions enter from one end and exit from another end
- Stages are pure combinational circuits for arithmetic or logic operations
- Result obtained from a stage is transferred to the next stage
- Final result is obtained after the instruction has passed through all the stages
- Stages are separated by high speed latches (or registers)
- All latches transfer data to the next stage simultaneously (on clock)



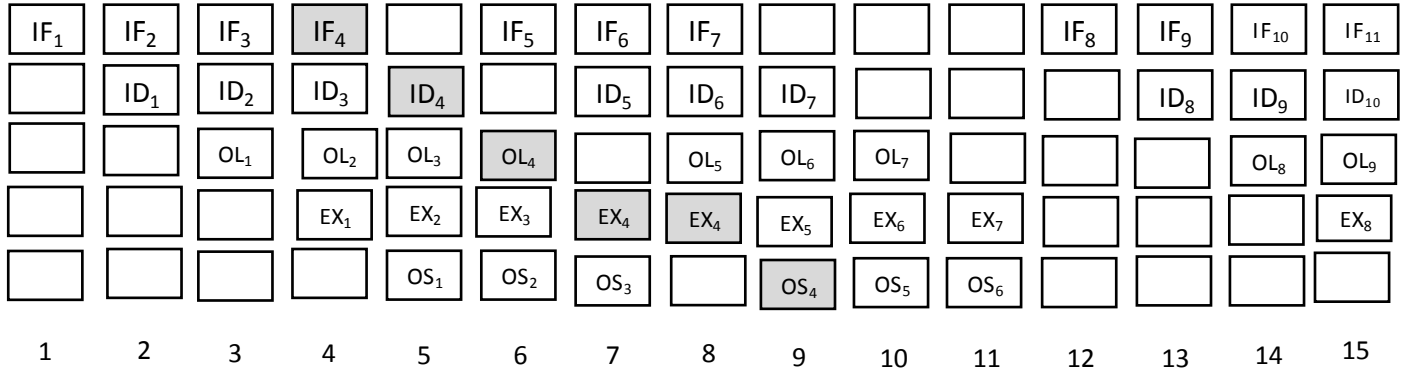
Instruction Processing :

(a) Non-pipelined (b) Pipelined

- Instruction Fetch (IF)
- Instruction Decode (ID)
- Operand Load (OL)
- Execution (EX)
- Operand Store (OS)
- Time (clock cycles)



- Instruction Fetch (IF)
- Instruction Decode (ID)
- Operand Load (OL)
- Execution (EX)
- Operand Store (OS)
- Time (clock cycles)



Pipelining

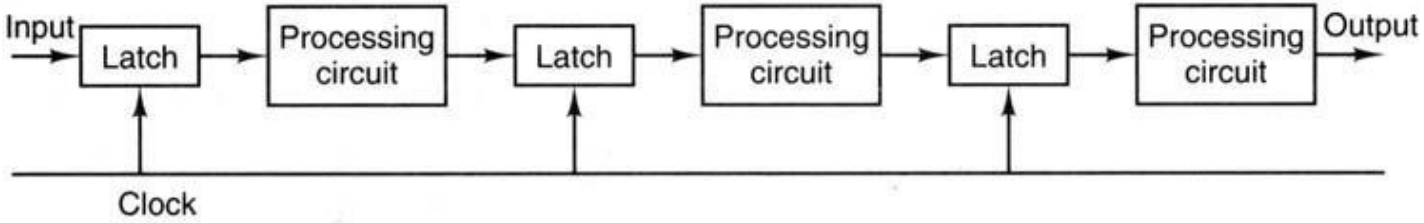
- In the example, upto five instructions can be overlapped,

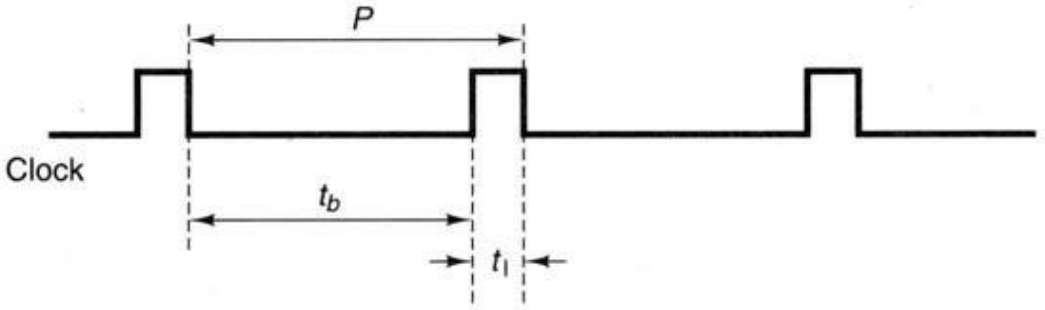
  provided, necessary pipeline stages are available

- Example of performance reducing delays
  - Instruction I_4 (shaded) uses the EX stage for two consecutive cycles
  - Instruction I_7 (branch), where the outcome of the I_7's EX step must be known before the location of the next instruction (I_8) to be processed can be identified.

Performance measure of a pipeline

- Speedup
- Efficiency
- Throughput





P: Clock period.
t_b: Maximum time for a stage to perform a function.
t_l: Time for latch to accept input data.

Speedup of a pipeline

- Let
  - *n* be the number of input tasks,
  - *m* the number of stages in the pipeline,
  - *P* the clock period
- The time required for the first input task to get through the pipeline = $1*m*P$
- The time required for the remaining tasks = $(n-1)*1*P$
- Note that after the pipeline has been filled, it generates an output on each clock cycle.
- Overall theoretical completion time (T_{pipe}) = $m*P + (n-1)*P$
- The pipeline will greatly outperform nonpipelined techniques, which require each task to complete before another task's execution sequence begins.

Speedup of a pipeline

- In a nonpipelined processor, the above sequential process requires a completion time of

  $T_{seq} = n * m * \tau$, where τ is the delay of each stage

- If we ignore the small storing time t_l that is required for latch storage (i.e., $t_l = 0$), then

  $T_{seq} = n * m * P$

- Now, *speedup* (S) may be represented as:

  $S = T_{seq} / T_{pipe} = n*m / (m+n-1)$

- The value S approaches m when $n \rightarrow \infty$. That is, the maximum speedup, also called ideal speedup, of a pipeline processor with m stages over an equivalent nonpipelined processor is m
- In other words, the ideal speedup is equal to the number of pipeline stages
- That is, when n is very large, a pipelined processor can produce output approximately m times faster than a nonpipelined processor
- When n is small, the speedup decreases. For n=1, the pipeline has the minimum speedup (= 1)

Efficiency and Throughput of a pipeline

- The efficiency E of a pipeline is defined as the speedup per stage
- If m is the number of stages, then

$$E = S/m = [n*m / (m+n -1)] / m = n / (m+n -1)$$

- The efficiency approaches its maximum value of 1 when $n \rightarrow \infty$
- When n=1, E will have the value $1/m$, which is the lowest obtainable value
- The throughput H, also called bandwidth, of a pipeline is defined as the number of input tasks it can process per unit of time
- When the pipeline has m stages, H is defined as

$$H = n / T_{pipe} = n / [m*P + (n-1)*P] = E / P = S / (mP)$$

- When $n \rightarrow \infty$, the throughput H approaches the maximum value of one task per clock cycle

Performance-cost-ratio of a pipeline

- The number of stages in a pipeline often depends on the tradeoff between performance and cost
- The optimal choice for such a number can be determined by obtaining the maximum value of a performance/cost ratio (PCR)
- PCR is defined as the ratio of maximum throughput to pipeline cost
- Throughput (H) =E /P
- As the maximum value of efficiency (E) is 1, so maximum throughput is 1/P
- Now P = $(t_{seq}/m) + t_l$
- Thus the maximum throughput that can be obtained with such a pipeline is

$$1/P = 1/[(t_{seq}/m) + t_l]$$

- The maximum throughput $1/P$ is also called the pipeline frequency
- The actual throughput may be less than $1/P$

Performance-cost-ratio of a pipeline

- The pipeline cost c_p can be expressed as the total cost of logic gates (c_g) and latches ($m*c_l$) used in m number of stages.

$$c_p = c_g + m*c_l$$

- Note that the cost of gates and latches may be interpreted in different ways:
  - Actual cost
  - design complexity
  - the area required on the chip or circuit board
- PCR = $1/\{[(t_{seq}/m) + t_l](c_g + m*c_l)\}$
- This equation has a maximum value when m = sqrt $\{(t_{seq} * c_g) / (t_l * c_l)\}$
- This value can be used as an optimal choice for the number of stages.

Thank You