Pipeline Basics

Lecture Notes # 14

ECE 366: Computer Architecture
Pipelining is the processing concept in which the entire processing flow is broken up into multiple stages, and a new data/instruction is processed by a stage potentially as soon as it is done with the current data/instruction, by contrast, the next data/instruction is processed after the entire processing of the previous data/instruction is complete.

1. Pipelining is the processing concept in which the entire processing flow is broken up into multiple stages, and a new data/instruction is processed by a stage potentially as soon as it is done with the current data/instruction.

2. In a non-pipelined processing, by contrast, the next data/instruction is processed after the entire processing of the previous data/instruction is complete.

3. A data/instruction’s processing in a pipeline is complete when it exits the last stage. If there are k stages each of equal delay, then the processing throughput (number of data/instruction processed per second) increases roughly by a factor of k.

4. The real-world has pipelining! E.g., water/oil pipelines.
1. Typically broken up into 3 to 5 stages:

   - Instruction Pipeline

   - Each stage now requires its own C.U., as control signals need to be generated simultaneously for the processing in each stage: E.g., while the Read stage is asserting the control signals for reading the operands for instruction \( i \), and other control signals are asserted in the Fetch stage for instruction \( i \); the Read stage is guaranteeing that all control signals for reading the operands for instruction \( i \) are asserted.

2. Input registers are required for each stage to store intermediate values and assuring the register while signals for instruction \( i \). 

3. Instruction pipeline is broken up into 3 to 5 stages: E.g., the input register
6. Some extra delay has been added to the total delay/latency for processing stage $j - 1$.

Stage $j - 1$.

Which, will be done, of course, only after stage $j$ assert its signal $done_j$ to inform it that new data/instruction has been written into its input register in its input register when it receives a signal $new_j$ from stage $j - 1$.

Also, the C.U. of stage $j$ can only start processing new data/instruction needed to communicate between the memory unit and the processor.

Further, the C.U. of stage $j$ can only store new data/instruction in the input register of instruction $i + 1$.

4. Further, the C.U. of stage $j$ can only store new data/instruction in the input register of instruction $i + 1$.

Store the ALU output and the write register held of instruction $i - 3$.

While the Write stage's input register would store the ALU output and the write register held of instruction $i - 3$, while the Write stage's input register would store the ALU output and the write register held of instruction $i - 3$ as the two input operands generated by the Read stage in its previous stage will store the opcode and write register helds of instruction $i - 2$, the input register of the Execute Read stage will store instruction $i - 2$, the input register of the Decode stage will store instruction $i - 1$, the input register of the
1. Let $t_i$ be the delay of stage $i$. Then, the non-pipelined time $T_{no\ pipe\ n}$ for processing $n$ instructions is

$$T_{no\ pipe\ n} = \sum_{i=1}^{n} t_i$$

2. Pipelined time $T_{pipe\ n}$ for processing $n$ instructions is

$$T_{pipe\ n} = \text{FillTime} + \{1\} \max_{k \leq u} (1 - u) + \text{FillTime} L$$

where the FillTime is the time taken for the first instruction to emerge from the pipeline, by which time, where the FillTime is the time taken to "fill" the pipeline, i.e., the time

$$\text{FillTime} = \sum_{i=1}^{n} t_i$$

1. Let $\mu$ be the delay of stage $i$. Then, the non-pipelined time $T_{no\ pipe\ n}$ for processing $n$ instructions is

$$T_{no\ pipe\ n} = \sum_{i=1}^{n} \mu$$

2. Pipelined time $T_{pipe\ n}$ for processing $n$ instructions is

$$T_{pipe\ n} = \text{FillTime} L$$
4. Example 1: If all $t_i$'s are equal and that value is $t$, then non-pipelined throughput is given by $(u)_{\text{no pipe}} = \frac{1}{1-t}$ and is in units of instructions/sec.

Instructions

3. Pipelined throughput is given by $(u)_{\text{pipe}} = \frac{1}{nT_{\text{pipe}}}$ for a large $n$ and is in units of instructions/sec.

(a) Timing view with "moving" pipeline

(b) A stationary-pipeline timing view

Done

Instructions

Time (current cc)

$T_{\text{pipe}} = \sum_{k=1}^{n} t_i$
5. Example 2: Consider a pipeline with 3 stages F, D, E with delays of 3 cc’s, 1 cc and 2 cc’s, respectively.

Non-pipelined throughput is $1/6$ instr/sec.

Pipelined throughput is $1/3$ instr/sec.

Pipelined throughput $\approx (3 \cdot (1 - u) + 6)/u = (u)_{\text{pipe}}.L$

Non-pipelined throughput $= (u)_{\text{non-pipe}}.L$

$6 \cdot u = (u)_{\text{pipe}} - (u)_{\text{non-pipe}}.L$

Throughput is $1/6$ instr/sec, while the pipelined throughput is $1/3$ instr/sec.