LPU
LYCEUM OF THE PHILIPPINES UNIVERSITY
CAVITE

Learn different. Live different.

CSCN07C – Computer Architecture and Organization

Pipelining, Parallelism and Instruction Level Parallelism

KLARENCE M. BAPTISTA, MIT

Pipelining

- Is a technique used in modern processors to improve performance by executing multiple instructions simultaneously.
- It breaks down the execution of instructions into several stages, where each stage completes a part of the instruction.
- These stages can overlap, allowing the processor to work on different instructions at various stages of completion, similar to an assembly line in manufacturing.

What is Pipelining?

- Pipelining is an arrangement of the CPU's hardware components to raise the CPU's general performance.
- In a pipelined processor, procedures called 'stages' are accomplished in parallel, and the execution of more than one line of instruction occurs.

What is Pipelining?

- Consider a water bottle packaging plant.
- For this case, let there be 3 processes that a bottle should go through, ensing the bottle(I), Filling water in the bottle(F), Sealing the bottle(S).
- It will be helpful for us to label these stages as stage 1, stage 2, and stage 3. Let each stage take 1 minute to complete its operation.
- Now, in a non-pipelined operation, a bottle is first inserted in the plant, and after 1 minute it is moved to stage 2 where water is filled.

What is Pipelining?

- Now, in stage 1 nothing is happening.
- Likewise, when the bottle is in stage 3 both stage 1 and stage 2 are inactive.
- But in pipelined operation, when the bottle is in stage 2, the bottle in stage 1 can be reloaded.
- In the same way, during the bottle 3 there could be one bottle in the 1st and 2nd stage accordingly.
- Therefore at the end of stage 3, we receive a new bottle for every minute. Hence, the average time taken to manufacture 1 bottle is:

What is Pipelining?

- Therefore, the average time intervals of manufacturing each bottle is:
- **Without pipelining** = 9/3 minutes = 3m

```
I F S | | | | | |
| | | I F S | | |
| | | | | | I F S (9 minutes)
```

- **With pipelining** = 5/3 minutes = 1.67m

```
I F S | |
| I F S |
| | I F S (5 minutes)
```

Design of a basic Pipeline

- In a pipelined processor, a pipeline has two ends, the input end and the output end. Between these ends, there are multiple stages/segments such that the output of one stage is connected to the input of the next stage and each stage performs a specific operation.
- Interface registers are used to hold the intermediate output between two stages. These interface registers are also called latch or buffer.
- All the stages in the pipeline along with the interface registers are controlled by a common clock.

Execution in a pipelined processor

- Execution sequence of instructions in a pipelined processor can be visualized using a space-time diagram.
- For example, consider a processor having 4 stages and let there be 2 instructions to be executed.

Execution in a pipelined processor

Stage / Cycle	1	2	3	4	5	6	7	8
S1	I_1				I_2			
S2		I_1				I_2		
S3			I_1				I_2	
S4				I_1				I_2

Non-Overlapped Execution

- Total time = **8 Cycle**

Execution in a pipelined processor

Stage / Cycle	1	2	3	4	5	6	7	8
S1	I_1				I_2			
S2		I_1				I_2		
S3			I_1				I_2	
S4				I_1				I_2

Non-Overlapped Execution

- Total time = **8 Cycle**

Stage / Cycle	1	2	3	4	5
S1	I_1	I_2			
S2		I_1	I_2		
S3			I_1	I_2	
S4				I_1	I_2

Overlapped Execution

- Total time = **5 Cycle**

Pipeline Stages

- **RISC** processor has 5 stage instruction pipeline to execute all the instructions in the RISC instruction set.

Pipeline Stages

Following are the 5 stages of the RISC pipeline with their respective operations:

- **Stage 1 (Instruction Fetch):** In this stage the **CPU** fetches the instructions from the address present in the memory location whose value is stored in the program counter.
- **Stage 2 (Instruction Decode):** In this stage, the instruction is decoded and register file is accessed to obtain the values of registers used in the instruction.
- **Stage 3 (Instruction Execute):** In this stage some of activities are done such as **ALU** operations.
- **Stage 4 (Memory Access):** In this stage, memory operands are read and written from/to the memory that is present in the instruction.
- **Stage 5 (Write Back):** In this stage, computed/fetched value is written back to the register present in the instructions.

Performance of a pipelined processor

- Consider a ‘k’ segment pipeline with clock cycle time as ‘Tp’.
- Let there be ‘n’ tasks to be completed in the pipelined processor.
- Now, the first instruction is going to take ‘k’ cycles to come out of the pipeline but the other ‘n – 1’ instructions will take only ‘1’ cycle each, i.e, a total of ‘n – 1’ cycles.

Performance of a pipelined processor

- So, time taken to execute ‘n’ instructions in a pipelined processor:

$$ET_{pipeline} = k + n - 1 \text{ cycles}$$
$$= (k + n - 1)\ Tp$$

- In the same case, for a non-pipelined processor, the execution time of ‘n’ instructions will be:

$$ET_{non\text{-}pipeline} = n * k * Tp$$

Performance of a pipelined processor

- So, speedup (S) of the pipelined processor over the non-pipelined processor, when ‘n’ tasks are executed on the same processor is:

$$S = \frac{\text{Performance of non-pipelined processor}}{\text{Performance of pipelined processor}}$$

- As the performance of a processor is inversely proportional to the execution time, we have,

$$S = ET_{\text{non-pipeline}} / ET_{\text{pipeline}}$$

$$\Rightarrow S = [n * k * Tp] / [(k + n - 1) * Tp]$$

$$S = [n * k] / [k + n - 1]$$

Performance of a pipelined processor

- When the number of tasks ‘n’ is significantly larger than k, that is, n >> k

$$S = n * k / n$$

$$S = k$$

- where ‘k’ are the number of stages in the pipeline. Also, **Efficiency** = Given speed up / Max speed up = S / S_{max} We know that Smax = k So, **Efficiency** = S / k **Throughput** = Number of instructions / Total time to complete the instructions So, **Throughput** = n / (k + n – 1) * Tp

Note: The cycles per instruction (CPI) value of an ideal pipelined processor is 1.

Performance of a pipelined processor

- Performance of pipeline is measured using two main metrices as **Throughput** and **latency**.

What is Throughput?

- It measure number of instruction completed per unit time.
- It represents overall processing speed of pipeline.
- Higher throughput indicate processing speed of pipeline.
- Calculated as, throughput= number of instruction executed/ execution time.
- It can be affected by pipeline length, clock frequency. efficiency of instruction execution and presence of pipeline hazards or stalls.

What is Latency?

- It measure time taken for a single instruction to complete its execution.
- It represents delay or time it takes for an instruction to pass through pipeline stages.
- Lower latency indicates better performance .
- It is calculated as, Latency= Execution time/ Number of instruction executed.
- It in influenced by pipeline length, depth, clock cycle time, instruction dependencies and pipeline hazards.

Advantages of Pipelining

- **Increased Throughput:** Pipelining enhance the throughput capacity of a CPU and enables a number of instruction to be processed at the same time at different stages.
- **Improved CPU Utilization:** From superimposing of instructions, pipelining helps to ensure that different sections of the CPU are useful.
- **Higher Instruction Throughput:** Pipelining occurring because when one particular instruction is in the execution stage it is possible for other instructions to be at varying stages of fetch, decode, execute, memory access, and write-back.

Advantages of Pipelining

- **Better Performance for Repeated Tasks:** Pipelining is particularly effective when all the tasks are accompanied by repetitive instructions, because the use of the pipeline shortens the amount of time each task takes to complete.
- **Scalability:** Pipelining is RSVP implemented in different types of processors hence it is scalable from simple CPU's to an advanced multi-core processor.

Disadvantages of Pipelining

- **Pipeline Hazards:** Pipelining may result to data hazards whereby instructions depends on other instructions; control hazards, which arise due to branch instructions; and structural hazards whereby there are inadequate hardware facilities.
- **Increased Complexity:** Pipelining enhances the complexity of processor design as well as its application as compared to non-pipelined structures.
- **Stall Cycles:** When risks are present, pipeline stalls or bubbles can be brought about, and this produces idle times in certain stages in the pipeline.

Disadvantages of Pipelining

- **Instruction Latency:** While pipelining increases the throughput of instructions the delay of each instruction may not necessarily be reduced. Every instruction must still go through all the pipeline stages and the time it takes for a single instruction to execute can neither reduce nor decrease significantly due to overheads.
- **Hardware Overhead:** It increases the complexity in designing the pipelining due to the presence of pipeline registers and the control logic used in managing the pipe stages and the data. This not only increases the cost of the wares but also forces integration of more complicated, and thus costly, hardware.

LPU
LYCEUM OF THE PHILIPPINES UNIVERSITY
CAVITE
Learn different. Live different.

Pipeline Processor

Pipeline Processor

- It consists of a sequence of m data-processing circuits, called stages or segments, which collectively perform a single operation on a stream of data operands passing through them.
- Some processing takes place in each stage, but a final result is obtained only after an operand set has passed through the entire pipeline.

Pipeline Processor

- As shown in figure, a stage $S_{(i)}$ contains a multiword input register or latch $R_{(i)}$, and a datapath circuit $C_{(i)}$, that is usually combinational.

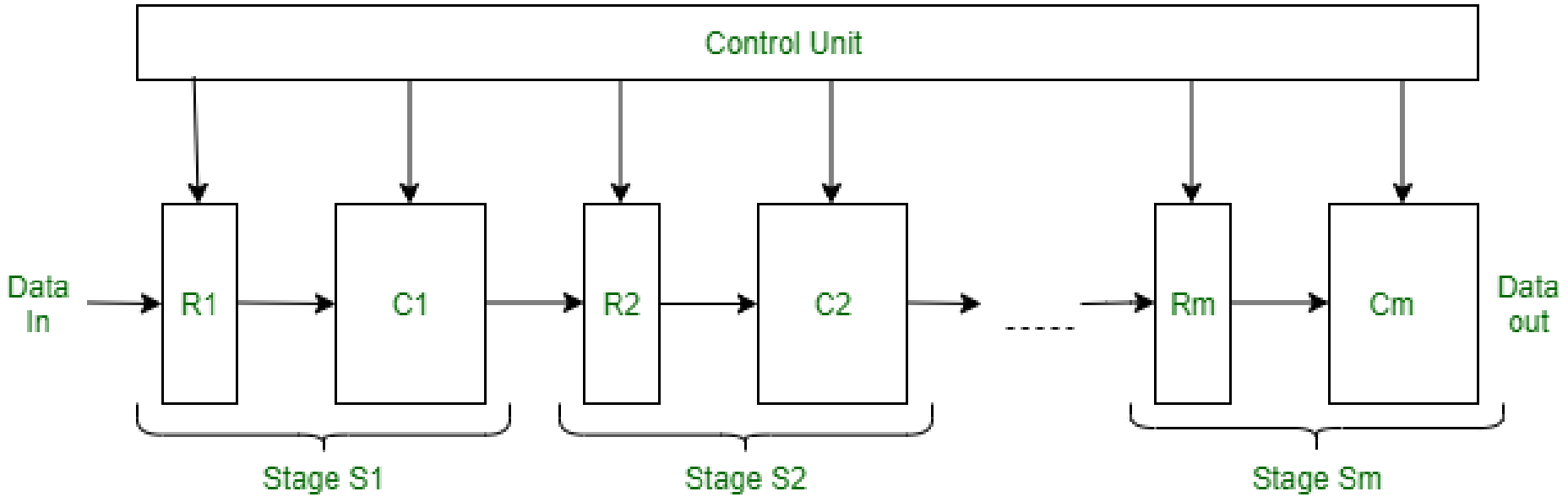

Figure - Structure of a Pipeline Processor

Principle of Pipelining

- pipeline is technique where multiple instructions are executed to overlapping fashion. Pipeline is divided into stages and their stages are connected to each other is cascade from to look like pipe like structure. In a pipeline system, each stage/segment consists of an input register followed by a combinational circuit. The register is used to hold data and combinational circuit perform operation on it.
- Here stages are pure combinational circuits performing arithmetic of logic operations. Over the data stream following through the pipe latches are high speed register for holding intermediate. Registers between the stages information flows between adjacent stages are under control of a common clock applied to all the latches simultaneously.

Performance evolution factor for pipelined computer:

1. clock period
2. Speedup
3. Efficiency
4. Throughput

LPU
LYCEUM OF THE PHILIPPINES UNIVERSITY
CAVITE
Learn different. Live different.

Instruction Level Parallelism

Instruction Level Parallelism

- **Instruction Level Parallelism (ILP)** is used to refer to the architecture in which multiple operations can be performed parallelly in a particular process, with its own set of resources – address space, registers, identifiers, state, and program counters.
- It refers to the compiler design techniques and processors designed to execute operations, like memory load and store, integer addition, and float multiplication, in parallel to improve the performance of the processors.

What is Instruction Level Parallelism?

- Instruction-level parallelism can also appear explicitly in the instruction set. VLIW (Very Long Instruction Word) machines have instructions that can issue multiple operations in parallel.
- The Intel IA64 is a well-known example of such an architecture.
- All high-performance, general-purpose microprocessors also include instructions that can operate on a vector of data at the same time.
- Compiler techniques have been developed to generate code automatically for such machines from sequential programs.

What is Instruction Level Parallelism?

- Examples of architectures that exploit ILP are VLIWs and superscalar Architecture.
- ILP processors have the same execution hardware as **RISC processors**.
- The machines without ILP have complex hardware which is hard to implement.
- A typical ILP allows multiple-cycle operations to be pipelined.

Example

Suppose, 4 operations can be carried out in a single clock cycle.

So there will be 4 functional units, each attached to one of the operations, branch unit, and common register file in the ILP execution hardware.

The sub-operations that can be performed by the functional units are Integer ALU, Integer Multiplication, Floating Point Operations, Load, and Store.

Let the respective latencies be 1, 2, 3, 2, 1.

Example

Let the sequence of instructions by

1. y1 = x1*1010
2. y2 = x2*1100
3. z1 = y1+0010
4. z2 = y2+0101
5. t1 = t1+1
6. p = q*1000
7. clr = clr+0010
8. r = r+0001

Sequential Record of Execution vs. Instruction-level Parallel Record of Execution

CYCLE	OPERATION
1	y1 = x1*1010
2	nop
3	nop
4	y2 = x2*1100
5	nop
6	nop
7	z1 = y1+0010
8	z2 = y2+0101
9	t1 = t1+1
10	p = q*1000
11	clr = clr+0010
12	r = r+0001

Fig. a

CYCLE	INT ALU	INT ALU	FLOAT ALU	FLOAT ALU
1	t1 = t1+1	clr = clr+0010	y1 = x1*1010	y2 = x2*1100
2	r = r+0001		p = q*1000	
3	nop			
4	z1 = y1+0010	z2 = y2+0101		

Fig. b

Instruction Level Parallelism (ILP) Architecture

- Instruction Level Parallelism is achieved when multiple operations are performed in a single cycle, which is done by either executing them simultaneously or by utilizing gaps between two successive operations that are created due to the latencies.
- Now, the decision of when to execute an operation depends largely on the compiler rather than the hardware.
- However, the extent of the compiler's control depends on the type of ILP architecture where information regarding parallelism given by the compiler to hardware via the program varies.

Classification of ILP Architectures

- The classification of ILP architectures can be done in the following ways –
- **Sequential Architecture:** Here, the program is not expected to explicitly convey any information regarding parallelism to hardware, like superscalar architecture.
- **Dependence Architectures:** Here, the program explicitly mentions information regarding dependencies between operations like dataflow architecture.
- **Independence Architecture:** Here, programme m gives information regarding which operations are independent of each other so that they can be executed instead of the ‘nops.

Advantages of Instruction-Level Parallelism

- **Improved Performance:** ILP can significantly improve the performance of processors by allowing multiple instructions to be executed simultaneously or out-of-order.

This can lead to faster program execution and better system throughput.

- **Efficient Resource Utilization:** ILP can help to efficiently utilize processor resources by allowing multiple instructions to be executed at the same time.

This can help to reduce resource wastage and increase efficiency.

Advantages of Instruction-Level Parallelism

- **Reduced Instruction Dependency:** ILP can help to reduce the number of instruction dependencies, which can limit the amount of instruction-level parallelism that can be exploited.

This can help to improve performance and reduce bottlenecks.

- **Increased Throughput:** ILP can help to increase the overall throughput of processors by allowing multiple instructions to be executed simultaneously or out-of-order.

This can help to improve the performance of multi-threaded applications and other parallel processing tasks.

Disadvantages of Instruction-Level Parallelism

- **Increased Complexity:** Implementing ILP can be complex and requires additional hardware resources, which can increase the complexity and cost of processors.
- **Instruction Overhead:** ILP can introduce additional instruction overhead, which can slow down the execution of some instructions and reduce performance.

Disadvantages of Instruction-Level Parallelism

- **Data Dependency:** Data dependency can limit the amount of instruction-level parallelism that can be exploited. This can lead to lower performance and reduced throughput.
- **Reduced Energy Efficiency:** ILP can reduce the energy efficiency of processors by requiring additional hardware resources and increasing instruction overhead. This can increase power consumption and result in higher energy costs.

FAQs on ILP

- **List some techniques to enhance Instruction-Level Parallelism?**

Some of the techniques include:

- *Reordering*
- *Out-of-Order Execution*
- *Speculative Execution*
- *Branch Prediction*

FAQs on ILP

- **State basic difference between ILP and Pipelining Process?**

Pipeline processing has the work of breaking down instruction execution into stages, where as ILP focuses on executing the multiple instructions at the same time.

Conclusion

- Pipelining is one of the most essential concepts and it improves CPU's capability to process several instructions at the same time across various stages. It increases immensely the system's throughput and overall efficiency by effectively determining the optimum use of hardware. On its own it enhances the processing speed but handling of pipeline hazards is critical for enhancing efficiency. It is thus crucial for any architect developing systems that will support HPC to have a war chest of efficient pipelining strategies that they can implement.