Overview of Pipelining Computing

Simmi Bagga[#1], Satinder Kaur [*2]

1. [#] *Assistant Professor, Sant Hira Dass Kanya Maha Vidyalaya, Kala Sanghian,Kapurthala, India,E-mail: simmibagga12@gmail.com*

2. [*] *Assistant Professor, GNDU, RC, Sathiala, India*

***Abstract*— Pipelining is one form of embedding parallelism or concurrency in a computer system. It refers to a segmentation of a computational process into several sub processes which are executed by dedicated autonomous units. In this paper, will discuss about the Pipelining computing and its various type and described the classification of pipelining. We also described various classifications of pipelining on the basis of Level of processing, Pipeline configuration and Type of instruction and data and how these types helps in effective computing. This paper also explains how pipeline allows multiple instructions to be processed at the same time.**

***Keywords*— Pipelining, Instruction Pipelining, Arithmetic Pipelining, Processor Pipelining.**

I. INTRODUCTION

The pipeline is divided into segments and each segment can execute its operation concurrently with the other segments. When a segment completes an operation, it passes the result to the next segment in the pipeline and fetches the next operation from the preceding segment. The final results of each instruction emerge at the end of the pipeline in rapid succession.

In computers, a pipeline is the continuous and somewhat overlapped movement of instruction to the processor or in the arithmetic steps taken by the processor to perform an instruction. An overlapped function that is used to increase the speed of system called Pipelining. It can divide tasks into subtasks and increase the speed of the entire system. Pipelining in a computer system is similar to assembly that is in industrial plant. To, achieve Pipelining one must subdivide the input task into sequence of subtasks. Pipelining is the use of a pipeline. Without a pipeline, a computer processor gets the first instruction from memory, performs the operation it calls for, and then goes to get the next instruction from memory, and so forth. While fetching (getting) the instruction, the arithmetic part of the processor is idle. It must wait until it gets the next instruction. With pipelining, the computer architecture allows the next instructions to be fetched while the processor is performing arithmetic operations, holding them in a buffer close to the processor until each instruction operation can be performed. The staging of instruction fetching is continuous. The result is an increase in the number of instructions that can be performed during a given time period

Pipelining is sometimes compared to a manufacturing assembly line in which different parts of a product are being assembled at the same time although ultimately there may be some parts that have to be assembled before others are. Even if there is some sequential dependency, the overall process can take advantage of those operations that can proceed concurrently. Computer processor pipelining is sometimes divided into an instruction pipeline and an arithmetic pipeline. The instruction pipeline represents the stages in which an instruction is moved through the processor, including its being fetched, perhaps buffered, and then executed. The arithmetic pipeline represents the parts of an arithmetic operation that can be broken down and overlapped as they are performed. Pipelines and pipelining also apply to computer memory controllers and moving data through various memory staging places.

Computer processors can handle millions of instructions each second. Once one instruction is processed, the next one in line is processed, and so on. A pipeline allows multiple instructions to be processed at the same time. While one stage of an instruction is being processed, other instructions may be undergoing processing at a different stage. Without a pipeline, each instruction would have to wait for the previous one to finish before it could even be accessed.

II. CLASSIFICATION OF PIPELINING

Classification of pipelining is based on:

1. Level of processing

2. Pipeline configuration

3. Type of instruction and data

1. Level Processing

- **Instruction pipelining:** This method is also called instruction look ahead. An instruction cycle has fetch opcode, decode opcode, compute operand addresses, fetch operands, and execute instructions. In instruction pipelining this process are over lapped.
- **Arithmetic pipelining:** The processor is segmented for pipelined operations in various formats.

2. Pipeline Configuration

- **Unifunction:** These are pipe line units with fixed or dedicated functions. They are not reconfigurable.
- **Multifunction pipelines:** They can be used for multiple purposes according to the need. They can also be reconfigured.

3. Type of instruction and data

- **Static pipelining:** It assumes only one functional configuration at a time. Static pipelines can be

unifunctional or multifunctional but piping needs continuous execution of instructions of the same type for its efficiency multifunctional would not be efficient.

- **Dynamic pipelining:** A dynamic pipeline processor allows multiple functional configurations to exist simultaneously. So a dynamic pipeline can be multifunctional.

III. TYPES OF PIPELINING

All Types of pipelining are as follows:

1. Instruction Pipelining.
2. Arithmetic Pipelining
3. Processor Pipelining.

Instruction Pipelining.

An instruction pipeline is a technique used in the design of computers and other digital electronic devices to increase their instruction throughput (the number of instructions that can be executed in a unit of time). Pipelining doesn't reduce the time it takes to complete an instruction; it increases the number of instructions that can be processed at once, thus reducing the delay between completed instructions.

The fundamental idea is to split the processing of a computer instruction into a series of independent steps, with storage at the end of each step. This allows the computer's control circuitry to issue instructions at the processing rate of the slowest step, which is much faster than the time needed to perform all steps at once. The term pipeline refers to the fact that each step is carrying data at once (like water), and each step is connected to the next (like the links of a pipe.)

The origin of pipelining is thought to be either the ILLIAC II project or the IBM Stretch project though a simple version was used earlier in the Z1in 1939 and the Z3 in 1941.

The IBM Stretch Project proposed the terms, "Fetch, Decode, and Execute" that became common usage. In this 'fetch' means that processor can fetch data from user and 'decode' means that it can convert data into machine language. After these terms processor can execute data.

The IBM Stretch Project proposed the terms, "Fetch, Decode, and Execute" that became common usage. In this 'fetch' means that processor can fetch data from user and 'decode' means that it can convert data into machine language. After these terms processor can execute data.

1. Instruction fetch
2. Instruction decode and register fetch
3. Execute
4. Memory access
5. Register write back

When a programmer (or compiler) writes assembly code, they make the assumption that each instruction is executed before execution of the subsequent instruction is begun. This assumption is invalidated by pipelining. When this causes a program to behave incorrectly, the situation is known as a hazard. Various techniques for resolving hazards such as forwarding and stalling exist.

Processors with pipelining are organized inside into stages which can semi-independently work on separate jobs. Each stage is organized and linked into a 'chain' so each stage's output is fed to another stage until the job is done. This organization of the processor allows overall processing time to be significantly reduced.

A deeper pipeline means that there are more stages in the pipeline, and therefore, fewer logic gates in each stage. This generally means that the processor's frequency can be increased as the cycle time is lowered. This happens because there are fewer components in each stage of the pipeline.

All instructions are not independent. In a simple pipeline, completing an instruction may require 5 stages. To operate at full performance, pipeline will need to run 4 subsequent independent instructions while the first is completing. If 4 instructions that do not depend on the output of the first instruction are not available, the pipeline control logic must insert a stall or wasted clock cycle into the pipeline until the dependency is resolved. Fortunately, techniques such as forwarding can significantly reduce the cases where stalling is required. While pipelining can in theory increase performance over an unpipelined core by a factor of the number of stages (assuming the clock frequency also scales with the number of stages), in reality, most code does not allow for ideal execution.

Pipelining does not help in all cases. There are several possible disadvantages. An instruction pipeline is said to be fully pipelined if it can accept a new instruction every clock cycle. A pipeline that is not has wait cycles that delay the progress of the pipeline. Some advantages are the cycle time of the processor is reduced, thus increasing instruction issue-rate in most cases. Some combinational circuits such as adders or multipliers can be made faster by adding more circuitry. If pipelining is used instead, it can save circuitry vs. a more complex combinational circuit. The main advantage of Pipelining is that all the components of the CPU will be active there by increasing the throughput.

Pipelining also has some disadvantages that are a non-pipelined processor executes only a single instruction at a time. This prevents branch delays (in effect, every branch is delayed) and problems with serial instructions being executed concurrently. Consequently the design is simpler and cheaper to manufacture. The instruction latency in a non-pipelined processor is slightly lower than in a pipelined equivalent. This is because extra flip flops must be added to the data path of a pipelined processor. A nonpipelined processor will have a stable instruction bandwidth. The performance of a pipelined processor is much harder to predict and may vary more widely between different programs.

Arithmetic pipelining

Arithmetic pipelines are differ from instruction pipelines in some tasks. They are generally synchronous. This means that each stage executes in a fixed number of clock cycles. In a synchronous pipeline, no buffering between stages is provided. Each stage can accept the data passed from last stage when data is produced. Arithmetic pipelining can be segmentized by

the processor. Well-known arithmetic pipelining processors are as the four-pipe stages are used in star-100, the eight pipe stages are used in the TI-ASC and up to 14 stages pipeline is used in cray-1 and upto.

The pipeline structures used for instruction pipelining may be applied in some cases to other processing tasks. If pipelining is to be useful, however, we must be faced with the need to perform a long sequence of essentially similar tasks. Large numerical applications often make use of repeated arithmetic operations for processing the elements of vectors and arrays. Architectures specialized for applications if this type often provides pipelines to speed processing of floating-point arithmetic sequences. This type of pipelining is called arithmetic pipelining.

Another important difference is that an arithmetic pipeline may be nonlinear. The "stages" in this type of pipeline are associated with key processing components such as adders, shifters, etc. Instead of a steady progression through a fixed sequence of stages, a task in a nonlinear pipeline may use more than one stage at a time, or may return to the same stage at several points in processing.

Processor pipelining

This refers to the pipeline processing of the same data whereas by a cascade of processors, each of which processors a specific task. The data stream passes the first processor with results stored in a secondary block which is also accessible by the second processor. The second processor passes the accessed result to the third processor and so on. This can proposed the following three pipelines:

- **Unifunction vs multifunction pipelines:** A pipeline unit with a fixed and dedecated function such as the floating point adder is called unifunction. The cray-1 has 12 unifunctional pipeline stages .a multifunction pipe may perform different functions, either at different time or the same time, by interconnecting different stages in the pipeline. the T1-ASC has four multifunction pipeline processors.
- **Static vs dynamic:** A static pipeline may ssume only one functional configuration at a time. Static pipeline can be either unifunctional or multifunctional. Pipelining is made possible in static pages only if instructions of the data type are to be executed continuously. The function performed by the static pipeline should not change frequently. A dynamic pipeline processor permits several functional configuration to exist simultaneously. In this sense, a dynamic pipeline must be multifunctional. On the other hand, a unifunctional pipeline must be static.
- **Scalar vs vector:** Depending on the instructions or data types, pipeline processors can be classified as scalar pipelines and vector pipelines. A scalar pipeline processors is sequence of scalar operands under the control of do loop. vector pipelines re commonly used to control the vector instruction. vector pipeline, use the hardware control and firmware control. The vector is the part of the scalar i.e. it is made from scalar.

IV. CONCLUSIONS

Pipelined processors represent an intelligent approach to speeding up instruction processing when the memory access time has improved to a certain extent. The pipeline structures used for pipelining may be applied in some cases to other processing tasks. If pipelining is to be useful, however, we must be faced with the need to perform a long sequence of essentially similar tasks. In this paper we described various classifications of pipelining on the basis of Level of processing, Pipeline configuration and Type of instruction and data.

REFERENCES

[1] MCINTYRE, D. "An introduction to the ILLIAC IV computer," Datamation (April 1970), 60-67.

[2] EVENSEN, A. J.; AND TROY, J.L. "Introduction to the architecture of a 288- element PEPE," in Proc. 1973 Sagamore Conf. on Parallel Processing, Springer-Verlag, N.Y. 1973, pp. 162-169.

[3] RUDOLPH, J. A. "A production implementation of an associattve array processor-- TARAN," in AFIPS 197~ Fall Jt. Computer Conf., AFIPS Press, Montvale, N.J., 1972.

[4] MAEVEL, O. E. "HAPPE--Honeywell associative parallel processing ensemble," in Proe. Symp. on Computer Architecture, Univ.of Florida, 1973.

[5] STANGA, D.C. "Univacll0 multiprocessor system," in AFIPS 1967 Spring Jr. Computer Conf., Thompson Book Co., Washington.

[6] Sunggu Lee ,Design of Computers and Other Complex Digital Devices, PrenticeHall

