Your Name

Instructor Name

Course Number

Date

Multi-threaded processing

Multithreading is a process that allows several threads to use a single processor's functional units in an overlapped fashion. Multithreading allows improved utilization of resources by allowing a ready thread to work and avoid these free resources. In order to understand multithreading, we must know about the difference between thread and a process. A process is an occurrence of a program executing on a computer. Thread is a dispatchable unit of effort within a process. A single-threaded application is which uses only one thread, and at a time, it is able to perform a single task. Context switching between threads and processes is the same, but for the thread, it is easier, faster, and cheaper as compared to a process. Every process allocates its own address space, but a thread within a process shares the same address space and other resources. It means to share data between threads is easy.

Multithreading works on thread-level parallelism (TLP). Parallelism denotes the concept of simultaneously executing multiple threads (Akhter and Jason). It duplicates the architectural state on every processor and shares only a single set of processor resources for implementation (Ed by Iannucci, Gao, and Halstead Jr.). The operating system schedules threads by treating different architectural positions as discrete logical processors. Logical processors provide support for sharing practical units of a single processor among different threads. Different architectures are using different sharing mechanisms. The kind of state a structure store chooses what sharing mechanism is required by this. The resources can be of replicated, partitioned, and shared type. A similar resource partitioning system is used in all multithreading methods, but difference occurs in their implementation in two cases, i.e., pipeline partitioning and thread scheduling policy. The main approaches to multithreading that TLP uses to improve resources utilization are (Nemirovsky and Tullsen):

Coarse-grained multithreading (CGMT)

Fine-grained multithreading (FGMT)

Simultaneous multithreading (SMT)

# **Coarse-grained multithreading (CGMT)**

It is also known as switch-on-event or block multithreading. In this type of multithreading, each core processor is attached to multiple hardware contexts. CGMT occurs on a switch of an event like cache miss, synchronization, FP operations, or quantum/timeout.

# **Fine-grained multithreading (FGMT)**

It is also known as interleaved multithreading; each processor is attached with multiple hardware contexts but can switch between them without any delay. Therefore, it can execute an instruction or a set of instructions from different threads in a cycle. In this multithreading, two instructions from thread cannot be in a pipeline simultaneously. FGMT switches to another thread cycle by cycle, and it improves pipeline consumption because of multiple threads. Moreover, in every cycle processor executes a different input/output thread. The example of FGMT is CDC 6600’s peripheral processing unit.

# **Simultaneous multithreading (SMT)**

In SMT, instructions are executed concurrently from multiple threads in the same cycle in order to keep multiple execution units operated. SMT share functional units flexibly and dynamically between numerous threads.

Multi-threaded technique is used to run numerous applications concurrently with enhanced speed. The multi-core processors, by using on-chip multithreading deliver better performance to cost ratios. Multiple threads can access common memory structures by using suitable [synchronization mechanisms](http://en.wikipedia.org/wiki/Synchronization_%28computer_science%29). For example, a program that downloads few documents simultaneously, then for each separate download thread is allocated. CPU-bound tasks utilize multithreading, which is beneficial for all core processors in modern computer world. Furthermore, there are many advantages of multi-threaded programming like:

* If a part of program is performing a extensive operation, or blocked due to some reason, by multitasking program can continue its running.
* Processes may share resources through shared memory and message passing; these techniques are prepared by programmer. However, by default, resources and memory of the process are shared by threads, which permits an application to have numerous threads of activity within identical address space.
* In processes, memory and resources need reasonable time and space due to which cost increases. On the other hand, threads share a memory of that process in which they are residing, so it is cheaper.
* **Multi-programming system having** multiprocessor architecture**,** where threads are running parallel, is more beneficent than a single threaded process. The multithreading technique divides a process in modules of small tasks performed by different processors.

# Works Cited

Akhter, Shameem and Roberts. Jason. *Multi-Core Programming: Increasing Performance through Software Multi-threading* . Intel Corporation, 2006.

Ed by Iannucci, Robert A., et al. *Multi-threaded Computer Architecture: A Summary of the State of the ART*. Springer Science & Business Media, 2012.

Nemirovsky, Mario and Dean Tullsen. *Multithreading Architecture*. Morgan & Claypool Publishers, 2013.