Chitkara

Programming Paradigms

OnTrack Submission

Task M2_S1P

Submitted By:
Ankit SINGH
2210994758
2024/02/10 23:46

Tutor:
Dr. Shanky Kansal

Outcome	**Weight**
ULO1	♦♦♦◊◊
ULO2	♦♦♦♦◊
ULO3	♦♦♦◊◊

..

February 10, 2024

SIT315
Student Name - Ankit singh
Student ID - 2210994758

Activity 1 - Parallel and concurrent programming

1. **Definitions and Examples:**
    - **Concurrent programming:** Concurrent programming involves the execution of multiple tasks or processes seemingly simultaneously. However, in reality, these tasks may be interleaved or executed in parallel but not necessarily simultaneously. Example: Multi-threaded programming, where different threads of execution operate independently but may share resources or communicate with each other.
    - **Parallel programming:** Parallel programming involves executing multiple tasks simultaneously using multiple processing units. These tasks may be broken down into smaller sub-tasks and executed concurrently for faster processing. Example: Parallel computing using GPU, where computations are distributed across multiple cores for faster processing.
    - **Distributed computing:** Distributed computing involves distributing computational tasks across multiple networked computers. Each computer works independently, and communication between them is necessary for coordination and sharing of resources. Example: Hadoop MapReduce, where data processing tasks are distributed across a cluster of computers for efficient processing of large datasets.
2. **Amdahl's Law:**
    - Amdahl's Law states that the speedup of a program from parallelization is limited by the proportion of the code that must be executed sequentially. It is expressed mathematically as: S(p)=1(1−f)+fpS(p)=(1−f)+pf1, where S(p)S(p) is the speedup with pp processors, and ff is the fraction of the code that cannot be parallelized.
3. **Tradeoff between Speedup of FPSQR and Floating-Point Operations:**
    - If 20% of the total execution time is spent in calculating FPSQR and 50% in executing floating-point operations:
        - Speedup execution of FPSQR by 10 times: FPSQR's contribution to the overall execution time becomes 20%×110=2%20%×101=2%.
        - Speedup execution of all floating-point operations by 2 times: Floating-point operations' contribution to the overall execution time becomes 50%×12=25%50%×21=25%.
    - Comparing the contributions, the second scenario yields a better tradeoff as it reduces the overall execution time more significantly.

Activity 2 - Flynn's taxonomy

1. **Architecture of Each Classification:**

    - **SISD:** Single-Instruction, Single-Data architecture operates with a single instruction stream and a single data stream. Example: Traditional von Neumann computers.
    - **MISD:** Multiple-Instruction, Single-Data architecture has multiple instruction streams operating on a single data stream. It's uncommon in practice.
    - **SIMD:** Single-Instruction, Multiple-Data architecture operates with a single instruction stream processing multiple data streams simultaneously. Example: GPUs.
    - **MIMD:** Multiple-Instruction, Multiple-Data architecture has multiple instruction streams operating on multiple data streams concurrently. Example: Cluster of networked computers.
2. **Array Processor:**
    - Array Processor is a subcategory of SIMD where multiple processing units operate on arrays of data simultaneously. Example: GPUs, which excel at processing large sets of data in parallel.
3. **MIMD Subcategories:**
    - **SPMD (Single Program, Multiple Data):** Each processing node runs the same program but works with different sets of data. Example: Parallel computing clusters performing weather simulations.
    - **MPMD (Multiple Program, Multiple Data):** Each processing component runs different programs on different data streams simultaneously. Example: Distributed computing environments running multiple applications on network nodes.

Activity 3 - Timers

1. **Blink LED every 2 seconds:** Modify code to set the timer to interrupt every 2 seconds and toggle the LED state in the interrupt service routine.
2. **Modify startTimer method:** Change the method signature to accept a double parameter representing the timer frequency. Use this parameter to calculate and set the correct values for the timer registers.
3. **Use potentiometer sensor:** Read the value from the potentiometer connected to pin A4 to determine the timer frequency dynamically.

Github link:- https://github.com/Ankitshreyasingh/315pp/upload/main

Thinkercad link:-
https://www.tinkercad.com/things/kipCr9KetWD-copy-of-sit315-seminar-3/editel?tenant=circuits