Structures and Enumerations

Instruction Structure

`Instruction` structure represents an instruction with a name and duration.

ProcessType Enumeration

`ProcessType` enumeration represents different types of processes: Platinum, Gold, Silver.

Process Structure

`Process` structure represents a process with various attributes such as name, arrival time, priority, process type, instructions, execution status, and other statistics.

Functions

readInstructions

`readInstructions` reads instructions from the "instructions.txt" file and stores them in an array of `Instruction` structures.

getInstructionDuration

`getInstructionDuration` returns the duration of an instruction given its name.

getNextArrivalTime

`getNextArrivalTime` finds and returns the next arrival time of a process when the CPU is idle.

readProcesses

`readProcesses` reads process definitions from the "definition.txt" file, allocates memory for instructions, and initializes process attributes.

executeProcesses

`executeProcesses` simulates the execution of processes based on the defined scheduling algorithm. It utilizes a ready queue, and processes of different types execute their instructions accordingly. The function takes four variables: array of processes, process count, array of instructions, and current time integer. The function starts by initializing some variables and enters a while loop until all processes are completed.

In the while loop, the function first checks if the CPU is idle for more than 4 cycles. If so, the current time is assigned the value of the next process' arrival time with a call to `getNextArrivalTime` function.

The function then iterates over all processes in the array to choose the highest priority one. If the process has arrived, not completed, and there is no currently running platinum process, it checks the type of the process. If it is a platinum process, it is added to the ready queue if not already in the queue. Another loop compares all platinum processes available, and the function adds the one with the highest priority to the front of the ready queue and skips to the execution part. If the type of the process is not platinum, it is added to the ready queue if not already in the queue, and the function finds the highest priority process with a loop and adds it to the front of the ready queue.

The execution part starts by decrementing the `idleCPUFlag` to indicate that the CPU is not idle. It then checks the type of the process. If it is a platinum process, the function starts executing by checking if a context switch occurred. It keeps the name of the executed process and checks if the process to be executed is different. If a context switch occurred, the current time is incremented by 10. The `platFlag` is set to indicate that a platinum process is executing and cannot be preempted. The `getInstructionDuration` function is called to get the current instruction duration, and it is added to both the current time and the execution time of the process. The instruction index is incremented by one, and the function checks whether the process is completed by comparing the instruction index with the exit index. If it is completed, necessary values are calculated, the platinum flag is set to 0 again, and the process is removed from the ready queue.

If the type is gold, the function first checks whether the time quantum has passed 120 ms. If so, the function sets the correct flags and checks if the process should be promoted to platinum, promoting it if so. Then the arrival time of the process is set to be equal to the current time, allowing the function to compare the processes accordingly when two processes with the same priorities have arrived. If the process has not passed 120 seconds, then the instruction of the process is executed similarly to platinum instructions, with the only difference being that this time `CPUBurst` and `timeQuantumUsed` variables are adjusted to keep track of the quantum used. If the type is silver, all things are the same as gold except this time the time quantum is 80, and the process is promoted to gold after taking 3 CPUBursts. After the execution of one instruction, the function checks whether all processes are completed and breaks the while loop if so. Finally, the function increments the `idleCPUFlag` to check whether the while loop is cycled without executing an instruction and then loops again until all processes are completed.

Main Function

The main function reads instructions and processes from files using the right functions and sets the variables. It then calls the `executeProcesses` function to

simulate the execution. After the execution, it calculates and prints the average waiting time and turnaround time for all completed processes. After printing, the program frees the allocated memory for instructions in each process before exiting.