SYSTEM UTILITIES Linux x64 (NASM)

Team Members:

1. **Joseph George Wahba**
2. **Menna-Allah Ahmed**
3. **Abdelhalim Ramadan**
4. **Zeyad Muhamed Yehya**

- Dr. Mahmoud Ouf
- ENG. Rahaf Mahmoud

1. System Overview

The System Utilities Suite is a high-performance command-line application developed in **x64 Assembly** for Linux systems. It provides direct hardware interaction and system monitoring by interfacing with the CPU via the CPUID instruction and performing direct Linux kernel syscalls.

2. Functional Specifications

2.1 System Information Discovery

- **Hardware Querying**: Utilizes the `CPUID` instruction to extract the CPU brand string, vendor ID, and architecture stepping.
- **Topology & Cache**: Identifies logical/physical core counts and calculates L1, L2, and L3 cache sizes.
- **Feature Detection**: Scans for advanced instruction set support including SSE4.1, AVX, AVX2, AES-NI, and RDRAND.
- **Memory Monitoring**: Parses the /proc/meminfo file to report real-time Total and Available RAM.

2.2 Stopwatch and High-Precision Timer

- **Terminal Control**: Modifies terminal flags (ICANON and ECHO) via SYS_IOCTL to enable real-time keyboard response without line buffering.
- **Accuracy**: Employs **`SYS_CLOCK_GETTIME`** with the **`CLOCK_MONOTONIC`** flag for nanosecond-level precision.

2.3 Scientific Calculator & Base Converter

- **Arithmetic**: Performs 64-bit signed addition, subtraction, multiplication, and division.
- **Radix Conversion**: Supports converting numerical data between Decimal, Binary, Octal, and Hexadecimal formats.

2.4 Memory Viewer (Hex Dump)

- **Introspection**: Displays a 256-byte hex dump of the application's own .text segment.
- **Visualization**: Uses ANSI color coding to distinguish printable ASCII from control characters and null bytes.
-

2.5 CPU Performance Benchmarking

- **Frequency Analysis**: Calculates approximate CPU GHz by comparing the RDTSC cycle counter against a wall-clock second.
- **Throughput Testing**: Benchmarks Millions of Operations per Second (Mop/s) for Scalar (64-bit), SSE (128-bit), and AVX (256-bit) execution paths.

3. Build and Deployment

3.1 Compilation Workflow

The suite includes an automated Bash script (compile.sh) that manages the following pipeline:

1. **Assembly**: Translates .asm source code into an ELF64 object file using nasm.
2. **Linking**: Utilizes the GNU Linker (ld) to produce a standalone binary.
3. **Visual Feedback**: Includes a progress bar to track the compilation status.

3.2 Build Instructions

To compile the system, execute the following commands in a Linux terminal:

```
chmod +x compile.sh

./compile.sh

./sysutil
```