What is new in release version 8.5.0?

It has been some time since I released the last version 7.5.50 of the CPU-OS Simulator. Work commitments and extended testing prevented me from making the next one available. I am hoping that the new version is a significant improvement and is found to be a useful educational resource by educators and students of computer architecture.

Below are the main changes since the last release.

1. Bug fixes

This work never finishes! Since the last release I have been busy fixing bugs as much as possible; especially bugs to do with the array compilation and implementation. The software has already been fairly stable so not many bugs had to be fixed. It is now even more stable, but I suspect very occasionally a bug will rear its ugly head but hopefully not as a showstopper. Unfortunately, any additional enhancement or a new feature runs the risk of introducing new bugs. This is one of the reasons why the big-time gap between the last release and this one...

2. MESI cache coherence protocol

MESI cache coherence protocol is now available for making sure multi-processor caches are kept in step with each other as the data caches are updated across multiple CPUs. The data cache simulators can be configured to use the MESI protocol as an option. This feature can be used to demonstrate the way the MESI protocol works and to demonstrate the need for a protocol like this.

3. Real-time scheduling

The OS simulator now includes three real-time scheduling simulations:

1. Rate monotonic (RMS)
2. Earliest deadline first (EDF)
3. Highest response ratio next

I am planning of including some more esoteric scheduling mechanisms in the future such as Lottery scheduling, Feedback and Fair-share for example!

4. Macro designer

This is a new feature that can be used to create new macro/pseudo instructions. This is intended to enable students to invent and experiment with new instruction sets such as zero, one, two or three address instructions. So, for example

The new 3-address add instruction "ADDX Opnd1, Opnd2, Opnd3", where Opnd1 can be either a literal or a register and Opnd2 and Opnd3 are registers, can be implemented using the following sequence of native CPU simulator instructions:
ADD Opnd1, Opnd2
ADD Opnd2, Opnd3

You can make this more flexible by using the following sequence, where both Opnd1 and Opnd2 can be either registers or literal values and R00 is used as a temporary register:
MOV Opnd1, R00
ADD Opnd2, R00
MOV R00, Opnd3

Alternatively, a stack-based, 0-address add instruction "ADDS" can be implemented as follows, where both Opnd1 and Opnd2 are registers and the two values to be added are assumed to be on top of the stack:

```
POP Opnd1
POP Opnd2
ADD Opnd1, Opnd2
PSH Opnd2
```

This new feature can also run the macro/pseudo instructions in a different window while the native instructions corresponding to these are run in the usual CPU window in parallel, all visually enabled.

The instructions for using the macro designer will be made available later but, in the meantime, students may wish to have some fun (and frustration?) while learning and experimenting with it!

5. Process/thread histogram

This is a new addition to OS simulator's **Views** tab that allows the tracing of up to 8 processes/threads as they run. The histograms of each process/thread are displayed in different colours. The histogram entries are labelled with the tick counts. These histograms can be useful particularly during the demonstration of the real-time schedulers.

6. Executable code

It is now possible to create an executable (.exe) file once the source is compiled to assembly level using the simulator's built-in compiler. This feature is available in the compiler window. The executable produced is a console program only and the runtime supports multi-threading for the executable code. The default executable is for Windows platforms. The executable version of a program can be executed significantly faster than the simulated version. This feature can be used to clearly demonstrate the speed advantage of the native code over the interpreted code.

An executable code for a different platform can be generated as long as a virtual machine suitable for that platform is made available. This can be achieved by replacing the **vm.exe** file in the application's installation directory with the virtual machine code for the desired platform. The C source files for the virtual machine are made available in this release to facilitate this feature.