

Simple: Demonstration Application for the Cmpware CMP-DK (Demo Version 2.0 for Eclipse 3.0)

Cmpware, Inc.

Introduction

The *Cmpware Configurable Multiprocessor Development Kit (CMP-DK)* is a multiprocessor simulation and software development environment. It provides fast and efficient modeling of multiprocessor architectures as well as support for software development on such systems. The goal of supporting software development is achieved by providing an interactive, display-rich environment that permits large amounts of information to be displayed in a fast, simple and uncluttered format. Such capabilities are essential in analyzing the behavior of multiprocessor systems.

This demonstration version of the *Cmpware CMP-DK* (version 2.0) for Eclipse 3.0 and higher contains all features of the standard toolkit, but restricts the simulation model to a 3 x 3 heterogeneous array of MIPS32 and SPARC-8 processors. All simulation capabilities and displays are included. This includes:

- Source Code Tracing
- Source Code Variables
- Disassembly
- Memory Display
- Power Estimator
- General Purpose Registers
- Special Purpose Registers
- Command Line Interface
- Link Utilization

Demonstration Applications

Available for use with the *Cmpware CMP-DK* version 2.0 is a series of demonstration applications which are presented to introduce some of the features in the *CMP-DK*. These applications start with small, simple programs gradually building up to more complex applications exploiting relatively low-level parallelism. These demonstrations stand alone and can be studied in any order, but it is best to start with the early examples, which are smaller and simpler and build up to the larger ones. This provides



a tutorial-like introduction to the features in the *Cmpware CMP-DK*.

While these demonstrations cover the application development aspects of this tool, much of the power in the *Cmpware CMP-DK* is in the ability to quickly model relatively complex multiprocessor systems. This modeling activity is reserved for licensed copies of the software. For more information on getting licensed copies of the *Cmpware CMP-DK*, contact Cmpware at info@cmpware.com.

The groups of files in this tutorial package are as follows:

- **Introduction** - An introduction to all of the applications
- **Simple** - A simple, single processor test application
- **Ping Pong** - a simple two processor application
- **Hetero** - the Ping Pong application on two different types of processors
- **FIR Filter** - A multiprocessor Finite Impulse Response (FIR) Filter
- **AES Encryption** - A multiprocessor AES encryption implementation
- **FFT Filter** - a multiprocessor FFT filter using shared memory
- **FFT Filter 2** - a multiprocessor FFT filter using communication channels

These example applications assume that the *Cmpware CMP-DK* has already been successfully installed on your system. For more information on acquiring and installing either the free demonstration version or the fully licensed version, see the Cmpware web site.

The source and compiled code for these demonstration applications can be downloaded from the Cmpware Web site as a compressed ZIP archive at:

http://www.cmpware.com/Apps/CmpwareApps_2_0.zip

The Simple Application

The first application in this tutorial is called *Simple*, and it is about as simple a uniprocessor application as can be written. It consists of a single infinite loop that increments a variable named *i* on each pass through the loop. This code is just a small test to help gradually explore the features of the *Cmpware IDE* without having to also address the various issues found when multiple processors begin operating in tandem.

This application will demonstrate all of the various 'views' available in the IDE as well as demonstrate the loading and control of executable code. This code is all in the compressed ZIP archive under the *Simple* directory, and contains source code,



Makefile, linker directives file, compiled relocatable object files and finally, fully linked executable ELF files. In fact, all of the demonstration applications will contain these types of files. All have been built using the Gnu GCC compiler with a version higher than 3.0. If you have access to a MIPS32 compiler which produces standard *ELF* executable with *DWARF2* debug information, you may modify these files and re-compile them and test the results and use them in the *Cmpware CMP-DK*.

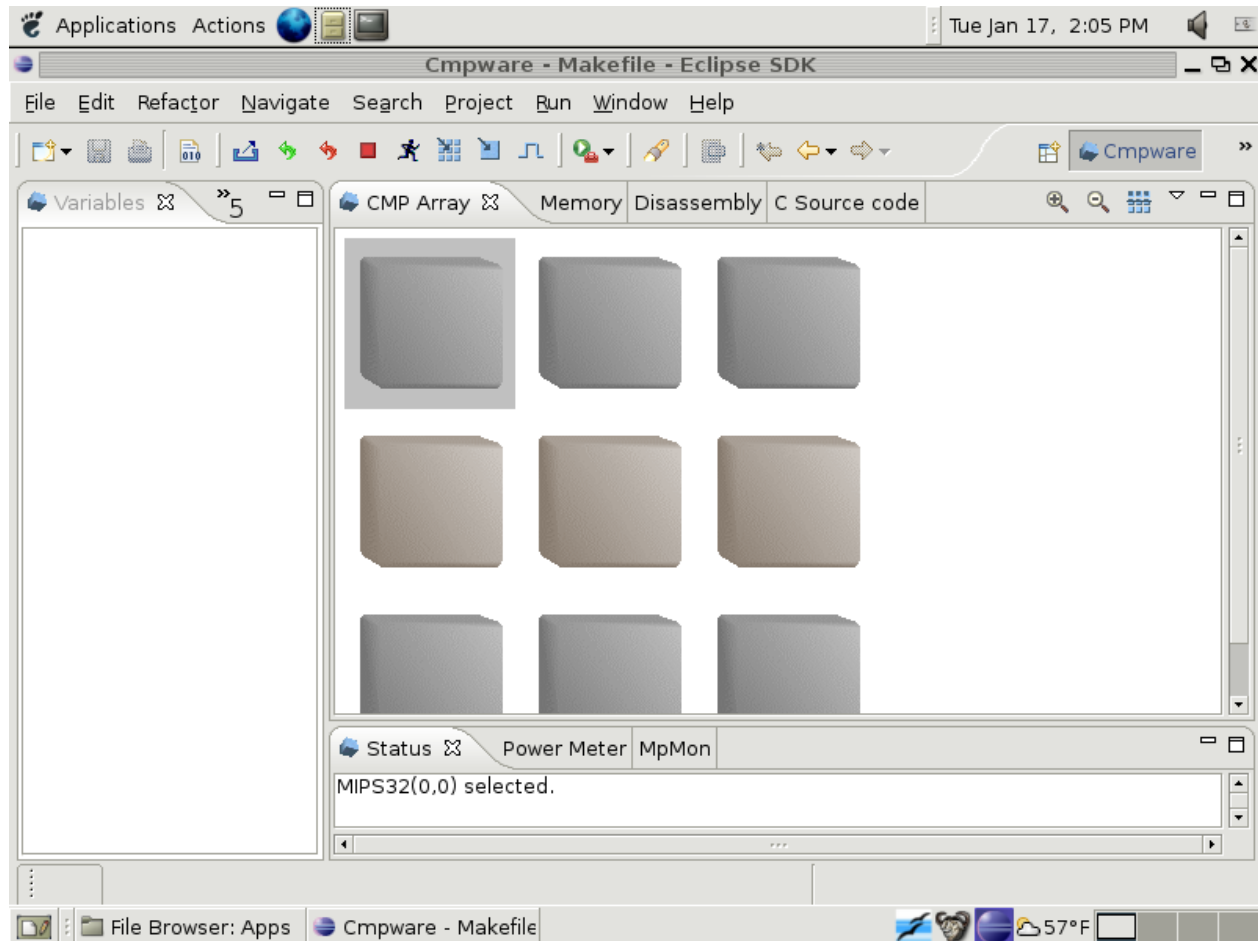


Figure 1: The Cmpware CMP-DK perspective.

Running the Simple Application

First, the Cmpware perspective in Eclipse should be opened. This is typically done from the Eclipse main menu using the **Window --> Open Perspective --> Cmpware** menu command. This brings up the view in Figure 1. If you have problems getting this



view to come up, or have not installed the *Cmpware CMP-DK*, see the installation guide available on the *Cmpware* web site. It will guide you in installing the software.

This view is of the demonstration version of the software and begins with the default 3 x 3 array of processors. The first row contains three *MIPS32* processors, the second row three *Sparc-8* processors, and the third row contains another three *MIPS32* processors. To load the first processor with executable code, select the processor at the upper left with the mouse. It should be highlighted with a grey background and the status window at the bottom should indicate that the processor **MIPS32(0,0)** is selected, as shown in Figure 1.

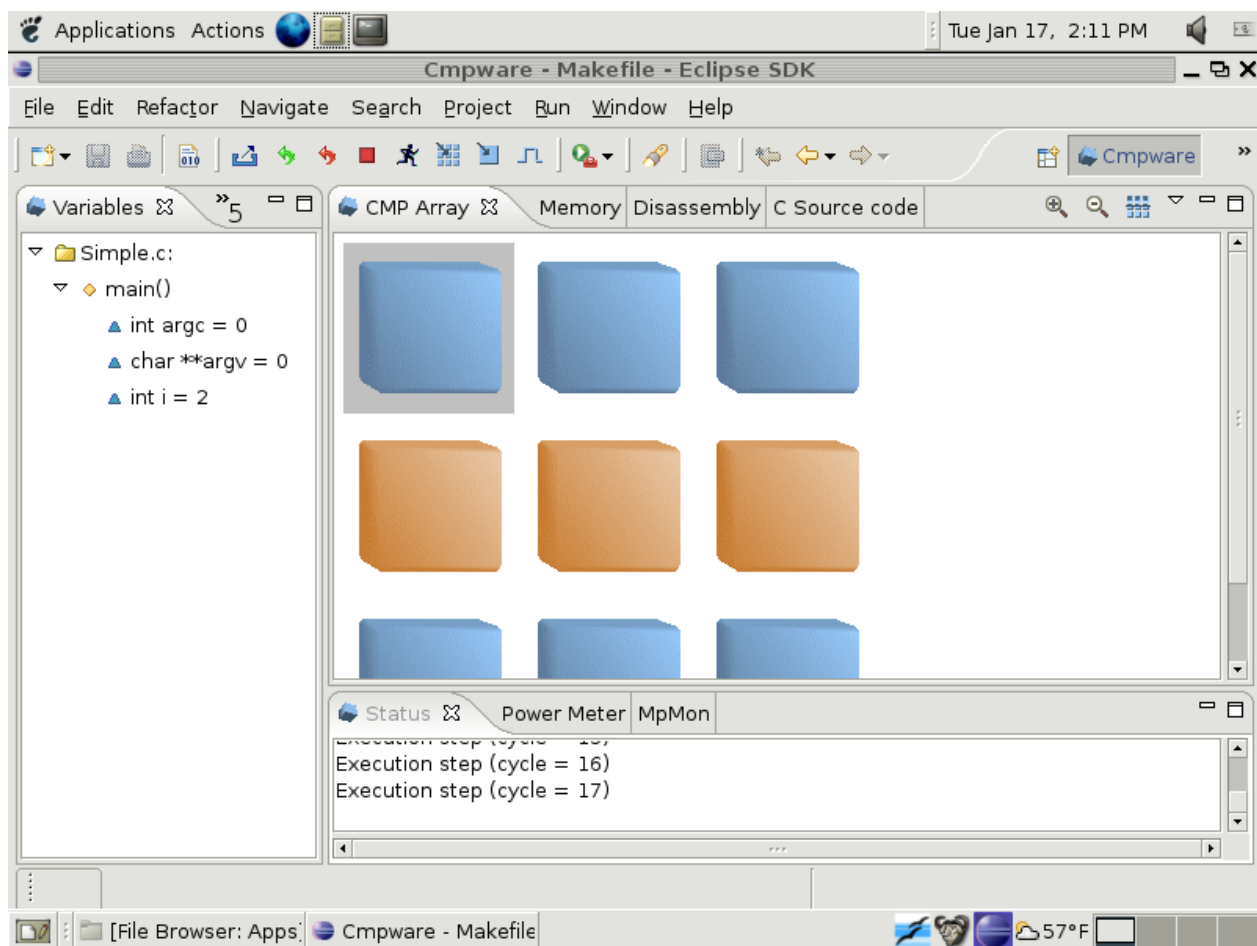



Figure 2: Loading the *Simple.elf* file.

Using the **Load** button () to bring up a file selection dialog. Using this file selection



dialog, select the *Simple.elf* file from the list of files for the Simple demonstration. This should result in the view in Figure 2. Note that the processors are now out of their initial idle state, indicate by their grey color, and are active. The *Variables* view on the left also displays the source code variable information in a tree format. Clicking on the **Step** button () steps the global clock. Each step updates the displays in the *Cmpware CMP-DK*. In the view in Figure 2, the variable *i* changes and the step count is displayed in the status window at the bottom.

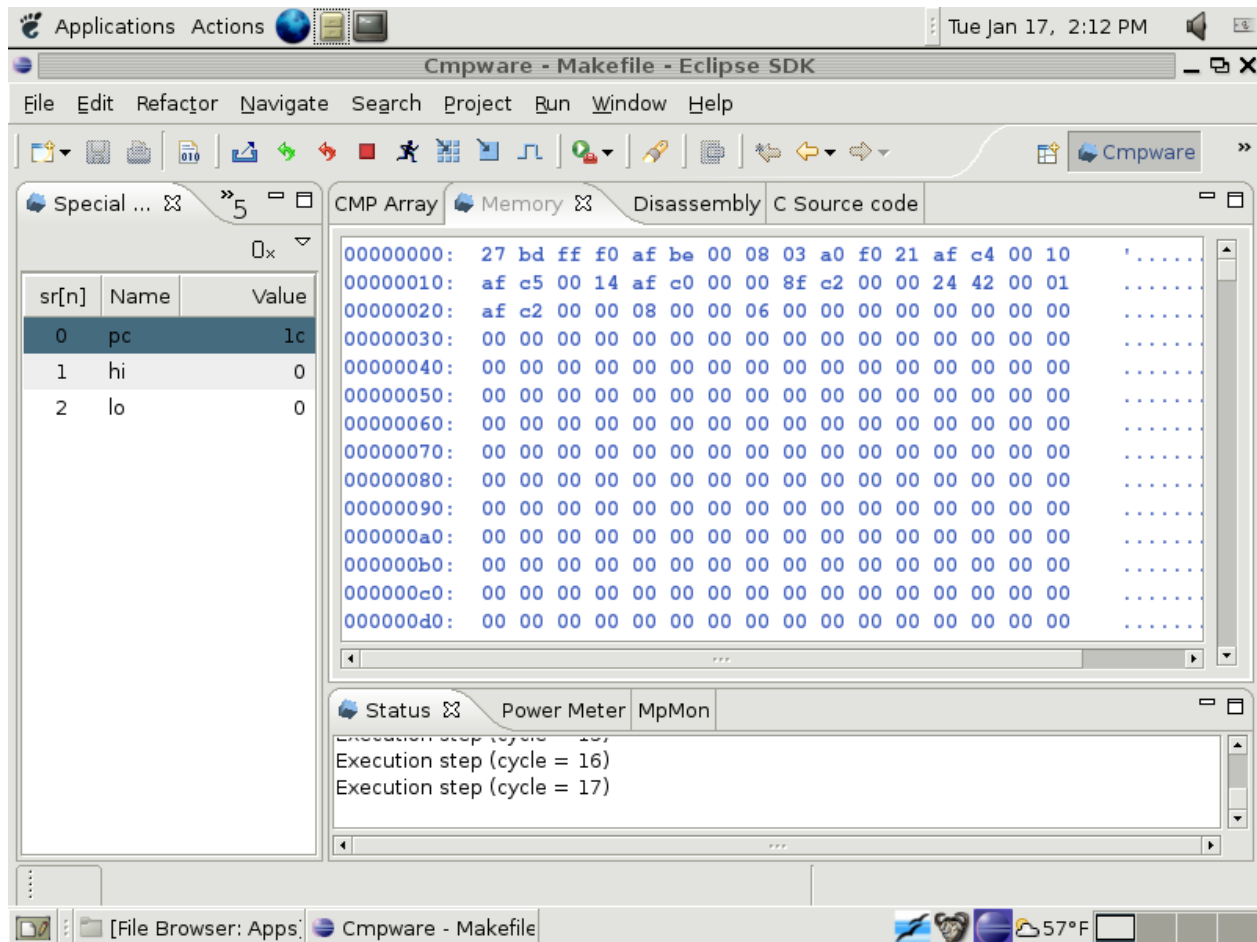


Figure 3: The Cmpware CMP-DK Special Registers and Memory views.

In addition, even the data in the hidden windows is updated on each step. By clicking on the tabs for the **Memory** view in the main window and the **Special Registers** on the left, the view as in Figure 3 is displayed. The memory view gives a standard hexadecimal dump format of the memory data, and the code at address zero can be



seen. The Special Registers are the processor Special Registers as defined for this specific processor. In this case, there are three Special Registers, the *Program Counter (PC)* which is at address 0x1c, and the *hi* and *lo* registers.

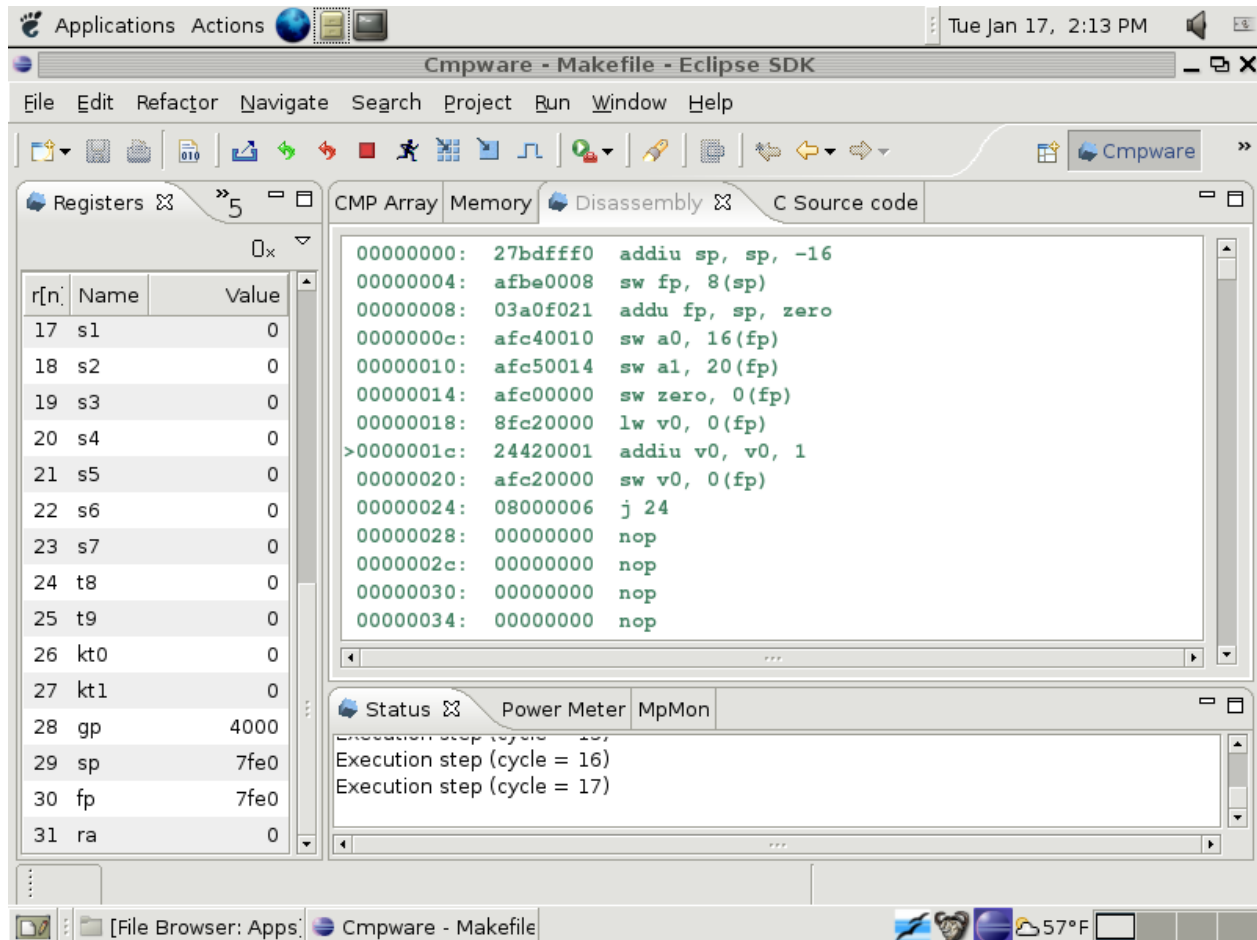


Figure 4: The Cmpware CMP-DK Registers and Disassembly views.

Similarly Figure 4 shows the view after the **Disassembly** and **Registers** tabs have been selected. This shows the general purpose registers in the window on the left and a disassembly of the program in the main window. Note that the current instruction at address 0x1c is indicated as the currently executing instruction with the ">" character in column zero.

If the **C Source Code** and the **Profile** tabs are selected, the C source code for the code being executed is displayed in the main window as in Figure 5. Note that the currently



executing line of source code is highlighted with a grey bar and an arrow (➔) in the leftmost column. The profile information in the left side window contains the address ranges and number of instruction fetches in the current execution profile. Note that unlike similar profiling tools on self-hosted systems, this data is not sampled and is 100% accurate. Each instruction fetch is counted and tallied in the simulation model.

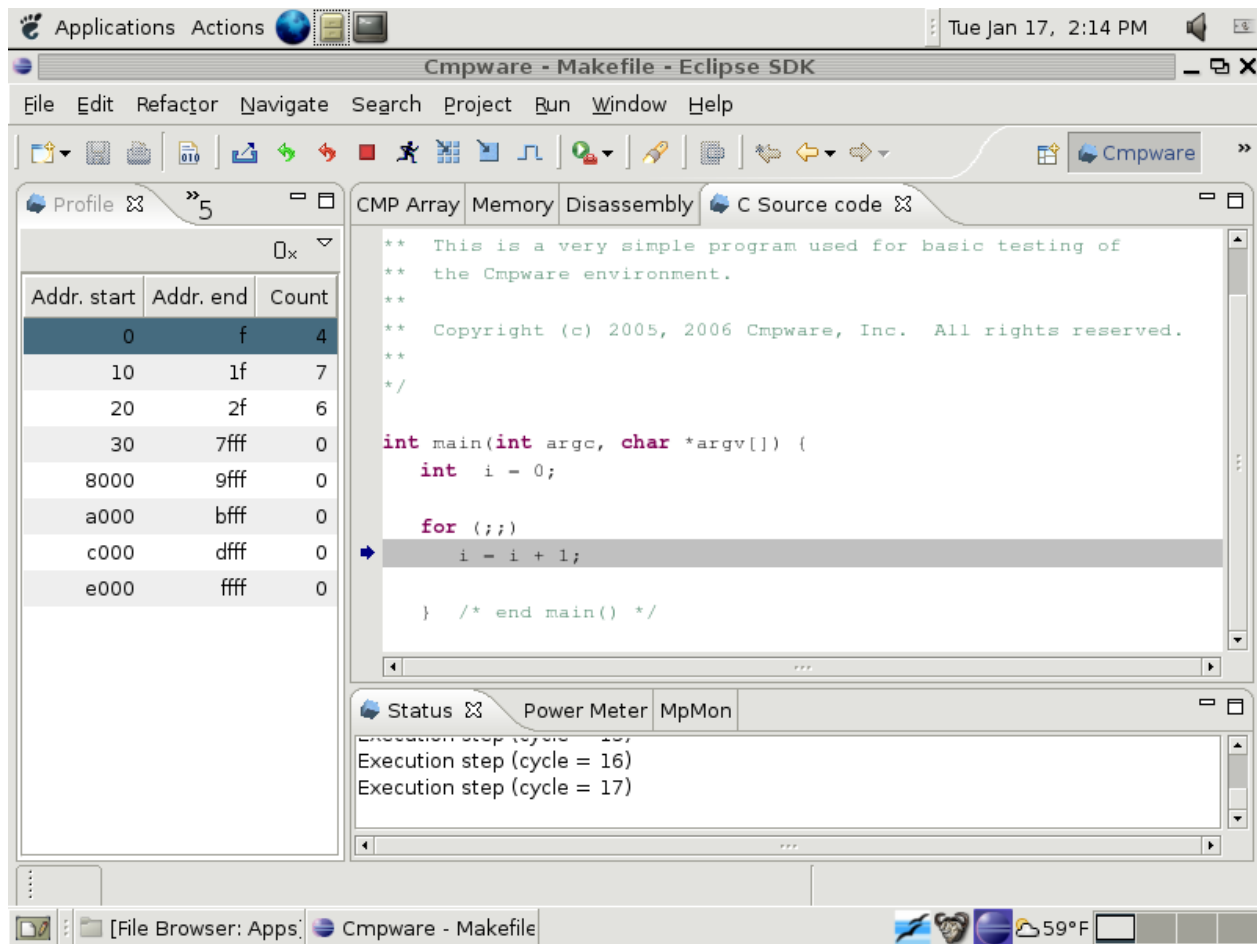


Figure 5: The Cmpware CMP-DK Profile and Source Code views.

By again selecting two tabs, this time the **Instruction Mix** on the right and the **Power Meter** at the bottom, two more views are available as in Figure 6. The Instruction Mix gives a list of all of the assembly language opcodes for the currently selected processor and the number of times that instruction has been executed. As with the profile information, this data is not sampled or estimated, but is 100% accurate and is maintained internally by the simulator. This can be useful, for instance, in finding



commonly used (or unused) instructions for optimization or to measure memory traffic through load and store instructions.

The screenshot shows the Eclipse IDE interface for the Cmpware CMP-DK perspective. The main editor displays C source code for a simple program. The 'Instr. Mix' window shows the following instruction counts:

Opcod	Mnemonic	Count
32	sw	6
2	addiu	3
2a	lw	3
1f	j	2
46	sll	2
3	addu	1
0	add	0
1	addi	0
4	clo	0
5	clz	0
6	div	0
7	divu	0
8	madd	0
9	maddu	0
a	msub	0

The 'Power Meter' window displays the following data:

P[n,m]	Run Cycles	Idle Cycles	Util (%)	Run (mW)	Idle (mW)	Node Po
P[0,0]	17	0	100	100.0	10.0	
P[0,1]	17	0	100	100.0	10.0	
P[0,2]	17	0	100	100.0	10.0	
P[1,0]	17	0	100	100.0	10.0	

Figure 6: The Cmpware CMP-DK perspective Instruction Mix and Power Meter.

The **Power Meter** window at the bottom is used to estimate power consumption in the multiprocessor, as well as to view utilization. In this case, all of the nodes are used 100% of the time, and the default power values are used. In more complex situations, where processors may be idled for some number of cycles to save power and when various combinations of processors are in a system, this tool can give a reliable estimate of power consumption. Perhaps more importantly, it can reliably provide relative numbers to compare the power consumption in two different hardware or software approaches.



Other Controls

In addition to these display windows, a command line monitor called **MpMon** is available in the bottom window group. Figure 7 shows the MpMon window when the tab is double clicked and enlarged to a full screen window. Note that any of the windows in the *Cmpware CMP-DK* may be enlarged and returned to their original size by double-clicking on the tab. The windows may also be dragged into different groupings, if the default grouping is not desired.

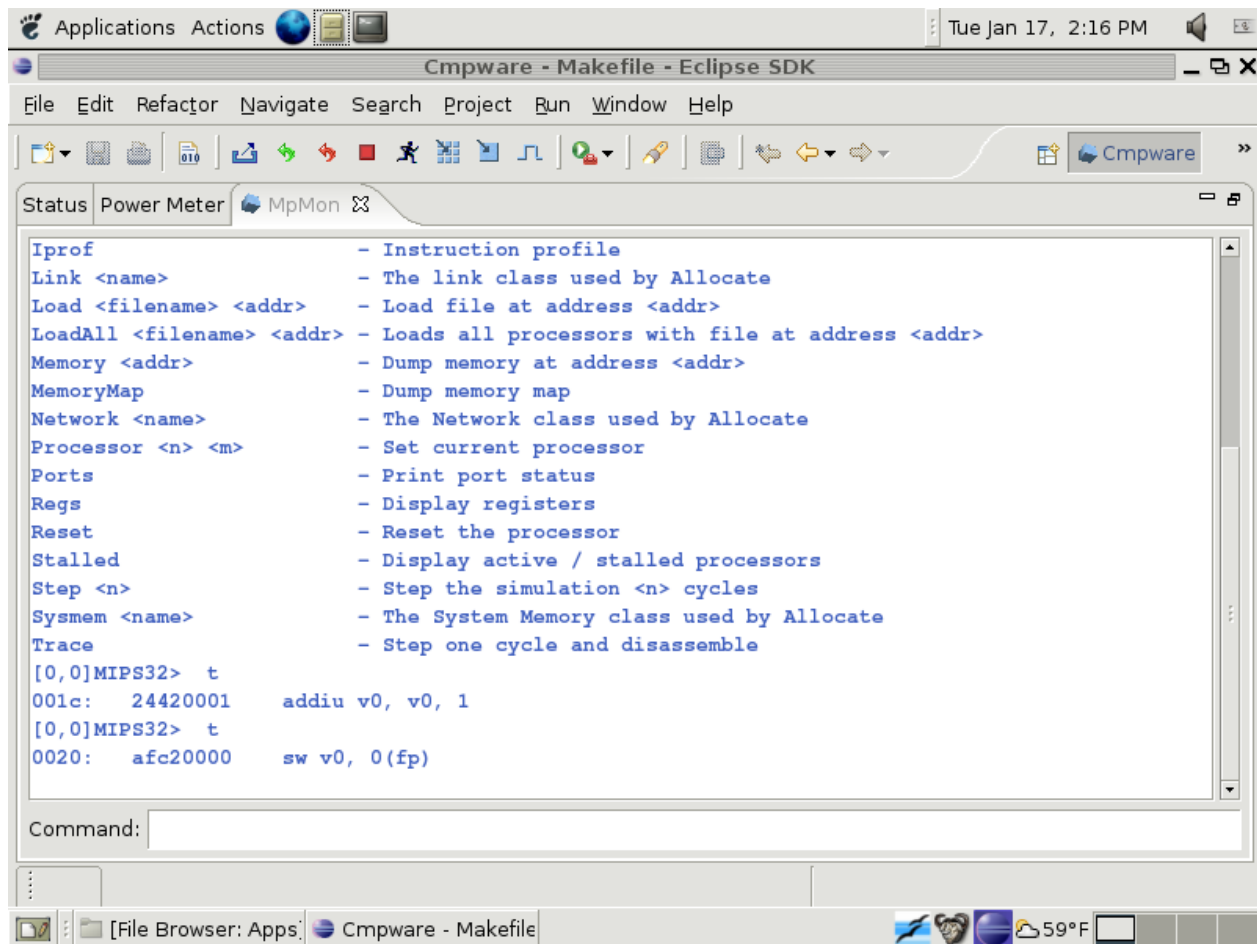


Figure 7: The Cmpware CMP-DK command line interface.

The MpMon command line interface provides most of the functionality available in the IDE and is somewhat redundant. But it is made available for situations when typing



commands may be more desirable. A stand-alone version of MpMon is also available for uses such as scripting, but its use is beyond the scope of this document.

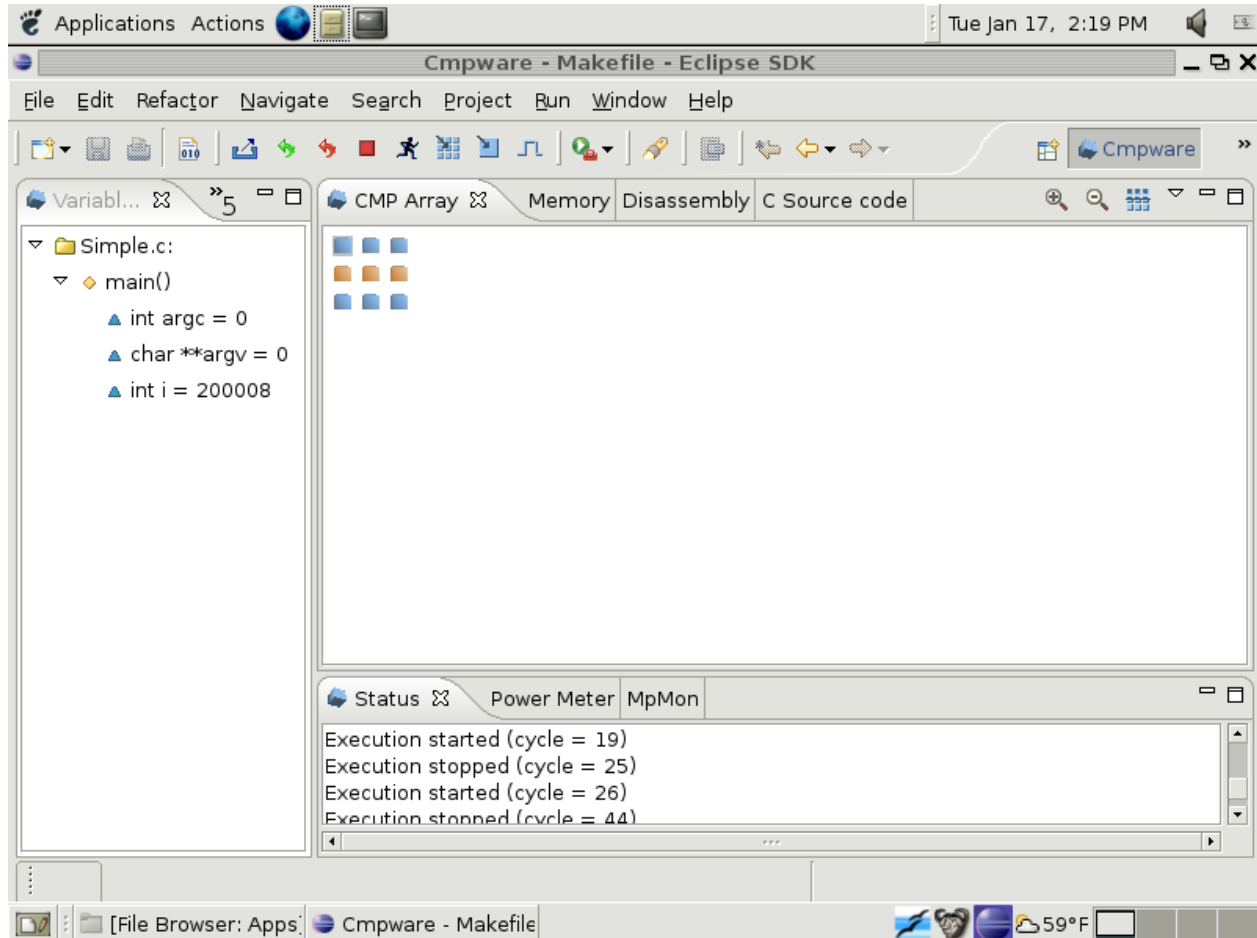


Figure 8: The Cmpware CMP-DK perspective.

Finally, some other control buttons are available with the *Cmpware CMP-DK*. While the **Load** (📁) and **Step** (⏸) buttons were discussed, several others are available and may be useful. These are:

Load All (📁) - This loads the same ELF or binary file into all processors. This is a fast way to get a large number of processors loaded with code.

Run (🏃) - This is similar to the step button, but the execution continues until a breakpoint is encountered, an error occurs or the stop button is pressed.



Stop (■) - This button stops execution if the run button has been pressed and code is currently executing.

Reset (↺) - This button resets the system. This sets all registers to zero and generally brings the system to its initial state.

Reload (↺) - This button resets the system and reloads the most recently loaded ELF or binary file. This is a good way to start over without having to reinitialize the system manually.

Export (📄) - this exports data from the system for external use. See the related documents on the Cmpware web site for more details on how to use the Export feature.

While these buttons are in the main control bar and are always available in the *Cmpware CMP-DK*, the main processor display window has three additional controls. These are:

Flip (🔄) - this button logically 'flips' the array on a horizontal axis. This changes the array origin of processor (0,0) from the upper left corner to the lower right corner. This can be useful in software while uses an orientation such as 'north' or 'south' and assumes some particular array orientation.

Zoom Out (🔍) - this button zooms the processor array view out, permitting more, but smaller array objects to be viewed. This is useful for larger arrays and is demonstrated in Figure 8.

Zoom In (🔍) - this button is the opposite of the Zoom Out button and permits fewer, but larger, processor array objects to be viewed.

Conclusions

The *Cmpware CMP-DK* is a rich display environment combining fast simulation and flexible multiprocessor modeling. This makes it an ideal environment for architecture modeling and software development for these systems.

While the execution and display features of the *Cmpware CMP-DK* are notable, much of the power of the system lies in its ability to quickly and flexibly construct processor, network, link and multiprocessor models. This modeling capability is a large part of the commercial version of the *Cmpware CMP-DK*.



For more information on the commercial version of the *Cmpware CMP-DK* see our web site at:

<http://www.cmpware.com/>

or send an email to:

info@cmpware.com



Appendix A: Simple.c Source Code

```
/*
** This is a very simple program used for basic testing of
** the Cmpware environment.
**
** Copyright (c) 2005, 2006 Cmpware, Inc. All rights reserved.
**
*/

int main(int argc, char *argv[]) {
    int i = 0;

    for (;;)
        i = i + 1;

} /* end main() */
```

