

Laboratorio di Sistemi Operativi
Anno Accademico 2006-2007

Software Development with uMPS

Mauro Morsiani

Copyright © 2007 Mauro Morsiani
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license can be found at:
<http://www.gnu.org/licenses/fdl.html#FOCS>

uMPS simulator and user interface

“Nothing is real”:

What is uMPS?

it is a computer system

it is a simulator (that is, a set of programs)

How to get it?

download from <http://mps.sourceforge.net>

compile and install (see INSTALL and README)

Suggested configuration on your own PC:

recent Linux distribution on x86 (Debian, Ubuntu, Fedora...)

release 1.23-RC2 (RC1 is identical, but is better suited for non-root installations)

© 2007 Mauro Morsiani

2

uMPS simulator and user interface

uMPS simulator main commands:

`umps`: the simulator itself

`umps-elf2umps`: to convert the output of the compiler to files the simulator will understand

`umps-objdump`: to analyze these files

`umps-mkdev`: to build disks and tapes for the simulator

uMPS simulator and user interface

uMPS simulator-related files:

In the `support/` and `example*/` directory:

`*.rom.umps`: the ROM files

`*.core.umps`: the kernel to be loaded

`*.stab.umps`: the kernel *symbol table*

`*.aout.umps`: for programs other than kernel

other `*.umps` files (`term0.umps`, `printer0.umps...`): files associated to devices

`/etc/umpsrc` and `.umpsrc` (`ls -a` to see it): the simulator configuration file

`elf32*.x` files: configuration files for the cross-compiler

uMPS simulator and user interface

uMPS other essential components:

some libraries (XForms, libelf) for building the simulator

libumps.e (and libumps.o) under support/: uMPS library for interfacing with ROM services, **CPO** registers and issue TLB-related and SYSCALL instructions

crtso.o and crti.o: kernel and program startup functions

const.h and types.h under support/h: some useful types and constants (eg. processor state definition)

a *cross-compiler* based on GNU gcc:

mipsel-linux-gcc for little-endian uMPS (on x86)

mips-linux-gcc for big-endian uMPS (on PPC)

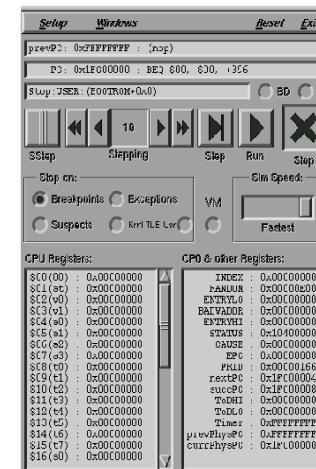
the make utility

© 2007 Mauro Morsiani

5

uMPS simulator and user interface

"Out of the Matrix": uMPS main window



© 2007 Mauro Morsiani

6

uMPS simulator and user interface

uMPS main window features:

a *menu bar* with *Setup*, *Windows*, *Reset*, and *Exit*

three *status lines* showing:

the last instruction executed

the instruction about to be executed

a status line with:

the current status of the simulator (running or stopped)

the reason the simulator stopped, if any (by user request or because some debugging event has happened)

the location of the instruction in a readable form

Load and Branch Delay status flags

© 2007 Mauro Morsiani

7

uMPS simulator and user interface

uMPS main window features (cont'd):

a series of buttons for controlling the simulation progression (*SingleStep*, *Stepping*, *Step*, *Run*, *Stop*)

a simulation speed slider (from *Slowest* to *Fastest*)

a *Virtual Memory* indicator

some *Stop on* buttons, to stop simulation on:

Exceptions

Breakpoints

Suspects

TLB-refill events in Kernel or User mode (requires also Exception to be selected)

© 2007 Mauro Morsiani

8

uMPS simulator and user interface

uMPS main window features (cont'd):

main CPU registers

CP0 registers

some useful "meta-registers":

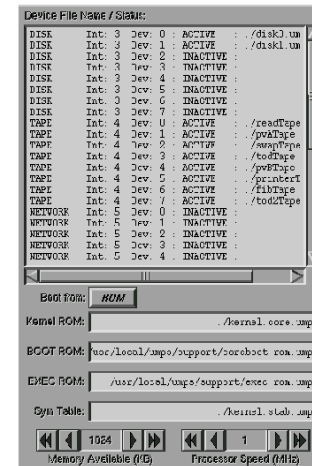
nextPC, succPC: next PC locations computed from the pipeline

prevPhysPC and **currPhysPC:** previous and current PC locations in physical memory (useful when VM is on)

register values may be changed just by double-clicking on them

uMPS simulator and user interface

uMPS Setup Window



uMPS simulator and user interface

uMPS setup window features:

the file list of `.umps` files associated to various devices

the “*Load core?*” flag: to decide if the kernel file is loaded in RAM before the start of the simulation or not

the file list of `.umps` files used for:

ROMs (`.rom.umps`)

kernel (`.core.umps` and `.stab.umps`)

the RAM size selector (64KB – 2 MB)

the processor speed selector (1-99 MHz)

uMPS simulator and user interface

uMPS setup menu contains:

the *Setup Window* option

the *Reset to Setup* option: to reset the simulator to the state described in the *Setup Window*, reloading all the files

the *Reset to Defaults* option: to reset the simulator to the state described in the `.umpsrc` file, reloading all the files

Some options may be configured only in the `.umpsrc` file:

TLB size

Expert Mode: no confirmation required for most operations

initial GUI setup (which buttons are on or off, etc.)

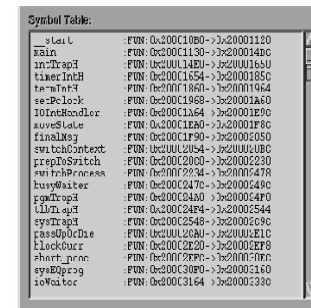
uMPS simulator and user interface

uMPS *Windows* menu contains:

- the *Symbol Table* option: to show the kernel Symbol Table
- the *Memory Browser* option: to show memory contents, and to set Breakpoints and Suspects
- the *TLB Display* option: to show the TLB contents
- Int x Device Status* options: to show the devices' status
- the *Terminal x* options: to show the terminals

uMPS simulator and user interface

uMPS *Symbol Table* Window



Symbol Name	Address
start	:FUN: 0x200C10E0 -> 3x20001120
main	:FUN: 0x200C1130 -> 3x200014DC
unTrapH	:FUN: 0x200C11E0 -> 3x20001650
finactinH	:FUN: 0x200C11E4 -> 3x2000185C
re=IntH	:FUN: 0x200C11E0 -> 3x20001964
se=DeLock	:FUN: 0x200C1968 -> 3x20001A60
IOIntrHandler	:FUN: 0x200C11E4 -> 3x20001E90
muveStatus	:FUN: 0x200C1E80 -> 3x20001F9C
finalMsg	:FUN: 0x200C1F90 -> 3x20002050
switchContext	:FUN: 0x200C2054 -> 3x200020BC
preIOSwitch	:FUN: 0x200C2050 -> 3x20002230
switchProcess	:FUN: 0x200C2054 -> 3x20002478
busyWaiter	:FUN: 0x200C247C -> 3x2000249C
rogTrapH	:FUN: 0x200C24A0 -> 3x200024F0
LLInqH	:FUN: 0x200C24F4 -> 3x20002544
sysTrapH	:FUN: 0x200C2548 -> 3x2000259C
passUpOrDie	:FUN: 0x200C2600 -> 3x20002E1C
clockOver	:FUN: 0x200C2E20 -> 3x20002EF8
short_proc	:FUN: 0x200C29F0 -> 3x200011FFC
sysUpProc	:FUN: 0x200C3090 -> 3x20002160
ioWaiter	:FUN: 0x200C3164 -> 3x20002330

uMPS simulator and user interface

uMPS TLB Display Window

TLB Status		
TLB [00]	: 0x00C0000	: 0x00C0000
TLB [01]	: 0x00JL0000	: 0x00JL0000
TLB [02]	: 0x00C0000	: 0x00C0000
TLB [03]	: 0x007F0000	: 0x007F0000
TLB [04]	: 0x00C0000	: 0x00C0000
TLB [05]	: 0x00C0000	: 0x00C0000
TLB [06]	: 0x00C0000	: 0x00C0000
TLB [07]	: 0x00C0000	: 0x00C0000
TLB [08]	: 0x00JL0000	: 0x00JL0000
TLB [09]	: 0x00C0000	: 0x00C0000
TLB [10]	: 0x007F0000	: 0x007F0000
TLB [11]	: 0x00C0000	: 0x00C0000
TLB [12]	: 0x00C0000	: 0x00C0000
TLB [13]	: 0x00C0000	: 0x00C0000
TLB [14]	: 0x00C0000	: 0x00C0000
TLB [15]	: 0x00JL0000	: 0x00JL0000

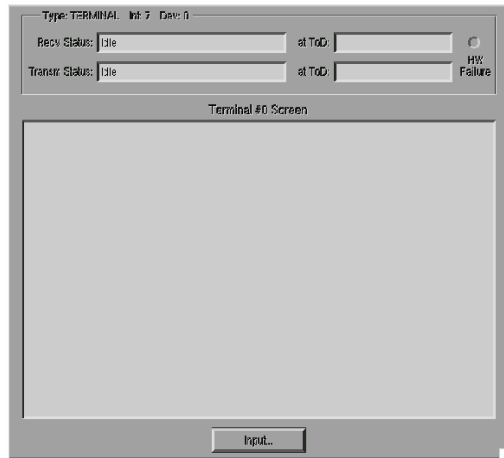
uMPS simulator and user interface

uMPS Device Status Window

Type: IDE0	Int 3	Dev 0
Name:	[Local Disk 3: SDC277113]	
Completion @TOD:	[150] [Fail]	
Type: IDE0	Int 3	Dev 1
Name:	[e]	
Completion @TOD:	[150] [Fail]	
Type: NULLDEV	Int 5	Dev 2
Name:	[Not Operational]	
Completion @TOD:	[150] [Fail]	
Type: NULLDEV	Int 5	Dev 3
Name:	[Not Operational]	
Completion @TOD:	[150] [Fail]	
Type: NULLDEV	Int 5	Dev 4
Name:	[Not Operational]	
Completion @TOD:	[150] [Fail]	
Type: NULLDEV	Int 5	Dev 5
Name:	[Not Operational]	
Completion @TOD:	[150] [Fail]	
Type: NULLDEV	Int 5	Dev 6
Name:	[Not Operational]	
Completion @TOD:	[150] [Fail]	
Type: NULLDEV	Int 5	Dev 7
Name:	[Not Operational]	
Completion @TOD:	[150] [Fail]	

uMPS simulator and user interface

uMPS Terminal Status Window



© 2007 Mauro Morsiani

17

uMPS simulator and user interface

uMPS Device Status Window details:

for each device:

- a status line describes what the device is currently doing
- a TOD field tells when the current operation (if any) will end
- a *HW Failure* button allows to set the device as "faulty" (all operations will not be performed and will return an error status)

Terminal devices have also:

- two device status lines (one for receiver, one for transmitter)
- a screen to show the output
- an *Input* button to allow line input

© 2007 Mauro Morsiani

18

uMPS simulator and user interface

uMPS Memory Browser Window



© 2007 Mauro Morsiani

19

uMPS simulator and user interface

uMPS Memory Browser Window details:

Breakpoint list and *Add / Delete* buttons: to show, set and remove Breakpoints

Suspect list, and *Add / Delete / Trace* buttons: to show, set and remove *Suspect areas*

Trace list and *Add / Delete / Modify* buttons: to show, set and remove memory traced ranges and to modify RAM contents

memory content window: to show memory contents

© 2007 Mauro Morsiani

20

uMPS simulator and user interface

Breakpoint, Suspect and Trace: the debugger's tools of trade

Breakpoint: a position (an address) in the code; simulation stops when reaches it (may be referred to with a *symbol* + offset)

Suspect area: a memory range (a set of addresses) containing data (array, variables...) under exam; may be a *Read* suspect and/or a *Write* suspect (may be referred with a *symbol*)

Suspect: simulation stops when an access of the appropriate type (R, W) is made to the suspect area

Traced range: a range of memory addresses selected for showing

Addresses may be physical or virtual ones

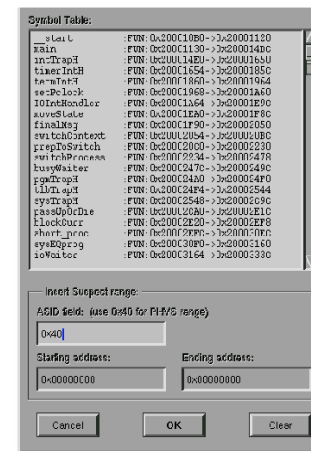
Only physical addresses may be traced

© 2007 Mauro Morsiani

21

uMPS simulator and user interface

uMPS: how to insert a Suspect range



© 2007 Mauro Morsiani

22