Introduction to Machine Problems

- Short C++ Tutorial
- MIPS Emulator: mips64emul / gxemul
- The OS infrastructure: What does the OS need to start?
- MPO: Writing / compiling / running simple program in emulator.

“The Machine”: mips64emul / gxemul

- Software instruction-level emulator for a wide variety of MIPS CPUs and machines.
- Provides some support for device emulation (stuff that happens around the CPU).
- For details, check out http://gavare.se/gxemul/

Availability:
- mips64emul is a previous version of gxemul and is installed on our Sun’s
- I have it running fine on my Mac (simply download and install)
- The TA’s have gotten gxemul to run on
  - Redhat LINUX V.9.0
  - Windows using Cygwin (Check project website for details.)
Compiling for MIPS

- We need a cross-compiler (meaning: C/C++ compiler and the tool chain, such as as/ld/id/...) that generates MIPS code.
- We have the following platforms available:
  - We have gcc 3.4.3 and binutils 2.15 installed on our Sun's.
  - I am running gcc 3.2.2 and binutils 2.13.2 on my Mac.
  - TA's got the gcc 3.4.3 and binutils 2.13 working on Linux RedHat 9.0 and Windows/Cygwin (again, check project website)
- If you want to install the cross compiler yourself, you need to run a script similar to the following (after downloading the packages)

```
$ export PREFIX=/usr/cross
$ export TARGET=mips64-unknown-elf
$ cd /home/username/OS/build-binutils-2.13
$ ../binutils-2.13/configure --target=TARGET --prefix=$PREFIX --disable-nls
$ make all install
$ cd /home/username/OS/build-gcc
$ export PATH=$PATH:$PREFIX/bin
$ ../gcc-3.4.3/configure --target=TARGET --prefix=$PREFIX --disable-nls --enable-languages=,c++,
  --with-header
$ make all-gcc install-gcc
```

Compiling for MIPS (2)

- A makefile would look similar to this one:

```
OPT = -DMIPS -fno-rtti -fno-exception
LDOPT = -static -nostdlib -Ttext 0x80030000 -e f
ASOPT = -I /usr/local/mips64/include

C++ = mips64-g++
LD = mips64-ld
AS = mips64-gcc

asmstuff.o: asmstuff.S
  $ (AS) $ (ASOPT) -o asmstuff.S

object.o: objekt.H objekt.C
  $ (C++) $ (OPT) -c objekt.C

  $ (C++) $ (OPT) -c main.C

main: utils.o assert.o < scheduler.o main.o
  $ (C++) $ (OPT) $ (LDOPT) main.o scheduler.o < util.o -o main
```
Running Code on mips64emul

```
rmcardo@muscat ~]$ mips64emul [-/MyDocuments/Projects/OSProject/TheOS] \$ ./mips64emu -D/mips64emu -o mtest
adding memory: 32 M
adding cpu: 8080 (L = 8x8 KB, L2 = 512 KB)
adding device 0 at 0x0000000010000000: cons
adding device 1 at 0x0000000110000000: op
calling "mips" test machine
net_init(): using namedeserver 128.194.138.1
loading files into emulation memory:
'memtest': 1632 M (64), entry point 0xffffffff80030000
'memtest': loadable chunk 0 @ 00001000, addr: 0xffffffff80030000 len:0x3020
'memtest': loadable chunk 1 @ 00004000, addr: 0xffffffff80070000 len:0x6f68
'memtest': 16K symbol entries at 0x8000
'memtest': 32K bytes of symbol strings at 0x8004
'memtest': found _sp address: 0xffffffff8007ba40
starting emulation: cpu pvr=0x6f1ff1f800030000 q=0x6f1ff1f8007ba40

Initialized console

SimpleMindedPhysMemory: init
Memory: init
Found and initialized memory. Testing memory allocation routines
using a heap intensive version of the Ackerman function
Initialized Exception and Interrupt subsystem.

n =  n = 2
m =  m = 2
Result: 7

Time for computation: Time: 1 sec / 358010 nanosec
Number of "new" calls: 27
Number of "delete" calls: 27

Starting program

```

The OS Infrastructure (1)

```
/*==================*/
/* Some System Components */
/*==================*/

SimpleMindedPhysMemory kernelMemory;
Scheduler * sched;
Dispatcher * timer_dispatcher;

/*------------------*/
/* MEMORY MANAGEMENT */
/*------------------*/

// replace the operators
// "new/alloc[]" and "delete/delete[]"
void * operator new (unsigned int size) {
    return kernelMemory.allocate(size);
}
void * operator new[] (unsigned int size) {
    return kernelMemory.allocate(size);
}
void operator delete (void * p) {
    kernelMemory.free((Addr)p);
}
void operator delete[] (void * p) {
    kernelMemory.free((Addr)p);
}
```
The OS Infrastructure (2)

/* ________________________________ */
/* MAIN */
/* ________________________________ */

extern "C" void f(void)
{
    /* -- INITIALIZE CONSOLE */
    Console::init();
    Console::putstr("Initialized console\n\n");
    /* -- INITIALIZE MEMORY SUBSYSTEM */
    kernelMemory.init((unsigned int)MIPS_KSEG0_START + 1 MB, 31 MB);
    Console::putstr("Found and initialized memory:\n");
    /* -- INITIALIZE PROCESSOR */

    /* -- INITIALIZE EXCEPTION AND INTERRUPT HANDLERS */
    ExceptionDispatcher::init();
    Console::putstr("Initialized Exception Dispatcher\n\n");
    DefaultExceptionHandler * default_handler = new DefaultExceptionHandler();
    InterruptDispatcher * interrupt_dispatcher = new InterruptDispatcher();
    ExceptionDispatcher::register_exception_handler(Int, interrupt_dispatcher);
    Console::putstr("Initialized Exception and Interrupt subsystem\.\n");
    InterruptDispatcher::disable_interrupts(); /* Don't ask */
    InterruptDispatcher::enable_interrupts();
}

OS Infrastructure (3)

/* -- INITIALIZE TIMER MANAGEMENT */
timer_dispatcher = new TimerDispatcher();
interrupt_dispatcher->register_interrupt_handler(TIMER_INTERRUPT, timer_dispatcher);
Timer::set_timer_dispatcher(timer_dispatcher);
InterruptDispatcher::unmask_interrupt_no(TIMER_INTERRUPT);
/* -- CREATE SCHEDULER AND IDLE THREAD */
//sched = new FIFOScheduler();
//sched = new RRSScheduler(1000000);
//idle_thread = new Thread("idle", idle_func, 1024, sched);
idle_thread->start();
/* -- START UP EXECUTION BY STARTING SCHEDULER */
sched->start();
/* -- CREATE ROOT PROCESS */
/* XXX TO DO */
Console::putstr("Created and started root process. NOT YET IMPLEMENTED\.\n");
/* WE SHOULD NEVER REACH THIS POINT! */
for(;;);
}