Operating Systems Lab - CS3510

Abhiram Ravi, Karthik Abinav

September 24, 2012

Abstract

This is a documentation of our progress in the Operating Systems lab in the fall semester(2012-13) under *Prof V. Kamakoti*

0.1 The Basics

The first part of lab is divided into four parts :

- Segmentation
- Interrupt and Exception Handling
- Task Switching
- Paging

Every group is given an *Intel Atom Processor* - powered board, a bootable USB Stick with the GDB software, and a connector cable to connect the intel board to the host machine. The gdb software installed on the host machine takes care of any interaction that is necessary with the atom board. Once the atom board is booted using the USB Stick, a connection is established to the host machine by running the following commands on the terminal with *root permissions*

```
 $ gdb
/* Enter the following commands in the gdb command line */
 $ set debug remote 0
 $ set remotebaud 38400
 $ target remote /dev/ttyS0
```

This establishes the required connection. In case of problems, reboot the atom board and make sure the cable is properly connected. In the worst case, try restarting the host machine.

Now we describe how to compile the assignment programs and load them onto the atom board. All code related to these assignments are written in *.s* files and are compiled using NASM. To compile the programs, the following commands are executed :

```
/* gdbkama_prot & prot_test.ld are scripts given to us to automate compilation */
$ ./gdbkama_prot code
/* where code.s is the program */
```

Once the compilation is complete, a *code.out* file is created. This file must be loaded onto the atom board. To do this, we now return to our gdb command line and execute

```
$ load code.out
```

Once the loading is complete, the program is automatically executed on the atom machine[1]. Breakpoints are already set in the code template. The program is paused by the gdb upon loading. To continue execution of the program enter 'c' in the gdb command line. If the code executed succesfully, a SIGTRAP is received. In order to view the register and memory state of the system, the following commands can be executed on the gdb command line

```
/* Gives the state of the registers at any point of time */
$ info reg
/* Viewing N bytes of data in memory starting from address XXXX */
$ x/Nx XXXX
```

The above commands are helpful during debugging.

[1]In case of repeated segmentation faults (SIGSEGV) try loading the file twice

0.2 Assignment 1

Aim

- To learn the concepts related to segmentation.
- To set up the Global Descriptor Table
- To understand the LGDT command, the working of segment registers, code *align*ment and to revisit NASM

Summary
In this assignment we had to write a C code that does matrix multiplication on two matrices A (10 x 20) and B (20 x 30), and output the answer in hexadecimal. The answer C was loaded onto the data segment of our NASM code. So were the matrices A and B. A matrix multiplication routine was then implemented in NASM, and the output generated was compared with the output generated by the C code(i.e with the values that were loaded into memory).

Implementation Details
The implementation of Matrix Multiplication in NASM was overlayed on a pre-written template *prot_start.s*

0.3 Assignment 3

```
Task Switching: TODO List
```

- Setup the GDT properly with all the code, data and stack segments defined. Refer Page 76 of the manual.
- Load the Task Register with the address in the GDT where the description of the Task State Segments begin.
-