X86 ATOM AND ARM SINGLE-BOARD-COMPUTER PLATFORM INSTRUCTION MODULES:

1. COMPLETE MODULE RESOURCES TO BE DEVELOPED

1) Module Source
2) Makefile
3) Build script
4) Example input and output test sessions
5) Kernel version
6) Platform

2. OBJECTIVE INSTRUCTION MODULES WITH REFERENCE TO ESSENTIAL LINUX DEVICE DRIVERS

1) Interrupt using serial port as interrupt input and serial output data from device driver as interrupt generation. This is based on the parallel port interrupt system, but replaces the parallel port with the serial port (since the parallel port is absent from most platforms.
2) CF flash driver – read and write to CF
3) PCI Access to device
4) USB Driver
5) Video Driver
6) Audio Driver
7) Block Driver
8) Network Device Driver
9) NAND Flash (MTD) Device Driver
10) EDAC Driver – Access to ECC information
11) Kprobes debugging examples

3. OBJECTIVE INSTRUCTION MODULES WITH REFERENCE TO LINUX FORUM

1) ioctl
http://www.linuxforu.com/2011/08/io-control-in-linux/

2) USB Drivers
http://www.linuxforu.com/2011/10/usb-drivers-in-linux-1/

http://www.linuxforu.com/2011/11/usb-drivers-in-linux-2/

http://www.linuxforu.com/2011/12/data-transfers-to-from-usb-devices/

3) Storage
http://www.linuxforu.com/2012/01/device-drivers-partitions-hard-disk/

http://www.linuxforu.com/2012/02/device-drivers-disk-on-ram-block-drivers/

4) MTD Devices
http://www.linuxforu.com/2012/01/working-with-mtd-devices/

5) proc file system

http://www.linuxforu.com/2012/03/device-drivers-kernel-window-peeping-through-proc/

6) Module interactions
http://www.linuxforu.com/2012/05/linux-device-drivers-module-interactions/

7) Generic Hardware Interface Drivers (video RAM)
http://www.linuxforu.com/2011/06/generic-hardware-access-in-linux/

8) Serial Port Driver
http://www.linuxforu.com/2011/07/accessing-x86-specific-io-mapped-hardware-in-linux/

Shouldn't worry about prior background, this is the opportunity to learn about OS internals. Differentiator in terms of personal experience. We've already made a great deal of progress in other modules. It has never been done before in any scale.

3 hours of prep for every unit of 199. Pick a module different than other students have done. Look at one for review purposes and then review accordingly. Then later on build a pathway toward the online final system. Objective: buy a 50 dollar atom motherboard and have all the capability and truly learn. Nobody has any prior experience at all, from 202 or 180D has any. Someone might have example source code, but from this code to an actual thing you can execute is a big leap, so no one has ever done it. To learn about system utilities about compiling user-space code etc etc you can get to speed in 3 days.

But the operating system is not designed that way. In userspace, you can drive a car anywhere in the world with no problem. But if someone is to take a car apart and change the RPM on the automatic transmission in 4th gear, that is what work on OS space is like. The engineer needs to be able to take the car apart and do it right there. Intel is still very very excited on this. We also want to move all the modules to ARM quad core next quarter, size of a credit card.

Go visit Digvijay today, then get a short intro module and also get assigned a module.

Engr IV 63-139