Lab 4.1

- Log on **Linux**
- **`cp /export/home/corsi/laface/Debian-6.0-rev4-2012.ova  /var/tmp`**
- **`cd /var/tmp`**
- **Right click** on icon **`Debian-6.0-rev4-2012.ova`** and open with **`virtualbox`**
- Select **Storage - Controller (SATA) – Virtual Disk Image**
- Replace the existing path **/export/…..** with
  `/var/tmp/sXXXXXX/Debian 6.0 laf-2012-disk1.vmdk`
- **Import** **(**wait 5 minutes**)**

- Check/install **`nasm`** (assembler)
- Check/install **`ddd`** (visual debugger)
- Check / install **`qemu`** (machine emulator and virtualizer)
- Read the file **`Booting a PC to run a kernel.pdf`**
- Using the **`ddd`** debugger, follow the steps from bootstrap to the **`GRUB`** kernel selection menu (read the file **`Installing GRUB on a hard disk image file.pdf`**

  No report is necessary for this.

Lab 4.2

Follows the steps for creating the kernel and the bootable hard **`hd.img`** file in directory **`MK/paging`**

Use the tutorial in http://www.jamesmolloy.co.uk/tutorial_html/ to understand the main and the functions of this minimal kernel.

- Change the file **`main.c`** to verify which is the first address that produces a page fault.

  Hint: Increment **`ptr`**, starting from **`0`**, by the dimension of a page (take care of the pointer arithmetic).

  Use the functions **`monitor_write`**, **`monitor_write_dec`**, **`monitor_write_hex`** to print lines such as:

  ```
  Normal access at address 0x0 at page 0
  Normal access at address 0x1000 at page 1
  .. .. .. .. .. ..
  ```

Take note of the **number of pages** that your kernel has allocated.

Lab 4.3

Write another file **main1.c** that

- Writes the page number on the first address of each page
- Prints the content of the content of the first address of all mapped pages (the number information is the one derived from previous main). Ex.:

  prt 0x0 (page 0) contains 0

  prt 0x1000 (page 1) contains 1

- Swap frame 0 and 1 in the page table
- Prints the content of the first address of these pages, so that you should obtain an output such as:

  page 0 now contains 1

  page 1 now contains 0

Produce a report that illustrates the data structure of the directory page table and page table, and the functions that you have used.

Use figures and tables to make your report easily readable.

Exercise Lab 4.3 - Barrier

Implement a sequential program in C that takes a single argument **k** from the command line. The program creates two vectors (**v1** and **v2**) of dimension **k**, and a matrix (**mat**) of dimension **k**x**k**, which are filled with random numbers in the range **[-0.5 0.5]**, then it performs the product **$v1^T$ * mat * v2**, and print the result. This is an example for **k=5**:

```
v1^T = [-0.0613   -0.1184   0.2655    0.2952   -0.3131]
mat=[  -0.3424   -0.3581    0.1557    0.2577    0.2060
        0.4706   -0.0782   -0.4643    0.2431   -0.4682
        0.4572    0.4157    0.3491   -0.1078   -0.2231
       -0.0146    0.2922    0.4340    0.1555   -0.4538
        0.3003    0.4595    0.1787   -0.3288   -0.4029]
v2^T = [-0.3235   0.1948   -0.1829    0.4502   -0.4656]

Result: 0.0194
```

Perform the product operation in two steps: **v = mat * v2**, which produces a new vector **v**, and **result = $v1^T$ * v**

Then, write a concurrent program using threads that performs the same task. The main thread creates the vectors, the matrix, and **k** threads. Then, it waits the termination of the other threads.

Each thread **i** performs the product of the **i-th** row vector of **mat** and **v2,** which produces the **i-th** element of vector **v**.

One of the created threads, the **last** one terminating its product operation, performs the final operation **result= $v1^T$ * v**, and prints the result.