# CIS 722 Fall,2014
# Project 1

**Due:**

1. Part 1 and Part 2: 5:00pm on 10/14/2014 (Tuesday)
2. Part 3: 5:00pm on 10/21/2014 (Tuesday)

**Prerequisite**

Assume that you work in directory "proj1" in the home directory (/root)

1. Map proj1.img to floppy 1 (/dev/fd1) and mount /dev/fd1 to /mnt ("mount /dev/fd1 /mnt")
2. Copy all the files in /mnt to your project directory (/root/proj1)
    1. This can be done by "cp –r /mnt/* proj1" (assuming you are in /root)
    2. Do not work in /mnt, since in order to create your Minix image in boot.img mapped to /dev/fd0, you must "umount" /dev/fd1 (= proj1.img) This is probably a bug in Minix, but we cannot fix it easily
3. umount /dev/fd1 (= proj1.img);
4. Confirm that your proj1 directory has the following files/directories
    - Proj1_1, proj1_2, proj1_3
5. Map boot.img to floppy 0 (/dev/fd0)

**Part 1**

After reading the source code carefully and checking many web pages, I have come to believe that the Minix 3 kernel runs with interrupts disabled. Furthermore, the kernel (including all the interrupt handlers) does not issue software interrupt. Therefore, an interrupt never occurs when the kernel uses the kernel stack (k_stack) [refer to Slide 5-4]. That is, the code that controls the stack structures using variable "k_reenter" is not necessary.

Remove this logic completely from the Minix source code.

Work in directory proj1_1, which contains directory "kernel."
In kernel, you will find

1. Makefie, .depend, REAME.txt (read this file)
2. mpx386.s (this is a copy of the original mpx386.s (/usr/src/kernel/mpx386.s)
3. mpx.s (you do not modify this file, but you need this to assemble mpx386.s

Here is how you make your new Minix

1. Go to proj1/proj1_1/kernel
2. Modify mpx386.s
    - Do not forget to change the file references on lines 60, 61, and 62 from relative to absolute.
3. Issue "make"; this will create file "kernel" in the directory.
4. Go back to proj1_1 (make sure you do not have file "image"; if you do, delete it) and issue "make fdboot". When asked, specify /fd0
5. Shutdown to go back to the monitor and issue "boot fd0" to boot your MINIX from the floppy image boot.img

When you submit your work,

1. write whether your program runs or not

2. submit a hard copy of the functions you have modified and mark the exact lines you have modified with a colored marker (give your hard copy to me or put it in my mailbox)

**Part 2**

Assume that you want to port Minix 3 to a machine with enough memory and speed, and decided to rewrite functions(not macros) hwint_master(int irq) and hwint_slave(int irq) in C to improve the readability and maintainability.

The new functions are called from interrupt entries for hardware interrupts 0-7 (lines 06530-6560) and 8-15 (lines 06582-6612), respectively, as follows:

```
 .align 16
_hwint00:
        call save     ! first, save the interrupted context and switch to
                      ! the kernel context
        push 0        ! interrupt routine for irq 0 (clock).
        call _hwint_master  !  you may use "jmp" instead of "call"
        pop eax             ! release the actual parameter
        ret
 .align 16
_hwint01:            ! Interrupt routine for irq 1 (keyboard)
        call save
        push 1
        call _hwint_master
        pop eax
        ret
                .
                .
```

Each interrupt entry point must follow the above calling sequence. You may use the following functions to access the I/O address space:

- unsigned inb(port_t port): refer to /usr/include/ibm/portio.h
- void outb(port_t port, u8_t value)

You must place your C functions in a new .c file named hwint.c (this file name is referred to in Makefile)
When you modify, follow the rules below:

1. You must not modify any *.h file.
2. You must not add any new variable in data or bss.

When you submit your work,

1. write whether your program runs or not
2. submit a hard copy of the functions you have modified and mark the exact lines you have modified with a colored marker (give your hard copy to me or put it in my mailbox)

Work in directory proj1_2, which contains directory “kernel.”
In kernel, you will find

1. Makefie, .depend, REAME.txt (read this file)

2. mpx.s (you do not modify this file, but you need this to assemble mpx386.s)
3. hwint.c (you write your C code in this file)

In addition, you must copy the mpx386.s file you developed in Part 1. If you could not complete Part 1, use the original mpx386.s in /usr/src/kernel/.

Here is how you make your new Minix

1. Go to proj1/proj1_2/kernel
2. Modify mpx386.s and whint.c
3. Issue “make”; this will create file “kernel” in the directory.
4. Go back to proj1_2 (make sure you do not have file “image”; if you do, delete it) and issue “make fdboot”. When asked, specify /fd0
5. Shutdown to go back to the monitor and issue "boot fd0" to boot your MINIX from the floppy image boot.img

**Part 3**

This time, assume that you want to port MINIX to a machine with a very small amount of memory. Recall that hwint_master(irq) (lines 06515-06527) and hwint_slave(irq) (lines 06566-06579) are macros (not functions). Thus, you have decided to change them to functions to save memory space. The new assembly functions are called from interrupt entries for hardware interrupts 0-7 (lines 06530-6560) and 8-15 (lines 06582-6612), respectively, as follows:

```
        .align 16
_hwint00:
        push 0            ! interrupt routine for irq 0 (the clock).
        call hwint_master !  you may use "jmp" instead of "call"
        .align 16
_hwint01:               ! Interrupt routine for irq 1 (keyboard)
         push 1
         call hwint_master
               .
               .
               .
```

You must follow the rules below:

1. Each interrupt entry point must follow the above calling sequence.
2. You may modify any function in mpx386.s.
3. You must not modify any *.h file.
4. You must not add any new variable in data or bss.
5. You are not allowed to put the body of "save" in line in the new functions.

Refer to the man entry for "as" and the Pentium Programmers manual for assembly instructions. I believe the following assembly instructions will be enough to complete the project

- move eax, k_stacktop ! move the address denoted by label "k_stacktop" to eax
- move eax, (_proc_ptr) ! move the CONTENTS of the address denoted by label "_porc_ptr" to eax
- call _irq_table (eax*4) ! the effective address = _irq_table (offset) + the contents of eax (index register) * 4 (index)
- mov eax,_irq_table (eax*4) ! move the contents of address _irq_table(eax*4) to eax
- lea eax,_irq_table (eax*4) ! move the value _irq_table(eax*4) to eax

- jmp _irq_table(eax*4) ! jump to the address specified in the contents of address _irq_table(eax*4)
- jmp eax ! jump to the address in eax

Notes on "pushad"
(from http://web.itu.edu.tr/kesgin/mul06/intel/instr/pusha_pushad.html)
**PUSHAD** – Push All Registers onto Stack
Pushes all general purpose registers onto the stack in the following order: (E)AX, (E)CX, (E)DX, (E)BX, (E)SP, (E)BP, (E)SI, (E)DI. *The value of SP is the value before the actual push of SP.*

Draw memory images of the stacks (the register saving area in the PCB and the kernel stack) as you modify your code.

Work in directory proj1_3, which contains directory "kernel."
In kernel, you will find
1. Makefie, .depend, REAME.txt (read this file)
2. mpx.s (you do not modify this file, but you need this to assemble mpx386.s

In addition, you must copy the mpx386.s file you developed in Part 1. If you could not complete Part 1, use the original mpx386.s in /usr/src/kernel/.

Here is how you make your new Minix
1. Go to proj1/proj1_3/kernel
2. Modify mpx386.s
3. Issue "make"; this will create file "kernel" in the directory.
4. Go back to proj1_3 (make sure you do not have file "image"; if you do, delete it) and issue "make fdboot". When asked, specify /fd0
5. Shutdown to go back to the monitor and issue "boot fd0" to boot your MINIX from the floppy image boot.img

When you submit your work,
1. write whether your program runs or not
2. submit a hard copy of the functions you have modified and mark the exact lines you have modified with a colored marker (give your hard copy to me or put it in my mailbox)

**How To Submit your code**

1. Change the name of the img file from proj1.img to your LastName_FirstName.img (e.g., mizuno_masaaki.img).
2. Map your_name.img to Floppy 1 (/dev/fd1) and issue "mount /dev/fd1 /mnt"
3. Copy only the files you have modified (mpx386.s, and for Part 2, also hwint.c) to the corresponding locations in /mnt
   - You are not supposed to modify other files
   - I will copy the entire file structure from your floppy (image) to a working directory and test your implementations.
4. Issue "umount /dev/fd1"
5. After shutting down VirtualBox, make a zip file of your .img file and submit it to the dropbox named Project1_12 (for Part 1 and Part 2) or Project1_3 (for Part 3).