Administrative things

Prof. Peter Desnoyers
pj d@ccs.neu.edu
Room - 334 WwH

TA: metpavel@ccs.neu.edu (Pavel Melnikov)

Prexeq: computer architecture, c, java

12 Lectures, Oct
Midterm: Oct 21, 09
Final: Dec 16, 09

Late homeworks — 15% penalty

Contents:

- OS basics
- Virtual memory
- Hardware virtualization
- File & block I/O, DMA
- OS Security
What is an Operating System? Manish Goel (mgo)  
- Resource Management: Memory allocation, managing disk space etc.  
- Virtual machine: platform for running programs.  
- Is a mediator between hardware and the programs.

Basics: 1) Program Loading & context switching  
2) Virtual memory  
3) Hardware Virtualization (supported by VMware)  
4) File & block I/O, DMA  
5) Security

Series of small programs using CS5600
1) First program uses the Frame Buffer Display hardware to write "Hello"

```plaintext
begin:  mov str, r1
        mov 5, r2
        mov fbuf, r3

loop:  mov *(r1++), r4
        mov r4, *(r3++)
        decr r2
        jnz loop

done:  jmp done
```

Program copies bytes 1 by 1 to memory to display the string.

```
Hello count fbuf
```

2) Another device in "toy" computer is Keyboard controller.

```plaintext
begin:  mov str, r1
        mov 5, r2
        mov fbuf, r3

loop:  mov *(status), r4
        cmp r4, 0  if *status=0
            jz loop1  goto loop1
        mov *(key), r4

loop1: mov *(status), r4
        mov r4, *(r3++)
        decr r2
        jnz loop

done:  jmp done
```

## Frame Buffer

```
H e l l o
```

 Kodak DCC-0000

```
F 800  status
F 801  key
```

Read the Key 

status goes to a Keyboard 

goes from 0 to 1.
3) Initialize $r_1, r_2 \rightarrow \mathbb{R}$

```
loop:
call GETKEY
mov +($r_++), r4
push r4
call PUTCHAR
pop r4
```

GETKEY is a function that reads a key from the keyboard. It is a reusable subroutine and can be used in other programs.

First primitive OS aka "library OS". It provides a basic set of subroutines to program.

OS is placed at a predefined address (8000). Program code is placed at lower level starting from 0000 to 8000. A boundary is created between the OS & program code. This scheme works alright if you don’t upgrade the OS.

All the subroutines have predefined memory locations & if address of one is changed others will be affected. Also, it might work differently on different architectures.
To prevent this we introduce indirection, so the new memory map is

A vector is used & code makes use of that vector to make calls. So, even if the OS is upgraded there calls won't break.

This OS can be made equivalent to MSDOS 1.0 if the vector is added as an Interrupt vector table

```
INT 1 GETKEY
INT 2 PUTCHAR
```

It is burned into a hardware as part of a BIOS to initialize the operating system.

→ Classic way of managing memory for a single process

```
Stock

Heap — malloc() or new in c++.
```

Region of memory used when you allocate memory using malloc or new.

Scheme results in maximal usage of the memory. If stack & heap regions meet system could crash if the allocation is not proper.
int function (int arg1, int arg2)
{
    int local1;

    we generally push the arguments backwards to handle variable arguments, we know & where the first argument is but not the last.

    PUSH arg2
    PUSH arg1

    CALL function

    POP 2 \rightarrow result in ro/eax1...

    function:
    
    decr SP
    ...
    ...
    ...

    mov result, ro
    incr SP
    RET

(Recursion)