CS 354 - Machine Organization & Programming
Thursday, November 21, 2019

Project p5 (4.5%): DUE at 10 pm on Monday, December 2nd
Project p6 (4.5%): Assigned on Tuesday, November 26th

Homework hw7 (1.5%): DUE at 10 pm Wednesday, November 27th

Last Time

- Unions
- Pointers
- Function Pointers
- Buffer Overflow & Stack Smashing
- Flow of Execution
- Exceptional Events
- Kinds of Exceptions

Today

- Kinds of Exceptions (from last time)
- Transferring Control via Exception Table
- Exceptions in IA-32 & Linux
- Processes and Context
- User/Kernel Modes
- Context Switch
- Context Switch Example

Next Time

- Signals
- **Read:** B&O 8.5 intro, 8.5.1 - 8.5.3, 8.5.4 p. 745

Transferring Control via Exception Table

Transferring Control to an Exception Handler

1. push

2. push

→ What stack is used for the push steps above?

3. do indirect function call

indirect function call

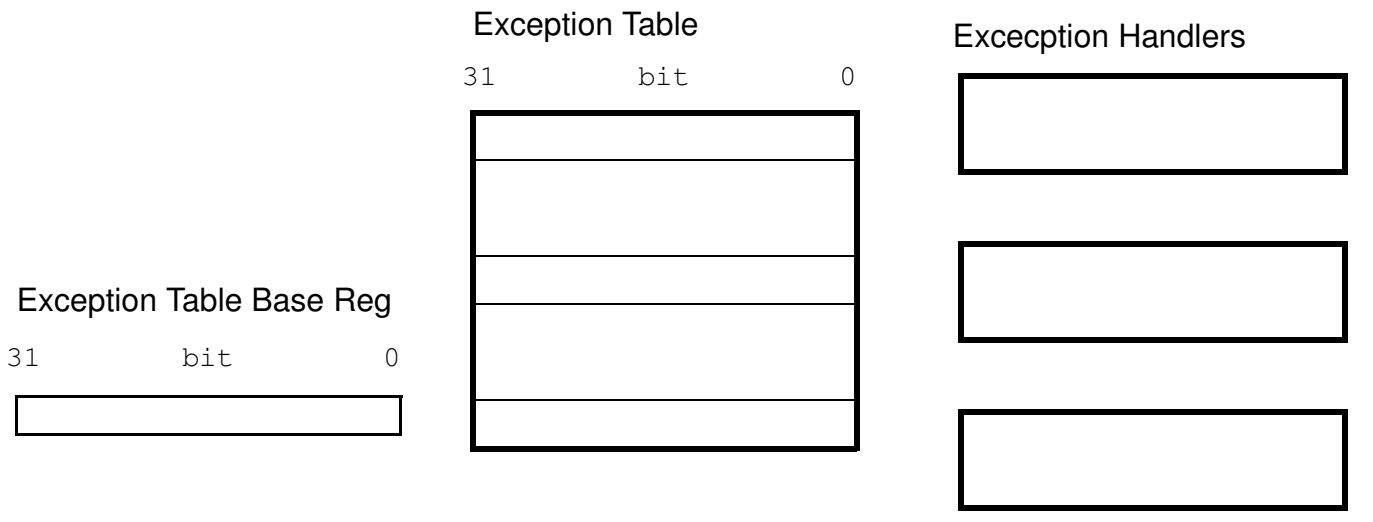
EHA is for exception handler's address

ETBR is for exception table base reg

ENUM is for exception number

Exception Table

exception number



Exceptions in IA-32 & Linux

Exception Numbers and Types

0 - 31 are defined by processor	0 13 14 18
32 - 255 are defined by OS	128 ($0x80)

System Calls and Service Numbers

1 exit
2 fork
3 read file 4 write file 5 open file 6 close file
11 execve

Making System Calls

1.)

2.)

3.) `int $0x80`

System Call Example

```
#include <stdlib.h>
int main(void) {
   write(1, "hello world\n", 12);
   exit(0);
}
```

Assembly Code:

```
   .section .data
   string:
      .ascii "hello world\n"
   string_end:
      .equ len, string_end - string
   .section .text
   .global main
   main:
      movl $4, %eax
      movl $1, %ebx
      movl $string, %ecx
      movl $len, %edx
      int $0x80
      movl $1, %eax
      movl $0, %ebx
```

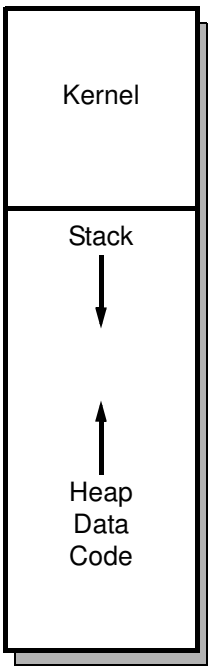
Processes & Context

Recall, a *process*

-
-

Why?

Process VAS



Key illusions

→ Who is the illusionist?

Concurrency

scheduler

interleaved execution

time slice

time	proc A	proc B	proc C

parallel execution

time	proc A	proc B	proc C

User/Kernel Modes

What? Processor *modes* are

mode bit

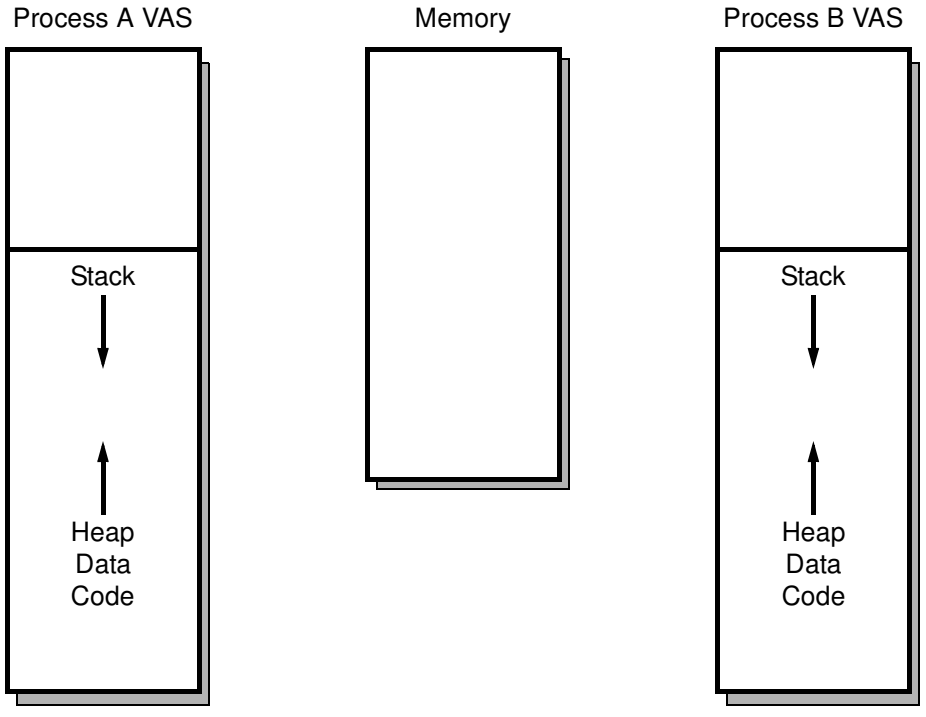
kernel mode

user mode

flipping modes

-
-
-

Sharing the Kernel



Context Switch

What? A *context switch*

-
-

When?

Why?

How?

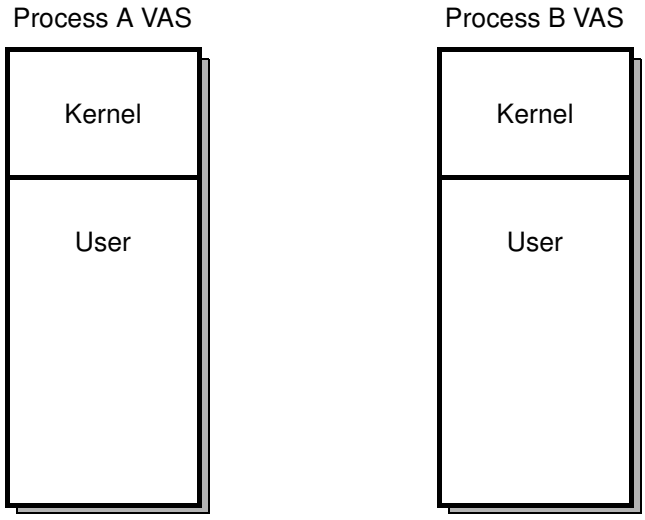
1.
2.
3.

❋ *Context switches*

→ What is the impact of a context switch on the cache?

Context Switch Example

read()



1.

2.

3.

4.

5.

6.

7.

8.