CS452/652 Real-Time Programming Course Notes

Daniel M. Berry, Cheriton School of Computer Science
University of Waterloo

Intel x86 Architecture

- Registers
- Segmentation
- Global Descriptor Table

8 General Purpose Registers

8 general-purpose registers (GPRs), each 32 bit:

EAX, EBX, ECX, EDX,

ESP, EBP, ESI, EDI

ESP is a.k.a. the Stack Pointer

EBP is a.k.a. the Base Pointer

16-bit Versions of 8 GPRs

AX, BX, CX, DX,

SP, BP, SI, DI

Each of these is nothing more than the lower 16 bits of the corresponding E register.

Each of the first four has a high 8 bits and a low 8 bits:

AH, AL, BH, BL, CH, CL, DH, DL,

Segmented Memory

Each address reference is confined to one segment, i.e., a slice of memory, and is represented as an offset from the start of a segment:

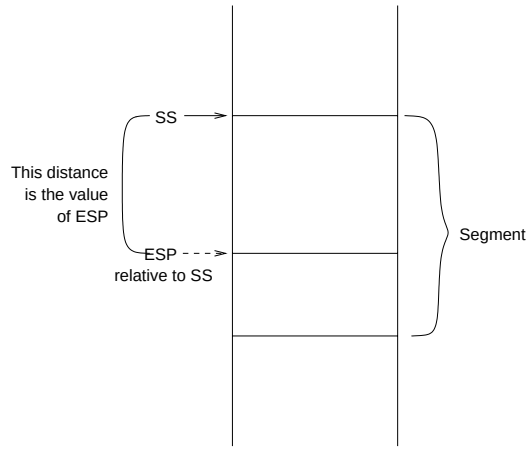
physicalAddress = startOfSegment + memoryOffset

Segment Registers

CS, DS, ES, FS, GS, SS

each 16 bits

Stack Segment (SS)



The Available Segments

SS	Stack Segment
CS	Code Segment
DS	Data Segment
ES	Extra Data Segment
FS	Extra Data Segment
GS	Extra Data Segment

Available Segments, Cont'd

A program may not reference addresses outside the bounds of its segments.

This is memory protection.

Segment Register Contents

Each segment register effectively specifies:

- lower bound for memory accesses,
- upper bound for memory accesses,
- access rights, i.e., read|write|execute,
- etc.,

for its segment.

All This in 16 Bits?

How do you pack *all* this information in a 16 bit segment register?

Global Descriptor Table

Each segment register is an index into a table called the Global Descriptor Table (GDT)

The GDT is an array of 8-byte entries.

Each entry indicates:

- lower bound for memory accesses,
- upper bound for memory accesses,
- access rights, i.e., read|write|execute,
- etc.,

for its segment.

Example

If DS = 0x28 (0d40), the memory reference:

Then, DS:0x34 means "Add 0x34 to the base address of GDT entry DS/8 = 40/8 = 5."

So, if GDT[5] has base address = 0x100, then DS:0x34 means physical address 0x134, …

provided that GDT[5] has an upper bound of at least 0x34.

Task Segments

Each task, including the kernel, needs 2 entries in the GDT:

1. CS
2. DS

There is no GDT in place when the kernel boots!

Compiler Assumptions

A compiler assumes that SS = DS.

Therefore you should set DS = ES = FS = GS = SS for each task.

Setting up GDT

The location of the GDT is stored in a register called GDTR.

x86 instructions
lgdt sets GDTR
sgdt reads GDTR

Setting up the GDT is the first thing your kernel should do.

EFLAGS

There is another register, EFLAGS, condition codes:

e.g., whether hardware interrupts are enabled, results of last comparison, etc.

Loading a Task: ELF Format

ELF = Executable and Linkable Format:

Set up CS segment to point to code segment in ELF file.

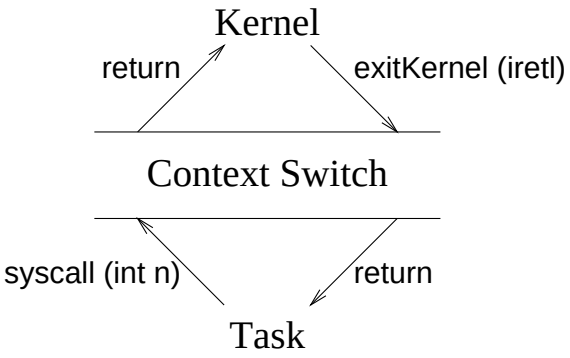
Allocate memory for task's data segment.

Copy data segment from ELF file to newly allocated memory.

Set up DS to point to the newly allocated memory.

Don't forget about uninitialized data.

Context Switch



int n Behavior

int *n*:

- pushes ELFAGS, CS, and EIP values into executing task's stack
- looks up *n*th entry in interrupt descriptor table (IDT)
- jumps to the address installed in IDT[*n*]

iretl Behavior

iretl:

- pops ELFAGS, CS, and EIP values from executing task's stack
- restores these popped values into the ELFAGS, CS, and EIP registers.

From Task1 to Kernel

1. Set up syscall parameters
2. int *n*

---

3. save task1's state on task1's stack: pushal saves all 8 GPRs
4. switch stacks to kernel's stack
5. restore kernel state from kernel stack
   - CS, EIP come from IDT
   - DS — whatever you used for the kernel in GDT
   - ESP — save as a global variable.

---

6. return from exitKernel

From Kernel to Task2

1. save kernel's state on kernel's stack
2. switch stack to task2's stack
3. restore task2's state from task2's stack: popal restores all 8 GPRs
4. set up return value of int *n*
5. iretl

---

6. return from syscall

First Time

The first time a task is loaded, put values on its stack so that on exitKernel, they will be popped like for any previously existing task.

Another example of faking it!