Intermediate IntlX 86 Processor Architecture & Assembly.

Part 2:

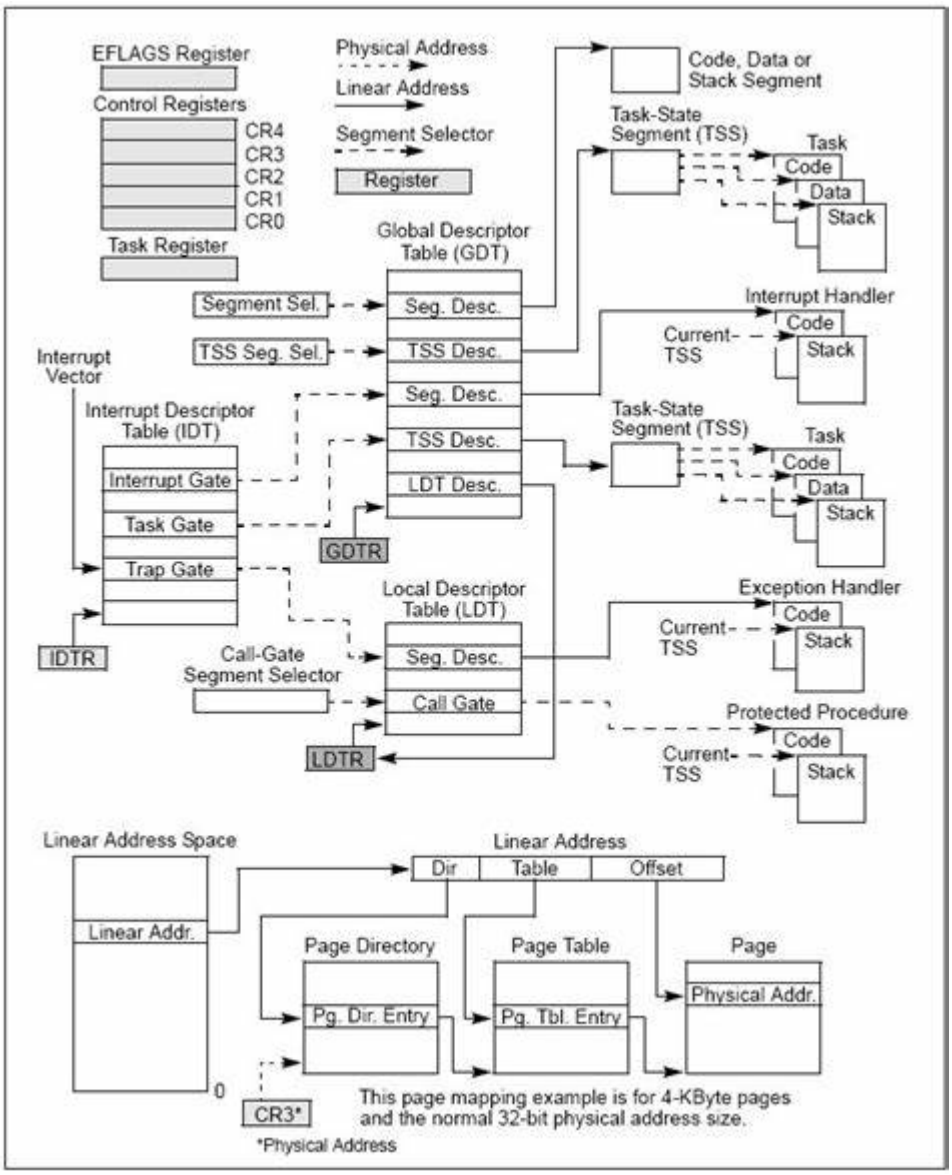
This course will cover the following :

> Memory Segmentation Paging Interrupts Debugging, I/O

In this lecture, we will dwell on IA-32, not 64 bit architecture yet.(that is for another one, wait for it 😃)

Also there wont be floating point assembly or registers since in reverse engineering or in malware analysis there is not much of a use for it.

This is what we are goint to learn 😃



CPUID- CPU Feature Identification.

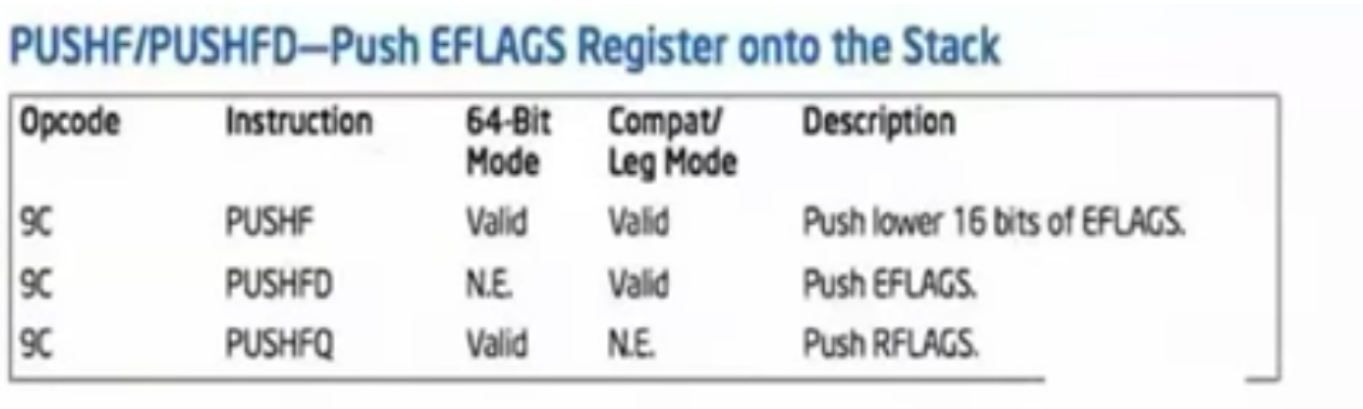
Different processors support different features. `CPUID` is how we know if the chip we are running on supports newer features, such as hardware virtualization, 64 bitmode(asdasdafa), Hyperthreading, thermal monitors etc.

CPUID does not have operands. Rather it **takes input** as value preloaded into `eax` (and possible ecx). After it finishes , the outputs are stored to `eax, ebx, ecx and edx`

so if you want your code to be compatible, you need to check some features before implementing.

ID flag in EFLAGS(bit 21), this is the CPUID flag. If it set to 0, you set to 1 and if it stays at 1, then it has CPUID, if it returned back to 0, then you dont have CPUID. But how do we read and write EFLAGS?

In order to manipulate the flags, we have 2 instructions.



PUSHF/PUSHFD—Push EFLAGS Register onto the Stack

Opcode	Instruction	64-Bit Mode	Compat/ Leg Mode	Description
9C	PUSHF	Valid	Valid	Push lower 16 bits of EFLAGS.
9C	PUSHFD	N.E.	Valid	Push EFLAGS.
9C	PUSHFQ	Valid	N.E.	Push RFLAGS.

there are FLAGS which are 16 bits and also there are E(xtended)FLAGS, which are 32 bits. make sure which one to push and pull. the problem occurs because all of them have the same opcode, which is `9C`. the instuction makes the difference.

`PUSHFD`: If you need to read the entire EFLAGS register, make usre you use PUSHFD not PUSHF. The difference is, PUSHFD uses Dword size flags so its not 16 bits but 32 bits.

`POPFD`: