

# Intermediate IntlX 86 Processor Architecture & Assembly.

## Part 2:

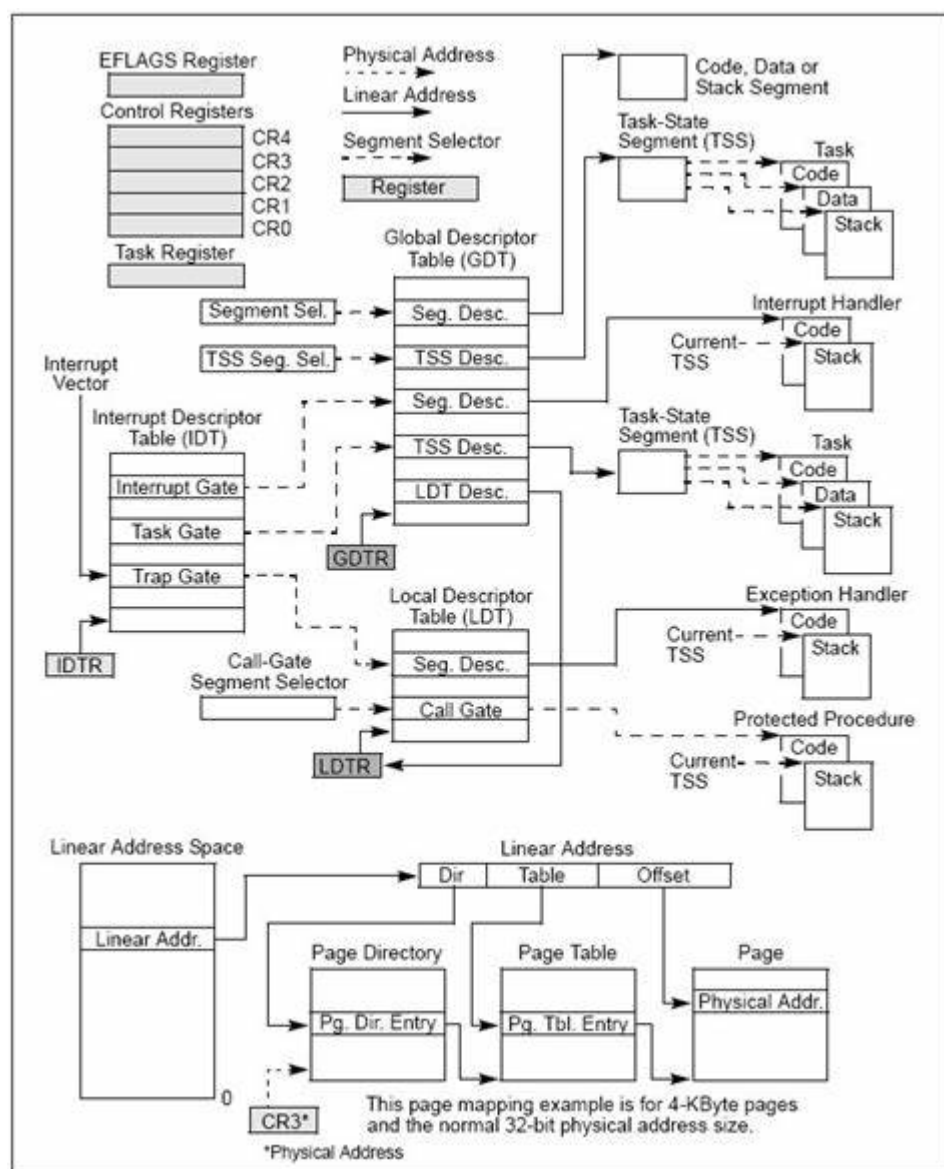
This course will cover the following :

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.

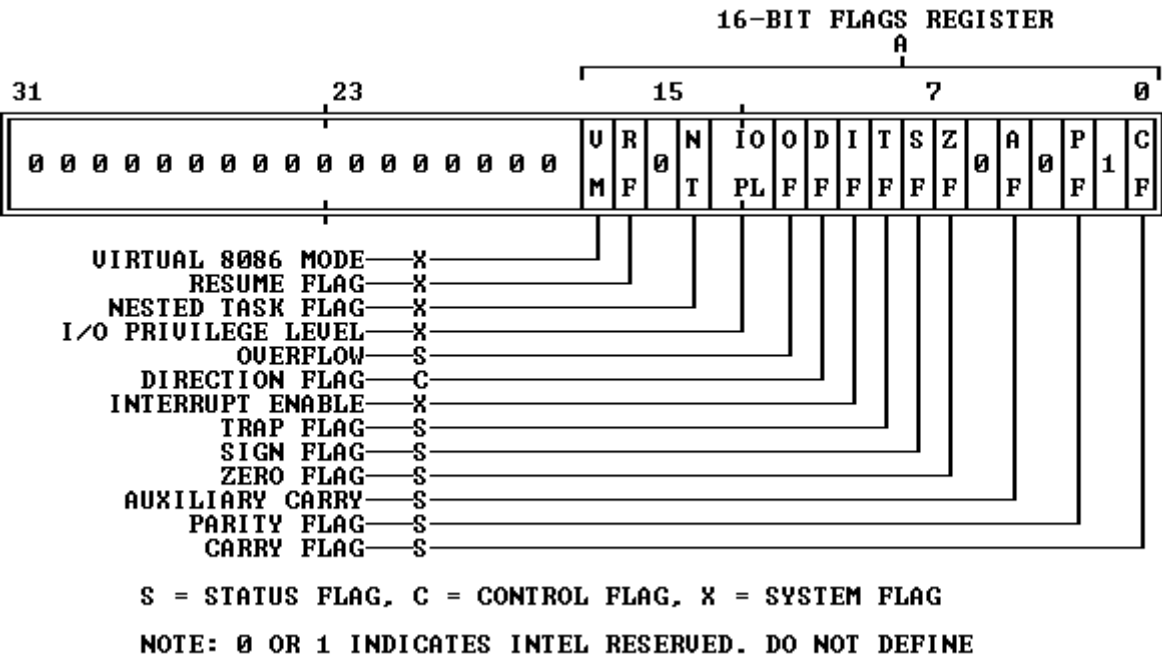
so PUSHFD takes the flag and puses it to the stack, just like anything else.

opcode for PUSHFD is **9C**

**POPFD**: There are some flags which will not be transferred from the stack to EFLAGS unless you are in ring 0. these are security purpose flags. we are generally operating on ring 3. If you need to set the entiure EFLAGS register, make sure you use **POPFD** not just **POPF**, same 16 bit issue.

opcode for POPFD is **9D**

Figure 2-8. EFLAGS Register



Information Returned by CPUID Instruction


Initial EAX Value	Information Provided about the Processor	
	Basic CPUID Information	
0H	EAX EBX ECX EDX	Maximum Input Value for Basic CPUID Information (see Table 3-7). "Genu" "ntel" "inel"
1H	EAX EBX   ECX EDX	Version Information (Type, Family, Model, and Stepping ID) Bits 7-0: Brand Index Bits 15-8: CLFLUSH line size. (Value * 8 = cache line size in bytes) Bits 23-16: Number of logical processors per physical processor. Bits 31-24: Local APIC ID Reserved Feature Information (see Figure 3-4 and Table 3-9)
2H	EAX EBX ECX EDX	Cache and TLB Information Cache and TLB Information Cache and TLB Information Cache and TLB Information
3H	EAX EBX ECX  EDX	Reserved. Reserved. Bits 00-31 of 96 bit processor serial number. (Available in Pentium® III processor only; otherwise, the value in this register is reserved.) Bits 32-63 of 96 bit processor serial number. (Available in Pentium III processor only; otherwise, the value in this register is reserved.)
	Extended Function CPUID Information	
80000000H	EAX  EBX ECX EDX	Maximum Input Value for Extended Function CPUID Information (see Table 3-7). Reserved. Reserved. Reserved.
80000001H	EAX  EBX ECX EDX	Extended Processor Signature and Extended Feature Bits. (Currently Reserved.) Reserved. Reserved. Reserved.
80000002H	EAX EBX ECX EDX	Processor Brand String. Processor Brand String Continued. Processor Brand String Continued. Processor Brand String Continued.
80000003H	EAX EBX ECX EDX	Processor Brand String Continued. Processor Brand String Continued. Processor Brand String Continued. Processor Brand String Continued.

Initial EAX Value	Information Provided about the Processor	
80000004H	EAX EBX ECX EDX	Processor Brand String Continued. Processor Brand String Continued. Processor Brand String Continued. Processor Brand String Continued.

CPUIDs are also returning processor manufacturer ID strings. for example,

```
AuthenticAMD
CyrixInstead
GenuineIntel
sis sis sis
```

This is what I got running the cpuid.c file in the source files



```
C:\Users\mehmetyavuziyagis\Desktop\IntermediateX86Code\Debug\CPUID.exe
maxBasicCPUID = 0x16, vendorString = GenuineIntel
```

here `0x16` is actually what is stored in my `eax`. so nothing big.