Batch: B1 **Roll No.: 1711072**

Experiment / assignment / tutorial No. 6

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Write a assembly program to find type of CPU inside the machine using CPUID instruction.

Aim: To learn instructions of Pentium.

Expected Outcome of Experiment:

CO4: Identify and describe multicore processors

Books/ Journals/ Websites referred:

1. **Advanced Microprocessor: By Roy & Bhurchandi (Tata McGraw Hill).**
2. **http://www.**

Pre Lab/ Prior Concepts:

Eflag Register:

The CPUID instruction supports two sets of functions. The first set returns basic processor information. The second set returns extended processor information.

CPUID instruction provides processor identification in register EAX, EBX, ECX, EDX. This information identifies INTEL as vendor, gives the family modes and stepping of processor.

The ID flag (bit 21) in the EFLAGS register indicates support for the CPUID instruction.

Condition codes (e.g., carry, sign, overflow) and mode bits are kept in a 32-bit register named EFLAGS. Figure 1 defines the bits within this register.

The flags control certain operations and indicate the status of the Pentium processor. Besides status and control flag bits, the flag register also contains system flags.

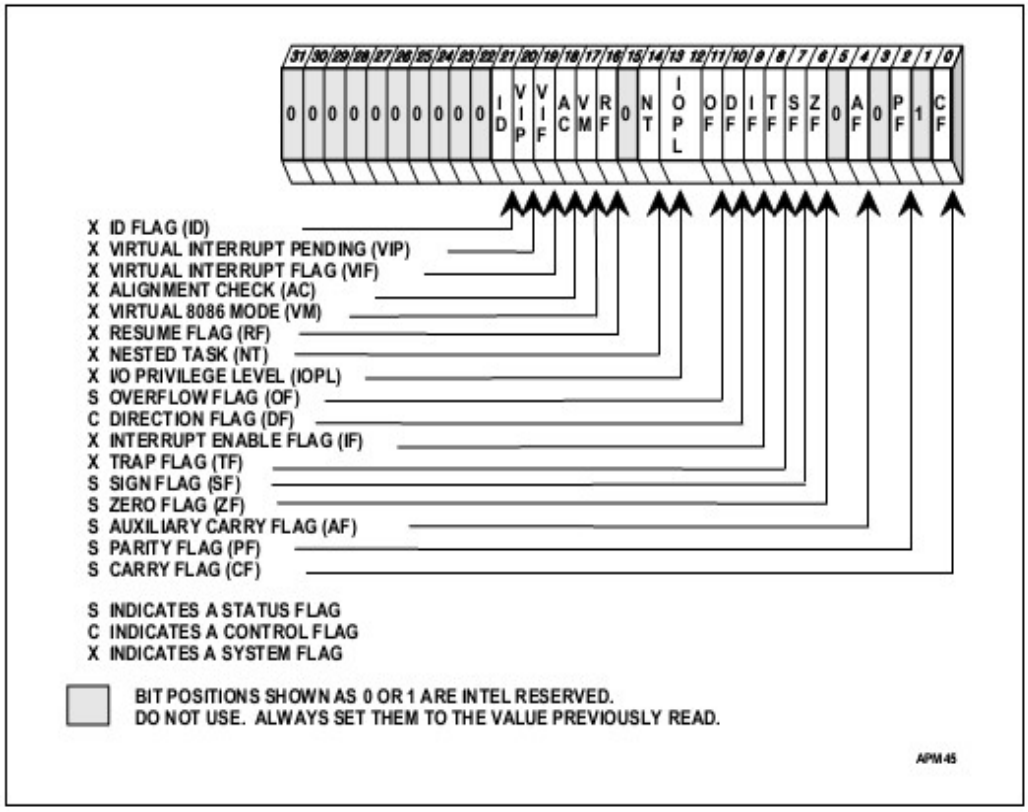


Figure 1 EFLAG Register

The output from the CPUID instruction is fully dependent upon the contents of the EAX register. This means, by placing different values in the EAX register and then executing CPUID, the CPUID instruction will perform a specific function dependent upon whatever value is resident in the EAX register.

1)In order to determine the highest acceptable value for the EAX register input and CPUID functions that return the basic processor information, the program should set the EAX register parameter value to "0" and then execute the CPUID instruction as follows:

MOV EAX, 00000000H

CPUID

After the execution of the CPUID instruction, a return value will be present in the EAX register.

Always use an EAX parameter value that is equal to or greater than zero and less than or equal to this highest EAX "returned" value.

So as to print numerical values, take two counters one is of 4 because each digit requires 4 bits & second is to count nos..Now move the contents of temporary to real register & rotate it to left by four places, because each digit requires four bits. Now again MOV the real register contents to temporary registers because afterwards we will lose those contents. ANDing operation is done with the contents of AL register to 0Fh, now compare the contents of al register with 0Ah if carry is generated then print that digit with adding 30h(so as to get real value not ASCII) else just add 07h to register contents with adding 30h to it. Repeat this procedure for all eight values.

A vendor identification string is retuned in EBX register for Intel processors ,the vendor identification string is 'Genuine Intel' as shown .vendor identification string is returned in the EBX, EDX, and ECX registers. For Intel(R) processors, the vendor identification string is "GenuineIntel" as follows:

EBX ←756e6547h (* "Genu", with G in the low nibble of BL *)

EDX ←49656e69h (* "ineI", with i in the low nibble of DL *)

ECX ←6c65746eh (* "ntel", with n in the low nibble of CL *)

So as to print character values, take two counters one is for total no of characters that is 4 because each char requires 8 bits. Now move the contents of temporary register to real register so as to retain original values. Now first char is printed using,

```
MOV dl, al

MOV ah, 02h

INT 21h
```

Now again move the contents of temporary to real register & rotate through right the contents of EAX to CL times .Again move the contents of real register to temporary register. Now

```
MOV dl, al

MOV ah, 02h

INT 21h
```

which will display the next character ,decrement the counter & repeat the procedure till counter becomes zero.

2. When the input value is 1, the processor returns versions information in EAX register and feature information in EDX register.EBX & ECX are reserved. When the input value is 1, the processor returns version information in the EAX register (see "Version

Information in the EAX Register"). The version information consists of an IA-32 processor family identifier, a model identifier, a stepping ID, and a processor type. The model, family, and processor type for the first processor in the Intel Pentium 4 processor family is as follows:

Model—0000B

Family—1111B

Processor Type—00B

The available processor types are given in the table "Processor Type Field". Intel releases information on stepping IDs as needed.

Processor Type Field

Type	Encoding
Original OEM processor	00B
Intel(R) OverDrive(R) processor	01B
Dual processor*	10B
Intel(R) reserved	11B

Since all the values are digits repeat the above digit display function.

3. When the input value is 2,the processor returns information about the processor, internal caches &ICB's in the EAX, EBX, ECX & EDX

When the EAX register contains a value of 2, the CPUID instruction loads the EAX, EBX, ECX and EDX registers with descriptors that indicate the processors cache and TLB characteristics.

The lower 8 bits of the EAX register (AL) contain a value that identifies the number of times the

CPUID has to be executed to obtain a complete image of the processor's caching systems. For example, the Pentium 4 processor returns a value of 1 in the lower 8 bits of the EAX register to indicate that the CPUID instruction need only be executed once (with EAX = 2) to obtain a complete image of the processor configuration.

The remainder of the EAX register, the EBX, ECX and EDX registers contain the cache and TLB descriptors..

When the input value is 2, the processor returns information about the processor's internal caches and TLBs in the EAX, EBX, ECX, and EDX registers. The encoding of these registers is as follows:

The least-significant byte in register EAX (register AL) indicates the number of times the CPUID instruction must be executed with an input value of 2 to get a complete description of the processor's caches and TLBs. The first member of the family of Pentium 4 processors will return a 1.

The most significant bit (bit 31) of each register indicates whether the register contains valid information (set to 0) or is reserved (set to 1).

If a register contains valid information, the information is contained in 1 byte descriptors. The table "Encoding of Cache and TLB Descriptors" shows the encoding of these descriptors. Note that the order of descriptors in the EAX, EBX, ECX, and EDX registers is not defined; that is, specific bytes are not designated to contain descriptors for specific cache or TLB types. The descriptors may appear in any order.

Encoding of Cache and TLB Descriptors:

The first member of the family of Pentium 4 processors will return the following information about caches and TLBs when the CPUID instruction is executed with an input value of 2:

EAX 66 5B 50 01H

EBX 0H

ECX 0H

EDX 00 7A 70 00H

These values are interpreted as follows:The least-significant byte (byte 0) of register EAX is set to 01H, indicating that the CPUID instruction needs to be executed only once with an input value of 2 to retrieve complete information about the processor's caches and TLBs.

The most-significant bit of all four registers (EAX, EBX, ECX, and EDX) is set to 0, indicating that each register contains valid 1-byte descriptors.Bytes 1, 2, and 3 of register EAX indicate that the processor contains the following:

50H—A 64-entry instruction TLB, for mapping 4-KByte and 2-MByte or 4-MByte pages.

5BH—A 64-entry data TLB, for mapping 4-KByte and 4-MByte pages.

66H—An 8-KByte 1st level data cache, 4-way set associative, with a 64-byte cache line size.

The descriptors in registers EBX and ECX are valid, but contain null descriptors.

Bytes 0, 1, 2, and 3 of register EDX indicate that the processor contains the following:

00H—Null descriptor.

70H—A 12-KByte 1st level code cache, 4-way set associative, with a 64-byte cache line size.

7AH—A 256-KByte 2nd level cache, 8-way set associative, with a sectored, 64-byte cache line size.

00H—Null descripton

Initial EAX Value	Information Provided about the Processor	
	Basic CPUID Information	
0H	EAX	Maximum Input Value for Basic CPUID Information (see Table 3-7).
	EBX	"Genu"
	ECX	"ntel"
	EDX	"inel"
1H	EAX	Version Information (Type, Family, Model, and Stepping ID)
	EBX	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
	ECX	Reserved
	EDX	Feature Information (see Figure 3-4 and Table 3-9)
2H	EAX	Cache and TLB Information
	EBX	Cache and TLB Information
	ECX	Cache and TLB Information
	EDX	Cache and TLB Information
3H	EAX	Reserved.
	EBX	Reserved.
	ECX	Bits 00-31 of 96 bit processor serial number. (Available in Pentium ®III processor only; otherwise, the value in this register is reserved.)
	EDX	Bits 32-63 of 96 bit processor serial number. (Available in Pentium III processor only; otherwise, the value in this register is reserved.)

Program:

```
.model small
.586p
.data
        teax dd ?
        tebx dd ?
        tecx dd ?
        tedx dd ?
        str1 db '00h :$'
        str2 db '01h :$'
        str3 db '02h :$'
        str4 db '03h :$'
.code
.startup
        mov eax,00000000h
        cpuid
        mov teax,eax
        mov tebx,ebx
        mov tecx,ecx
        mov tedx,edx
        lea dx,str1
        mov ah,09h
        int 21h
        call newline
        call dispd
        call newline
        mov eax,tebx
        mov teax,eax
        call dispc
        mov edx,tedx
        mov teax,edx
        call dispc
        mov ecx,tecx
```

```
mov teax,ecx
call dispc
call newline
call newline
mov eax,00000001h
cpuid
mov teax,eax
mov tebx,ebx
mov tecx,ecx
mov tedx,edx
lea dx,str2
mov ah,09h
int 21h
call newline
call dispd
call newline
mov eax,tebx
mov teax,eax
call dispd
call newline
mov eax,tecx
mov teax,eax
call dispd
call newline
mov eax,tedx
mov teax,eax
call dispd
call newline
call newline
mov eax,00000002h
cpuid
mov teax,eax
mov tebx,ebx
mov tecx,ecx
mov tedx,edx
```

```
lea dx,str3
mov ah,09h
int 21h
call newline
call dispd
call newline
mov eax,tebx
mov teax,eax
call dispd
call newline
mov eax,tecx
mov teax,eax
call dispd
call newline
mov eax,tedx
mov teax,eax
call dispd
call newline
call newline
mov eax,00000003h
cpuid
mov teax,eax
mov tebx,ebx
mov tecx,ecx
mov tedx,edx
lea dx,str4
mov ah,09h
int 21h
call newline
call dispd
call newline
mov eax,tebx
mov teax,eax
call dispd
call newline
```

```
mov eax,tecx
mov teax,eax
call dispd
call newline
mov eax,tedx
mov teax,eax
call dispd
call newline
mov ah,4ch
int 21h

newline proc near
mov al,0ah
mov dl,al
mov ah,02h
int 21h
ret
endp
dispd proc near
mov ch,08h
mov cl,04h
up: mov eax,teax
rol eax,cl
mov teax,eax
and al,0fh
cmp al,0ah
jc digit
add al,06h
        digit:
        add al,30h
        mov dl,al
        mov ah,02h
        int 21h
        dec ch
        jnz up
```

```
        ret
        endp
        dispc proc near
        mov cl,08h
        mov ch,03h
        mov eax,teax
        mov dl,al
        mov ah,02h
        int 21h
        up1:
        mov eax,teax
        ror teax,cl
        mov eax,teax
        mov dl,al
        mov ah,02h
        int 21h
        dec ch
        jnz up1
        ret
        endp
end
```

Screenshot:

```
C:\USERS\STUDENT\DESKTOP>cpu.exe
00h :
0000000C
GenuineIntel

01h :
000306B3
00100800
7EC8EAAE
AEDAEAEE

02h :
76036301
00E0A5EE
00000000
00B10000

03h :
00000000
00000000
00000000
00000000
```

Conclusion: The CPUID program ran successfully as we were able to extract "GenuineIntel" from the CPU information.

Post Lab Descriptive Questions (Add questions from examination point view)

Explain BrandID instruction of Pentium processor.

Ans.

```
.model small
.586p
.data
        teax dd ?
        tebx dd ?
        tecx dd ?
        tedx dd ?
        str1 db '00h :$'
        str2 db '01h :$'
        str3 db '02h  03h  :$'
        str4 db '$'
        str5 db '04 :$'

.code
.startup
        mov eax,80000000h
        cpuid
        mov teax,eax
        mov tebx,ebx
        mov tecx,ecx
        mov tedx,edx
        lea dx,str1
        mov ah,09h
        int 21h
        call newline
        call dispd
        call newline
        mov eax,tebx
        mov teax,eax
        call dispd
        mov edx,tedx
        mov teax,edx
        call dispd
        mov ecx,tecx
        mov teax,ecx
        call dispd
```

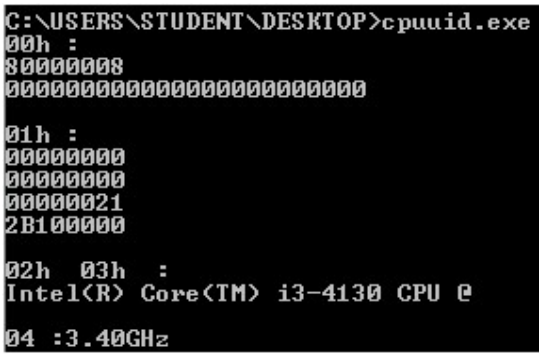
```
call newline
call newline
mov eax,80000001h
cpuid
mov teax,eax
mov tebx,ebx
mov tecx,ecx
mov tedx,edx
lea dx,str2
mov ah,09h
int 21h
call newline
call dispd
call newline
mov eax,tebx
mov teax,eax
call dispd
call newline
mov eax,tecx
mov teax,eax
call dispd
call newline
mov eax,tedx
mov teax,eax
call dispd
call newline
call newline
mov eax,80000002h
cpuid
mov teax,eax
mov tebx,ebx
mov tecx,ecx
mov tedx,edx
lea dx,str3
mov ah,09h
int 21h
call newline
call dispc

mov eax,tebx
mov teax,eax
call dispc

mov eax,tecx
mov teax,eax
call dispc
```

```
        mov eax,tedx
        mov teax,eax
        call dispc

        mov eax,80000003h
        cpuid
        mov teax,eax
        mov tebx,ebx
        mov tecx,ecx
        mov tedx,edx
        lea dx,str4
        mov ah,09h
        int 21h

        call dispc

        mov eax,tebx
        mov teax,eax
        call dispc

        mov eax,tecx
        mov teax,eax
        call dispc

        mov eax,tedx
        mov teax,eax
        call dispc
call newline
        call newline
        mov eax,80000004h
        cpuid
        mov teax,eax
        mov tebx,ebx
        mov tecx,ecx
        mov tedx,edx
        lea dx,str5
        mov ah,09h
        int 21h

        call dispc

        mov eax,tebx
        mov teax,eax
        call dispc

        mov eax,tecx
        mov teax,eax
```

```
call dispc

mov eax,tedx
mov teax,eax
call dispc

mov ah,4ch
int 21h


newline proc near
mov al,13

mov dl,al
mov ah,02h
int 21h
mov al,10

mov dl,al
mov ah,02h
int 21h
ret
endp
dispd proc near
mov ch,08h
mov cl,04h
up: mov eax,teax
rol eax,cl
mov teax,eax
and al,0fh
cmp al,0ah
jc digit
add al,06h
        digit:
        add al,30h
        mov dl,al
        mov ah,02h
        int 21h
        dec ch
        jnz up
        ret
        endp
        dispc proc near
        mov cl,08h
        mov ch,03h
        mov eax,teax
        mov dl,al
```

```
        mov ah,02h
        int 21h
        up1:
        mov eax,teax
        ror teax,cl
        mov eax,teax
        mov dl,al
        mov ah,02h
        int 21h
        dec ch
        jnz up1
        ret
        endp
end
```

Screenshot:



Date: 25/03/2019 **Signature of faculty in-charge**