

Sources

Intel Pentium® Processor Family Developer's Manual

Page 19.3.1.6 - INTERRUPT DISTRIBUTION MODES

Figure 19-9

[Link to manual](#)

Processor Programming Reference (PPR) for AMD Family 17h Model 01h, Revision B1 Processors

2.1.10.2 Local APIC - Local APIC Functional Description

[Link to manual](#)

Solaris on x86

[Link to manual](#)

Linux Kernel source code:

[Link to APIC source](#)

```
225      /*
226      * Quirk: some x86_64 machines can only use ph
ysical APIC mode
227      * regardless of how many processors are prese
nt (x86_64 ES7000
228      * is an example).
229      */
```

For instance - structure as descriptor for LVT -thermal Sensor

```
339/*330*/ struct { /* LVT - Thermal Sensor */
340      u32  vector      : 8,
341      delivery_mode    : 3,
342      __reserved_1     : 1,
343      delivery_status  : 1,
344      __reserved_2     : 3,
345      mask             : 1,
346      __reserved_3     : 15;
347      u32 __reserved_4[3];
348  } lvt_thermal;
```

Bochs x86 Emulator

[Link to APIC Emultaion in Bochs x86Emulator]

(<http://bochs.sourceforge.net/cgi-bin/lxr/source/cpu/apic.cc>)

Bochs GUI Debugger

<http://bochs.sourceforge.net/doc/docbook/user/internal-debugger.html#DEBUGGER-GUI>

NASM x86 Win32 Bootloader Bochs Emulator

[Link to APIC Emultaion in Bochs x86Emulator]

<https://forum.nasm.us/index.php?topic=1791.0>

Tutorial with simple bootloader on NASM

[Link to video tutorial with NASM bootloader]

<https://www.youtube.com/watch?v=ZA5MpQVtFeI>

NASM Installer

[NASM installer]

<https://www.nasm.us/pub/nasm/releasebuilds/2.14.01/win64/>

To compile NASM file - install NASM for all users, open nasm-shell console and navigate to file bootable.asm (create it before running

the compiler :D)

Execute the following command in NASM console:

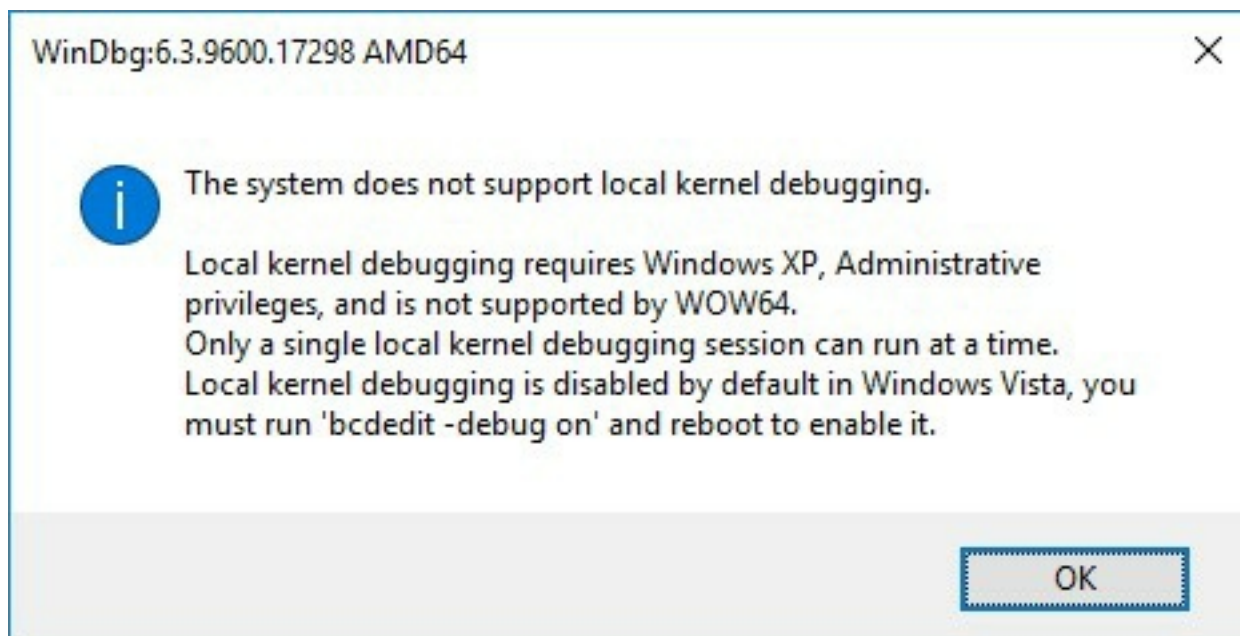
```
nasm -f bin -o bootloader.bin bootable.asm
```

MSR's OSDEV

https://wiki.osdev.org/Model_Specific_Registers

WinDbg install and execute

<https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/getting-started-with-windbg-kernel-mode->



Possible trouble - windbg can't be launched in kernel debugging mode on Win10.

Execute in PowerShell or CMD with administrator permissions

```
bcdedit -debug on
```

```
PS C:\Program Files (x86)\Windows Kits\8.1\Debuggers\x64>  
.\windbg.exe -o notepad.exe
```

xAPIC/ x2APIC

https://xem.github.io/minix86/manual/intel-x86-and-64-manual-vol3/o_fe12b1e2a880e0ce-371.html

Read MSR

<https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/rdmsr-read-msr->

MSR-Tools

<https://01.org/msr-tools/downloads/msr-tools-source-code>

Find and execute windbg

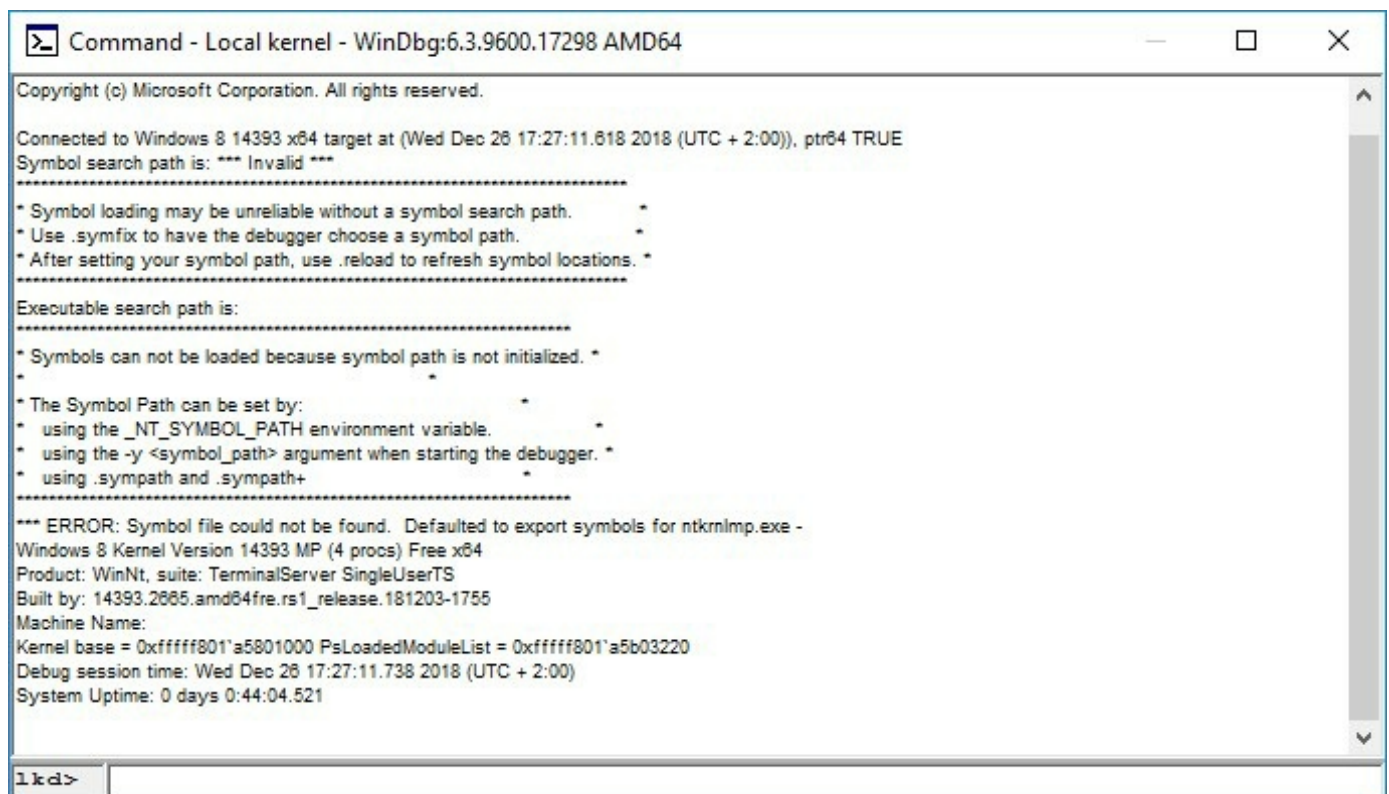
<https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/getting-started-with-windbg>

Local debugging in windows kernel

<https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/performing-local-kernel-debugging>

Local computer setup windebug

<https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/setting-up-local-kernel-debugging-of-a-single-computer-manually>



```
Command - Local kernel - WinDbg:6.3.9600.17298 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.

Connected to Windows 8 14393 x64 target at (Wed Dec 26 17:27:11.618 2018 (UTC + 2:00)), ptr64 TRUE
Symbol search path is: *** Invalid ***

*****
* Symbol loading may be unreliable without a symbol search path. *
* Use .symfix to have the debugger choose a symbol path. *
* After setting your symbol path, use .reload to refresh symbol locations. *
*****

Executable search path is:
*****
* Symbols can not be loaded because symbol path is not initialized. *
*
* The Symbol Path can be set by:
*   using the _NT_SYMBOL_PATH environment variable.
*   using the -y <symbol_path> argument when starting the debugger.
*   using .sympath and .sympath+
*****

*** ERROR: Symbol file could not be found. Defaulted to export symbols for ntkrnlmp.exe -
Windows 8 Kernel Version 14393 MP (4 procs) Free x64
Product: WinNt, suite: TerminalServer SingleUserTS
Built by: 14393.2685.amd64fre.rs1_release.181203-1755
Machine Name:
Kernel base = 0xfffff801'a5801000 PsLoadedModuleList = 0xfffff801'a5b03220
Debug session time: Wed Dec 26 17:27:11.738 2018 (UTC + 2:00)
System Uptime: 0 days 0:44:04.521

!kd>
```

Install QEMU on Ubuntu

<https://www.unixmen.com/how-to-install-and-configure-qemu-in-ubuntu/>

ACPI and Local APIC for x86

<https://github.com/p4nthr0/draft-x86-acpi-lapic>

How to:

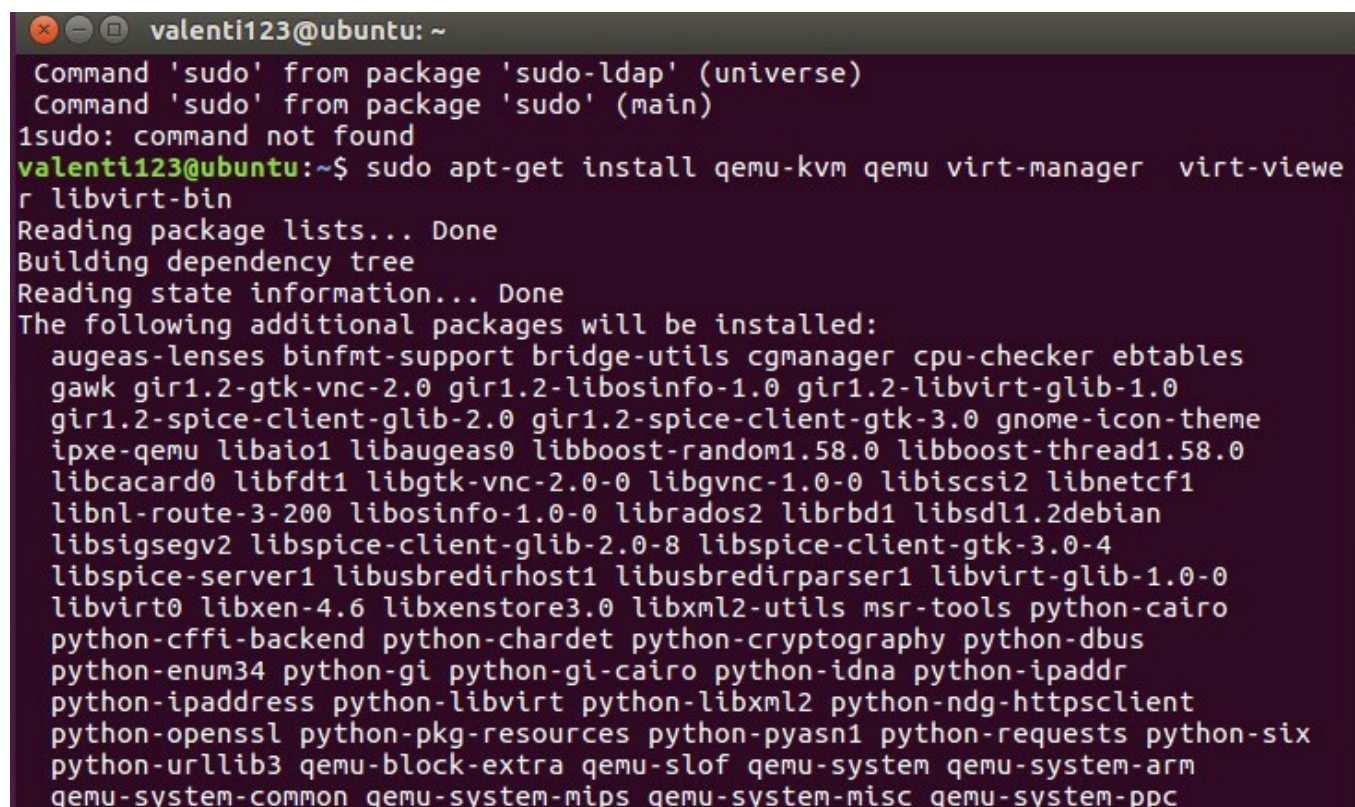
Mount image of Ubuntu LTS , open terminal and execute

```
sudo apt-get update
```

then execute

```
sudo apt-get install qemu-kvm qemu virt-manager virt-viewer  
libvirt-bin
```

for installing QEMU



```
valenti123@ubuntu: ~  
Command 'sudo' from package 'sudo-ldap' (universe)  
Command 'sudo' from package 'sudo' (main)  
1sudo: command not found  
valenti123@ubuntu:~$ sudo apt-get install qemu-kvm qemu virt-manager virt-viewer  
libvirt-bin  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  augeas-lenses binfmt-support bridge-utils cgmanager cpu-checker ebttables  
  gawk gir1.2-gtk-vnc-2.0 gir1.2-libosinfo-1.0 gir1.2-libvirt-glib-1.0  
  gir1.2-spice-client-glib-2.0 gir1.2-spice-client-gtk-3.0 gnome-icon-theme  
  ipxe-qemu libaio1 libaugeas0 libboost-random1.58.0 libboost-thread1.58.0  
  libcacard0 libfdt1 libgtk-vnc-2.0-0 libgvnc-1.0-0 libiscsi2 libnetcf1  
  libnl-route-3-200 libosinfo-1.0-0 librados2 librbd1 libSDL1.2debian  
  libsigsegv2 libspice-client-glib-2.0-8 libspice-client-gtk-3.0-4  
  libspice-server1 libusbredirhost1 libusbredirparser1 libvirt-glib-1.0-0  
  libvirt0 libxen-4.6 libxenstore3.0 libxml2-utils msr-tools python-cairo  
  python-cffi-backend python-chardet python-cryptography python-dbus  
  python-enum34 python-gi python-gi-cairo python-idna python-ipaddr  
  python-ipaddress python-libvirt python-libxml2 python-ndg-httpsclient  
  python-openssl python-pkg-resources python-pyasn1 python-requests python-six  
  python-urllib3 qemu-block-extra qemu-slof qemu-system qemu-system-arm  
  qemu-system-common qemu-system-mips qemu-system-misc qemu-system-ppc
```


Check the python version for running example. Execute

```
python --version
```

The output must be Python 2.7.12 or later version

Install git. Execute

```
sudo apt-get install git
```

Create directory for example binary

```
cd Desktop  
mkdir QemuExample  
cd QemuExample
```

Clone the repository. Execute and then go into directory draft-x86-acpi-lapic and build the example.

```
git clone https://github.com/p4nthr0/draft-x86-acpi-lapic.  
git  
cd draft-x86-acpi-lapic  
make
```

For running QEMU emulator:

In folder with examples execute


```
qemu-system-x86_64 -hda kernel.bin -cpu core2duo -smp cores=4 -s -S
```

Another QEMU configurations:

```
qemu-system-x86_64 -hda kernel.bin -cpu phenom -smp cores=4 -s -S
```

Where:

- -hda - bootable image
- -cpu - - Specify a processor architecture to emulate. To see a list of supported architectures, run:

```
qemu-system-x86_64 -cpu ?
```

- -smp <NUMBER> - Specify the number of cores the guest is permitted to use. The number can be higher than the available cores on the host system. Use -smp \$(nproc) to use all currently available cores.

[QEMU documentation] (<https://wiki.gentoo.org/wiki/QEMU/Options>)

QEMU waiting for connection from GDB debugger, for running it - open the second Terminal and execute

```
gdb --command debug.py --batch
```

Where

- -batch

*Run in batch mode. Exit with status 0 after processing all the command files specified with `-x'` (and all commands from initialization files, if not inhibited with `-n`). Exit with nonzero status if an error occurs in executing the GDB commands in the command files. Batch mode may be useful for running GDB as a filter; for example to download and run a program on another computer; in order to make this more useful, the message *Program exited normally*. (which is ordinarily issued whenever a program running under GDB control terminates) is not issued when running in batch mode.*

[Official documentation for GDB debugger](#)

Possible output:

```

valenti123@ubuntu: ~/Desktop/QemuExample/draft-x86-acpi-lapic
Thread 1 hit Breakpoint 1, 0x00007c77 in ?? ()
valenti123@ubuntu:~/Desktop/QemuExample/draft-x86-acpi-lapic$ gdb --nh --command
debug.py --batch
0x0000ffff in ?? ()
Breakpoint 1 at 0x7c77
add symbol table from file "kernel.elf" at
    .text_addr = 0x21000

cpu_info[0].isbsp=0x1
cpu_info[0].lapic_id=0x0
cpu_info[0].lapic_base.low=0xfe00000
cpu_info[0].lapic_base.high=0x0

cpu_info[1].isbsp=0x0
cpu_info[1].lapic_id=0x1
cpu_info[1].lapic_base.low=0xfe00000
cpu_info[1].lapic_base.high=0x0

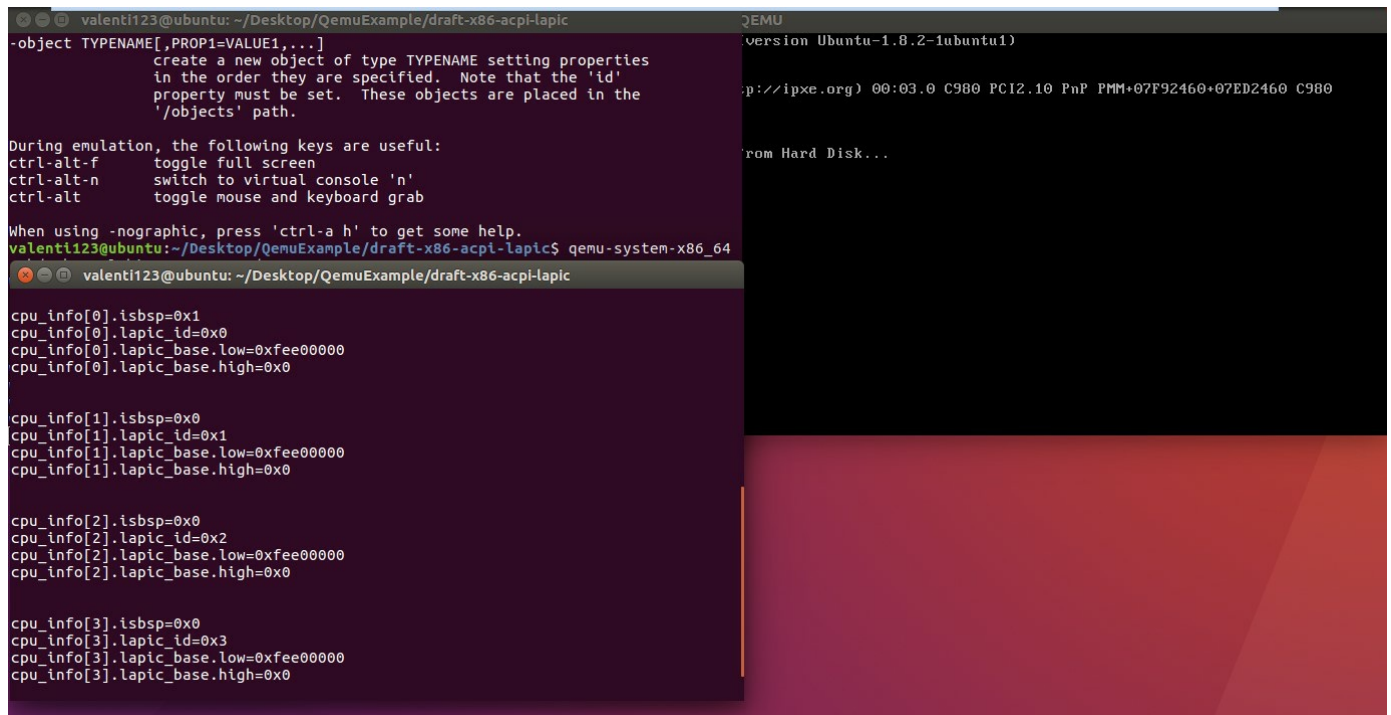
cpu_info[2].isbsp=0x0
cpu_info[2].lapic_id=0x2
cpu_info[2].lapic_base.low=0xfe00000
cpu_info[2].lapic_base.high=0x0

```

```

valenti123@ubuntu: ~/Desktop/QemuExample/draft-x86-acpi-lapic
-hda kernel.bin -cpu core2duo -smp cores=4 -s -S
WARNING: Image format was not specified for 'kernel.bin' and probing guessed raw
.
    Automatically detecting the format is dangerous for raw images, write o
perations on block 0 will be restricted.
    Specify the 'raw' format explicitly to remove the restrictions.
warning: TCG doesn't support requested feature: CPUID.01H:EDX.vme [bit 1]
warning: TCG doesn't support requested feature: CPUID.01H:EDX.vme [bit 1]
warning: TCG doesn't support requested feature: CPUID.01H:EDX.vme [bit 1]
warning: TCG doesn't support requested feature: CPUID.01H:EDX.vme [bit 1]
^Cqemu-system-x86_64: terminating on signal 2
valenti123@ubuntu:~/Desktop/QemuExample/draft-x86-acpi-lapic$ qemu-system-x86_64
-hda kernel.bin -cpu core2duo -smp cores=4 -s -S
WARNING: Image format was not specified for 'kernel.bin' and probing guessed raw
.
    Automatically detecting the format is dangerous for raw images, write o
perations on block 0 will be restricted.
    Specify the 'raw' format explicitly to remove the restrictions.
warning: TCG doesn't support requested feature: CPUID.01H:EDX.vme [bit 1]
warning: TCG doesn't support requested feature: CPUID.01H:EDX.vme [bit 1]
warning: TCG doesn't support requested feature: CPUID.01H:EDX.vme [bit 1]
warning: TCG doesn't support requested feature: CPUID.01H:EDX.vme [bit 1]
^Cqemu-system-x86_64: terminating on signal 2
valenti123@ubuntu:~/Desktop/QemuExample/draft-x86-acpi-lapic$

```



Launch APIC timer on ASM

<https://forum.osdev.org/viewtopic.php?t=10686>

APIC code examples

(Article apic1.pdf)

http://www.o3one.org/sources/oz_hw_smproc_486.s

<https://forum.osdev.org/viewtopic.php?p=107868#107868>

Additional code:

#define LAPICapicID	0x0020
#define LAPICversion	0x0030
#define LAPICtaskPriority	0x0080
#define LAPICendOfInterrupt	0x00B0
#define LAPIClogicalDestination	0x00D0

```

#define LAPICdestinationFormat      0x00E0
#define LAPICspurious               0x00F0
#define LAPICinterruptCommandLow    0x0300
#define LAPICinterruptCommandHigh   0x0310
#define LAPICLVTtimer               0x0320
#define LAPICLVTperfCounter         0x0340
#define LAPICLVT_LINT0              0x0350
#define LAPICLVT_LINT1              0x0360
#define LAPICLVTerror               0x0370
#define LAPICTIMERinitialCount      0x0380
#define LAPICTIMERcurrentCount      0x0390
#define LAPICTIMERdivider           0x03E0

#define LTIMERdivideBy1             0x0000000B
#define LTIMERdivideBy16           0x00000003
#define LTIMERdivideBy128          0x0000000A

#define LTIMERperiodic              0x00020000

#define APIC_DISABLE 0x10000
#define APIC_SW_ENABLE 0x10000
#define APIC_NMI (4<<8)
#define TMR_PERIODIC 0x20000
#define TMR_BASE_DIV (1<<20)

static int incrementable_counter = 0;

```



```

void cpuWriteIoApic(u32 *ioapicaddr, u32 reg, u32 value)
{
    u32 volatile *ioapic = ioapicaddr;
    ioapic[0] = (reg & 0xff);
    ioapic[4] = value;
}

void init_apic_counter( u32* _apicAdress )
{
    cpuWriteIoApic( _apicAdress ,LAPICdestinationFormat, 0
    xFFFFFFFFF0 );
    cpuWriteIoApic( _apicAdress ,LAPIClogicalDestination,
    0x1 );
    cpuWriteIoApic( _apicAdress ,LAPICLVTtimer, APIC_DISAB
    LE );
    cpuWriteIoApic( _apicAdress ,LAPICLVTperfCounter, APIC
    _NMI );
    cpuWriteIoApic( _apicAdress ,LAPICLVT_LINT0, APIC_DISA
    BLE );
    cpuWriteIoApic( _apicAdress ,LAPICLVT_LINT1, APIC_DISA
    BLE );
    cpuWriteIoApic( _apicAdress ,LAPICtaskPriority, 0 );
    cpuWriteIoApic( _apicAdress ,LAPICspurious, 39+APIC_SW
    _ENABLE);
    apic_counter_initial_value = cpuReadIoApic(_apicAdress
    ,LAPICTIMERinitialCount );
    cpuWriteIoApic( _apicAdress ,LAPICLVTtimer, 32);

```

```

        cpuWriteIoApic( _apicAddress ,LAPICTIMERdivider, LTIMER
divideBy1);

        cpuWriteIoApic( _apicAddress ,LAPICLVTtimer, 32|LTIMERp
eriodic);

        apic_counter_counted_value = cpuReadIoApic(_apicAddress
,LAPICTIMERcurrentCount );

        incrementable_counter = 1;
    }

int smp_init_cpu_info()
{
    rsdp_t *rsdp;
    madt_t *madt;

    u32 apicFirstCoreBase;

    ....
    apicFirstCoreBase = cpu_info[0].lapic_base.low;
    init_apic_counter(&apicFirstCoreBase);
    return num_of_cpus;
}

```

26/12/2018

Valentyn Korniienko

Ivan Semenenko

Tkachenko Danyil