# Paging System

The paging system is capable of mapping the entire 64 bit address space. A multi-level system of page directories and subdirectories is used. In most cases the address space mapped will be less than a full 64 bit address space. The paging system accommodates this by using a smaller directory hierarchy. For instance, if an application is less than 1GB in size, a two level page system is used. If the application can fit within 2MB only a single level is required. The depth of the directory system is controllable on an application basis. The page memory management unit takes care of walking the page tables in hardware in order to find a translation. Translations are stored in a translation look-aside buffer (TLB) which is a translation cache, so that the page tables don’t have to be walked for every translation.

## Page Tables

Page tables are 4kB in size and contain 512, 8 byte entries. The page table format is the same for both page directories, and page mapping tables.

## The Page Table Address Register

This register contains the base address of the page table in memory. The page table must be aligned on a 4kB boundary. The lowest three bits of the register identify the translation space.

|  |  |  |
| --- | --- | --- |
| Address63..12 | ~9 | TS3 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TS3 | Translation Space | Address Processed | | | | | |
| 0 | 2MB |  | | | | | Address20..12 |
| 1 | 1GB |  | | | | Address29..12 | |
| 2 | 512 GB |  | | | Address38..12 | | |
| 3 | 256 TB |  | | Address47..12 | | | |
| 4 | 128 XB |  | Address56..12 | | | | |
| 5 | 2^64 | Address63..12 | | | | | |
| 6 | not used |  | | | | | |
| 7 | not used |  | | | | | |

Page Directory Entries (PDE’s)

The page directory entries identify where in memory further subdirectories or translation pages are.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address63..12 | ~2 | a1 | d1 | p4 | c1 | r1 | x1 | w1 |

Page Table Entries (PTE’s)

Page table entries map the virtual address to a physical one.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address63..12 | ~2 | a1 | d1 | p4 | c1 | r1 | x1 | w1 |

a1: accessed

d1: dirty

p4: privilege level

c1: cacheable

r1: readable

x1: executable

w1: writeable

NOP Ramps

In a paged memory management unit, address translations take place continuously, not just at segment load time as would be for a segmentation unit. One nice feature about segmentation is that one can be sure that if the segment is loaded it is a continuously available memory range.

With paging on the other hand, page faults may occur at 4kB boundaries. When the page isn’t present in memory, it has to be loaded then the instruction can be executed. In Table888 most instructions are a single instruction word in length so they won’t cross a page boundary. However there are several prefix instructions, when combined with a prefix an instruction might cross a page boundary. This is bad news. The problem is that the prefixing would get lost in the shuffle to move the missing page into memory. There are two solutions, one is to go back a page in memory and re-execute the prefix after the missing page is loaded. This is complicated by the fact that both pages are required to be present in memory, otherwise the processor would thrash back and forth trying to execute the instruction. The second solution is to force the instruction stream to output the prefixes so that they don’t cross page boundaries. This can be handled by the assembler. The assembler handles the occasional case where a prefix instruction would cause an instruction to span a page boundary by outputting a series of NOPs to force the instruction onto the next memory page. The following example shows that the prefix instruction to a store byte operation is forced onto the next page of memory.

|  |
| --- |
| 00008FF0 41 F8 2A 90 00 bne fl0,kbdi2  00008FF5 16 01 24 00 00 ldi r1,#36  00008FFA EA EA EA EA EA ; imm  00009000 EA EA EA EA EA  00009005 FD 70 FF 03 10  0000900A A0 00 01 00 18 sb r1,LEDS |

Note that the NOP ramp’s won’t work if address space defined by a segment is paged out of memory and the segment isn’t aligned on a 4kB boundary.