Large Pages

Rodrigo Etchegaray Campisi 96856
Federico Soldo 98288

```
diff --git a/kern/entry.S b/kern/entry.S
index 100e92c..ede8908 100644
--- a/kern/entry.S
+++ b/kern/entry.S
@@ -58,15 +58,16 @@ entry:
        movl  $(RELOC(entry_pgdir)), %eax
        movl  %eax, %cr3

-       movl  %eax, %cr4
+       movl  %cr4, %eax
+       orl   $(CR4_PSE), %eax
+       movl  %eax, %cr4
+
        # Turn on paging.
        movl  %cr0, %eax
        orl   $(CR0_PE|CR0_PG|CR0_WP), %eax
        movl  %eax, %cr0

-       movl  %cr4, %eax
-       orl   $(CR4_PVI|CR4_PSE|CR4_VME), %eax
-       movl  %eax, %cr4
+

        # Now paging is enabled, but we're still running at a low EIP
        # (why is this okay?).  Jump up above KERNBASE before entering
diff --git a/kern/entrypgdir.c b/kern/entrypgdir.c
index 4f324d1..e203961 100644
--- a/kern/entrypgdir.c
+++ b/kern/entrypgdir.c
@@ -1,8 +1,6 @@
 #include <inc/mmu.h>
 #include <inc/memlayout.h>

-pte_t entry_pgtable[NPTENTRIES];
-
 // The entry.S page directory maps the first 4MB of physical memory
 // starting at virtual address KERNBASE (that is, it maps virtual
 // addresses [KERNBASE, KERNBASE+4MB) to physical addresses [0,
4MB)).
@@ -21,1039 +19,8 @@ __attribute__((__aligned__(PGSIZE)))
 pde_t entry_pgdir[NPDENTRIES] = {
        // Map VA's [0, 4MB) to PA's [0, 4MB)
        [0]
-               = ((uintptr_t)entry_pgtable - KERNBASE) + PTE_P,
+               = 0x000000 + PTE_P + PTE_PS,
        // Map VA's [KERNBASE, KERNBASE+4MB) to PA's [0, 4MB)
        [KERNBASE>>PDXSHIFT]
-               = ((uintptr_t)entry_pgtable - KERNBASE) + PTE_P + PTE_W
+               = 0x000000 + PTE_P + PTE_W + PTE_PS
 };
-
-// Entry 0 of the page table maps to physical page 0, entry 1 to
-// physical page 1, etc.
-};
-


+
 //
 // Map [va, va+size) of virtual address space to physical [pa, pa+size)
 // in the page table rooted at pgdir.  Size is a multiple of PGSIZE, and
@@ -349,7 +409,30 @@ pgdir_walk(pde_t *pgdir, const void *va, int
create)
 static void
 boot_map_region(pde_t *pgdir, uintptr_t va, size_t size, physaddr_t pa,
int perm)
 {
-       // Fill this function in
+#ifndef TP1_PSE
+       pte_t *pte;
+       for (size_t j = 0; j < size / PGSIZE; j++) {
+               pte = pgdir_walk(pgdir, (void *) (va + (uintptr_t)(j * PGSIZE)),
1);
+               *pte = (pa + j * PGSIZE) | perm | PTE_P;
+       }
+#else
+       pte_t *pte;
+       pde_t *pde;
+
+       if (pa % PTSIZE == 0) {
+               for (size_t i = 0; i < size / PTSIZE; i++) {
+                       pde = &pgdir[PDX(va + (uintptr_t)(i * PTSIZE))];
+                       *pde = (pa + i * PTSIZE) | perm | PTE_P | PTE_PS;
+               }
+       } else {
+               for (size_t j = 0; j < size / PGSIZE; j++) {
+                       pte = pgdir_walk(pgdir,
+                                    (void *) (va + (uintptr_t)(j * PGSIZE)),
+                                    1);
+                       *pte = (pa + j * PGSIZE) | perm | PTE_P;
+               }
+       }
+#endif
```

```
 }
```

```
  Physical page allocator: OK
  Page management: OK
  Kernel page directory: OK
  Page management 2: OK
  Large pages: OK (1.9s)
Score: 5/5
```

map_region_large

Con 2 niveles de indireccion, se usan 4MB para el page directory. Tenemos 2^10 paginas de 4 bytes cada una y a su vez 1024 page tables como maximo de 4MB, es decir 4294967296 bytes.
En el caso de Large Pages tenemos 4194304 bytes. Por lo tanto estamos ahorrando 4MB.

Es una cantidad fija, ya que sin importar la cantidad de memoria fisica que tenga la maquina, la parte del sistema operativo que trabaja con los page directories siempre es la misma.