

[PATCH 2/4] x86_64: changes to x86_64 architecture for alternative instruction improvements

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-02/msg10058.html>

- *From:* "Joerg Roedel" <joerg.roedel@xxxxxxx>
 - *Date:* Wed, 28 Feb 2007 15:18:00 +0100
-

From: Joerg Roedel <joerg.roedel@xxxxxxx>

In this patch updates the x86_64 architecture to work with the changes to alternative instructions in i386

Signed-off-by: Joerg Roedel <joerg.roedel@xxxxxxx>

```
---
Joerg Roedel
Operating System Research Center
AMD Saxony LLC & Co. KG
diff --git a/arch/x86_64/lib/clear_page.S b/arch/x86_64/lib/clear_page.S
index 9a10a78..ab525ee 100644
--- a/arch/x86_64/lib/clear_page.S
+++ b/arch/x86_64/lib/clear_page.S
@@ -53,7 +53,10 @@ ENDPROC(clear_page)
.align 8
.quad clear_page
.quad 1b
+ .quad 0
.byte X86_FEATURE_REP_GOOD
+ .byte 0
.byte .Lclear_page_end - clear_page
.byte 2b - 1b
+ .byte 0
.previous
diff --git a/arch/x86_64/lib/copy_page.S b/arch/x86_64/lib/copy_page.S
index 727a5d4..b4d0329 100644
--- a/arch/x86_64/lib/copy_page.S
+++ b/arch/x86_64/lib/copy_page.S
@@ -113,7 +113,10 @@ ENDPROC(copy_page)
.align 8
.quad copy_page
.quad 1b
+ .quad 0
.byte X86_FEATURE_REP_GOOD
```

[PATCH 2/4] x86_64: changes to x86_64 architecture for alternative instruction improvements

```
+ .byte 0
.byte .Lcopy_page_end - copy_page
.byte 2b - 1b
+ .byte 0
.previous
diff --git a/arch/x86_64/lib/copy_user.S b/arch/x86_64/lib/copy_user.S
index 70bebd3..d505df3 100644
--- a/arch/x86_64/lib/copy_user.S
+++ b/arch/x86_64/lib/copy_user.S
@@ -27,9 +27,12 @@
.align 8
.quad 0b
.quad 2b
+ .quad 0
.byte \feature /* when feature is set */
+ .byte 0
.byte 5
.byte 5
+ .byte 0
.previous
.endm
```

```
diff --git a/arch/x86_64/lib/memcpy.S b/arch/x86_64/lib/memcpy.S
index 0ea0ddc..b1e1686 100644
--- a/arch/x86_64/lib/memcpy.S
+++ b/arch/x86_64/lib/memcpy.S
@@ -123,7 +123,10 @@ ENDPROC(__memcpy)
.align 8
.quad memcpy
.quad 1b
+ .quad 0
.byte X86_FEATURE_REP_GOOD
+ .byte 0
.byte .Lfinal - memcpy
.byte 2b - 1b
+ .byte 0
.previous
```

```
diff --git a/arch/x86_64/lib/memset.S b/arch/x86_64/lib/memset.S
index 2c59481..566e179 100644
--- a/arch/x86_64/lib/memset.S
+++ b/arch/x86_64/lib/memset.S
@@ -127,7 +127,10 @@ ENDPROC(__memset)
.align 8
.quad memset
.quad 1b
+ .quad 0
.byte X86_FEATURE_REP_GOOD
+ .byte 0
.byte .Lfinal - memset
.byte 2b - 1b
+ .byte 0
```

[PATCH 2/4] x86_64: changes to x86_64 architecture for alternative instruction improvements

```
.previous
diff --git a/include/asm-x86_64/alternative.h b/include/asm-x86_64/alternative.h
index a6657b4..63cd8e5 100644
--- a/include/asm-x86_64/alternative.h
+++ b/include/asm-x86_64/alternative.h
@@ -10,11 +10,14 @@
struct alt_instr {
u8 *instr; /* original instruction */
u8 *replacement;
+ u8 *replacement2;
u8 cpuid; /* cpuid bit set for replacement */
+ u8 cpuid2; /* cpuid bit set for replacement2 */
u8 instrlen; /* length of original instruction */
u8 replacementlen; /* length of new instruction, <= instrlen */
- u8 pad[5];
-};
+ u8 replacementlen2;
+ u8 pad[3];
+} __attribute__((packed));

extern void apply_alternatives(struct alt_instr *start, struct alt_instr *end);

@@ -36,6 +39,12 @@ static inline void alternatives_smp_switch(int smp) {}

#endif

+/*
+ * use this macro(s) if you need more than one output parameter
+ * in alternative_io_*
+ */
+#define ASM_OUTPUT2(a, b) a, b
+
+/*
+ * Alternative instructions for different CPU types or capabilities.
+ */
@@ -54,9 +63,12 @@ static inline void alternatives_smp_switch(int smp) {}
".align 8\n" \
".quad 661b\n" /* label */ \
".quad 663f\n" /* new instruction */ \
+ ".quad 0x00\n" \
".byte %c0\n" /* feature bit */ \
+ ".byte 0x00\n" \
".byte 662b-661b\n" /* sourcelen */ \
".byte 664f-663f\n" /* replacementlen */ \
+ ".byte 0x00\n" \
".previous\n" \
".section .altinstr_replacement,\"ax\"\\n" \
"663:\\n\t" newinstr "\\n664:\\n" /* replacement */ \
@@ -78,9 +90,12 @@ static inline void alternatives_smp_switch(int smp) {}
".align 8\n" \
".quad 661b\n" /* label */ \
```

[PATCH 2/4] x86_64: changes to x86_64 architecture for alternative instruction improvements

```

.quad 663f\n" /* new instruction */\
+ ".quad 0x00\n" \
.byte %c0\n" /* feature bit */\
+ ".byte 0x00\n" \
.byte 662b-661b\n" /* sourcelen */\
.byte 664f-663f\n" /* replacementlen */\
+ ".byte 0x00\n" \
.previous\n" \
.section .altinstr_replacement,\"ax\"\n" \
"663:\n\t" newinstr "\n664:\n" /* replacement */\
@@ -93,14 +108,37 @@ static inline void alternatives_smp_switch(int smp) {}
.align 8\n" \
.quad 661b\n" /* label */\
.quad 663f\n" /* new instruction */\
+ ".quad 0x00\n" \
.byte %c[feat]\n" /* feature bit */\
+ ".byte 0x00\n" \
.byte 662b-661b\n" /* sourcelen */\
.byte 664f-663f\n" /* replacementlen */\
+ ".byte 0x00\n" \
.previous\n" \
.section .altinstr_replacement,\"ax\"\n" \
"663:\n\t" newinstr "\n664:\n" /* replacement */\
.previous" : output : [feat] "i" (feature), ##input

+/* Like alternative_io, but supports 2 possible alternatives */
+#define alternative_io_two(oldinstr, newinstr, feat, newinstr2, feat2,\
+ output, input...) \
+ asm volatile ("661:\n\t" oldinstr "\n662:\n" \
+ ".section .altinstructions,\"a\"\n" \
+ ".align 8\n" \
+ ".quad 661b\n" /* label */\
+ ".quad 663f\n" /* new instruction */\
+ ".quad 665f\n" /* new instruction 2 */\
+ ".byte %c[f]\n" /* feature bit */\
+ ".byte %c[f2]\n" /* feature bit 2 */\
+ ".byte 662b-661b\n" /* sourcelen */\
+ ".byte 664f-663f\n" /* replacementlen */\
+ ".byte 666f-665f\n" /* replacementlen 2 */\
+ ".previous\n" \
+ ".section .altinstr_replacement,\"ax\"\n" \
+ "663:\n\t" newinstr "\n664:\n" /* replacement */\
+ "665:\n\t" newinstr2 "\n666:\n" /* replacement2 */\
+ ".previous" : output : [f] "i" (feat), \
+ [f2] "i" (feat2),##input
)/*
* Alternative inline assembly for SMP.
*

```