

[PATCH][5/6] A different KGDB stub

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-02/4199.html>

From: Tom Rini (trini_at_kernel.crashing.org)

Date: 02/17/04

Date: Tue, 17 Feb 2004 15:04:08 -0700

To: Andrew Morton <akpm@osdl.org>, Kernel Mailing List <linux-kernel@vger.kernel.org>

The following is the ppc32-specific bits to this KGDB stub.

```
arch/ppc/8260_io/uart.c | 23
arch/ppc/Kconfig | 46 -
arch/ppc/kernel/entry.S | 69 ++
arch/ppc/kernel/irq.c | 13
arch/ppc/kernel/ppc-stub.c | 985 ++++++-----
arch/ppc/kernel/setup.c | 16
arch/ppc/mm/fault.c | 10
arch/ppc/platforms/lopec_setup.c | 44 -
arch/ppc/platforms/pmac_setup.c | 3
arch/ppc/platforms/prep_setup.c | 5
arch/ppc/platforms/sandpoint.c | 38 -
arch/ppc/syslib/Makefile | 2
arch/ppc/syslib/gen550_dbg.c | 23
arch/ppc/syslib/gen550_kgdb.c | 2
include/asm-ppc/kgdb.h | 21
include/asm-ppc/processor.h | 3
16 files changed, 373 insertions(+), 930 deletions(-)
--- linux-2.6.3-rc4/arch/ppc/8260_io/uart.c 2004-02-17 09:50:56.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/arch/ppc/8260_io/uart.c 2004-02-17 11:33:48.965044753 -0700
@@ -43,6 +43,7 @@
#include <linux/slab.h>
#include <linux/init.h>
#include <linux/delay.h>
+#include <linux/kgdb.h>
#include <asm/uaccess.h>
#include <asm/immap_8260.h>
#include <asm/mpc8260.h>
@@ -394,8 +395,21 @@ static _INLINE_ void receive_chars(ser_i
    status = bdp->cbd_sc;
#ifdef CONFIG_KGDB
    if (info->state->smc_scc_num == KGDB_SER_IDX) {
- if (*cp == 0x03 || *cp == '$')
- breakpoint();
+ while (i-- > 0) {
+ if (*cp == 0x03) {
```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```
+ breakpoint();
+ return;
+ }
+ if (*cp == '$') {
+ atomic_set(&kgdb_might_be_resumed, 1);
+ breakpoint();
+ atomic_set(&kgdb_might_be_resumed, 0);
+ return;
+ }
+ cp++;
+ }
+ bdp->cbd_sc |= BD_SC_EMPTY;
+ bdp->cbd_sc &= ~(BD_SC_BR | BD_SC_FR | BD_SC_PR | BD_SC_OV);
+ return;
+ }
#endif
@@ -2275,9 +2289,8 @@ static void my_console_write(int idx, co
static void serial_console_write(struct console *c, const char *s,
                               unsigned count)
{
-#if defined(CONFIG_KGDB_CONSOLE) && !defined(CONFIG_USE_SERIAL2_KGDB)
- /* Try to let stub handle output. Returns true if it did. */
- if (kgdb_output_string(s, count))
+#ifdef CONFIG_KGDB
+ if (kgdb_initialized)
+ return;
#endif
    my_console_write(c->index, s, count);
--- linux-2.6.3-rc4/arch/ppc/Kconfig 2004-02-17 09:54:25.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/arch/ppc/Kconfig 2004-02-17 11:34:27.618272112 -0700
@@ -609,7 +609,7 @@ config PPC_OF

config PPC_GEN550
    bool
- depends on SANDPOINT || MCPN765 || SPRUCE
+ depends on SANDPOINT || MCPN765 || SPRUCE || LOPEC || PPC_PREP
    default y

config FORCE
@@ -1170,41 +1170,7 @@ config DEBUG_SPINLOCK_SLEEP
    If you say Y here, various routines which may sleep will become very
    noisy if they are called with a spinlock held.

-config KGDB
- bool "Include kgdb kernel debugger"
- depends on DEBUG_KERNEL
- select DEBUG_INFO
- help
- Include in-kernel hooks for kgdb, the Linux kernel source level
- debugger. See <http://kgdb.sourceforge.net/> for more information.
- Unless you are intending to debug the kernel, say N here.
```

```

-
-choice
- prompt "Serial Port"
- depends on KGDB
- default KGDB_TTYS1
-
-config KGDB_TTYS0
- bool "ttyS0"
-
-config KGDB_TTYS1
- bool "ttyS1"
-
-config KGDB_TTYS2
- bool "ttyS2"
-
-config KGDB_TTYS3
- bool "ttyS3"
-
-endchoice
-
-config KGDB_CONSOLE
- bool "Enable serial console thru kgdb port"
- depends on KGDB && 8xx || 8260
- help
- If you enable this, all serial console messages will be sent
- over the gdb stub.
- If unsure, say N.
+source "kernel/Kconfig.kgdb"

config XMON
    bool "Include xmon kernel debugger"
@@ -1231,6 +1197,14 @@ config DEBUG_INFO
    debug the kernel.
    If you don't debug the kernel, you can say N.

+config FRAME_POINTER
+ bool "Compile the kernel with frame pointers"
+ help
+ If you say Y here the resulting kernel image will be slightly larger
+ and slower, but it will give very useful debugging information.
+ If you don't debug the kernel, you can say N, but we may not be able
+ to solve problems without frame pointers.
+
config BOOTX_TEXT
    bool "Support for early boot text console (BootX or OpenFirmware only)"
    depends PPC_OF
--- linux-2.6.3-rc4/arch/ppc/kernel/entry.S 2004-02-17 09:53:59.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/arch/ppc/kernel/entry.S 2004-02-17 11:34:22.828359208 -0700
@@ -296,7 +296,11 @@ syscall_exit_work:
    ori r10,r10,MSR_EE
    SYNC

```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```
    MTMSRD(r10) /* re-enable interrupts */
- bl schedule
+#ifdef CONFIG_KGDB_THREAD
+ bl user_schedule
+#else
+ bl do_schedule
+#endif
    b 2b

#ifdef SHOW_SYSCALLS
@@ -547,7 +551,11 @@ resume_kernel:
    ori r10,r10,MSR_EE
    SYNC
    MTMSRD(r10) /* hard-enable interrupts */
- bl schedule
+#ifdef CONFIG_KGDB_THREAD
+ bl user_schedule
+#else
+ bl do_schedule
+#endif
    LOAD_MSR_KERNEL(r10,MSR_KERNEL)
    SYNC
    MTMSRD(r10) /* disable interrupts */
@@ -751,7 +759,11 @@ do_resched: /* r10 contains MSR_KERNEL
    ori r10,r10,MSR_EE
    SYNC
    MTMSRD(r10) /* hard-enable interrupts */
- bl schedule
+#ifdef CONFIG_KGDB_THREAD
+ bl user_schedule
+#else
+ bl do_schedule
+#endif
recheck:
    LOAD_MSR_KERNEL(r10,MSR_KERNEL)
    SYNC
@@ -815,6 +827,57 @@ END_FTR_SECTION_IFSET(CPU_FTR_601)
    /* shouldn't return */
    b 4b

+#ifdef CONFIG_KGDB_THREAD
+/*
+ * These are hooks used by KGDB because switch_to does not save registers
+ * in pt_regs. The registers are saved on the stack on behalf of the caller
+ * of these funtions.
+ */
+
+
+_GLOBAL(kern_schedule)
+ stwu r1,-INT_FRAME_SIZE(r1) /* Allocate exception frame */
+ SAVE_8GPRS( 0, r1)
+ SAVE_8GPRS( 8, r1)
```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```

+ SAVE_8GPRS(16, r1)
+ SAVE_8GPRS(24, r1)
+ addi r3,r1,INT_FRAME_SIZE
+ stw r3,GPR1(r1)
+ mfcrr3
+ stw r3,_CCR(r1)
+ mflr r3
+ stw r3,_NIP(r1)
+ mfctr r3
+ stw r3,_CTR(r1)
+ mfxer r3
+ stw r3,_XER(r1)
+ mfmsr r3
+ stw r3,_MSR(r1)
+ lwz r3,INT_FRAME_SIZE+4(r1)
+ stw r3,_LINK(r1)
+
+ addi r3,r1,STACK_FRAME_OVERHEAD
+ bl kern_do_schedule
+
+ lwz r3,_CCR(r1)
+ mtcrr3
+ lwz r3,_XER(r1)
+ mtixer3
+ lwz r3,_NIP(r1)
+ mtlr r3
+ lwz r3,_CTR(r1)
+ mtctr r3
+ lwz r3,_MSR(r1)
+ mtmsr r3
+ lwz r0,GPR0(r1)
+ REST_2GPRS ( 2, r1)
+ REST_4GPRS ( 4, r1)
+ REST_8GPRS ( 8, r1)
+ REST_8GPRS (16, r1)
+ REST_8GPRS (24, r1)
+ addi r1,r1,INT_FRAME_SIZE
+ blr
+#endif
+
    .comm ee_restarts,4

/*
--- linux-2.6.3-rc4/arch/ppc/kernel/irq.c 2004-02-17 09:53:48.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/arch/ppc/kernel/irq.c 2004-02-17 11:34:18.365372110 -0700
@@ -46,6 +46,7 @@
#include <linux/random.h>
#include <linux/seq_file.h>
#include <linux/cpumask.h>
+#include <linux/kgdb.h>

```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```

#include <asm/uaccess.h>
#include <asm/bitops.h>
@@ -513,6 +514,14 @@ out:
void do_IRQ(struct pt_regs *regs)
{
    int irq, first = 1;
+ #if 0
+ unsigned long *lrptr;
+ if (!(regs->msr & MSR_PR)) {
+ /* Came in from the kernel. Save call link. */
+ lrptr = (unsigned long *) (regs->gpr[1] + 4);
+ *lrptr = regs->nip;
+ }
+ #endif
    irq_enter();

    /*
@@ -530,7 +539,9 @@ void do_IRQ(struct pt_regs *regs)
    if (irq != -2 && first)
        /* That's not SMP safe ... but who cares ? */
        ppc_spurious_interrupts++;
- irq_exit();
+ irq_exit();
+
+ kgdb_process_breakpoint();
}

unsigned long probe_irq_on (void)
--- linux-2.6.3-rc4/arch/ppc/kernel/ppc-stub.c 2004-02-17 09:54:01.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/arch/ppc/kernel/ppc-stub.c 2004-02-17 11:34:23.163283196 -0700
@@ -1,857 +1,292 @@
/*
- * ppc-stub.c: KGDB support for the Linux kernel.
+ * arch/ppc/kernel/ppc-stub.c
+ *
- * adapted from arch/sparc/kernel/sparc-stub.c for the PowerPC
- * some stuff borrowed from Paul Mackerras' xmon
- * Copyright (C) 1998 Michael AK Tesch (tesch@cs.wisc.edu)
+ * PowerPC-specific bits to work with the common KGDB stub.
+ *
- * Modifications to run under Linux
- * Copyright (C) 1995 David S. Miller (davem@caip.rutgers.edu)
- *
- * This file originally came from the gdb sources, and the
- * copyright notices have been retained below.
+ * 1998 (c) Michael AK Tesch (tesch@cs.wisc.edu)
+ * 2003 (c) TimeSys Corporation
+ * 2004 (c) MontaVista Software, Inc.
+ * This file is licensed under the terms of the GNU General Public License
+ * version 2. This program is licensed "as is" without any warranty of any
+ * kind, whether express or implied.

```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```
*/
_/******
-
- THIS SOFTWARE IS NOT COPYRIGHTED
-
- HP offers the following for use in the public domain. HP makes no
- warranty with regard to the software or its performance and the
- user accepts the software "AS IS" with all faults.
-
- HP DISCLAIMS ANY WARRANTIES, EXPRESS OR IMPLIED, WITH REGARD
- TO THIS SOFTWARE INCLUDING BUT NOT LIMITED TO THE WARRANTIES
- OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
-
_*****/
-
_/******
- * Header: remcom.c,v 1.34 91/03/09 12:29:49 glenne Exp $
- *
- * Module name: remcom.c $
- * Revision: 1.34 $
- * Date: 91/03/09 12:29:49 $
- * Contributor: Lake Stevens Instrument Division$
- *
- * Description: low level support for gdb debugger. $
- *
- * Considerations: only works on target hardware $
- *
- * Written by: Glenn Engel $
- * ModuleState: Experimental $
- *
- * NOTES: See Below $
- *
- * Modified for SPARC by Stu Grossman, Cygnus Support.
- *
- * This code has been extensively tested on the Fujitsu SPARClite demo board.
- *
- * To enable debugger support, two things need to happen. One, a
- * call to set_debug_traps() is necessary in order to allow any breakpoints
- * or error conditions to be properly intercepted and reported to gdb.
- * Two, a breakpoint needs to be generated to begin communication. This
- * is most easily accomplished by a call to breakpoint(). Breakpoint()
- * simulates a breakpoint by executing a trap #1.
- *
_*****
- *
- * The following gdb commands are supported:
- *
- * command function Return value
- *
- * g return the value of the CPU registers hex data or ENN
```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```
- * G set the value of the CPU registers OK or ENN
- * qOffsets Get section offsets. Reply is Text=xxx;Data=yyy;Bss=zzz
- *
- * mAA..AA,LLLL Read LLLL bytes at address AA..AA hex data or ENN
- * MAA..AA,LLLL: Write LLLL bytes at address AA.AA OK or ENN
- *
- * c Resume at current address SNN ( signal NN)
- * cAA..AA Continue at address AA..AA SNN
- *
- * s Step one instruction SNN
- * sAA..AA Step one instruction from AA..AA SNN
- *
- * k kill
- *
- * ? What was the last signal ? SNN (signal NN)
- *
- * bBB..BB Set baud rate to BB..BB OK or BNN, then sets
- * baud rate
- *
- * All commands and responses are sent with a packet which includes a
- * checksum. A packet consists of
- *
- * $<packet info>#<checksum>.
- *
- * where
- * <packet info> :: <characters representing the command or response>
- * <checksum> :: <two hex digits computed as modulo 256 sum of <packetinfo>>
- *
- * When a packet is received, it is first acknowledged with either '+' or '-'.
- * '+' indicates a successful transfer. '-' indicates a failed transfer.
- *
- * Example:
- *
- * Host: Reply:
- * $m0,10#2a+$00010203040506070809101112131415#42
- *
- *****/
-
-#include <linux/config.h>
#include <linux/kernel.h>
-#include <linux/string.h>
-#include <linux/mm.h>
-#include <linux/smp.h>
-#include <linux/smp_lock.h>
-
-#include <asm/cacheflush.h>
#include <asm/system.h>
#include <asm/signal.h>
#include <asm/kgdb.h>
#include <asm/pgtable.h>
#include <asm/ptrace.h>
```

Linux–Kernel: [PATCH][5/6] A different KGDB stub

```

+#include <linux/config.h>
+#include <linux/kgdb.h>

-void breakinst(void);
+#include <asm/current.h>
+#include <asm/ptrace.h>
+#include <asm/processor.h>
+#include <asm/machdep.h>

+/* Convert the hardware trap type code to a unix signal number. */
+/*
- * BUFMAX defines the maximum number of characters in inbound/outbound buffers
- * at least NUMREGBYTES*2 are needed for register packets
+ * This table contains the mapping between PowerPC hardware trap types, and
+ * signals, which are primarily what GDB understands.
+ */
-#define BUFMAX 2048
-static char remcomInBuffer[BUFMAX];
-static char remcomOutBuffer[BUFMAX];
-
-static int initialized;
-static int kgdb_active;
-static int kgdb_started;
-static u_int fault_jmp_buf[100];
-static int kdebug;
-
-
-static const char hexchars[]="0123456789abcdef";
-
-/* Place where we save old trap entries for restoration – sparc*/
-/* struct tt_entry kgdb_savetable[256]; */
-/* typedef void (*trapfunc_t)(void); */
-
-static void kgdb_fault_handler(struct pt_regs *regs);
-static int handle_exception (struct pt_regs *regs);
-
-#if 0
-/* Install an exception handler for kgdb */
-static void exceptionHandler(int tnum, unsigned int *tfunc)
+static struct hard_trap_info
+{
- /* We are dorking with a live trap table, all irq's off */
-}
+ unsigned int tt; /* Trap type code for powerpc */
+ unsigned char signo; /* Signal that we map this trap into */
+} hard_trap_info[] = {
+#if defined(CONFIG_4xx)
+ { 0x0100, 0x02 /* SIGINT */ }, /* critical input interrupt */
+ { 0x0200, 0x0b /* SIGSEGV */ }, /* machine check */
+ { 0x0300, 0x0b /* SIGSEGV */ }, /* data storage */
+ { 0x0400, 0x0a /* SIGBUS */ }, /* instruction storage */

```

Linux–Kernel: [PATCH][5/6] A different KGDB stub

```

+ { 0x0500, 0x02 /* SIGINT */ }, /* interrupt */
+ { 0x0600, 0x0a /* SIGBUS */ }, /* alignment */
+ { 0x0700, 0x04 /* SIGILL */ }, /* program */
+ { 0x0800, 0x04 /* SIGILL */ }, /* reserved */
+ { 0x0900, 0x04 /* SIGILL */ }, /* reserved */
+ { 0x0a00, 0x04 /* SIGILL */ }, /* reserved */
+ { 0x0b00, 0x04 /* SIGILL */ }, /* reserved */
+ { 0x0c00, 0x14 /* SIGCHLD */ }, /* syscall */
+ { 0x0d00, 0x04 /* SIGILL */ }, /* reserved */
+ { 0x0e00, 0x04 /* SIGILL */ }, /* reserved */
+ { 0x0f00, 0x04 /* SIGILL */ }, /* reserved */
+ { 0x2000, 0x05 /* SIGTRAP */}, /* debug */
+ #else
+ { 0x0200, 0x0b /* SIGSEGV */ }, /* machine check */
+ { 0x0300, 0x0b /* SIGSEGV */ }, /* address error (store) */
+ { 0x0400, 0x0a /* SIGBUS */ }, /* instruction bus error */
+ { 0x0500, 0x02 /* SIGINT */ }, /* interrupt */
+ { 0x0600, 0x0a /* SIGBUS */ }, /* alingment */
+ { 0x0700, 0x05 /* SIGTRAP */ }, /* breakpoint trap */
+ { 0x0800, 0x08 /* SIGFPE */}, /* fpu unavail */
+ { 0x0900, 0x0e /* SIGALRM */ }, /* decremter */
+ { 0x0a00, 0x04 /* SIGILL */ }, /* reserved */
+ { 0x0b00, 0x04 /* SIGILL */ }, /* reserved */
+ { 0x0c00, 0x14 /* SIGCHLD */ }, /* syscall */
+ { 0x0d00, 0x05 /* SIGTRAP */ }, /* single–step/watch */
+ { 0x0e00, 0x08 /* SIGFPE */ }, /* fp assist */
+ #endif
+ { 0x0000, 0x000 } /* Must be last */
+ };

```

```

–int
–kgdb_setjmp(long *buf)
–{
– asm ("mflr 0; stw 0,0(%0);"
– "stw 1,4(%0); stw 2,8(%0);"
– "mfcrl 0; stw 0,12(%0);"
– "stmw 13,16(%0)"
– : : "r" (buf));
– /* XXX should save fp regs as well */
– return 0;
–}
–void
–kgdb_longjmp(long *buf, int val)
–{
– if (val == 0)
– val = 1;
– asm ("lmw 13,16(%0);"
– "lwz 0,12(%0); mtrcrf 0x38,0;"
– "lwz 0,0(%0); lwz 1,4(%0); lwz 2,8(%0);"
– "mtrr 0; mr 3,%1"
– : : "r" (buf), "r" (val));

```

```

-}
-/* Convert ch from a hex digit to an int */
-static int
-hex(unsigned char ch)
-{
- if (ch >= 'a' && ch <= 'f')
- return ch-'a'+10;
- if (ch >= '0' && ch <= '9')
- return ch-'0';
- if (ch >= 'A' && ch <= 'F')
- return ch-'A'+10;
- return -1;
-}
-
-/* Convert the memory pointed to by mem into hex, placing result in buf.
- * Return a pointer to the last char put in buf (null), in case of mem fault,
- * return 0.
- */
-static unsigned char *
-mem2hex(const char *mem, char *buf, int count)
+static int computeSignal(unsigned int tt)
{
- unsigned char ch;
- unsigned short tmp_s;
- unsigned long tmp_l;
-
- if (kgdb_setjmp((long*)fault_jmp_buf) == 0) {
- debugger_fault_handler = kgdb_fault_handler;
-
- /* Accessing 16 bit and 32 bit objects in a single
- ** load instruction is required to avoid bad side
- ** effects for some IO registers.
- */
-
- if ((count == 2) && (((long)mem & 1) == 0)) {
- tmp_s = *(unsigned short *)mem;
- mem += 2;
- *buf++ = hexchars[(tmp_s >> 12) & 0xf];
- *buf++ = hexchars[(tmp_s >> 8) & 0xf];
- *buf++ = hexchars[(tmp_s >> 4) & 0xf];
- *buf++ = hexchars[tmp_s & 0xf];
-
- } else if ((count == 4) && (((long)mem & 3) == 0)) {
- tmp_l = *(unsigned int *)mem;
- mem += 4;
- *buf++ = hexchars[(tmp_l >> 28) & 0xf];
- *buf++ = hexchars[(tmp_l >> 24) & 0xf];
- *buf++ = hexchars[(tmp_l >> 20) & 0xf];
- *buf++ = hexchars[(tmp_l >> 16) & 0xf];
- *buf++ = hexchars[(tmp_l >> 12) & 0xf];
- *buf++ = hexchars[(tmp_l >> 8) & 0xf];

```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```

- *buf++ = hexchars[(tmp_1 >> 4) & 0xf];
- *buf++ = hexchars[tmp_1 & 0xf];
-
- } else {
- while (count-- > 0) {
- ch = *mem++;
- *buf++ = hexchars[ch >> 4];
- *buf++ = hexchars[ch & 0xf];
- }
- }
-
- } else {
- /* error condition */
- }
- debugger_fault_handler = 0;
- *buf = 0;
- return buf;
-}
-
-/* convert the hex array pointed to by buf into binary to be placed in mem
- * return a pointer to the character AFTER the last byte written.
-*/
-static char *
-hex2mem(char *buf, char *mem, int count)
-{
- unsigned char ch;
- int i;
- char *orig_mem;
- unsigned short tmp_s;
- unsigned long tmp_l;
-
- orig_mem = mem;
-
- if (kgdb_setjmp((long*)fault_jmp_buf) == 0) {
- debugger_fault_handler = kgdb_fault_handler;
-
- /* Accessing 16 bit and 32 bit objects in a single
- ** store instruction is required to avoid bad side
- ** effects for some IO registers.
- */
-
- if ((count == 2) && (((long)mem & 1) == 0)) {
- tmp_s = hex(*buf++) << 12;
- tmp_s |= hex(*buf++) << 8;
- tmp_s |= hex(*buf++) << 4;
- tmp_s |= hex(*buf++);
-
- *(unsigned short *)mem = tmp_s;
- mem += 2;
-
- } else if ((count == 4) && (((long)mem & 3) == 0)) {

```

```

- tmp_1 = hex(*buf++) << 28;
- tmp_1 |= hex(*buf++) << 24;
- tmp_1 |= hex(*buf++) << 20;
- tmp_1 |= hex(*buf++) << 16;
- tmp_1 |= hex(*buf++) << 12;
- tmp_1 |= hex(*buf++) << 8;
- tmp_1 |= hex(*buf++) << 4;
- tmp_1 |= hex(*buf++);
-
- *(unsigned long *)mem = tmp_1;
- mem += 4;
-
- } else {
- for (i=0; i<count; i++) {
- ch = hex(*buf++) << 4;
- ch |= hex(*buf++);
- *mem++ = ch;
- }
- }
-
+ struct hard_trap_info *ht;

- /*
- ** Flush the data cache, invalidate the instruction cache.
- */
- flush_icache_range((int)orig_mem, (int)orig_mem + count - 1);
+ for (ht = hard_trap_info; ht->tt && ht->signo; ht++)
+ if (ht->tt == tt)
+ return ht->signo;

- } else {
- /* error condition */
- }
- debugger_fault_handler = 0;
- return mem;
+ return SIGHUP; /* default for things we don't know about */
}

/*
- * While we find nice hex chars, build an int.
- * Return number of chars processed.
+ * Routines
+ */
-static int
-hexToInt(char **ptr, int *intValue)
+static void
+kgdb_debugger(struct pt_regs *regs)
{
- int numChars = 0;
- int hexValue;
-

```

```

- *intValue = 0;
-
- if (kgdb_setjmp((long*)fault_jump_buf) == 0) {
- debugger_fault_handler = kgdb_fault_handler;
- while (**ptr) {
- hexValue = hex(**ptr);
- if (hexValue < 0)
- break;
-
- *intValue = (*intValue << 4) | hexValue;
- numChars ++;
-
- (*ptr)++;
- }
- } else {
- /* error condition */
- }
- debugger_fault_handler = 0;
-
- return (numChars);
+ (*linux_debug_hook) (0, computeSignal(regs->trap), 0, regs);
+ return;
}

-/* scan for the sequence $<data>#<checksum> */
-static void
-getpacket(char *buffer)
+static int
+kgdb_breakpoint(struct pt_regs *regs)
{
- unsigned char checksum;
- unsigned char xmitsum;
- int i;
- int count;
- unsigned char ch;
-
- do {
- /* wait around for the start character, ignore all other
- * characters */
- while ((ch = (getDebugChar() & 0x7f)) != '$') ;
-
- checksum = 0;
- xmitsum = -1;
-
- count = 0;
-
- /* now, read until a # or end of buffer is found */
- while (count < BUFMAX) {
- ch = getDebugChar() & 0x7f;
- if (ch == '#')
- break;

```

```

- checksum = checksum + ch;
- buffer[count] = ch;
- count = count + 1;
- }
+ (*linux_debug_hook) (0, SIGTRAP, 0, regs);

- if (count >= BUFMAX)
- continue;
-
- buffer[count] = 0;
+ if (atomic_read(&kgdb_setting_breakpoint))
+ regs->nip += 4;

- if (ch == '#') {
- xmitcsum = hex(getDebugChar() & 0x7f) << 4;
- xmitcsum |= hex(getDebugChar() & 0x7f);
- if (checksum != xmitcsum)
- putDebugChar('-'); /* failed checksum */
- else {
- putDebugChar('+'); /* successful transfer */
- /* if a sequence char is present, reply the ID */
- if (buffer[2] == ':') {
- putDebugChar(buffer[0]);
- putDebugChar(buffer[1]);
- /* remove sequence chars from buffer */
- count = strlen(buffer);
- for (i=3; i <= count; i++)
- buffer[i-3] = buffer[i];
- }
- }
- } while (checksum != xmitcsum);
+ return 1;
}

-/* send the packet in buffer. */
-static void putpacket(unsigned char *buffer)
+static int
+kgdb_singlestep(struct pt_regs *regs)
{
- unsigned char checksum;
- int count;
- unsigned char ch, rcv;
-
- /* $<packet info>#<checksum>. */
- do {
- putDebugChar('$');
- checksum = 0;
- count = 0;
-
- while ((ch = buffer[count])) {

```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```

- putDebugChar(ch);
- checksum += ch;
- count += 1;
- }
-
- putDebugChar('#');
- putDebugChar(hexchars[checksum >> 4]);
- putDebugChar(hexchars[checksum & 0xf]);
- recv = getDebugChar();
- } while ((recv & 0x7f) != '+');
+ (*linux_debug_hook) (0, SIGTRAP, 0, regs);
+ return 1;
}

-static void kgdb_flush_cache_all(void)
+int
+kgdb_iabr_match(struct pt_regs *regs)
{
- flush_instruction_cache();
+ (*linux_debug_hook) (0, computeSignal(regs->trap), 0, regs);
+ return 1;
}

-/* Set up exception handlers for tracing and breakpoints
- * [could be called kgdb_init()]
- */
-void set_debug_traps(void)
+int
+kgdb_dabr_match(struct pt_regs *regs)
{
-#if 0
- unsigned char c;
-
- save_and_cli(flags);
-
- /* In case GDB is started before us, ack any packets (presumably
- * "$?#xx") sitting there.
- *
- * I've found this code causes more problems than it solves,
- * so that's why it's commented out. GDB seems to work fine
- * now starting either before or after the kernel -bwb
- */
-
- while((c = getDebugChar()) != '$');
- while((c = getDebugChar()) != '#');
- c = getDebugChar(); /* eat first csum byte */
- c = getDebugChar(); /* eat second csum byte */
- putDebugChar('+'); /* ack it */
-#endif
- debugger = kgdb;
- debugger_bpt = kgdb_bpt;

```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```

- debugger_sstep = kgdb_sstep;
- debugger_iabr_match = kgdb_iabr_match;
- debugger_dabr_match = kgdb_dabr_match;
-
- initialized = 1;
+ (*linux_debug_hook) (0, computeSignal(regs->trap), 0, regs);
+ return 1;
}

-static void kgdb_fault_handler(struct pt_regs *regs)
+void
+regs_to_gdb_regs(unsigned long *gdb_regs, struct pt_regs *regs)
{
- kgdb_longjmp((long*)fault_jmp_buf, 1);
-}
+ int reg;
+ unsigned long *ptr = gdb_regs;

-int kgdb_bpt(struct pt_regs *regs)
- {
- return handle_exception(regs);
- }
+ memset(gdb_regs, 0, MAXREG * 4);

-int kgdb_sstep(struct pt_regs *regs)
- {
- return handle_exception(regs);
- }
+ for (reg = 0; reg < 32; reg++)
+ *(ptr++) = regs->gpr[reg];

-void kgdb(struct pt_regs *regs)
- {
- handle_exception(regs);
- }
+ for (reg = 0; reg < 64; reg++)
+ *(ptr++) = 0;

-int kgdb_iabr_match(struct pt_regs *regs)
- {
- printk(KERN_ERR "kgdb doesn't support iabr, what!?\n");
- return handle_exception(regs);
- }
+ *(ptr++) = regs->nip;
+ *(ptr++) = regs->msr;
+ *(ptr++) = regs->ccr;
+ *(ptr++) = regs->link;
+ *(ptr++) = regs->ctr;
+ *(ptr++) = regs->xer;

-int kgdb_dabr_match(struct pt_regs *regs)

```

Linux–Kernel: [PATCH][5/6] A different KGDB stub

```

- {
- printk(KERN_ERR "kgdb doesn't support dabr, what?!?\n");
- return handle_exception(regs);
- }
+ return;
+ } /* regs_to_gdb_regs */

-/* Convert the hardware trap type code to a unix signal number. */
-/*
- * This table contains the mapping between PowerPC hardware trap types, and
- * signals, which are primarily what GDB understands.
- */
-static struct hard_trap_info
+void
+sleeping_thread_to_gdb_regs(unsigned long *gdb_regs, struct task_struct *p)
{
- unsigned int tt; /* Trap type code for powerpc */
- unsigned char signo; /* Signal that we map this trap into */
- } hard_trap_info[] = {
-#if defined(CONFIG_40x)
- { 0x100, SIGINT }, /* critical input interrupt */
- { 0x200, SIGSEGV }, /* machine check */
- { 0x300, SIGSEGV }, /* data storage */
- { 0x400, SIGBUS }, /* instruction storage */
- { 0x500, SIGINT }, /* interrupt */
- { 0x600, SIGBUS }, /* alignment */
- { 0x700, SIGILL }, /* program */
- { 0x800, SIGILL }, /* reserved */
- { 0x900, SIGILL }, /* reserved */
- { 0xa00, SIGILL }, /* reserved */
- { 0xb00, SIGILL }, /* reserved */
- { 0xc00, SIGCHLD }, /* syscall */
- { 0xd00, SIGILL }, /* reserved */
- { 0xe00, SIGILL }, /* reserved */
- { 0xf00, SIGILL }, /* reserved */
- /*
- ** 0x1000 PIT
- ** 0x1010 FIT
- ** 0x1020 watchdog
- ** 0x1100 data TLB miss
- ** 0x1200 instruction TLB miss
- */
- { 0x2000, SIGTRAP }, /* debug */
-#else
- { 0x200, SIGSEGV }, /* machine check */
- { 0x300, SIGSEGV }, /* address error (store) */
- { 0x400, SIGBUS }, /* instruction bus error */
- { 0x500, SIGINT }, /* interrupt */
- { 0x600, SIGBUS }, /* alingment */
- { 0x700, SIGTRAP }, /* breakpoint trap */
- { 0x800, SIGFPE }, /* fpu unavail */

```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```

- { 0x900, SIGALRM }, /* decrementer */
- { 0xa00, SIGILL }, /* reserved */
- { 0xb00, SIGILL }, /* reserved */
- { 0xc00, SIGCHLD }, /* syscall */
- { 0xd00, SIGTRAP }, /* single-step/watch */
- { 0xe00, SIGFPE }, /* fp assist */
-#endif
- { 0, 0 } /* Must be last */
+ struct pt_regs *regs = (struct pt_regs *) (p->thread.ksp +
+ STACK_FRAME_OVERHEAD);
+ int reg;
+ unsigned long *ptr = gdb_regs;
+
+ memset(gdb_regs, 0, MAXREG * 4);
+
+ /* Regs GPR0-2 */
+ for (reg = 0; reg < 3; reg++)
+ *(ptr++) = regs->gpr[reg];
+
+ /* Regs GPR3-13 are not saved */
+ for (reg = 3; reg < 14; reg++)
+ *(ptr++) = 0;
+
+ /* Regs GPR14-31 */
+ for (reg = 14; reg < 32; reg++)
+ *(ptr++) = regs->gpr[reg];
+
+ for (reg = 0; reg < 64; reg++)
+ *(ptr++) = 0;
+
+ *(ptr++) = regs->nip;
+ *(ptr++) = regs->msr;
+ *(ptr++) = regs->ccr;
+ *(ptr++) = regs->link;
+ *(ptr++) = regs->ctr;
+ *(ptr++) = regs->xer;

-};
+ return;
+}

-static int computeSignal(unsigned int tt)
+void
+gdb_regs_to_regs(unsigned long *gdb_regs, struct pt_regs *regs)
+{
- struct hard_trap_info *ht;
+ int reg;
+ unsigned long *ptr = gdb_regs;

- for (ht = hard_trap_info; ht->tt && ht->signo; ht++)
- if (ht->tt == tt)

```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```

- return ht->signo;
+ for (reg = 0; reg < 32; reg++)
+ regs->gpr[reg] = *(ptr++);

- return SIGHUP; /* default for things we don't know about */
-}
+ for (reg = 0; reg < 64; reg++)
+ ptr++;
+
+ regs->nip = *(ptr++);
+ regs->msr = *(ptr++);
+ regs->ccr = *(ptr++);
+ regs->link = *(ptr++);
+ regs->ctr = *(ptr++);
+ regs->xer = *(ptr++);

-#define PC_REGNUM 64
-#define SP_REGNUM 1
+ return;
+} /* gdb_regs_to_regs */

/*
- * This function does all command processing for interfacing to gdb.
+ * This function does PoerPC specific processing for interfacing to gdb.
*/
-static int
-handle_exception (struct pt_regs *regs)
+int
+kgdb_arch_handle_exception(int vector, int signo, int err_code,
+ char *remcomInBuffer, char *remcomOutBuffer,
+ struct pt_regs *linux_regs)
{
- int sigval;
- int addr;
- int length;
- char *ptr;
- unsigned int msr;
-
- /* We don't handle user-mode breakpoints. */
- if (user_mode(regs))
- return 0;
+ unsigned long addr;

- if (debugger_fault_handler) {
- debugger_fault_handler(regs);
- panic("kgdb longjump failed!\n");
- }
- if (kgdb_active) {
- printk(KERN_ERR "interrupt while in kgdb, returning\n");
- return 0;
- }

```

```

-
- kgdb_active = 1;
- kgdb_started = 1;
-
-#ifdef KGDB_DEBUG
- printk("kgdb: entering handle_exception; trap [0x%x]\n",
- (unsigned int)regs->trap);
-#endif
-
- kgdb_interruptible(0);
- lock_kernel();
- msr = mfmsr();
- mtmsr(msr & ~MSR_EE); /* disable interrupts */
-
- if (regs->nip == (unsigned long)breakinst) {
- /* Skip over breakpoint trap insn */
- regs->nip += 4;
- }
-
- /* reply to host that an exception has occurred */
- signal = computeSignal(regs->trap);
- ptr = remcomOutBuffer;
-
-#if defined(CONFIG_40x)
- *ptr++ = 'S';
- *ptr++ = hexchars[signal >> 4];
- *ptr++ = hexchars[signal & 0xf];
-#else
- *ptr++ = 'T';
- *ptr++ = hexchars[signal >> 4];
- *ptr++ = hexchars[signal & 0xf];
- *ptr++ = hexchars[PC_REGNUM >> 4];
- *ptr++ = hexchars[PC_REGNUM & 0xf];
- *ptr++ = ':';
- ptr = mem2hex((char *)&regs->nip, ptr, 4);
- *ptr++ = ':';
- *ptr++ = hexchars[SP_REGNUM >> 4];
- *ptr++ = hexchars[SP_REGNUM & 0xf];
- *ptr++ = ':';
- ptr = mem2hex(((char *)regs) + SP_REGNUM*4, ptr, 4);
- *ptr++ = ':';
-#endif
-
- *ptr++ = 0;
-
- putpacket(remcomOutBuffer);
- if (kdebug)
- printk("remcomOutBuffer: %s\n", remcomOutBuffer);
-
- /* XXX We may want to add some features dealing with poking the
- * XXX page tables, ... (look at sparc-stub.c for more info)

```

Linux–Kernel: [PATCH][5/6] A different KGDB stub

```
– * XXX also required hacking to the gdb sources directly...
– */
–
– while (1) {
– remcomOutBuffer[0] = 0;
–
– getpacket(remcomInBuffer);
– switch (remcomInBuffer[0]) {
– case '?': /* report most recent signal */
– remcomOutBuffer[0] = 'S';
– remcomOutBuffer[1] = hexchars[signal >> 4];
– remcomOutBuffer[2] = hexchars[signal & 0xf];
– remcomOutBuffer[3] = 0;
– break;
–#if 0
– case 'q': /* this screws up gdb for some reason...*/
– {
– extern long _start, sdata, __bss_start;
–
– ptr = &remcomInBuffer[1];
– if (strncmp(ptr, "Offsets", 7) != 0)
– break;
–
– ptr = remcomOutBuffer;
– sprintf(ptr, "Text=%8.8x;Data=%8.8x;Bss=%8.8x",
– &_start, &sdata, &__bss_start);
– break;
– }
–#endif
– case 'd':
– /* toggle debug flag */
– kdebug ^= 1;
– break;
–
– case 'g': /* return the value of the CPU registers.
– * some of them are non–PowerPC names :(
– * they are stored in gdb like:
– * struct {
– * u32 gpr[32];
– * f64 fpr[32];
– * u32 pc, ps, cnd, lr; (ps=msr)
– * u32 cnt, xer, mq;
– * }
– */
– {
– int i;
– ptr = remcomOutBuffer;
– /* General Purpose Regs */
– ptr = mem2hex((char *)regs, ptr, 32 * 4);
– /* Floating Point Regs – FIXME */
– /*ptr = mem2hex((char *), ptr, 32 * 8);*/
```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```

- for(i=0; i<(32*8*2); i++) { /* 2chars/byte */
- ptr[i] = '0';
- }
- ptr += 32*8*2;
- /* pc, msr, cr, lr, ctr, xer, (mq is unused) */
- ptr = mem2hex((char *)&regs->nip, ptr, 4);
- ptr = mem2hex((char *)&regs->msr, ptr, 4);
- ptr = mem2hex((char *)&regs->ccr, ptr, 4);
- ptr = mem2hex((char *)&regs->link, ptr, 4);
- ptr = mem2hex((char *)&regs->ctr, ptr, 4);
- ptr = mem2hex((char *)&regs->xer, ptr, 4);
- }
+ switch (remcomInBuffer[0]) {
+ /*
+ * sAA..AA Step one instruction from AA..AA
+ * This will return an error to gdb ..
+ */
+ case 's':
+ case 'c':
+ if (kgdb_contthread && kgdb_contthread != current) {
+ strcpy(remcomOutBuffer, "E00");
+         break;
+ }
-
- case 'G': /* set the value of the CPU registers */
- {
- ptr = &remcomInBuffer[1];
-
- /*
- * If the stack pointer has moved, you should pray.
- * (cause only god can help you).
- */
-
- /* General Purpose Regs */
- hex2mem(ptr, (char *)regs, 32 * 4);
-
- /* Floating Point Regs - FIXME?? */
- /*ptr = hex2mem(ptr, ??, 32 * 8);*/
- ptr += 32*8*2;
-
- /* pc, msr, cr, lr, ctr, xer, (mq is unused) */
- ptr = hex2mem(ptr, (char *)&regs->nip, 4);
- ptr = hex2mem(ptr, (char *)&regs->msr, 4);
- ptr = hex2mem(ptr, (char *)&regs->ccr, 4);
- ptr = hex2mem(ptr, (char *)&regs->link, 4);
- ptr = hex2mem(ptr, (char *)&regs->ctr, 4);
- ptr = hex2mem(ptr, (char *)&regs->xer, 4);
-
- strcpy(remcomOutBuffer,"OK");
+ }
- break;
- case 'H':

```

Linux–Kernel: [PATCH][5/6] A different KGDB stub

```

- /* don't do anything, yet, just acknowledge */
- hexToInt(&ptr, &addr);
- strcpy(remcomOutBuffer,"OK");
- break;

- case 'm': /* mAA..AA,LLLL Read LLLL bytes at address AA..AA */
- /* Try to read %x,%x. */
+ kgdb_contthread = NULL;

- ptr = &remcomInBuffer[1];
-
- if (hexToInt(&ptr, &addr) && *ptr++ == ','
- && hexToInt(&ptr, &length)) {
- if (mem2hex((char *)addr, remcomOutBuffer,
- length))
- break;
- strcpy(remcomOutBuffer, "E03");
- } else
- strcpy(remcomOutBuffer, "E01");
- break;
-
- case 'M': /* MAA..AA,LLLL: Write LLLL bytes at address AA.AA return OK */
- /* Try to read '%x,%x:'. */
-
- ptr = &remcomInBuffer[1];
-
- if (hexToInt(&ptr, &addr) && *ptr++ == ','
- && hexToInt(&ptr, &length)
- && *ptr++ == ':') {
- if (hex2mem(ptr, (char *)addr, length))
- strcpy(remcomOutBuffer, "OK");
- else
- strcpy(remcomOutBuffer, "E03");
- flush_icache_range(addr, addr+length);
- } else
- strcpy(remcomOutBuffer, "E02");
- break;
-
-
- case 'k': /* kill the program, actually just continue */
- case 'c': /* cAA..AA Continue; address AA..AA optional */
- /* try to read optional parameter, pc unchanged if no parm */
-
- ptr = &remcomInBuffer[1];
- if (hexToInt(&ptr, &addr))
- regs->nip = addr;
+ /* handle the optional parameter */
+ ptr = &remcomInBuffer[1];
+ if (kgdb_hex2long(&ptr, &addr))
+ linux_regs->nip = addr;

```

Linux–Kernel: [PATCH][5/6] A different KGDB stub

```

/* Need to flush the instruction cache here, as we may have deposited a
 * breakpoint, and the icache probably has no way of knowing that a data ref to
 * some location may have changed something that is in the instruction cache.
 */
- kgdb_flush_cache_all();
-#if defined(CONFIG_40x)
- strcpy(remcomOutBuffer, "OK");
- putpacket(remcomOutBuffer);
-#endif
- mtmsr(msr);
+ flush_instruction_cache();

- kgdb_interruptible(1);
- unlock_kernel();
- kgdb_active = 0;
- if (kdebug) {
- printk("remcomInBuffer: %s\n", remcomInBuffer);
- printk("remcomOutBuffer: %s\n", remcomOutBuffer);
- }
- return 1;
-
- case 's':
- kgdb_flush_cache_all();
-#if defined(CONFIG_40x)
- regs->msr |= MSR_DE;
- regs->dbcr0 |= (DBCR0_IDM | DBCR0_IC);
- mtmsr(msr);
+ /* set the trace bit if we're stepping */
+ if (remcomInBuffer[0] == 's') {
+ #if defined (CONFIG_4xx)
+ linux_regs->msr |= MSR_DE;
+ current->thread.dbcr0 |= (DBCR_IDM | DBCR_IC);
+ #else
- regs->msr |= MSR_SE;
+ linux_regs->msr |= MSR_SE;
+ #endif
- unlock_kernel();
- kgdb_active = 0;
- if (kdebug) {
- printk("remcomInBuffer: %s\n", remcomInBuffer);
- printk("remcomOutBuffer: %s\n", remcomOutBuffer);
- }
- return 1;
-
- case 'r': /* Reset (if user process..exit ???)*/
- panic("kgdb reset.");
- break;
- } /* switch */
- if (remcomOutBuffer[0] && kdebug) {
- printk("remcomInBuffer: %s\n", remcomInBuffer);
- printk("remcomOutBuffer: %s\n", remcomOutBuffer);

```

```

    }
- /* reply to the request */
- putpacket(remcomOutBuffer);
- } /* while(1) */
-}
+ return 0;
+ }

-/* This function will generate a breakpoint exception. It is used at the
- beginning of a program to sync up with a debugger and can be used
- otherwise as a quick means to stop program execution and "break" into
- the debugger. */
+ return -1;
+}

#ifdef CONFIG_PPC_SIMPLE_SERIAL
void
-breakpoint(void)
+kgdb_put_debug_char(int chr)
{
- if (!initialized) {
- printk("breakpoint() called b4 kgdb init\n");
- return;
- }
-
- asm(" .globl breakinst \n\
- breakinst: .long 0x7d821008");
+ putDebugChar(chr);
}

#ifdef CONFIG_KGDB_CONSOLE
-/* Output string in GDB O-packet format if GDB has connected. If nothing
- output, returns 0 (caller must then handle output). */
int
-kgdb_output_string (const char* s, unsigned int count)
+kgdb_get_debug_char(void)
{
- char buffer[512];
-
- if (!kgdb_started)
- return 0;
+ return getDebugChar();
+}

- count = (count <= (sizeof(buffer) / 2 - 2))
- ? count : (sizeof(buffer) / 2 - 2);
+int
+kgdb_hook_io(void)
+{
+ if (ppc_md.kgdb_map_scc)
+ ppc_md.kgdb_map_scc();

```

```

+ return 0;
+}
+#endif

- buffer[0] = 'O';
- mem2hex (s, &buffer[1], count);
- putpacket(buffer);
+int
+kgdb_arch_init(void)
+{
+ debugger = kgdb_debugger;
+ debugger_bpt = kgdb_breakpoint;
+ debugger_sstep = kgdb_singlestep;
+ debugger_iabr_match = kgdb_iabr_match;
+ debugger_dabr_match = kgdb_dabr_match;

- return 1;
+ return 0;
}
-#endif
+
+/*
+ * Global data
+ */
+struct kgdb_arch arch_kgdb_ops = {
+ .gdb_bpt_instr = {0x7d, 0x82, 0x10, 0x08},
+};
--- linux-2.6.3-rc4/arch/ppc/kernel/setup.c 2004-02-17 09:54:15.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/arch/ppc/kernel/setup.c 2004-02-17 11:34:24.696935125 -0700
@@ -38,10 +38,6 @@
#include <asm/nvram.h>
#include <asm/xmon.h>

-#if defined CONFIG_KGDB
-#include <asm/kgdb.h>
-#endif
-
extern void platform_init(unsigned long r3, unsigned long r4,
                        unsigned long r5, unsigned long r6, unsigned long r7);
extern void bootx_init(unsigned long r4, unsigned long phys);
@@ -642,18 +638,6 @@ void __init setup_arch(char **cmdline_p)
#endif /* CONFIG_XMON */
    if ( ppc_md.progress ) ppc_md.progress("setup_arch: enter", 0x3eab);

-#if defined(CONFIG_KGDB)
- if (ppc_md.kgdb_map_scc)
- ppc_md.kgdb_map_scc();
- set_debug_traps();
- if (strstr(cmd_line, "gdb")) {
- if (ppc_md.progress)
- ppc_md.progress("setup_arch: kgdb breakpoint", 0x4000);

```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```

- printk("kgdb breakpoint activated\n");
- breakpoint();
- }
-#endif
-
-     /*
-      * Set cache line size based on type of cpu as a default.
-      * Systems with OF can look in the properties on the cpu node(s)
---- linux-2.6.3-rc4/arch/ppc/mm/fault.c 2004-02-17 09:51:50.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/arch/ppc/mm/fault.c 2004-02-17 11:33:54.831713256 -0700
@@ -28,6 +28,7 @@
#include <linux/interrupt.h>
#include <linux/highmem.h>
#include <linux/module.h>
+#include <linux/debugger.h>

#include <asm/page.h>
#include <asm/pgtable.h>
@@ -139,6 +140,13 @@ void do_page_fault(struct pt_regs *regs,
        bad_page_fault(regs, address, SIGSEGV);
        return;
    }
+
+ if (debugger_memerr_expected) {
+ /* This fault was caused by memory access through a debugger.
+ * Don't handle it like user accesses */
+ goto no_context;
+ }
+
        down_read(&mm->mmap_sem);
        vma = find_vma(mm, address);
        if (!vma)
@@ -293,6 +301,7 @@ bad_area:
        return;
    }

+no_context:
        bad_page_fault(regs, address, SIGSEGV);
        return;

@@ -346,6 +355,7 @@ bad_page_fault(struct pt_regs *regs, uns
        if (debugger_kernel_faults)
            debugger(regs);
#endif
+
+ die("kernel access of bad area", regs, sig);
    }

---- linux-2.6.3-rc4/arch/ppc/platforms/lopec_setup.c 2004-02-17 09:52:46.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/arch/ppc/platforms/lopec_setup.c 2004-02-17 11:34:03.669707396 -0700
@@ -32,6 +32,7 @@

```

```

#include <asm/mpc10x.h>
#include <asm/hw_irq.h>
#include <asm/prep_nvram.h>
+#include <asm/kgdb.h>

extern char saved_command_line[];
extern void lopec_find_bridges(void);
@@ -261,44 +262,6 @@ lopec_set_bat(void)
    : "=r" (batu), "=r" (batl));
}

-#ifdef CONFIG_SERIAL_TEXT_DEBUG
-#include <linux/serial.h>
-#include <linux/serialP.h>
-#include <linux/serial_reg.h>
-#include <asm/serial.h>
-
-static struct serial_state rs_table[RS_TABLE_SIZE] = {
- SERIAL_PORT_DFNS /* Defined in <asm/serial.h> */
-};
-
-volatile unsigned char *com_port;
-volatile unsigned char *com_port_lsr;
-
-static void
-serial_writechar(char c)
-{
- while ((*com_port_lsr & UART_LSR_THRE) == 0)
- ;
- *com_port = c;
-}
-
-void
-lopec_progress(char *s, unsigned short hex)
-{
- volatile char c;
-
- com_port = (volatile unsigned char *) rs_table[0].port;
- com_port_lsr = com_port + UART_LSR;
-
- while ((c = *s++) != 0)
- serial_writechar(c);
-
- /* Most messages don't have a newline in them */
- serial_writechar('\n');
- serial_writechar('\r');
-}
-#endif /* CONFIG_SERIAL_TEXT_DEBUG */
-
TODC_ALLOC();

```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```

static void __init
@@ -383,7 +346,10 @@ platform_init(unsigned long r3, unsigned
    ppc_ide_md.default_io_base = lopec_ide_default_io_base;
    ppc_ide_md.ide_init_hwif = lopec_ide_init_hwif_ports;
#endif
+#ifdef CONFIG_PPC_SIMPLE_SERIAL
+ ppc_md.kgdb_map_scc = gen550_kgdb_map_scc;
+#endif
#ifdef CONFIG_SERIAL_TEXT_DEBUG
- ppc_md.progress = lopec_progress;
+ ppc_md.progress = gen550_progress;
#endif
}
--- linux-2.6.3-rc4/arch/ppc/platforms/pmac_setup.c 2004-02-17 09:53:13.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/arch/ppc/platforms/pmac_setup.c 2004-02-17 11:34:07.360869658 -0700
@@ -297,9 +297,6 @@ pmac_setup_arch(void)
    ppc_override_l2cr_value, (ppc_override_l2cr_value & 0x80000000)
        ? "enabled" : "disabled");

-#ifdef CONFIG_KGDB
- zs_kgdb_hook(0);
-#endif

#ifdef CONFIG_ADB_CUDA
    find_via_cuda();
--- linux-2.6.3-rc4/arch/ppc/platforms/prep_setup.c 2004-02-17 09:53:56.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/arch/ppc/platforms/prep_setup.c 2004-02-17 11:34:22.411453827 -0700
@@ -58,6 +58,7 @@
#include <asm/i8259.h>
#include <asm/open_pic.h>
#include <asm/pci-bridge.h>
+#include <asm/kgdb.h>
#include <asm/todc.h>

TODC_ALLOC();
@@ -1036,6 +1037,10 @@ prep_init(unsigned long r3, unsigned lon

    ppc_md.setup_io_mappings = prep_map_io;

+#ifdef CONFIG_SERIAL_TEXT_DEBUG
+ ppc_md.progress = gen550_progress;
+#endif
+
+#if defined(CONFIG_BLK_DEV_IDE) || defined(CONFIG_BLK_DEV_IDE_MODULE)
    ppc_ide_md.default_irq = prep_ide_default_irq;
    ppc_ide_md.default_io_base = prep_ide_default_io_base;
--- linux-2.6.3-rc4/arch/ppc/platforms/sandpoint.c 2004-02-17 09:54:28.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/arch/ppc/platforms/sandpoint.c 2004-02-17 11:34:29.160922001 -0700
@@ -252,38 +252,6 @@ sandpoint_find_bridges(void)
    return;
}

```

```

-#if defined(CONFIG_SERIAL_8250) && \
- (defined(CONFIG_KGDB) || defined(CONFIG_SERIAL_TEXT_DEBUG))
-static void __init
-sandpoint_early_serial_map(void)
-{
- struct uart_port serial_req;
-
- /* Setup serial port access */
- memset(&serial_req, 0, sizeof(serial_req));
- serial_req.uartclk = UART_CLK;
- serial_req.irq = 4;
- serial_req.flags = STD_COM_FLAGS;
- serial_req.iotype = SERIAL_IO_MEM;
- serial_req.membase = (u_char *)SANDPOINT_SERIAL_0;
-
- gen550_init(0, &serial_req);
-
- if (early_serial_setup(&serial_req) != 0)
- printk(KERN_ERR "Early serial init of port 0 failed\n");
-
- /* Assume early_serial_setup() doesn't modify serial_req */
- serial_req.line = 1;
- serial_req.irq = 3; /* XXXX */
- serial_req.membase = (u_char *)SANDPOINT_SERIAL_1;
-
- gen550_init(1, &serial_req);
-
- if (early_serial_setup(&serial_req) != 0)
- printk(KERN_ERR "Early serial init of port 1 failed\n");
-}
-#endif
-
-static void __init
-sandpoint_setup_arch(void)
-{
@@ -701,16 +669,12 @@ platform_init(unsigned long r3, unsigned
- ppc_md.nvram_read_val = todc_mc146818_read_val;
- ppc_md.nvram_write_val = todc_mc146818_write_val;

-#if defined(CONFIG_SERIAL_8250) && \
- (defined(CONFIG_KGDB) || defined(CONFIG_SERIAL_TEXT_DEBUG))
- sandpoint_early_serial_map();
-#ifdef CONFIG_KGDB
+#ifdef CONFIG_PPC_SIMPLE_SERIAL
- ppc_md.kgdb_map_scc = gen550_kgdb_map_scc;
-#endif
-#ifdef CONFIG_SERIAL_TEXT_DEBUG
- ppc_md.progress = gen550_progress;
-#endif
-#endif

```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```

#if defined(CONFIG_BLK_DEV_IDE) || defined(CONFIG_BLK_DEV_IDE_MODULE)
    ppc_ide_md.default_irq = sandpoint_ide_default_irq;
--- linux-2.6.3-rc4/arch/ppc/syslib/Makefile 2004-02-17 09:50:55.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/arch/ppc/syslib/Makefile 2004-02-17 11:33:48.820077655 -0700
@@ -68,7 +68,7 @@ obj-$(CONFIG_SPRUCE) += cpc700_pic.o in
    todc_time.o
obj-$(CONFIG_8260) += m8260_setup.o ppc8260_pic.o
ifeq ($(CONFIG_PPC_GEN550),y)
-obj-$(CONFIG_KGDB) += gen550_kgdb.o gen550_dbg.o
+obj-$(CONFIG_PPC_SIMPLE_SERIAL) += gen550_kgdb.o gen550_dbg.o
obj-$(CONFIG_SERIAL_TEXT_DEBUG) += gen550_dbg.o
endif
obj-$(CONFIG_BOOTX_TEXT) += btext.o
--- linux-2.6.3-rc4/arch/ppc/syslib/gen550_dbg.c 2004-02-17 09:52:15.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/arch/ppc/syslib/gen550_dbg.c 2004-02-17 11:33:59.464661767 -0700
@@ -27,7 +27,19 @@
#include <asm/serial.h>
#include <asm/io.h>

-#define SERIAL_BAUD 9600
+#ifdef CONFIG_KGDB_9600BAUD
+#define SERIAL_BAUD 9600
+#elif defined(CONFIG_KGDB_19200BAUD)
+#define SERIAL_BAUD 19200
+#elif defined(CONFIG_KGDB_38400BAUD)
+#define SERIAL_BAUD 38400
+#elif defined(CONFIG_KGDB_57600BAUD)
+#define SERIAL_BAUD 57600
+#elif defined(CONFIG_KGDB_115200BAUD)
+#define SERIAL_BAUD 115200
+#else /* default to 9600 */
+#define SERIAL_BAUD 9600
+#endif

static struct serial_state rs_table[RS_TABLE_SIZE] = {
    SERIAL_PORT_DFNS /* defined in <asm/serial.h> */
@@ -109,12 +121,15 @@ unsigned long serial_init(int chan, void
    serial_outb(com_port + (UART_DLM << shift),
                (rs_table[chan].baud_base / SERIAL_BAUD) >> 8);
    /* 8 data, 1 stop, no parity */
- serial_outb(com_port + (UART_LCR << shift), 0x03);
+ serial_outb(com_port + (UART_LCR << shift), UART_LCR_WLEN8);
    /* RTS/DTR */
- serial_outb(com_port + (UART_MCR << shift), 0x03);
+ serial_outb(com_port + (UART_MCR << shift),
+ UART_MCR_RTS | UART_MCR_DTR);

    /* Clear & enable FIFOs */
- serial_outb(com_port + (UART_FCR << shift), 0x07);
+ serial_outb(com_port + (UART_FCR << shift),

```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```

+ UART_FCR_ENABLE_FIFO | UART_FCR_TRIGGER_1 |
+ UART_FCR_CLEAR_XMIT | UART_FCR_CLEAR_RCVR);
    }

    return (com_port);
--- linux-2.6.3-rc4/arch/ppc/syslib/gen550_kgdb.c 2004-02-17 09:52:04.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/arch/ppc/syslib/gen550_kgdb.c 2004-02-17 11:33:57.761048417 -0700
@@ -74,7 +74,7 @@ void putDebugString(char* str)

/*
 * Note: gen550_init() must be called already on the port we are going
- * to use.
+ * to use, or <asm/serial.h> must provide static definitions.
 */
void
gen550_kgdb_map_scc(void)
--- linux-2.6.3-rc4/include/asm-ppc/kgdb.h 2004-02-17 09:53:13.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/include/asm-ppc/kgdb.h 2004-02-17 11:34:07.456847875 -0700
@@ -2,6 +2,8 @@
 * kgdb.h: Defines and declarations for serial line source level
 * remote debugging of the Linux kernel using gdb.
 *
+ * PPC Mods (C) 2004 Tom Rini (trini@mvista.com)
+ * PPC Mods (C) 2003 John Whitney (john.whitney@timesys.com)
 * PPC Mods (C) 1998 Michael Tesch (tesch@cs.wisc.edu)
 *
 * Copyright (C) 1995 David S. Miller (davem@caip.rutgers.edu)
@@ -12,6 +14,16 @@

#ifdef __ASSEMBLY__

+#define BREAK_INSTR_SIZE 4
+#define MAXREG (PT_FPSCR+1)
+#define NUMREGBYTES (MAXREG * sizeof(int))
+#define BUFMAX ((NUMREGBYTES * 2) + 512)
+#define OUTBUFMAX ((NUMREGBYTES * 2) + 512)
+#define PC_REGNUM 64
+#define SP_REGNUM 1
+#define PTRACE_PC nip /* Program Counter, in ptrace regs. */
+#define BREAKPOINT() asm(".long 0x7d821008") /* twge r2, r2 */
+
+ /* Things specific to the gen550 backend. */
struct uart_port;

@@ -19,15 +31,6 @@ extern void gen550_progress(char *, unsi
extern void gen550_kgdb_map_scc(void);
extern void gen550_init(int, struct uart_port *);

-/* Things specific to the pmac backend. */
-extern void zs_kgdb_hook(int tty_num);
-

```

Linux-Kernel: [PATCH][5/6] A different KGDB stub

```

-/* To init the kgdb engine. (called by serial hook)*/
-extern void set_debug_traps(void);
-
-/* To enter the debugger explicitly. */
-extern void breakpoint(void);
-
-/* For taking exceptions
- * these are defined in traps.c
- */
--- linux-2.6.3-rc4/include/asm-ppc/processor.h 2004-02-17 09:49:40.000000000 -0700
+++ linux-2.6.3-rc4-kgdb/include/asm-ppc/processor.h 2004-02-17 11:33:46.627575266 -0700
@@ -119,6 +119,9 @@ struct thread_struct {
    unsigned long vrsave;
    int used_vr; /* set if process has used altivec */
-#endif /* CONFIG_ALTIVEC */
+#ifdef CONFIG_KGDB
+ void *debuggerinfo;
+#endif
};

#define INIT_SP (sizeof(init_stack) + (unsigned long) &init_stack)

--
Tom Rini
http://gate.crashing.org/~trini/
-
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@vger.kernel.org
More majordomo info at http://vger.kernel.org/majordomo-info.html
Please read the FAQ at http://www.tux.org/lkml/
```