

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-10/msg04857.html>

- *From:* Jason Wessel <jason.wessel@xxxxxxxxxxxxxx>
 - *Date:* Mon, 15 Oct 2007 13:33:31 -0500
-

Signed-off-by: Jason Wessel <jason.wessel@xxxxxxxxxxxxxx>
sh-lite.patch

From: Jason Wessel <jason.wessel@xxxxxxxxxxxxxx>
CC: lethal@xxxxxxxxxxxxxx
Subject: [PATCH] This adds basic support for KGDB on SuperH

This patch originally came from Tom Rini. It adds the unified kgdb support to the sh architechure covering sh3 and sh4. This patch also adds some architecture specific notes to the DocBook file and converting the 7751 to use this.

Signed-off-by: Milind Dumbare <milind@xxxxxxxxxxxxxx>
Signed-off-by: Tom Rini <trini@xxxxxxxxxxxxxx>

Documentation/DocBook/kgdb.tmpl | 16
arch/sh/Kconfig.debug | 79 ---
arch/sh/kernel/Makefile | 2
arch/sh/kernel/kgdb-jmp.S | 32 +
arch/sh/kernel/kgdb.c | 366 ++++++++
arch/sh/kernel/kgdb_jmp.S | 33 -
arch/sh/kernel/kgdb_stub.c | 1054 -----
arch/sh/kernel/time.c | 7
arch/sh/kernel/traps.c | 24
arch/sh/mm/extable.c | 7
arch/sh/mm/fault.c | 12
drivers/serial/sh-sci.c | 14
include/asm-sh/kgdb.h | 95 +---
lib/Kconfig.kgdb | 11
14 files changed, 479 insertions(+), 1273 deletions(-)
create mode 100644 arch/sh/kernel/kgdb-jmp.S
create mode 100644 arch/sh/kernel/kgdb.c
delete mode 100644 arch/sh/kernel/kgdb_jmp.S
delete mode 100644 arch/sh/kernel/kgdb_stub.c

--- a/Documentation/DocBook/kgdb.tmpl
+++ b/Documentation/DocBook/kgdb.tmpl

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

@@ -124,6 +124,10 @@

serial driver, pass in: <constant>kgdbwait</constant>.

</para>

<para>

+ To specify the values of the SH SCI(F) serial port at boot:

+ <constant>kgdbsci=0,115200</constant>.

+ </para>

+ <para>

To specify the values of the serial port at boot:

<constant>kgdb8250=io,3f8,115200,3</constant>.

On IA64 this could also be:

@@ -181,6 +185,18 @@

application program.

</para>

</chapter>

+ <chapter id="ArchitectureNotes">

+ <title>Architecture specific notes</title>

+ <para>

+ SuperH: The NMI switch found on some boards can be used to trigger an

+ initial breakpoint. Subsequent triggers do nothing. If console

+ is enabled on the SCI(F) serial port, and that is the port being used

+ for KGDB, then you must trigger a breakpoint via sysrq, NMI, or

+ some other method prior to connecting, or echo a control-c to the

+ serial port. Also, to use the SCI(F) port for KGDB, the

+ <symbol>CONFIG_SERIAL_SH_SCI</symbol> driver must be enabled.

+ </para>

+ </chapter>

<chapter id="CommonBackEndReq">

<title>The common backend (required)</title>

<para>

--- a/arch/sh/Kconfig.debug

+++ b/arch/sh/Kconfig.debug

@@ -86,83 +86,4 @@ config 4KSTACKS

on the VM subsystem for higher order allocations. This option

will also use IRQ stacks to compensate for the reduced stackspace.

-config SH_KGDB

- bool "Include KGDB kernel debugger"

- select FRAME_POINTER

- select DEBUG_INFO

- depends on CPU_SH3 || CPU_SH4

- 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.

-

-menu "KGDB configuration options"

- depends on SH_KGDB

-

-config MORE_COMPILE_OPTIONS

- bool "Add any additional compile options"

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

- help
- If you want to add additional CFLAGS to the kernel build, enable this
- option and then enter what you would like to add in the next question.
- Note however that -g is already appended with the selection of KGDB.
-
- config COMPILE_OPTIONS
- string "Additional compile arguments"
- depends on MORE_COMPILE_OPTIONS
-
- config KGDB_NMI
- bool "Enter KGDB on NMI"
- default n
-
- config SH_KGDB_CONSOLE
- bool "Console messages through GDB"
- depends on !SERIAL_SH_SCI_CONSOLE
- select SERIAL_CORE_CONSOLE
- default n
-
- config KGDB_SYSRQ
- bool "Allow SysRq 'G' to enter KGDB"
- default y
-
- comment "Serial port setup"
-
- config KGDB_DEFPORT
- int "Port number (ttySCn)"
- default "1"
-
- config KGDB_DEFBAUD
- int "Baud rate"
- default "115200"
-
- choice
- prompt "Parity"
- depends on SH_KGDB
- default KGDB_DEFPARITY_N
-
- config KGDB_DEFPARITY_N
- bool "None"
-
- config KGDB_DEFPARITY_E
- bool "Even"
-
- config KGDB_DEFPARITY_O
- bool "Odd"
-
- endchoice
-
- choice
- prompt "Data bits"

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
- depends on SH_KGDB
- default KGDB_DEFBITS_8
-
-config KGDB_DEFBITS_8
- bool "8"
-
-config KGDB_DEFBITS_7
- bool "7"
-
-endchoice
-
-endmenu
-
endmenu
---- a/arch/sh/kernel/Makefile
+++ b/arch/sh/kernel/Makefile
@@ -14,7 +14,7 @@ obj-$(CONFIG_VSYSCALL) += vsyscall/
obj-$(CONFIG_SMP) += smp.o
obj-$(CONFIG_CF_ENABLER) += cf-enabler.o
obj-$(CONFIG_SH_STANDARD_BIOS) += sh_bios.o
-obj-$(CONFIG_SH_KGDB) += kgdb_stub.o kgdb_jump.o
+obj-$(CONFIG_KGDB) += kgdb.o kgdb-jmp.o
obj-$(CONFIG_SH_CPU_FREQ) += cpufreq.o
obj-$(CONFIG_MODULES) += sh_ksyms.o module.o
obj-$(CONFIG_EARLY_PRINTK) += early_printk.o
---- /dev/null
+++ b/arch/sh/kernel/kgdb-jmp.S
@@ -0,0 +1,32 @@
+#include <linux/linkage.h>
+
+ENTRY(kgdb_fault_setjmp)
+ add #(9*4), r4
+ sts.l pr, @-r4
+ mov.l r15, @-r4
+ mov.l r14, @-r4
+ mov.l r13, @-r4
+ mov.l r12, @-r4
+ mov.l r11, @-r4
+ mov.l r10, @-r4
+ mov.l r9, @-r4
+ mov.l r8, @-r4
+ rts
+ mov #0, r0
+
+ENTRY(kgdb_fault_longjmp)
+ mov.l @r4+, r8
+ mov.l @r4+, r9
+ mov.l @r4+, r10
+ mov.l @r4+, r11
+ mov.l @r4+, r12
+ mov.l @r4+, r13
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
+ mov.l @r4+, r14
+ mov.l @r4+, r15
+ lds.l @r4+, pr
+ mov r5, r0
+ tst r0, r0
+ bf 1f
+ mov #1, r0
+1: rts
+ nop
--- /dev/null
+++ b/arch/sh/kernel/kgdb.c
@@ -0,0 +1,366 @@
+/*
+ * arch/sh/kernel/kgdb.c
+ *
+ * Contains SH-specific low-level support for KGDB.
+ *
+ * Contains extracts from code by Glenn Engel, Jim Kingdon,
+ * David Grothe <dave@xxxxxxx>, Tigran Aivazian <tigran@xxxxxxx>,
+ * Amit S. Kale <akale@xxxxxxxxxxxx>, William Gatliff <bgat@xxxxxxxxxxxxxxxxxxxx>,
+ * Ben Lee, Steve Chamberlain and Benoit Miller <fulg@xxxxxxx>,
+ * Henry Bell <henry.bell@xxxxxx> and Jeremy Siegel <jsiegel@xxxxxxxxxxxx>
+ *
+ * Maintainer: Tom Rini <trini@xxxxxxxxxxxxxxxxxxxx>
+ *
+ * 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.
+ */
+
+#include <linux/string.h>
+#include <linux/kernel.h>
+#include <linux/sched.h>
+#include <linux/smp.h>
+#include <linux/spinlock.h>
+#include <linux/delay.h>
+#include <linux/linkage.h>
+#include <linux/init.h>
+#include <linux/kgdb.h>
+#include <linux/signal.h>
+#include <linux/ptrace.h>
+
+#include <asm/system.h>
+#include <asm/current.h>
+#include <asm/pgtable.h>
+#include <asm/mmu_context.h>
+
+extern void per_cpu_trap_init(void);
+extern atomic_t cpu_doing_single_step;
+
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
+/* Function pointers for linkage */
+static struct kgdb_regs trap_registers;
+
+/* Globals. */
+char in_nmi; /* Set during NMI to prevent reentry */
+
+/* TRA differs sh3/4 */
+#if defined(CONFIG_CPU_SH3)
+#define TRA 0xfffffd0
+#elif defined(CONFIG_CPU_SH4)
+#define TRA 0xff000020
+#endif
+
+/* Macros for single step instruction identification */
+#define OPCODE_BT(op) (((op) & 0xff00) == 0x8900)
+#define OPCODE_BF(op) (((op) & 0xff00) == 0x8b00)
+#define OPCODE_BTF_DISP(op) (((op) & 0x80) ? (((op) | 0xfffff80) << 1) : \
+ (((op) & 0x7f) << 1))
+#define OPCODE_BFS(op) (((op) & 0xff00) == 0x8f00)
+#define OPCODE_BTS(op) (((op) & 0xff00) == 0x8d00)
+#define OPCODE_BRA(op) (((op) & 0xf000) == 0xa000)
+#define OPCODE_BRA_DISP(op) (((op) & 0x800) ? (((op) | 0xffff800) << 1) : \
+ (((op) & 0x7ff) << 1))
+#define OPCODE_BRAF(op) (((op) & 0xf0ff) == 0x0023)
+#define OPCODE_BRAF_REG(op) (((op) & 0x0f00) >> 8)
+#define OPCODE_BSR(op) (((op) & 0xf000) == 0xb000)
+#define OPCODE_BSR_DISP(op) (((op) & 0x800) ? (((op) | 0xffff800) << 1) : \
+ (((op) & 0x7ff) << 1))
+#define OPCODE_BSRF(op) (((op) & 0xf0ff) == 0x0003)
+#define OPCODE_BSRF_REG(op) (((op) >> 8) & 0xf)
+#define OPCODE_JMP(op) (((op) & 0xf0ff) == 0x402b)
+#define OPCODE_JMP_REG(op) (((op) >> 8) & 0xf)
+#define OPCODE_JSR(op) (((op) & 0xf0ff) == 0x400b)
+#define OPCODE_JSR_REG(op) (((op) >> 8) & 0xf)
+#define OPCODE_RTS(op) ((op) == 0xb)
+#define OPCODE_RTE(op) ((op) == 0x2b)
+
+#define SR_T_BIT_MASK 0x1
+#define STEP_OPCODE 0xc320
+#define BIOS_CALL_TRAP 0x3f
+
+/* Exception codes as per SH-4 core manual */
+#define ADDRESS_ERROR_LOAD_VEC 7
+#define ADDRESS_ERROR_STORE_VEC 8
+#define TRAP_VEC 11
+#define INVALID_INSN_VEC 12
+#define INVALID_SLOT_VEC 13
+#define NMI_VEC 14
+#define SERIAL_BREAK_VEC 58
+
+/* Misc static */
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
+static int stepped_address;
+static short stepped_opcode;
+
+/* Translate SH-3/4 exception numbers to unix-like signal values */
+static int compute_signal(const int excep_code)
+{
+ switch (excep_code) {
+ case INVALID_INSN_VEC:
+ case INVALID_SLOT_VEC:
+ return SIGILL;
+ case ADDRESS_ERROR_LOAD_VEC:
+ case ADDRESS_ERROR_STORE_VEC:
+ return SIGSEGV;
+ case SERIAL_BREAK_VEC:
+ case NMI_VEC:
+ return SIGINT;
+ default:
+ /* Act like it was a break/trap. */
+ return SIGTRAP;
+ }
+}
+
+/*
+ * Translate the registers of the system into the format that GDB wants. Since
+ * we use a local structure to store things, instead of getting them out
+ * of pt_regs, we can just do a memcpy.
+ */
+void regs_to_gdb_regs(unsigned long *gdb_regs, struct pt_regs *ign)
+{
+ memcpy(gdb_regs, &trap_registers, sizeof(trap_registers));
+}
+
+/*
+ * On SH we save: r1 (prev->thread.sp) r2 (prev->thread.pc) r4 (prev) r5 (next)
+ * r6 (next->thread.sp) r7 (next->thread.pc)
+ */
+void sleeping_thread_to_gdb_regs(unsigned long *gdb_regs, struct task_struct *p)
+{
+ int count;
+
+ for (count = 0; count < 16; count++)
+ *(gdb_regs++) = 0;
+ *(gdb_regs++) = p->thread.pc;
+ *(gdb_regs++) = 0;
+ *(gdb_regs++) = 0;
+ *(gdb_regs++) = 0;
+ *(gdb_regs++) = 0;
+ *(gdb_regs++) = 0;
+ *(gdb_regs++) = 0;
+}
+
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
+/*
+ * Translate the registers values that GDB has given us back into the
+ * format of the system. See the comment above about memcpy.
+ */
+void gdb_regs_to_regs(unsigned long *gdb_regs, struct pt_regs *ign)
+{
+ memcpy(&trap_registers, gdb_regs, sizeof(trap_registers));
+}
+
+/* Calculate the new address for after a step */
+static short *get_step_address(void)
+{
+ short op = *(short *)trap_registers.pc;
+ long addr;
+
+ /* BT */
+ if (OPCODE_BT(op)) {
+ if (trap_registers.sr & SR_T_BIT_MASK)
+ addr = trap_registers.pc + 4 + OPCODE_BT_DISP(op);
+ else
+ addr = trap_registers.pc + 2;
+ }
+
+ /* BTS */
+ else if (OPCODE_BTS(op)) {
+ if (trap_registers.sr & SR_T_BIT_MASK)
+ addr = trap_registers.pc + 4 + OPCODE_BT_DISP(op);
+ else
+ addr = trap_registers.pc + 4; /* Not in delay slot */
+ }
+
+ /* BF */
+ else if (OPCODE_BF(op)) {
+ if (!(trap_registers.sr & SR_T_BIT_MASK))
+ addr = trap_registers.pc + 4 + OPCODE_BT_DISP(op);
+ else
+ addr = trap_registers.pc + 2;
+ }
+
+ /* BFS */
+ else if (OPCODE_BFS(op)) {
+ if (!(trap_registers.sr & SR_T_BIT_MASK))
+ addr = trap_registers.pc + 4 + OPCODE_BT_DISP(op);
+ else
+ addr = trap_registers.pc + 4; /* Not in delay slot */
+ }
+
+ /* BRA */
+ else if (OPCODE_BRA(op))
+ addr = trap_registers.pc + 4 + OPCODE_BRA_DISP(op);
+
+}
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
+ /* BRAF */
+ else if (OPCODE_BRAF(op))
+ addr = trap_registers.pc + 4
+ + trap_registers.regs[OPCODE_BRAF_REG(op)];
+
+ /* BSR */
+ else if (OPCODE_BSR(op))
+ addr = trap_registers.pc + 4 + OPCODE_BSR_DISP(op);
+
+ /* BSRF */
+ else if (OPCODE_BSRF(op))
+ addr = trap_registers.pc + 4
+ + trap_registers.regs[OPCODE_BSRF_REG(op)];
+
+ /* JMP */
+ else if (OPCODE_JMP(op))
+ addr = trap_registers.regs[OPCODE_JMP_REG(op)];
+
+ /* JSR */
+ else if (OPCODE_JSR(op))
+ addr = trap_registers.regs[OPCODE_JSR_REG(op)];
+
+ /* RTS */
+ else if (OPCODE_RTS(op))
+ addr = trap_registers.pr;
+
+ /* RTE */
+ else if (OPCODE_RTE(op))
+ addr = trap_registers.regs[15];
+
+ /* Other */
+ else
+ addr = trap_registers.pc + 2;
+
+ kgdb_flush_icache_range(addr, addr + 2);
+ return (short *)addr;
+}
+
+/* The command loop, read and act on requests */
+int kgdb_arch_handle_exception(int e_vector, int signo, int err_code,
+ char *remcom_in_buffer, char *remcom_out_buffer,
+ struct pt_regs *ign)
+{
+ unsigned long addr;
+ char *ptr = &remcom_in_buffer[1];
+
+ /* Examine first char of buffer to see what we need to do */
+ switch (remcom_in_buffer[0]) {
+ case 'c': /* Continue at address AA..AA (optional) */
+ case 's': /* Step one instruction from AA..AA */
+ /* Try to read optional parameter, PC unchanged if none */
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
+ if (kgdb_hex2long(&ptr, &addr))
+ trap_registers.pc = addr;
+
+ atomic_set(&cpu_doing_single_step, -1);
+ if (remcom_in_buffer[0] == 's') {
+ /* Replace the instruction immediately after the
+ * current instruction (i.e. next in the expected
+ * flow of control) with a trap instruction, so that
+ * returning will cause only a single instruction to
+ * be executed. Note that this model is slightly
+ * broken for instructions with delay slots
+ * (e.g. B[TF]S, BSR, BRA etc), where both the branch
+ * and the instruction in the delay slot will be
+ * executed.
+ */
+ /* Determine where the target instruction will send
+ * us to */
+ unsigned short *next_addr = get_step_address();
+ stepped_address = (int)next_addr;
+
+ /* Replace it */
+ stepped_opcode = *(short *)next_addr;
+ *next_addr = STEP_OPCODE;
+
+ /* Flush and return */
+ kgdb_flush_icache_range((long)next_addr,
+ (long)next_addr + 2);
+ if (kgdb_contthread)
+ atomic_set(&cpu_doing_single_step,
+ smp_processor_id());
+ }
+ return 0;
+ }
+ return -1;
+ }
+
+ /*
+ * When an exception has occurred, we are called. We need to set things
+ * up so that we can call kgdb_handle_exception to handle requests from
+ * the remote GDB.
+ */
+void kgdb_exception_handler(struct pt_regs *regs)
+{
+ int excep_code, vbr_val;
+ int count;
+
+ /* Copy kernel regs (from stack) */
+ for (count = 0; count < 16; count++)
+ trap_registers.regs[count] = regs->regs[count];
+ trap_registers.pc = regs->pc;
+ trap_registers.pr = regs->pr;
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
+ trap_registers.sr = regs->sr;
+ trap_registers.gbr = regs->gbr;
+ trap_registers.mach = regs->mach;
+ trap_registers.macl = regs->macl;
+
+ __asm__ __volatile__("stc vbr, %0":="r"(vbr_val));
+ trap_registers.vbr = vbr_val;
+
+ /* Get the exception code. */
+ __asm__ __volatile__("stc r2_bank, %0":="r"(excep_code));
+
+ excep_code >>= 5;
+
+ /* If we got an NMI, and KGDB is not yet initialized, call
+ * breakpoint() to try and initialize everything for us. */
+ if (excep_code == NMI_VEC && !kgdb_initialized) {
+ breakpoint();
+ return;
+ }
+
+ #ifdef TRA
+ /* TRAP_VEC exception indicates a software trap inserted in place of
+ * code by GDB so back up PC by one instruction, as this instruction
+ * will later be replaced by its original one. Do NOT do this for
+ * trap 0xff, since that indicates a compiled-in breakpoint which
+ * will not be replaced (and we would retake the trap forever) */
+ if (excep_code == TRAP_VEC &&
+ *(volatile unsigned long *)TRA != (0xff << 2))
+ trap_registers.pc -= 2;
+ #endif
+
+ /* If we have been single-stepping, put back the old instruction.
+ * We use stepped_address in case we have stopped more than one
+ * instruction away. */
+ if (stepped_opcode != 0) {
+ *(short *)stepped_address = stepped_opcode;
+ kgdb_flush_icache_range(stepped_address, stepped_address + 2);
+ }
+ stepped_opcode = 0;
+
+ /* Call the stub to do the processing. Note that not everything we
+ * need to send back and forth lives in pt_regs. */
+ kgdb_handle_exception(excep_code, compute_signal(excep_code), 0, regs);
+
+ /* Copy back the (maybe modified) registers */
+ for (count = 0; count < 16; count++)
+ regs->regs[count] = trap_registers.regs[count];
+ regs->pc = trap_registers.pc;
+ regs->pr = trap_registers.pr;
+ regs->sr = trap_registers.sr;
+ regs->gbr = trap_registers.gbr;
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
+ regs->mach = trap_registers.mach;
+ regs->macl = trap_registers.macl;
+
+ vbr_val = trap_registers.vbr;
+ __asm__ __volatile__("ldc %0, vbr": : "r"(vbr_val));
+}
+
+int __init kgdb_arch_init(void)
+{
+ per_cpu_trap_init();
+
+ return 0;
+}
+
+struct kgdb_arch arch_kgdb_ops = {
+#ifdef CONFIG_CPU_LITTLE_ENDIAN
+ .gdb_bpt_instr = {0xff, 0xc3},
+#else /* ! CONFIG_CPU_LITTLE_ENDIAN */
+ .gdb_bpt_instr = {0xc3, 0xff},
+#endif
+};
---- a/arch/sh/kernel/kgdb_jmp.S
+++ /dev/null
@@ -1,33 +0,0 @@
-#include <linux/linkage.h>
-
-ENTRY(setjmp)
- add #(9*4), r4
- sts.l pr, @-r4
- mov.l r15, @-r4
- mov.l r14, @-r4
- mov.l r13, @-r4
- mov.l r12, @-r4
- mov.l r11, @-r4
- mov.l r10, @-r4
- mov.l r9, @-r4
- mov.l r8, @-r4
- rts
- mov #0, r0
-
-ENTRY(longjmp)
- mov.l @r4+, r8
- mov.l @r4+, r9
- mov.l @r4+, r10
- mov.l @r4+, r11
- mov.l @r4+, r12
- mov.l @r4+, r13
- mov.l @r4+, r14
- mov.l @r4+, r15
- lds.l @r4+, pr
- mov r5, r0
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
- tst r0, r0
- bf 1f
- mov #1, r0 ! in case val==0
-1: rts
- nop
-
--- a/arch/sh/kernel/kgdb_stub.c
+++ /dev/null
@@ -1,1054 +0,0 @@
-/*
- * May be copied or modified under the terms of the GNU General Public
- * License. See linux/COPYING for more information.
- *
- * Contains extracts from code by Glenn Engel, Jim Kingdon,
- * David Grothe <dave@xxxxxxxx>, Tigran Aivazian <tigran@xxxxxxxx>,
- * Amit S. Kale <akale@xxxxxxxxxxxx>, William Gatliff <bgat@xxxxxxxxxxxxxxxxxxxx>,
- * Ben Lee, Steve Chamberlain and Benoit Miller <fulg@xxxxxxxx>.
- *
- * This version by Henry Bell <henry.bell@xxxxxx>
- * Minor modifications by Jeremy Siegel <jsiegel@xxxxxxxxxxxx>
- *
- * Contains low-level support for remote debug using GDB.
- *
- * To enable debugger support, two things need to happen. 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.
- * A breakpoint also needs to be generated to begin communication. This
- * is most easily accomplished by a call to breakpoint() which does
- * a trapa if the initialisation phase has been successfully completed.
- *
- * In this case, set_debug_traps() is not used to "take over" exceptions;
- * other kernel code is modified instead to enter the kgdb functions here
- * when appropriate (see entry.S for breakpoint traps and NMI interrupts,
- * see traps.c for kernel error exceptions).
- *
- * The following gdb commands are supported:
- *
- * Command Function Return value
- *
- * g return the value of the CPU registers hex data or ENN
- * G set the value of the CPU registers OK or ENN
- *
- * 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
- * XAA..AA,LLLL: Same, but data is binary (not hex) OK or ENN
- *
- * c Resume at current address SNN ( signal NN)
- * cAA..AA Continue at address AA..AA SNN
- * CNN; Resume at current address with signal SNN
- * CNN;AA..AA Resume at address AA..AA with signal SNN
- *
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
– * s Step one instruction SNN
– * sAA..AA Step one instruction from AA..AA SNN
– * SNN; Step one instruction with signal SNN
– * SNNAA..AA Step one instruction from AA..AA w/NN SNN
– *
– * k kill (Detach GDB)
– *
– * d Toggle debug flag
– * D Detach GDB
– *
– * Hct Set thread t for operations, OK or ENN
– * c = 'c' (step, cont), c = 'g' (other
– * operations)
– *
– * qC Query current thread ID QCpid
– * qfThreadInfo Get list of current threads (first) m<id>
– * qsThreadInfo " " " " " (subsequent)
– * qOffsets Get section offsets Text=x;Data=y;Bss=z
– *
– * TXX Find if thread XX is alive OK or ENN
– * ? What was the last signal ? SNN (signal NN)
– * O Output to GDB console
– *
– * Remote communication protocol.
– *
– * A debug packet whose contents are <data> is encapsulated for
– * transmission in the form:
– *
– * $ <data> # CSUM1 CSUM2
– *
– * <data> must be ASCII alphanumeric and cannot include characters
– * '$' or '#'. If <data> starts with two characters followed by
– * ':', then the existing stubs interpret this as a sequence number.
– *
– * CSUM1 and CSUM2 are ascii hex representation of an 8-bit
– * checksum of <data>, the most significant nibble is sent first.
– * the hex digits 0–9,a–f are used.
– *
– * Receiver responds with:
– *
– * + – if CSUM is correct and ready for next packet
– * – – if CSUM is incorrect
– *
– * Responses can be run-length encoded to save space. A '*' means that
– * the next character is an ASCII encoding giving a repeat count which
– * stands for that many repetitions of the character preceding the '*'.
– * The encoding is n+29, yielding a printable character where n >=3
– * (which is where RLE starts to win). Don't use an n > 126.
– *
– * So "0*" means the same as "0000".
– */
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
-
-#include <linux/string.h>
-#include <linux/kernel.h>
-#include <linux/sched.h>
-#include <linux/smp.h>
-#include <linux/spinlock.h>
-#include <linux/delay.h>
-#include <linux/linkage.h>
-#include <linux/init.h>
-#include <linux/console.h>
-#include <linux/sysrq.h>
-#include <asm/system.h>
-#include <asm/cacheflush.h>
-#include <asm/current.h>
-#include <asm/signal.h>
-#include <asm/pgtable.h>
-#include <asm/ptrace.h>
-#include <asm/kgdb.h>
-#include <asm/io.h>
-
-/* Function pointers for linkage */
-kgdb_debug_hook_t *kgdb_debug_hook;
-kgdb_bus_error_hook_t *kgdb_bus_err_hook;
-
-int (*kgdb_getchar)(void);
-void (*kgdb_putchar)(int);
-
-static void put_debug_char(int c)
-{
- if (!kgdb_putchar)
- return;
- (*kgdb_putchar)(c);
-}
-static int get_debug_char(void)
-{
- if (!kgdb_getchar)
- return -1;
- return (*kgdb_getchar)();
-}
-
-/* Num chars in in/out bound buffers, register packets need NUMREGBYTES * 2 */
-#define BUFMAX 1024
-#define NUMREGBYTES (MAXREG*4)
-#define OUTBUFMAX (NUMREGBYTES*2+512)
-
-enum regs {
- R0 = 0, R1, R2, R3, R4, R5, R6, R7,
- R8, R9, R10, R11, R12, R13, R14, R15,
- PC, PR, GBR, VBR, MACH, MACL, SR,
- /* */
- MAXREG
}
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
-};  
-  
-static unsigned int registers[MAXREG];  
-struct kgdb_regs trap_registers;  
-  
-char kgdb_in_gdb_mode;  
-char in_nmi; /* Set during NMI to prevent reentry */  
-int kgdb_nofault; /* Boolean to ignore bus errs (i.e. in GDB) */  
-  
-/* Default values for SCI (can override via kernel args in setup.c) */  
-#ifndef CONFIG_KGDB_DEFPORT  
-#define CONFIG_KGDB_DEFPORT 1  
-#endif  
-  
-#ifndef CONFIG_KGDB_DEFBAUD  
-#define CONFIG_KGDB_DEFBAUD 115200  
-#endif  
-  
-#if defined(CONFIG_KGDB_DEFPARITY_E)  
-#define CONFIG_KGDB_DEFPARITY 'E'  
-#elif defined(CONFIG_KGDB_DEFPARITY_O)  
-#define CONFIG_KGDB_DEFPARITY 'O'  
-#else /* CONFIG_KGDB_DEFPARITY_N */  
-#define CONFIG_KGDB_DEFPARITY 'N'  
-#endif  
-  
-#ifndef CONFIG_KGDB_DEFBITS_7  
-#define CONFIG_KGDB_DEFBITS '7'  
-#else /* CONFIG_KGDB_DEFBITS_8 */  
-#define CONFIG_KGDB_DEFBITS '8'  
-#endif  
-  
-/* SCI/UART settings, used in kgdb_console_setup() */  
-int kgdb_portnum = CONFIG_KGDB_DEFPORT;  
-int kgdb_baud = CONFIG_KGDB_DEFBAUD;  
-char kgdb_parity = CONFIG_KGDB_DEFPARITY;  
-char kgdb_bits = CONFIG_KGDB_DEFBITS;  
-  
-/* Jump buffer for setjmp/longjmp */  
-static jmp_buf rem_com_env;  
-  
-/* TRA differs sh3/4 */  
-#if defined(CONFIG_CPU_SH3)  
-#define TRA 0xfffffd0  
-#elif defined(CONFIG_CPU_SH4)  
-#define TRA 0xff000020  
-#endif  
-  
-/* Macros for single step instruction identification */  
-#define OPPOSITE_BT(op) (((op) & 0xff00) == 0x8900)  
-#define OPPOSITE_BF(op) (((op) & 0xff00) == 0x8b00)
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```

-#define OPCODE_BTF_DISP(op) (((op) & 0x80) ? (((op) | 0xfffff80) << 1) : \
- (((op) & 0x7f) << 1))
-#define OPCODE_BFS(op) (((op) & 0xff00) == 0x8f00)
-#define OPCODE_BTS(op) (((op) & 0xff00) == 0x8d00)
-#define OPCODE_BRA(op) (((op) & 0xf000) == 0xa000)
-#define OPCODE_BRA_DISP(op) (((op) & 0x800) ? (((op) | 0xffff800) << 1) : \
- (((op) & 0x7ff) << 1))
-#define OPCODE_BRAF(op) (((op) & 0xf0ff) == 0x0023)
-#define OPCODE_BRAF_REG(op) (((op) & 0x0f00) >> 8)
-#define OPCODE_BSR(op) (((op) & 0xf000) == 0xb000)
-#define OPCODE_BSR_DISP(op) (((op) & 0x800) ? (((op) | 0xffff800) << 1) : \
- (((op) & 0x7ff) << 1))
-#define OPCODE_BSRF(op) (((op) & 0xf0ff) == 0x0003)
-#define OPCODE_BSRF_REG(op) (((op) >> 8) & 0xf)
-#define OPCODE_JMP(op) (((op) & 0xf0ff) == 0x402b)
-#define OPCODE_JMP_REG(op) (((op) >> 8) & 0xf)
-#define OPCODE_JSR(op) (((op) & 0xf0ff) == 0x400b)
-#define OPCODE_JSR_REG(op) (((op) >> 8) & 0xf)
-#define OPCODE_RTS(op) ((op) == 0xb)
-#define OPCODE_RTE(op) ((op) == 0x2b)
-
-#define SR_T_BIT_MASK 0x1
-#define STEP_OPCODE 0xc320
-#define BIOS_CALL_TRAP 0x3f
-
-/* Exception codes as per SH-4 core manual */
-#define ADDRESS_ERROR_LOAD_VEC 7
-#define ADDRESS_ERROR_STORE_VEC 8
-#define TRAP_VEC 11
-#define INVALID_INSN_VEC 12
-#define INVALID_SLOT_VEC 13
-#define NMI_VEC 14
-#define USER_BREAK_VEC 15
-#define SERIAL_BREAK_VEC 58
-
-/* Misc static */
-static int stepped_address;
-static short stepped_opcode;
-static char in_buffer[BUFMAX];
-static char out_buffer[OUTBUFMAX];
-
-static void kgdb_to_gdb(const char *s);
-
-/* Convert ch to hex */
-static int hex(const char ch)
-{
- if ((ch >= 'a') && (ch <= 'f'))
- return (ch - 'a' + 10);
- if ((ch >= '0') && (ch <= '9'))
- return (ch - '0');
- if ((ch >= 'A') && (ch <= 'F'))

```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
- return (ch - 'A' + 10);
- return (-1);
-}
-
-/* Convert the memory pointed to by mem into hex, placing result in buf.
- Returns a pointer to the last char put in buf (null) */
-static char *mem_to_hex(const char *mem, char *buf, const int count)
-{
- int i;
- int ch;
- unsigned short s_val;
- unsigned long l_val;
-
- /* Check for 16 or 32 */
- if (count == 2 && ((long) mem & 1) == 0) {
- s_val = *(unsigned short *) mem;
- mem = (char *) &s_val;
- } else if (count == 4 && ((long) mem & 3) == 0) {
- l_val = *(unsigned long *) mem;
- mem = (char *) &l_val;
- }
- for (i = 0; i < count; i++) {
- ch = *mem++;
- *buf++ = highhex(ch);
- *buf++ = lowhex(ch);
- }
- *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 *hex_to_mem(const char *buf, char *mem, const int count)
-{
- int i;
- unsigned char ch;
-
- for (i = 0; i < count; i++) {
- ch = hex(*buf++) << 4;
- ch = ch + hex(*buf++);
- *mem++ = ch;
- }
- return (mem);
-}
-
-/* While finding valid hex chars, convert to an integer, then return it */
-static int hex_to_int(char **ptr, int *int_value)
-{
- int num_chars = 0;
- int hex_value;
-
-
```

```

- *int_value = 0;
-
- while (**ptr) {
- hex_value = hex(**ptr);
- if (hex_value >= 0) {
- *int_value = (*int_value << 4) | hex_value;
- num_chars++;
- } else
- break;
- (*ptr)++;
- }
- return num_chars;
-}
-
-/* Copy the binary array pointed to by buf into mem. Fix $, #,
- and 0x7d escaped with 0x7d. Return a pointer to the character
- after the last byte written. */
-static char *ebin_to_mem(const char *buf, char *mem, int count)
-{
- for (; count > 0; count--, buf++) {
- if (*buf == 0x7d)
- *mem++ = *(++buf) ^ 0x20;
- else
- *mem++ = *buf;
- }
- return mem;
-}
-
-/* Pack a hex byte */
-static char *pack_hex_byte(char *pkt, int byte)
-{
- *pkt++ = hexchars[(byte >> 4) & 0xf];
- *pkt++ = hexchars[(byte & 0xf)];
- return pkt;
-}
-
-/* Scan for the start char '$', read the packet and check the checksum */
-static void get_packet(char *buffer, int buflen)
-{
- unsigned char checksum;
- unsigned char xmitcsum;
- int i;
- int count;
- char ch;
-
- do {
- /* Ignore everything until the start character */
- while ((ch = get_debug_char()) != '$');
-
- checksum = 0;
- xmitcsum = -1;

```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
- count = 0;
-
- /* Now, read until a # or end of buffer is found */
- while (count < (buflen - 1)) {
- ch = get_debug_char();
-
- if (ch == '#')
- break;
-
- checksum = checksum + ch;
- buffer[count] = ch;
- count = count + 1;
- }
-
- buffer[count] = 0;
-
- /* Continue to read checksum following # */
- if (ch == '#') {
- xmitcsum = hex(get_debug_char()) << 4;
- xmitcsum += hex(get_debug_char());
-
- /* Checksum */
- if (checksum != xmitcsum)
- put_debug_char('-'); /* Failed checksum */
- else {
- /* Ack successful transfer */
- put_debug_char('+');
-
- /* If a sequence char is present, reply
- the sequence ID */
- if (buffer[2] == ':') {
- put_debug_char(buffer[0]);
- put_debug_char(buffer[1]);
-
- /* Remove sequence chars from buffer */
- count = strlen(buffer);
- for (i = 3; i <= count; i++)
- buffer[i - 3] = buffer[i];
- }
- }
- }
- while (checksum != xmitcsum); /* Keep trying while we fail */
- }
-
- /* Send the packet in the buffer with run-length encoding */
- static void put_packet(char *buffer)
- {
- int checksum;
- char *src;
- int runlen;
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
- int encode;
-
- do {
- src = buffer;
- put_debug_char('$');
- checksum = 0;
-
- /* Continue while we still have chars left */
- while (*src) {
- /* Check for runs up to 99 chars long */
- for (runlen = 1; runlen < 99; runlen++) {
- if (src[0] != src[runlen])
- break;
- }
-
- if (runlen > 3) {
- /* Got a useful amount, send encoding */
- encode = runlen + ' ' - 4;
- put_debug_char(*src); checksum += *src;
- put_debug_char('*'); checksum += '*';
- put_debug_char(encode); checksum += encode;
- src += runlen;
- } else {
- /* Otherwise just send the current char */
- put_debug_char(*src); checksum += *src;
- src += 1;
- }
- }
-
- /* '#' Separator, put high and low components of checksum */
- put_debug_char('#');
- put_debug_char(highhex(checksum));
- put_debug_char(lowhex(checksum));
- }
- while ((get_debug_char()) != '+'); /* While no ack */
- }
-
- /* A bus error has occurred - perform a longjmp to return execution and
- allow handling of the error */
- static void kgdb_handle_bus_error(void)
- {
- longjmp(rem_com_env, 1);
- }
-
- /* Translate SH-3/4 exception numbers to unix-like signal values */
- static int compute_signal(const int excep_code)
- {
- int signal;
-
- switch (excep_code) {
-
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
- case INVALID_INSN_VEC:
- case INVALID_SLOT_VEC:
- signal = SIGILL;
- break;
- case ADDRESS_ERROR_LOAD_VEC:
- case ADDRESS_ERROR_STORE_VEC:
- signal = SIGSEGV;
- break;
-
- case SERIAL_BREAK_VEC:
- case NMI_VEC:
- signal = SIGINT;
- break;
-
- case USER_BREAK_VEC:
- case TRAP_VEC:
- signal = SIGTRAP;
- break;
-
- default:
- signal = SIGBUS; /* "software generated" */
- break;
- }
-
- return (signal);
-}
-
-/* Make a local copy of the registers passed into the handler (bletch) */
-static void kgdb_regs_to_gdb_regs(const struct kgdb_regs *regs,
- int *gdb_regs)
-{
- gdb_regs[R0] = regs->regs[R0];
- gdb_regs[R1] = regs->regs[R1];
- gdb_regs[R2] = regs->regs[R2];
- gdb_regs[R3] = regs->regs[R3];
- gdb_regs[R4] = regs->regs[R4];
- gdb_regs[R5] = regs->regs[R5];
- gdb_regs[R6] = regs->regs[R6];
- gdb_regs[R7] = regs->regs[R7];
- gdb_regs[R8] = regs->regs[R8];
- gdb_regs[R9] = regs->regs[R9];
- gdb_regs[R10] = regs->regs[R10];
- gdb_regs[R11] = regs->regs[R11];
- gdb_regs[R12] = regs->regs[R12];
- gdb_regs[R13] = regs->regs[R13];
- gdb_regs[R14] = regs->regs[R14];
- gdb_regs[R15] = regs->regs[R15];
- gdb_regs[PC] = regs->pc;
- gdb_regs[PR] = regs->pr;
- gdb_regs[GBR] = regs->gbr;
- gdb_regs[MACH] = regs->mach;
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
- gdb_regs[MACL] = regs->macl;
- gdb_regs[SR] = regs->sr;
- gdb_regs[VBR] = regs->vbr;
-}
-
-/* Copy local gdb registers back to kgdb regs, for later copy to kernel */
-static void gdb_regs_to_kgdb_regs(const int *gdb_regs,
- struct kgdb_regs *regs)
-{
- regs->regs[R0] = gdb_regs[R0];
- regs->regs[R1] = gdb_regs[R1];
- regs->regs[R2] = gdb_regs[R2];
- regs->regs[R3] = gdb_regs[R3];
- regs->regs[R4] = gdb_regs[R4];
- regs->regs[R5] = gdb_regs[R5];
- regs->regs[R6] = gdb_regs[R6];
- regs->regs[R7] = gdb_regs[R7];
- regs->regs[R8] = gdb_regs[R8];
- regs->regs[R9] = gdb_regs[R9];
- regs->regs[R10] = gdb_regs[R10];
- regs->regs[R11] = gdb_regs[R11];
- regs->regs[R12] = gdb_regs[R12];
- regs->regs[R13] = gdb_regs[R13];
- regs->regs[R14] = gdb_regs[R14];
- regs->regs[R15] = gdb_regs[R15];
- regs->pc = gdb_regs[PC];
- regs->pr = gdb_regs[PR];
- regs->gbr = gdb_regs[GBR];
- regs->mach = gdb_regs[MACH];
- regs->macl = gdb_regs[MACL];
- regs->sr = gdb_regs[SR];
- regs->vbr = gdb_regs[VBR];
-}
-
-/* Calculate the new address for after a step */
-static short *get_step_address(void)
-{
- short op = *(short *) trap_registers.pc;
- long addr;
-
- /* BT */
- if (OPCODE_BT(op)) {
- if (trap_registers.sr & SR_T_BIT_MASK)
- addr = trap_registers.pc + 4 + OPCODE_BT_DISP(op);
- else
- addr = trap_registers.pc + 2;
- }
-
- /* BTS */
- else if (OPCODE_BTS(op)) {
- if (trap_registers.sr & SR_T_BIT_MASK)
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
- addr = trap_registers.pc + 4 + OPCODE_BTF_DISP(op);
- else
- addr = trap_registers.pc + 4; /* Not in delay slot */
- }
-
- /* BF */
- else if (OPCODE_BF(op)) {
- if (!(trap_registers.sr & SR_T_BIT_MASK))
- addr = trap_registers.pc + 4 + OPCODE_BTF_DISP(op);
- else
- addr = trap_registers.pc + 2;
- }
-
- /* BFS */
- else if (OPCODE_BFS(op)) {
- if (!(trap_registers.sr & SR_T_BIT_MASK))
- addr = trap_registers.pc + 4 + OPCODE_BTF_DISP(op);
- else
- addr = trap_registers.pc + 4; /* Not in delay slot */
- }
-
- /* BRA */
- else if (OPCODE_BRA(op))
- addr = trap_registers.pc + 4 + OPCODE_BRA_DISP(op);
-
- /* BRAF */
- else if (OPCODE_BRAF(op))
- addr = trap_registers.pc + 4
- + trap_registers.regs[OPCODE_BRAF_REG(op)];
-
- /* BSR */
- else if (OPCODE_BSR(op))
- addr = trap_registers.pc + 4 + OPCODE_BSR_DISP(op);
-
- /* BSRF */
- else if (OPCODE_BSRF(op))
- addr = trap_registers.pc + 4
- + trap_registers.regs[OPCODE_BSRF_REG(op)];
-
- /* JMP */
- else if (OPCODE_JMP(op))
- addr = trap_registers.regs[OPCODE_JMP_REG(op)];
-
- /* JSR */
- else if (OPCODE_JSR(op))
- addr = trap_registers.regs[OPCODE_JSR_REG(op)];
-
- /* RTS */
- else if (OPCODE_RTS(op))
- addr = trap_registers.pr;
-
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
- /* RTE */
- else if (OPCODE_RTE(op))
- addr = trap_registers.regs[15];
-
- /* Other */
- else
- addr = trap_registers.pc + 2;
-
- flush_icache_range(addr, addr + 2);
- return (short *) addr;
-}
-
-/* Set up a single-step. Replace the instruction immediately after the
- current instruction (i.e. next in the expected flow of control) with a
- trap instruction, so that returning will cause only a single instruction
- to be executed. Note that this model is slightly broken for instructions
- with delay slots (e.g. B[TF]S, BSR, BRA etc), where both the branch
- and the instruction in the delay slot will be executed. */
-static void do_single_step(void)
-{
- unsigned short *addr = 0;
-
- /* Determine where the target instruction will send us to */
- addr = get_step_address();
- stepped_address = (int)addr;
-
- /* Replace it */
- stepped_opcode = *(short *)addr;
- *addr = STEP_OPCODE;
-
- /* Flush and return */
- flush_icache_range((long) addr, (long) addr + 2);
-}
-
-/* Undo a single step */
-static void undo_single_step(void)
-{
- /* If we have stepped, put back the old instruction */
- /* Use stepped_address in case we stopped elsewhere */
- if (stepped_opcode != 0) {
- *(short*)stepped_address = stepped_opcode;
- flush_icache_range(stepped_address, stepped_address + 2);
- }
- stepped_opcode = 0;
-}
-
-/* Send a signal message */
-static void send_signal_msg(const int signum)
-{
- out_buffer[0] = 'S';
- out_buffer[1] = highhex(signum);
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
- out_buffer[2] = lowhex(signum);
- out_buffer[3] = 0;
- put_packet(out_buffer);
-}
-
-/* Reply that all was well */
-static void send_ok_msg(void)
-{
- strcpy(out_buffer, "OK");
- put_packet(out_buffer);
-}
-
-/* Reply that an error occurred */
-static void send_err_msg(void)
-{
- strcpy(out_buffer, "E01");
- put_packet(out_buffer);
-}
-
-/* Empty message indicates unrecognised command */
-static void send_empty_msg(void)
-{
- put_packet("");
-}
-
-/* Read memory due to 'm' message */
-static void read_mem_msg(void)
-{
- char *ptr;
- int addr;
- int length;
-
- /* Jmp, disable bus error handler */
- if (setjmp(rem_com_env) == 0) {
-
- kgdb_nofault = 1;
-
- /* Walk through, have m<addr>,<length> */
- ptr = &in_buffer[1];
- if (hex_to_int(&ptr, &addr) && (*ptr++ == ','))
- if (hex_to_int(&ptr, &length)) {
- ptr = 0;
- if (length * 2 > OUTBUFMAX)
- length = OUTBUFMAX / 2;
- mem_to_hex((char *) addr, out_buffer, length);
- }
- if (ptr)
- send_err_msg();
- else
- put_packet(out_buffer);
- } else
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
- send_err_msg();
-
- /* Restore bus error handler */
- kgdb_nofault = 0;
- }
-
- /* Write memory due to 'M' or 'X' message */
- static void write_mem_msg(int binary)
- {
- char *ptr;
- int addr;
- int length;
-
- if (setjmp(rem_com_env) == 0) {
-
- kgdb_nofault = 1;
-
- /* Walk through, have M<addr>,<length>:<data> */
- ptr = &in_buffer[1];
- if (hex_to_int(&ptr, &addr) && (*ptr++ == ','))
- if (hex_to_int(&ptr, &length) && (*ptr++ == ':')) {
- if (binary)
- ebin_to_mem(ptr, (char*)addr, length);
- else
- hex_to_mem(ptr, (char*)addr, length);
- flush_icache_range(addr, addr + length);
- ptr = 0;
- send_ok_msg();
- }
- if (ptr)
- send_err_msg();
- } else
- send_err_msg();
-
- /* Restore bus error handler */
- kgdb_nofault = 0;
- }
-
- /* Continue message */
- static void continue_msg(void)
- {
- /* Try to read optional parameter, PC unchanged if none */
- char *ptr = &in_buffer[1];
- int addr;
-
- if (hex_to_int(&ptr, &addr))
- trap_registers.pc = addr;
- }
-
- /* Continue message with signal */
- static void continue_with_sig_msg(void)
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
-{
- int signal;
- char *ptr = &in_buffer[1];
- int addr;
-
- /* Report limitation */
- kgdb_to_gdb("Cannot force signal in kgdb, continuing anyway.\n");
-
- /* Signal */
- hex_to_int(&ptr, &signal);
- if (*ptr == ';')
- ptr++;
-
- /* Optional address */
- if (hex_to_int(&ptr, &addr))
- trap_registers.pc = addr;
-}
-
-/* Step message */
-static void step_msg(void)
-{
- continue_msg();
- do_single_step();
-}
-
-/* Step message with signal */
-static void step_with_sig_msg(void)
-{
- continue_with_sig_msg();
- do_single_step();
-}
-
-/* Send register contents */
-static void send_regs_msg(void)
-{
- kgdb_regs_to_gdb_regs(&trap_registers, registers);
- mem_to_hex((char *) registers, out_buffer, NUMREGBYTES);
- put_packet(out_buffer);
-}
-
-/* Set register contents – currently can't set other thread's registers */
-static void set_regs_msg(void)
-{
- kgdb_regs_to_gdb_regs(&trap_registers, registers);
- hex_to_mem(&in_buffer[1], (char *) registers, NUMREGBYTES);
- gdb_regs_to_kgdb_regs(registers, &trap_registers);
- send_ok_msg();
-}
-
-#ifdef CONFIG_SH_KGDB_CONSOLE
-/*
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
- * Bring up the ports..
- */
-static int __init kgdb_serial_setup(void)
-{
- struct console dummy;
- return kgdb_console_setup(&dummy, 0);
-}
-#else
-#define kgdb_serial_setup() 0
-#endif
-
-/* The command loop, read and act on requests */
-static void kgdb_command_loop(const int excep_code, const int trapa_value)
-{
- int sigval;
-
- /* Enter GDB mode (e.g. after detach) */
- if (!kgdb_in_gdb_mode) {
- /* Do serial setup, notify user, issue preemptive ack */
- printk(KERN_NOTICE "KGDB: Waiting for GDB\n");
- kgdb_in_gdb_mode = 1;
- put_debug_char('+');
- }
-
- /* Reply to host that an exception has occurred */
- sigval = compute_signal(excep_code);
- send_signal_msg(sigval);
-
- /* TRAP_VEC exception indicates a software trap inserted in place of
- code by GDB so back up PC by one instruction, as this instruction
- will later be replaced by its original one. Do NOT do this for
- trap 0xff, since that indicates a compiled-in breakpoint which
- will not be replaced (and we would retake the trap forever) */
- if ((excep_code == TRAP_VEC) && (trapa_value != (0x3c << 2)))
- trap_registers.pc -= 2;
-
- /* Undo any stepping we may have done */
- undo_single_step();
-
- while (1) {
- out_buffer[0] = 0;
- get_packet(in_buffer, BUFMAX);
-
- /* Examine first char of buffer to see what we need to do */
- switch (in_buffer[0]) {
- case '?': /* Send which signal we've received */
- send_signal_msg(sigval);
- break;
-
- case 'g': /* Return the values of the CPU registers */
- send_regs_msg();
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
- break;
-
- case 'G': /* Set the value of the CPU registers */
- set_regs_msg();
- break;
-
- case 'm': /* Read LLLL bytes address AA..AA */
- read_mem_msg();
- break;
-
- case 'M': /* Write LLLL bytes address AA..AA, ret OK */
- write_mem_msg(0); /* 0 = data in hex */
- break;
-
- case 'X': /* Write LLLL bytes esc bin address AA..AA */
- if (kgdb_bits == '8')
- write_mem_msg(1); /* 1 = data in binary */
- else
- send_empty_msg();
- break;
-
- case 'C': /* Continue, signum included, we ignore it */
- continue_with_sig_msg();
- return;
-
- case 'c': /* Continue at address AA..AA (optional) */
- continue_msg();
- return;
-
- case 'S': /* Step, signum included, we ignore it */
- step_with_sig_msg();
- return;
-
- case 's': /* Step one instruction from AA..AA */
- step_msg();
- return;
-
- case 'k': /* 'Kill the program' with a kernel ? */
- break;
-
- case 'D': /* Detach from program, send reply OK */
- kgdb_in_gdb_mode = 0;
- send_ok_msg();
- get_debug_char();
- return;
-
- default:
- send_empty_msg();
- break;
- }
- }
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
-}
-
-/* There has been an exception, most likely a breakpoint. */
-static void handle_exception(struct pt_regs *regs)
-{
- int excep_code, vbr_val;
- int count;
- int trapa_value = ctrl_inl(TRA);
-
- /* Copy kernel regs (from stack) */
- for (count = 0; count < 16; count++)
- trap_registers.regs[count] = regs->regs[count];
- trap_registers.pc = regs->pc;
- trap_registers.pr = regs->pr;
- trap_registers.sr = regs->sr;
- trap_registers.gbr = regs->gbr;
- trap_registers.mach = regs->mach;
- trap_registers.macl = regs->macl;
-
- asm("stc vbr, %0":="r"(vbr_val));
- trap_registers.vbr = vbr_val;
-
- /* Get excode for command loop call, user access */
- asm("stc r2_bank, %0":="r"(excep_code));
-
- /* Act on the exception */
- kgdb_command_loop(excep_code, trapa_value);
-
- /* Copy back the (maybe modified) registers */
- for (count = 0; count < 16; count++)
- regs->regs[count] = trap_registers.regs[count];
- regs->pc = trap_registers.pc;
- regs->pr = trap_registers.pr;
- regs->sr = trap_registers.sr;
- regs->gbr = trap_registers.gbr;
- regs->mach = trap_registers.mach;
- regs->macl = trap_registers.macl;
-
- vbr_val = trap_registers.vbr;
- asm("ldc %0, vbr":="r"(vbr_val));
-}
-
-asm linkage void kgdb_handle_exception(unsigned long r4, unsigned long r5,
- unsigned long r6, unsigned long r7,
- struct pt_regs __regs)
-{
- struct pt_regs *regs = RELOC_HIDE(&__regs, 0);
- handle_exception(regs);
-}
-
-/* Initialise the KGDB data structures and serial configuration */
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
-int __init kgdb_init(void)
- {
-     in_nmi = 0;
-     kgdb_nofault = 0;
-     stepped_opcode = 0;
-     kgdb_in_gdb_mode = 0;
-
-     if (kgdb_serial_setup() != 0) {
-         printk(KERN_NOTICE "KGDB: serial setup error\n");
-         return -1;
-     }
-
-     /* Init ptr to exception handler */
-     kgdb_debug_hook = handle_exception;
-     kgdb_bus_err_hook = kgdb_handle_bus_error;
-
-     /* Enter kgdb now if requested, or just report init done */
-     printk(KERN_NOTICE "KGDB: stub is initialized.\n");
-
-     return 0;
- }
-
- /* Make function available for "user messages"; console will use it too. */
-
- char gdbmsgbuf[BUFMAX];
- #define MAXOUT ((BUFMAX-2)/2)
-
- static void kgdb_msg_write(const char *s, unsigned count)
- {
-     int i;
-     int wcount;
-     char *bufptr;
-
-     /* 'O'utput */
-     gdbmsgbuf[0] = 'O';
-
-     /* Fill and send buffers... */
-     while (count > 0) {
-         bufptr = gdbmsgbuf + 1;
-
-         /* Calculate how many this time */
-         wcount = (count > MAXOUT) ? MAXOUT : count;
-
-         /* Pack in hex chars */
-         for (i = 0; i < wcount; i++)
-             bufptr = pack_hex_byte(bufptr, s[i]);
-         *bufptr = '\0';
-
-         /* Move up */
-         s += wcount;
-         count -= wcount;
-     }
- }
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
-
- /* Write packet */
- put_packet(gdbmsgbuf);
- }
-}
-
-static void kgdb_to_gdb(const char *s)
-{
- kgdb_msg_write(s, strlen(s));
-}
-
-#ifdef CONFIG_SH_KGDB_CONSOLE
-void kgdb_console_write(struct console *co, const char *s, unsigned count)
-{
- /* Bail if we're not talking to GDB */
- if (!kgdb_in_gdb_mode)
- return;
-
- kgdb_msg_write(s, count);
-}
-#endif
-
-#ifdef CONFIG_KGDB_SYSRQ
-static void sysrq_handle_gdb(int key, struct tty_struct *tty)
-{
- printk("Entering GDB stub\n");
- breakpoint();
-}
-
-static struct sysrq_key_op sysrq_gdb_op = {
- .handler = sysrq_handle_gdb,
- .help_msg = "Gdb",
- .action_msg = "GDB",
-};
-
-static int gdb_register_sysrq(void)
-{
- printk("Registering GDB sysrq handler\n");
- register_sysrq_key('g', &sysrq_gdb_op);
- return 0;
-}
-module_init(gdb_register_sysrq);
-#endif
--- a/arch/sh/kernel/time.c
+++ b/arch/sh/kernel/time.c
@@ -259,11 +259,4 @@ void __init time_init(void)
((sh_hpt_frequency + 500) / 1000) / 1000,
((sh_hpt_frequency + 500) / 1000) % 1000);

-#if defined(CONFIG_SH_KGDB)
- /*
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
- * Set up kgdb as requested. We do it here because the serial
- * init uses the timer vars we just set up for figuring baud.
- */
- kgdb_init();
-#endif
}
--- a/arch/sh/kernel/traps.c
+++ b/arch/sh/kernel/traps.c
@@ -25,16 +25,10 @@
#include <linux/limits.h>
#include <asm/system.h>
#include <asm/uaccess.h>
+#include <linux/kgdb.h>

-#ifdef CONFIG_SH_KGDB
-#include <asm/kgdb.h>
-#define CHK_REMOTE_DEBUG(regs) \
- { \
- if (kgdb_debug_hook && !user_mode(regs)) \
- (*kgdb_debug_hook)(regs); \
- }
-#else
-#define CHK_REMOTE_DEBUG(regs)
+#ifndef CONFIG_KGDB
+#define kgdb_handle_exception(t, s, e, r)
#endif

#ifdef CONFIG_CPU_SH2
@@ -91,7 +85,9 @@ void die(const char * str, struct pt_reg

printk("%s: %04lx [#%d]\n", str, err & 0xffff, ++die_counter);

- CHK_REMOTE_DEBUG(regs);
+#ifdef CONFIG_KGDB
+ kgdb_handle_exception(1, SIGTRAP, err, regs);
+#endif
print_modules();
show_regs(regs);

@@ -700,7 +696,9 @@ asmlinkage void do_reserved_inst(unsigne
lookup_exception_vector(error_code);

local_irq_enable();
- CHK_REMOTE_DEBUG(regs);
+#ifdef CONFIG_KGDB
+ kgdb_handle_exception(1, SIGILL, error_code, regs);
+#endif
force_sig(SIGILL, ts);
die_if_no_fixup("reserved instruction", regs, error_code);
}
@@ -771,7 +769,9 @@ asmlinkage void do_illegal_slot_inst(uns
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
lookup_exception_vector(error_code);

local_irq_enable();
- CHK_REMOTE_DEBUG(regs);
+#ifdef CONFIG_KGDB
+ kgdb_handle_exception(1, SIGILL, error_code, regs);
+#endif
force_sig(SIGILL, tsk);
die_if_no_fixup("illegal slot instruction", regs, error_code);
}
--- a/arch/sh/mm/extable.c
+++ b/arch/sh/mm/extable.c
@@ -5,6 +5,7 @@
*/

#include <linux/module.h>
#include <linux/kgdb.h>
#include <asm/uaccess.h>

int fixup_exception(struct pt_regs *regs)
@@ -16,6 +17,12 @@ int fixup_exception(struct pt_regs *regs)
regs->pc = fixup->fixup;
return 1;
}
+#ifdef CONFIG_KGDB
+ if (atomic_read(&debugger_active) && kgdb_may_fault)
+ /* Restore our previous state. */
+ kgdb_fault_longjmp(kgdb_fault_jmp_regs);
+ /* Never reached. */
+#endif

return 0;
}
--- a/arch/sh/mm/fault.c
+++ b/arch/sh/mm/fault.c
@@ -18,7 +18,6 @@
#include <asm/system.h>
#include <asm/mmu_context.h>
#include <asm/tlbflush.h>
-#include <asm/kgdb.h>

/*
* This routine handles page faults. It determines the address,
@@ -39,11 +38,6 @@ asmlinkage void __kprobes do_page_fault(
trace_hardirqs_on();
local_irq_enable();

-#ifdef CONFIG_SH_KGDB
- if (kgdb_nofault && kgdb_bus_err_hook)
- kgdb_bus_err_hook();
-#endif
```

```

-
tsk = current;
mm = tsk->mm;
si_code = SEGV_MAPERR;
@@ -200,6 +194,7 @@ no_context:
die("Oops", regs, writeaccess);
bust_spinlocks(0);
do_exit(SIGKILL);
+ dump_stack();

/*
* We ran out of memory, or some other thing happened to us that made
@@ -262,11 +257,6 @@ asmlinkage int __kprobes __do_page_fault
spinlock_t *ptl = NULL;
int ret = 1;

-#ifdef CONFIG_SH_KGDB
- if (kgdb_nofault && kgdb_bus_err_hook)
- kgdb_bus_err_hook();
-#endif
-
/*
* We don't take page faults for P1, P2, and parts of P4, these
* are always mapped, whether it be due to legacy behaviour in
--- a/drivers/serial/sh-sci.c
+++ b/drivers/serial/sh-sci.c
@@ -120,7 +120,8 @@ static int get_char(struct uart_port *po
do {
status = sci_in(port, SCxSR);
if (status & SCxSR_ERRORS(port)) {
- handle_error(port);
+ /* Clear error flags. */
+ sci_out(port, SCxSR, SCxSR_ERROR_CLEAR(port));
continue;
}
} while (!(status & SCxSR_RDxF(port)));
@@ -162,6 +163,7 @@ static void put_string(struct sci_port *

#if defined(CONFIG_SH_STANDARD_BIOS) || defined(CONFIG_SH_KGDB)
int checksum;
+ const char hexchars[] = "0123456789abcdef";
int usegdb=0;

#ifdef CONFIG_SH_STANDARD_BIOS
@@ -186,18 +188,18 @@ static void put_string(struct sci_port *
int h, l;

c = *p++;
- h = highhex(c);
- l = lowhex(c);
+ h = hexchars[c >> 4];

```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
+ l = hexchars[c % 16];
put_char(port, h);
put_char(port, l);
checksum += h + l;
}
put_char(port, '#');
- put_char(port, highhex(checksum));
- put_char(port, lowhex(checksum));
+ put_char(port, hexchars[checksum >> 4]);
+ put_char(port, hexchars[checksum & 16]);
} while (get_char(port) != '+');
} else
-#endif /* CONFIG_SH_STANDARD_BIOS || CONFIG_SH_KGDB */
+#endif /* CONFIG_SH_STANDARD_BIOS */
for (i=0; i<count; i++) {
if (*p == 10)
put_char(port, '\r');
--- a/include/asm-sh/kgdb.h
+++ b/include/asm-sh/kgdb.h
@@ -2,82 +2,41 @@
* May be copied or modified under the terms of the GNU General Public
* License. See linux/COPYING for more information.
*
- * Based on original code by Glenn Engel, Jim Kingdon,
- * David Grothe <dave@xxxxxxxx>, Tigran Aivazian, <tigran@xxxxxxxx> and
- * Amit S. Kale <akale@xxxxxxxxxxxx>
- *
- * Super-H port based on sh-stub.c (Ben Lee and Steve Chamberlain) by
- * Henry Bell <henry.bell@xxxxxx>
- *
- * Header file for low-level support for remote debug using GDB.
+ * Based on a file that was modified or based on files by: Glenn Engel,
+ * Jim Kingdon, David Grothe <dave@xxxxxxxx>, Tigran Aivazian <tigran@xxxxxxxx>,
+ * Amit S. Kale <akale@xxxxxxxxxxxx>, sh-stub.c from Ben Lee and
+ * Steve Chamberlain, Henry Bell <henry.bell@xxxxxx>
+ *
+ * Maintainer: Tom Rini <trini@xxxxxxxxxxxxxxxxxxxxxxxx>
*
*/

#ifndef __KGDB_H
#define __KGDB_H

-#include <asm/ptrace.h>
+#include <asm-generic/kgdb.h>
+
+/* Based on sh-gdb.c from gdb-6.1, Glenn
+ Engel at HP Ben Lee and Steve Chamberlain */
+#define NUMREGBYTES 112 /* 92 */
+#define NUMCRITREGBYTES (9 << 2)
+#define BUFMAX 400
```

```

-/* Same as pt_regs but has vbr in place of syscall_nr */
+#ifndef __ASSEMBLY__
struct kgdb_regs {
- unsigned long regs[16];
- unsigned long pc;
- unsigned long pr;
- unsigned long sr;
- unsigned long gbr;
- unsigned long mach;
- unsigned long macl;
- unsigned long vbr;
+ unsigned long regs[16];
+ unsigned long pc;
+ unsigned long pr;
+ unsigned long gbr;
+ unsigned long vbr;
+ unsigned long mach;
+ unsigned long macl;
+ unsigned long sr;
};

-/* State info */
-extern char kgdb_in_gdb_mode;
-extern int kgdb_nofault; /* Ignore bus errors (in gdb mem access) */
-extern char in_nmi; /* Debounce flag to prevent NMI reentry*/
-
-/* SCI */
-extern int kgdb_portnum;
-extern int kgdb_baud;
-extern char kgdb_parity;
-extern char kgdb_bits;
-
-/* Init and interface stuff */
-extern int kgdb_init(void);
-extern int (*kgdb_getchar)(void);
-extern void (*kgdb_putchar)(int);
-
-/* Trap functions */
-#typedef void (kgdb_debug_hook_t)(struct pt_regs *regs);
-#typedef void (kgdb_bus_error_hook_t)(void);
-extern kgdb_debug_hook_t *kgdb_debug_hook;
-extern kgdb_bus_error_hook_t *kgdb_bus_err_hook;
-
-/* Console */
-struct console;
-void kgdb_console_write(struct console *co, const char *s, unsigned count);
-extern int kgdb_console_setup(struct console *, char *);
-
-/* Prototypes for jmp fns */
-#define _JBLEN 9

```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
-typedef int jmp_buf[_JBLEN];
-extern void longjmp(jmp_buf __jmpb, int __retval);
-extern int setjmp(jmp_buf __jmpb);
-
-/* Forced breakpoint */
-#define breakpoint() __asm__ __volatile__("trapa #0x3c")
-
-/* Taken from sh-stub.c of GDB 4.18 */
-static const char hexchars[] = "0123456789abcdef";
-
-/* Get high hex bits */
-static inline char highhex(const int x)
-{
- return hexchars[(x >> 4) & 0xf];
-}
-
-/* Get low hex bits */
-static inline char lowhex(const int x)
-{
- return hexchars[x & 0xf];
-}
+#define BREAKPOINT() asm("trapa #0xff");
+#define BREAK_INSTR_SIZE 2
+#define CACHE_FLUSH_IS_SAFE 1
+
+#endif /* !__ASSEMBLY__ */
#endif
--- a/lib/Kconfig.kgdb
+++ b/lib/Kconfig.kgdb
@@ -3,7 +3,7 @@ config WANT_EXTRA_DEBUG_INFORMATION
bool
select DEBUG_INFO
select UNWIND_INFO
- select FRAME_POINTER if X86
+ select FRAME_POINTER if X86 || SUPERH
default n

config UNWIND_INFO
@@ -14,7 +14,7 @@ config KGDB
bool "KGDB: kernel debugging with remote gdb"
select WANT_EXTRA_DEBUG_INFORMATION
select KGDB_ARCH_HAS_SHADOW_INFO if X86_64
- depends on DEBUG_KERNEL && (X86 || MIPS || IA64 || PPC)
+ depends on DEBUG_KERNEL && (X86 || MIPS || (SUPERH && !SUPERH64) || IA64 || PPC)
help
If you say Y here, it will be possible to remotely debug the
kernel using gdb. Documentation of kernel debugger is available
@@ -44,6 +44,7 @@ choice
default KGDB_CPM_UART if (CPM2 || 8xx)
default KGDB_SIBYTE if SIBYTE_SB1xxx_SOC
default KGDB_TXX9 if CPU_TX49XX
```

[PATCH 13/21] KGDB: This adds basic support for KGDB on SuperH

```
+ default KGDB_SH_SCI if SERIAL_SH_SCI
default KGDB_8250_NOMODULE
help
There are a number of different ways in which you can communicate
@@ -104,6 +105,12 @@ config KGDB_TXX9
depends on MIPS && CPU_TX49XX
help
Uses TX49xx serial port to communicate with the host GDB.
+
+config KGDB_SH_SCI
+ bool "KGDB: On SH SCI(F) serial port"
+ depends on SUPERH && SERIAL_SH_SCI
+ help
+ Uses the SH SCI(F) serial port to communicate with the host GDB.
endchoice

choice
```