By Vlad Lungu vlad.lungu@windriver.com 2007-Oct-01

Qemu is a full system emulator. See

http://www.nongnu.org/qemu/

Limitations & comments

Supports the "-M mips" configuration of qemu: serial,NE2000,IDE.
Supports little and big endian as well as 32 bit and 64 bit.
Derived from au1x00 with a lot of things cut out.

Supports emulated flash (patch Jean-Christophe PLAGNIOL-VILLARD) with
recent qemu versions. When using emulated flash, launch with
-pflash <filename> and erase mips_bios.bin.

Notes for the Qemu MIPS port

I) Example usage:

Using u-boot.bin as ROM (replaces Qemu monitor):

32 bit, big endian:
make qemu_mips
qemu-system-mips -M mips -bios u-boot.bin -nographic

32 bit, little endian:
make qemu_mipsel
qemu-system-mipsel -M mips -bios u-boot.bin -nographic

64 bit, big endian:
make qemu_mips64
qemu-system-mips64 -cpu MIPS64R2-generic -M mips -bios u-boot.bin -nographic

64 bit, little endian:
make qemu_mips64el
qemu-system-mips64el -cpu MIPS64R2-generic -M mips -bios u-boot.bin -nographic

or using u-boot.bin from emulated flash:

if you use a qemu version after commit 4224

create image:
dd of=flash bs=1k count=4k if=/dev/zero
dd of=flash bs=1k conv=notrunc if=u-boot.bin
start it (see above):
qemu-system-mips[64][el] [-cpu MIPS64R2-generic] -M mips -pflash flash -nographic

2) Download kernel + initrd

On ftp://ftp.denx.de/pub/contrib/Jean-Christophe_Plagniol-Villard/qemu_mips/
you can downland

#config to build the kernel
qemu_mips_defconfig
#patch to fix mips interrupt init on 2.6.24.y kernel
qemu_mips_kernel.patch
initrd.gz
vmlinux
vmlinux.bin
System.map

4) Generate uImage

tools/mkimage -A mips -O linux -T kernel -C gzip -a 0x80010000 -e 0x80245650 -n "Linux 2.6.24.y" -d

```
vmlinux.bin.gz uImage

5) Copy uImage to Flash
# dd if=uImage bs=1k conv=notrunc seek=224 of=flash

6) Generate Ide Disk

# dd of=ide bs=1k cout=100k if=/dev/zero

# sfdisk -C 261 -d ide
# partition table of ide
unit: sectors

     ide1 : start=       63, size=    32067, Id=83
     ide2 : start=    32130, size=    32130, Id=83
     ide3 : start=    64260, size=  4128705, Id=83
     ide4 : start=        0, size=        0, Id= 0

7) Copy to ide

# dd if=uImage bs=512 conv=notrunc seek=63 of=ide

8) Generate ext2 on part 2 on Copy uImage and initrd.gz

# Attached as loop device ide offset = 32130 * 512
# losetup -o 16450560 -f ide
# Format as ext2 ( arg2 : nb blocks)
# mke2fs /dev/loop0 16065
# losetup -d /dev/loop0
# Mount and copy uImage and initrd.gz to it
# mount -o loop,offset=16450560 -t ext2 ide /mnt
# mkdir /mnt/boot
# cp {initrd.gz,uImage} /mnt/boot/
# Umount it
# umount /mnt

9) Set Environment

setenv rd_start 0x80800000
setenv rd_size 2663940
setenv kernel BFC38000
setenv oad_addr 80500000
setenv load_addr2 80F00000
setenv kernel_flash BFC38000
setenv load_addr_hello 80200000
setenv bootargs 'root=/dev/ram0 init=/bin/sh'
setenv load_rd_ext2 'ide res; ext2load ide 0:2 ${rd_start} /boot/initrd.gz'
setenv load_rd_tftp 'tftp ${rd_start} /initrd.gz'
setenv load_kernel_hda 'ide res; diskboot ${load_addr} 0:2'
setenv load_kernel_ext2 'ide res; ext2load ide 0:2 ${load_addr} /boot/uImage'
setenv load_kernel_tftp 'tftp ${load_addr} /qemu_mips/uImage'
setenv boot_ext2_ext2 'run load_rd_ext2; run load_kernel_ext2; run addmisc; bootm ${load_addr}'
setenv boot_ext2_flash 'run load_rd_ext2; run addmisc; bootm ${kernel_flash}'
setenv boot_ext2_hda 'run load_rd_ext2; run load_kernel_hda; run addmisc; bootm ${load_addr}'
setenv boot_ext2_tftp 'run load_rd_ext2; run load_kernel_tftp; run addmisc; bootm ${load_addr}'
setenv boot_tftp_hda 'run load_rd_tftp; run load_kernel_hda; run addmisc; bootm ${load_addr}'
setenv boot_tftp_ext2 'run load_rd_tftp; run load_kernel_ext2; run addmisc; bootm ${load_addr}'
setenv boot_tftp_flash 'run load_rd_tftp; run addmisc; bootm ${kernel_flash}'
setenv boot_tftp_tftp 'run load_rd_tftp; run load_kernel_tftp; run addmisc; bootm ${load_addr}'
setenv load_hello_tftp 'tftp ${load_addr_hello} /examples/hello_world.bin'
setenv go_tftp 'run load_hello_tftp; go ${load_addr_hello}'
setenv addmisc 'setenv bootargs ${bootargs} console=ttyS0,${baudrate} rd_start=${rd_start} rd_size=
${rd_size} ethaddr=${ethaddr}'
setenv bootcmd 'run boot_tftp_flash'

10) Now you can boot from flash, ide, ide+ext2 and tfp
```

```
# qemu-system-mips -M mips -pflash flash -monitor null -nographic -net nic -net user -tftp `pwd` -hda ide

II) How to debug U-Boot

In order to debug U-Boot you need to start qemu with gdb server support (-s)
and waiting the connection to start the CPU (-S)

# qemu-system-mips -S -s -M mips -pflash flash -monitor null -nographic -net nic -net user -tftp `pwd` -
hda ide

in an other console you start gdb

1) Debugging of U-Boot Before Relocation

Before relocation, the addresses in the ELF file can be used without any problems
by connecting to the gdb server localhost:1234

# mipsel-unknown-linux-gnu-gdb u-boot
GNU gdb 6.6
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "--host=i486-linux-gnu --target=mipsel-unknown-linux-gnu"...
(gdb)  target remote localhost:1234
Remote debugging using localhost:1234
_start () at start.S:64
64              RVECENT(reset,0)        /* U-boot entry point */
Current language:  auto; currently asm
(gdb)  b board.c:289
Breakpoint 1 at 0xbfc00cc8: file board.c, line 289.
(gdb) c
Continuing.

Breakpoint 1, board_init_f (bootflag=<value optimized out>) at board.c:290
290             relocate_code (addr_sp, id, addr);
Current language:  auto; currently c
(gdb) p/x addr
$1 = 0x87fa0000

2) Debugging of U-Boot After Relocation

For debugging U-Boot after relocation we need to know the address to which
U-Boot relocates itself to 0x87fa0000 by default.
And replace the symbol table to this offset.

(gdb) symbol-file
Discard symbol table from `/private/u-boot-arm/u-boot'? (y or n) y
Error in re-setting breakpoint 1:
No symbol table is loaded.  Use the "file" command.
No symbol file now.
(gdb) add-symbol-file u-boot 0x87fa0000
add symbol table from file "u-boot" at
        .text_addr = 0x87fa0000
(y or n) y
Reading symbols from /private/u-boot-arm/u-boot...done.
Breakpoint 1 at 0x87fa0cc8: file board.c, line 289.
(gdb) c
Continuing.

Program received signal SIGINT, Interrupt.
0xffffffff87fa0de4 in udelay (usec=<value optimized out>) at time.c:78
78              while ((tmo - read_c0_count()) < 0x7fffffff)
```