binutils support for Xtensa

- git tree: https://github.com/jcmvbkbc/binutils-gdb-xtensa

FDPIC support

- [+] static linking
- [+] PDE
- [+] PIE
- [±] PLT and lazy binding
- [-] TLS

FDPIC instruction sequences

Local call

Default local call

```
+0:     movi    tmp, target@GOT
+3:     add     tmp, tmp, localGOTptr
+5:     l32i    tmp, tmp, 0
+7:     mov     GOTptr, localGOTptr
+9:     callx0  tmp
```

No-GOT local call

```
+0:     movi    tmp1, target
+3:     l32i    tmp2, localGOTptr, TEXT_SEGMENT_OFFSET
+5:     add     tmp1, tmp1, tmp2
+7:     mov     GOTptr, localGOTptr
+9:     callx0  tmp1
```

When call0 reaches the target it can be transformed to

```
+0:     mov     GOTptr, localGOTptr
+2:     call0   target
```

In the j.l style it could probably be done like

```
        call0.l target, tmp1, tmp2, localGOTptr
```

PLT call

Obvious version:

```
+0:     movi    tmp, target@PLTGOT
+3:     add     tmp, tmp, localGOTptr
+5:     l32i    tmp, tmp, 0
+7:     l32i    GOTptr, tmp, 4
+9:     l32i    tmp, tmp, 0
+11:    callx0  tmp
```

```
target@PLT:
+0:     movi    a8, target@PLTGOT
+3:     add     a8, a8, GOTptr
+5:     movi    a9, target-symbol
+8:     l32i    a10, GOTptr, RESOLVER_FN
+10:    l32i    GOTptr, GOTptr, RESOLVER_GOT
+12:    jx0     a10
```

The inline part calls the PLT part only once, after resolution the inline part calls the target directly. The adjustment is done to a single GOT entry, so it's atomic. The inline part can be reduced to a fixed direct call to the PLT:

```
+0:     mov     GOTptr, localGOTptr
+2:     call0   target@PLT
```

that reduces the inline part from 14 to 5 bytes, but adds two jumps to each call and some special logic to the resolver to avoid name resolution on each call.

TLS General Dynamic

```
+0:     movi    a2, x@GOTTLSDESC
+3:     add     a2, a2, localGOTptr
+5:     l32i    tmp, a2, 0
+7:     l32i    GOTptr, tmp, 4
+9:     l32i    tmp, tmp, 0
+11:    callx0  tmp
```

This TLSDESC is not the same as the descriptor of the default xtensa toolchain. It contains two pointers, one to the resolver function, the other to that other descriptor containing DTPOFF and module index in the dtv.

Manual toolchain building script

```
#! /bin/bash -ex

target=${TARGET:-xtensa-linux-uclibcfdpic}
build_base=`pwd`/build
src_base=$(dirname $(readlink -f "$0"))
binutils_src=$HOME/ws/tensilica/binutils-gdb/binutils-gdb
gcc_src=$HOME/ws/tensilica/gcc/gcc
```

```
linux_src="$src_base/linux"
uclibc_src="$src_base/uclibc-ng"
uclibc_config_src="$src_base/uclibc-ng-config"

prefix=`pwd`
sysroot="$prefix/$target/sysroot"
linux_headers="$sysroot/usr"

_FLAGS_FOR_HOST=${FLAGS_FOR_HOST:--Og -g}
_FLAGS_FOR_TARGET=${FLAGS_FOR_TARGET:--mauto-litpools -mfdpic -Oz -g}
CROSS_COMPILE=${CROSS_COMPILE:-$prefix/bin/$target-}
TARGET_CFLAGS="$_FLAGS_FOR_TARGET -D_LARGEFILE64_SOURCE -
D_FILE_OFFSET_BITS=64"

if [ "$1" = "-r" ]; then
        reconfigure=1
fi

mkdir -p .build

mkdir .build/binutils && (
        cd .build/binutils
        "$binutils_src/configure" --prefix="$prefix" \
                --target=$target \
                --with-sysroot="$sysroot" \
                --disable-shared --disable-werror --disable-gdb --disable-
gdbstub \
                CFLAGS="$_FLAGS_FOR_HOST"

        make -j8
        make -j8 install
)

mkdir .build/gcc-initial && (
        cd .build/gcc-initial
        "$gcc_src/configure" --prefix="$prefix" \
                --target=$target \
                --with-sysroot="$sysroot" \
                --enable-languages=c \
                --disable-shared \
                --enable-__cxa_atexit \
                --disable-tls --disable-threads \
                --without-headers --with-newlib \
                CFLAGS_FOR_TARGET="$_FLAGS_FOR_TARGET" \
                CXXFLAGS_FOR_TARGET="$_FLAGS_FOR_TARGET" \
                CFLAGS="$_FLAGS_FOR_HOST" \
                CXXFLAGS="$_FLAGS_FOR_HOST"

        make -j8 all-gcc
        make -j8 all-target-libgcc
        make -j8 install-gcc
```

```
        make -j8 install-target-libgcc
)

mkdir .build/linux && (
        cd .build/linux
        make -C "$linux_src" ARCH=xtensa \
                CROSS_COMPILE="$CROSS_COMPILE" O=`pwd` \
                defconfig
        make -C "$linux_src" ARCH=xtensa \
                CROSS_COMPILE="$CROSS_COMPILE" O=`pwd` \
                INSTALL_HDR_PATH="$linux_headers" \
                -j8 headers_install
)

mkdir .build/uclibc && (
        cd .build/uclibc
        cp "$uclibc_config_src/.config" .
        if [ -n "$reconfigure" ]; then
                make -C "$uclibc_src" ARCH=xtensa \
                        CROSS_COMPILE="$CROSS_COMPILE" \
                        O=`pwd` KERNEL_HEADERS="$linux_headers/include" \
                        UCLIBC_EXTRA_CFLAGS="${TARGET_CFLAGS}" \
                        menuconfig
                cp .config "$uclibc_config_src"
        fi

        make -C "$uclibc_src" ARCH=xtensa \
                CROSS_COMPILE="$CROSS_COMPILE" \
                O=`pwd` KERNEL_HEADERS="$linux_headers/include" \
                UCLIBC_EXTRA_CFLAGS="${TARGET_CFLAGS}" \
                -j8 "$@"
        make -C "$uclibc_src" ARCH=xtensa \
                CROSS_COMPILE="$CROSS_COMPILE" \
                O=`pwd` KERNEL_HEADERS="$linux_headers/include" \
                UCLIBC_EXTRA_CFLAGS="${TARGET_CFLAGS}" \
                DESTDIR="$sysroot" \
                install
)

mkdir .build/gcc-final && (
        cd .build/gcc-final
        "$gcc_src/configure" --prefix="$prefix" \
                --target=$target \
                --with-sysroot="$sysroot" \
                --enable-languages=c,c++ \
                --disable-shared \
                --enable-__cxa_atexit \
                --disable-tls --disable-threads \
                --with-uclibc \
                CFLAGS_FOR_TARGET="$_FLAGS_FOR_TARGET" \
```

```
                CXXFLAGS_FOR_TARGET="$_FLAGS_FOR_TARGET" \
                CFLAGS="$_FLAGS_FOR_HOST" \
                CXXFLAGS="$_FLAGS_FOR_HOST"

        make -j8 all
        make -j8 install
)
```