

Retargeting GCC to ArchC models

Mini-Howto
August 21, 2003

ArchC Project
LSC - UNICAMP
<http://www.archc.org>

1 Introduction

This mini-howto aims to facilitate the configuration of the GNU Compiler Collection (GCC) for ArchC use. This method tries to minimize the changes on a working GCC installation. Only one file is added to the original GCC tree, while other support files reside in a separate tree.

After configuration, to compile code for ArchC, it's just necessary to add the command line argument `-specs=archc` to the `gcc` command.

The next section explains how to install GCC and the support tools Binutils and Newlib. The GNU Debugger (GDB) is also presented as an optional package to be installed. If you have these tools already installed go directly to section 3, where the GCC configuration is documented. The last section indicates how to recompile the ArchC System Call Library.

2 Installing GCC and support tools

The next steps install the tools necessary to cross-compile programs with GCC. They are the GNU Compiler proper, the Newlib, a implementation of the C library for embedded systems, and the Binutils, the package which contains the assembler, the linker and other useful tools. The debugger tool, GDB, is optional. The commands assume the tools will be installed in the `/1/archc/compilers` directory (to install in another location, change the commands accordingly). If there is any question about cross-compilation not covered in this section, the CrossGCC Frequently Asked Questions may be useful (<http://www.objsw.com/CrossGCC/>).

GCC Homepage: <http://gcc.gnu.org>

Main FTP host: <ftp://gcc.gnu.org/pub/gcc>

Mirrors page: <http://gcc.gnu.org/mirrors.html>

Binutils Homepage: <http://sources.redhat.com/binutils/>

FTP: <ftp://ftp.gnu.org/pub/gnu/binutils>

Newlib Homepage: <http://sources.redhat.com/newlib/>

FTP: <ftp://sources.redhat.com/pub/newlib/index.html>

GDB Homepage: <http://www.gnu.org/software/gdb>

FTP: <ftp://sources.redhat.com/pub/gdb/releases>

1. Get recent GCC, Binutils, Newlib and GDB packages:

```
ftp://gcc.gnu.org/pub/gcc/releases/gcc-3.3.1/gcc-3.3.1.tar.bz2
ftp://ftp.gnu.org/pub/gnu/binutils/binutils-2.14.tar.bz2
ftp://sources.redhat.com/pub/newlib/newlib-1.11.0.tar.gz
ftp://sources.redhat.com/pub/gdb/releases/gdb-5.3.tar.bz2
```

2. Decompress all on default directory:

```
cd /l/archc/compilers
tar -xvjf gcc-3.3.1.tar.bz2
tar -xvjf binutils-2.14.tar.bz2
tar -xvzf newlib-1.11.0.tar.gz
tar -xvjf gdb-5.3.tar.bz2
```

3. Set the shell environment variable `$ARCH` to the target system (sparc, mips, etc). The line below uses mips as an example.

```
ARCH=mips      (in sh and bash shells)
set ARCH=mips  (in csh shell)
```

4. Configure and install binutils:

```
mkdir build-$ARCH-binutils; cd build-$ARCH-binutils
../binutils-2.14/configure --prefix=/l/archc/compilers/$ARCH --target=$ARCH-elf
make
make install
cd ..
```

5. Configure and install gcc:

```
PATH=$PATH:$PWD/$ARCH/bin
ln -s $PWD/newlib-1.11.0/newlib/ $PWD/gcc-3.3.1/newlib
mkdir build-$ARCH-gcc-3.3.1; cd build-$ARCH-gcc-3.3.1
../gcc-3.3.1/configure --prefix=/l/archc/compilers/$ARCH --target=$ARCH-elf \
    --with-gnu-as --with-gnu-ld --with-newlib --enable-languages=c++,c
make
make install
cd ..
```

6. Configure and install newlib.

NOTE: If you want support for long long (%lld) formatting to newlib's printf and similar functions, change the line
newlib_cflags=
to
newlib_cflags="-DWANT_PRINTF_LONG_LONG"
in the file newlib-1.11.0/newlib/configure.host

```
mkdir build-$ARCH-newlib-1.11.0; cd build-$ARCH-newlib-1.11.0
../newlib-1.11.0/configure --prefix=/l/archc/compilers/$ARCH --target=$ARCH-elf
make
make install
cd ..
```

7. Configure and install gdb:

```
mkdir build-$ARCH-gdb-5.3; cd build-$ARCH-gdb-5.3
../gdb-5.3/configure --prefix=/l/archc/compilers/$ARCH --target=$ARCH-elf
make
make install
cd ..
```

8. Temporary directories can be safely removed

```
rm -rf build-$ARCH-*
```

9. Put the dir /l/archc/compilers/\$ARCH/bin in the PATH (change \$ARCH to match architecture!!!). For system wide use, in RedHat9, append this line in /etc/profile.d/cross-compilers.sh (scripts in this directory get executed for all users logins). Consult your operating system manuals to change the PATH system wide for other operating systems.

```
PATH=$PATH:/l/archc/compilers/$ARCH/bin
```

3 Configuring GCC for ArchC models

After the cross-GCC installation for the architecture simulated by ArchC, follow the steps below to configure this installation to create a ArchC compatible binary file.

1. Set the system variable \$AC_ARCH to the target system (sparc, mips, etc). The line below uses mips as an example.

```
AC_ARCH=mips1          (in sh and bash shells)
```

```
set AC_ARCH=mips1 (in csh shell)
```

2. Create directory for the auxiliary files and change to this directory (the recommended is `/l/archc/compilers/ac_specs/$AC_ARCH`).

```
cd /l/archc/compilers
```

```
mkdir -p ac_specs/$AC_ARCH; cd ac_specs/$AC_ARCH
```

3. Create the file `ac_specs` as in Figure 1. Change `$AC_ARCH` explicitly to match architecture!

```
*link:
-L/l/archc/compilers/ac_specs/$AC_ARCH -Tac_link.lds

*startfile:

*endfile:

*lib:
-lc -lac_sysc
```

Figure 1: Spec file for ArchC

4. Link this file in the GCC specs directory with the name `archc`

```
ln -s /l/archc/compilers/ac_specs/$AC_ARCH/ac_specs \
    /l/archc/compilers/$ARCH/lib/gcc-lib/$ARCH-elf/3.3.1/archc
```

5. Create a `ac_link.lds` file based on the basic linker script in `/l/archc/compilers/$ARCH/$ARCH-elf/lib/ldscripts/` (the one with extension `.x`). The changes necessary:

- Change `STARTUP` directive (if it doesn't exist create one in the first line):
`STARTUP(ac_start.o)`
- Move the section block `.text` to the first position in the `SECTIONS` block
- Remove absolute address manipulations inside the `SECTIONS` block (the lines which attribute a hex number to the `“.”` (a dot) variable)
- Remove the `SEARCH_DIR` directive, for the multi-lib functionality to work (ex: `-msoft-float` links with the appropriate library from another directory)

6. Create `ac_start.s` based on the GNU Assembler for MIPS example in Figure 2. (ArchC models are using 5MB of memory in the time this document was written)

```
.text
.align 2
.globl _start
.ent _start
_start:
    ;;
    ;;; Put architecture assembler here
    ;;
    ;;; 1. Initialize stack pointer and other registers having in mind 5MB
    ;;; of procesor memory. Note that the last 1KB of memory are reserved
    ;;; for ArchC to pass command line arguments

    li    $sp,5242880                # 0x500000
    addi  $sp,$sp,-1024
    la    $gp,_gp

    ;;; 2. Call function "main"

    jal   main
    nop

    ;;; 3. Call system function "_exit"

    jal   _exit
    nop

    ;;; End of function _start
    .end  _start

    ;;; Align to address 0x40
    .align 7
```

Figure 2: Example for file `start.s`

7. Compile the `ac_start.s` file:

```
$ARCH-elf-gcc -c ac_start.s
```

4 Recompile the ArchC System Calls Library

The last task in configuring GCC for ArchC is to recompile the ArchC System Calls Library. The last line in `ac_specs` indicates a link to this library by `-lac_sysc`. Get the source code for the library in <http://www.archc.org>. Expand it in a directory (recomended `/1/archc/compilers/ac_specs/libac_sysc`) and compile.

```
cd /l/archc/compilers/ac_specs
tar -xvzf libac_sysc.tar.gz
cd libac_sysc
make TARGET="$ARCH-elf"
cp libac_sysc.a ../$AC_ARCH
```

NOTE: Makefile variable ARCH_CFLAGS may need to be changed also.