How to generate Verilog testvector from ARM assembly code

To verify your CPU design, you need to write testvectors. For that, you should write some test programs and use those programs as testvectors. You will write the test programs in ARM assembly language. The test programs are stored in Verilog memory model.

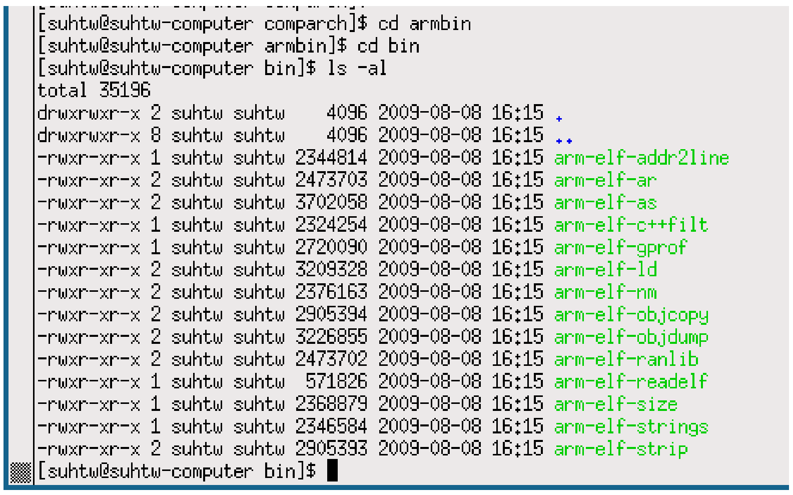
ARM assembler and linker convert your assembly programs into machine codes. Thus, you should have ARM assembler and linker installed first.

This note explains how to install ARM assembler and linker using GNU bin utilities under Linux and how to generate Verilog testvector from ARM assembly code

There are 2 options to install ARM assembler and linker: The first option is to download the GNU binutils and build the ARM assembler and linker. The second option is to simply download the ARM assembler and linker pre-bulit under x86 (**32-bit Fedora**).

Option 1: Download binutils and build the ARM assembler and linker

1. Download the latest GNU bin utility, binutils-2.191.tar.bz2, either from http://ftp.gnu.org/gnu/binutils/ or from the class website at http://comedu.korea.ac.kr/~suhtw/teaching/comp212_CA/binutils-2.19.1.tar.bz2
2. Create a directory where the downloaded file is copied
   - "mkdir classes"
   - "cd classes"
   - "mkdir comparch"
   - "cd comparch"
   - "mkdir armbin"   // this is a directory where compiled bin utilities are located
   - "cp ~/Downloads/binutils-2.19.1.tar.bz2 ."
3. Untar (Uncompress) the file
   - "tar jxvf binutils-2.19.1.tar.bz2 "
4. Move to the bin utility directory
   - "cd binutils-2.19.1"
5. Set environment for compilation
   - "export HOST=i686-linux"
   - "export TARGET=arm-elf"
   - "export binLocal=~/classes/comparch/armbin"
6. Check if your environment set correctly
   - "set"   // scroll up and down to check HOST, TARGET, and binLocal
7. Compile the bin utilities
   - "./configure --host=$HOST --target=$TARGET --prefix=$binLocal   --exec-prefix=$binLocal"
   - "make"
   - "make install"
8. Check if the bin utilities are located in ~/classes/comparch/armbin"
   - "cd ~/classes/comparch/armbin"
   - "ls -al"   // You should be able to see the following files. The ARM assembler and linker are "arm-elf-as" and "arm-elf-ld"



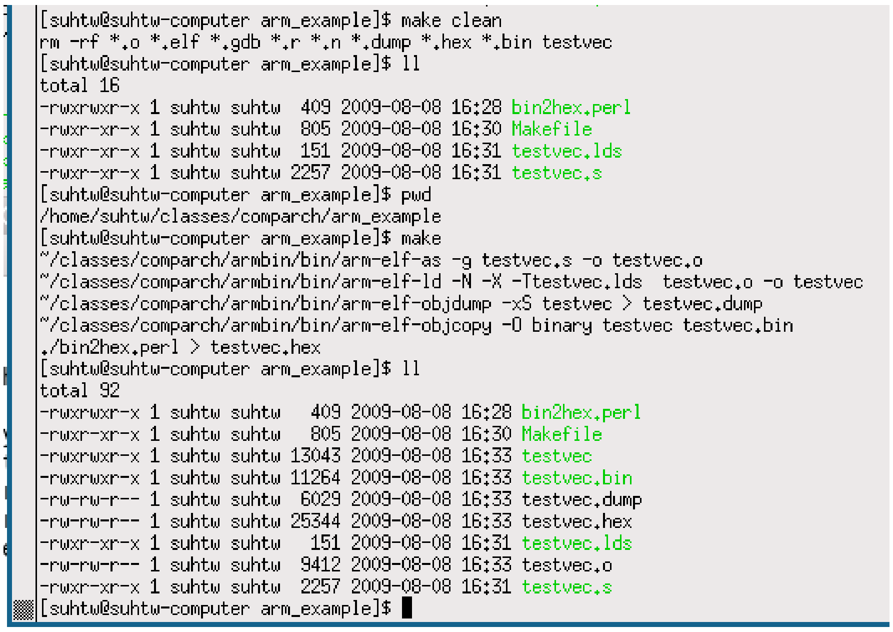
```
[suhtw@suhtw-computer comparch]$ cd armbin
[suhtw@suhtw-computer armbin]$ cd bin
[suhtw@suhtw-computer bin]$ ls -al
total 35196
drwxrwxr-x 2 suhtw suhtw    4096 2009-08-08 16:15 .
drwxrwxr-x 8 suhtw suhtw    4096 2009-08-08 16:15 ..
-rwxr-xr-x 1 suhtw suhtw 2344814 2009-08-08 16:15 arm-elf-addr2line
-rwxr-xr-x 2 suhtw suhtw 2473703 2009-08-08 16:15 arm-elf-ar
-rwxr-xr-x 2 suhtw suhtw 3702058 2009-08-08 16:15 arm-elf-as
-rwxr-xr-x 1 suhtw suhtw 2324254 2009-08-08 16:15 arm-elf-c++filt
-rwxr-xr-x 1 suhtw suhtw 2720090 2009-08-08 16:15 arm-elf-gprof
-rwxr-xr-x 2 suhtw suhtw 3209328 2009-08-08 16:15 arm-elf-ld
-rwxr-xr-x 2 suhtw suhtw 2376163 2009-08-08 16:15 arm-elf-nm
-rwxr-xr-x 2 suhtw suhtw 2905394 2009-08-08 16:15 arm-elf-objcopy
-rwxr-xr-x 2 suhtw suhtw 3226855 2009-08-08 16:15 arm-elf-objdump
-rwxr-xr-x 2 suhtw suhtw 2473702 2009-08-08 16:15 arm-elf-ranlib
-rwxr-xr-x 1 suhtw suhtw  571826 2009-08-08 16:15 arm-elf-readelf
-rwxr-xr-x 1 suhtw suhtw 2368879 2009-08-08 16:15 arm-elf-size
-rwxr-xr-x 1 suhtw suhtw 2346584 2009-08-08 16:15 arm-elf-strings
-rwxr-xr-x 2 suhtw suhtw 2905393 2009-08-08 16:15 arm-elf-strip
[suhtw@suhtw-computer bin]$ █
```

Option 2: Simply download pre-built (under x86 with 32-bit Fedora) ARM assembler and linker

1. Download the pre-built (under x86 with 32-bit Fedora) ARM assembler and linker from http://comedu.korea.ac.kr/~suhtw/Research/ARM/armbin.tar.bz2
2. Create a directory where the downloaded file is copied
   - "mkdir classes"
   - "cd classes"
   - "mkdir comparch"
   - "cd comparch"
3. Copy the download file
   - "cp ~/Download/armbin.tar.bz2 ."
4. Untar (Uncompress) it
   - "tar jxvf armbin.tar.bz2"
5. Change the directory and check if you have necessary files as shown in the above figure.
   - "cd armbin/bin"
   - "ls -al" // The ARM assembler and linker are "arm-elf-as" and "arm-elf-ld"

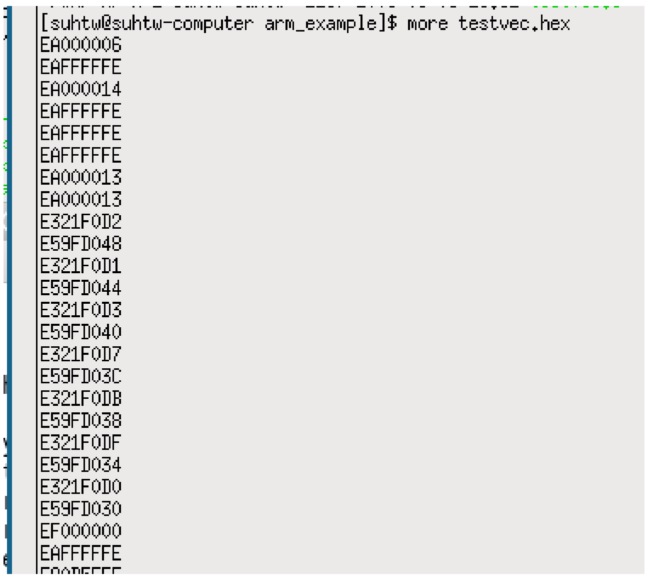
Download example assembly code and generate testvector

1. Download an example from
   http://comedu.korea.ac.kr/~suhtw/Research/ARM/arm_example.tar.bz2
2. Untar (uncompress) it
   - "cd ~/classes/comparch/"
   - "cp ~/Download/arm_example.tar.bz2 ."
   - "tar jxvf arm_example.tar.bz2"
3. Generate a Verilog testvector
   - "cd arm_example"
   - "more testvec.s" // Check out what kind of assembly program you are assembling
   - "make" // you should be able to see the following files



```
[suhtw@suhtw-computer arm_example]$ make clean
rm -rf *.o *.elf *.gdb *.r *.n *.dump *.hex *.bin testvec
[suhtw@suhtw-computer arm_example]$ ll
total 16
-rwxrwxr-x 1 suhtw suhtw  409 2009-08-08 16:28 bin2hex.perl
-rwxr-xr-x 1 suhtw suhtw  805 2009-08-08 16:30 Makefile
-rwxr-xr-x 1 suhtw suhtw  151 2009-08-08 16:31 testvec.lds
-rwxr-xr-x 1 suhtw suhtw 2257 2009-08-08 16:31 testvec.s
[suhtw@suhtw-computer arm_example]$ pwd
/home/suhtw/classes/comparch/arm_example
[suhtw@suhtw-computer arm_example]$ make
~/classes/comparch/armbin/bin/arm-elf-as -g testvec.s -o testvec.o
~/classes/comparch/armbin/bin/arm-elf-ld -N -X -Ttestvec.lds  testvec.o -o testvec
~/classes/comparch/armbin/bin/arm-elf-objdump -xS testvec > testvec.dump
~/classes/comparch/armbin/bin/arm-elf-objcopy -O binary testvec testvec.bin
./bin2hex.perl > testvec.hex
[suhtw@suhtw-computer arm_example]$ ll
total 92
-rwxrwxr-x 1 suhtw suhtw   409 2009-08-08 16:28 bin2hex.perl
-rwxr-xr-x 1 suhtw suhtw   805 2009-08-08 16:30 Makefile
-rwxrwxr-x 1 suhtw suhtw 13043 2009-08-08 16:33 testvec
-rwxrwxr-x 1 suhtw suhtw 11264 2009-08-08 16:33 testvec.bin
-rw-rw-r-- 1 suhtw suhtw  6029 2009-08-08 16:33 testvec.dump
-rw-rw-r-- 1 suhtw suhtw 25344 2009-08-08 16:33 testvec.hex
-rwxr-xr-x 1 suhtw suhtw   151 2009-08-08 16:31 testvec.lds
-rw-rw-r-- 1 suhtw suhtw  9412 2009-08-08 16:33 testvec.o
-rwxr-xr-x 1 suhtw suhtw  2257 2009-08-08 16:31 testvec.s
[suhtw@suhtw-computer arm_example]$
```

   - Check out "testvec.hex" // You can use testvec.hex as a Verilog testvector for ARM



```
[suhtw@suhtw-computer arm_example]$ more testvec.hex
EA000006
EAFFFFFE
EA000014
EAFFFFFE
EAFFFFFE
EAFFFFFE
EA000013
EA000013
E321F0D2
E59FD048
E321F0D1
E59FD044
E321F0D3
E59FD040
E321F0D7
E59FD03C
E321F0DB
E59FD038
E321F0DF
E59FD034
E321F0D0
E59FD030
EF000000
EAFFFFFE
```