ARM Assembly Programming by Using GAS

Peng-Sheng Chen

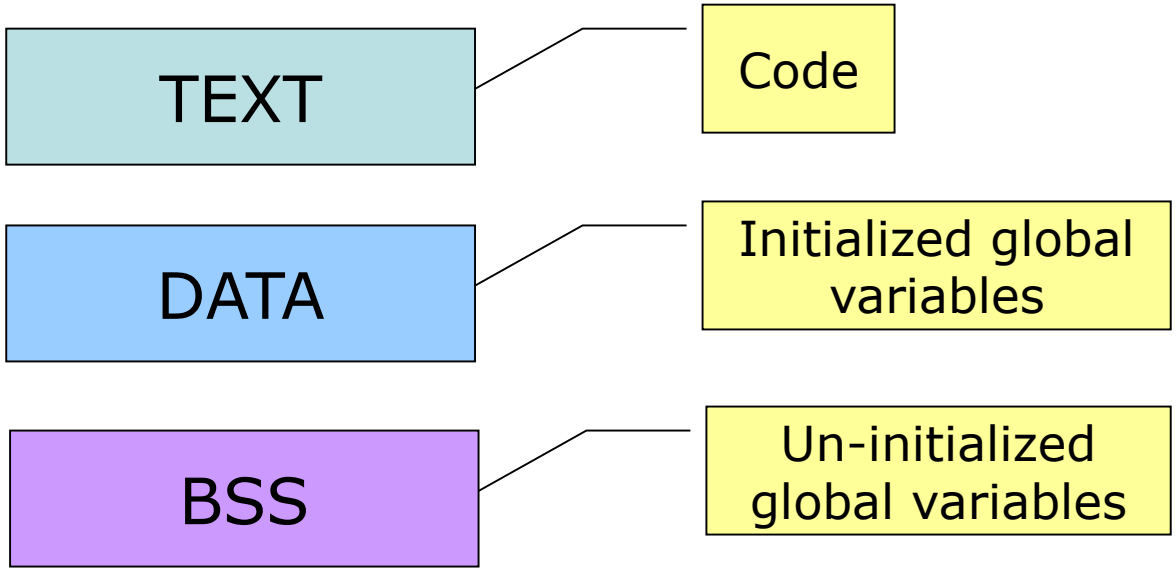
Fall, 2017

Outline

- Object File Format
- GDB & ARM Emulator
- Run Your Program

Object File Format

- COFF (Common Object File Format)
- ELF (Extended Linker Format)
- The segments (sections) in the object file



Basic Format (1)

```
        .section .text
        .global main
        .type main,%function
main:
         MOV r0, #100
         ADD r0, r0, r0
        .end
```

Filename: test.s

- Assemble the following code into a section
- Similar to “AREA” in armasm

http://sourceware.org/binutils/docs-2.23/as/index.html

Basic Format (2)

```
        .section .text
        .global main
        .type main,%function
main:
         MOV r0, #100
         ADD r0, r0, r0
        .end
```

Filename: test.s

- ".global" makes the symbol visible to ld
- Similar to "EXPORT" in armasm

Basic Format (3)

```
        .section .text
        .global main
        .type main,%function
main:
         MOV r0, #100
         ADD r0, r0, r0
        .end
```

Filename: test.s

- This sets the type of symbol name to be either a function symbol or an object symbol

- ".end" marks the end of the assembly file
- Assembler does not process anything in the file past the ".end" directive

Basic Format (4)

```
        .section .text
        .global main
        .type main,%function
main:
         MOV r0, #100
         ADD r0, r0, r0
        .end
```

Filename: test.s

- LABEL透過 ”:” 來做識別
- armasm則是透過指令和保留字的縮排來做識別

- Comments
  - ***/*** …your comments... ****/***
  - @ your comments (line comment)

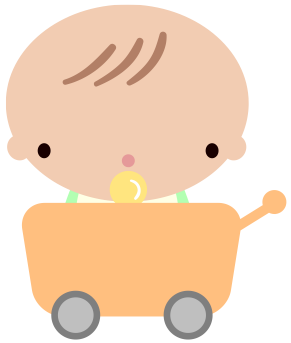
How to Run a ARM program?

ARM assembly program

ARM object code

ARM executable code

Run ARM program



GCC cross-compiler & binutils

GDB & ARM emulator

GDB & ARM Emulator

GDB & ARM Emulator

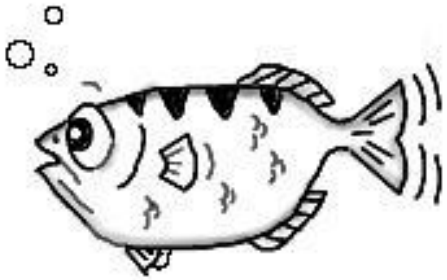
- GNU Debugger, GDB
- GUI GDB
- Cross-debugger & ARM Emulator

GNU Project Debugger (GDB)

- Allows you to see what is going on `**inside**' another program while it executes
  - Start your program, specifying anything that might affect its behavior
  - Make your program stop on specified conditions
  - Examine what has happened, when your program has stopped
  - Change things in your program, so you can experiment with correcting the effects of one bug and go on to learn about another

GDB

- GDB 8.01 was released at September 7th, 2017
- Debug programs written in
  - C, C++, Fortran, Ada …… **OK**
  - Modula-2, Pascal ……Partial
- Official website
  - http://sourceware.org/gdb/



Running Programs under GDB

- Compiling for debugging
  - Add option "**-g**"
  - Add option "**-gdwarf-2**" or "**-g3**" for macro information

Starting GDB

- gdb hello.exe

```
~/test

pschen@pelican ~/test
$ gdb
GNU gdb 6.3.50_2004-12-28-cvs (cygwin-special)
Copyright 2004 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 "i686-pc-cygwin".
(gdb)
```

Basic Commands (1)

- target exec *executable-filename*
- run *command-line-arguments*
  - Start your program under GDB
  - run test1.exe < foo.txt
- break *place*
  - *break FUNCTION*
    - Set a breakpoint at entry to function *FUNCTION*
  - *break LINENUM*
    - Set a breakpoint at line *LINENUM* in the current source file
  - *break FILENAME:LINENUM*
    - Set a breakpoint at line *LINENUM* in source file *FILENAME*
  - …

Basic Commands (2)

- file *executable-filename*
  - Load symbol information
- step [COUNT]
  - Continue running your program until control reaches a different source line, then stop it and return control to GDB
- next [COUNT]
  - Continue to the next source line in the current (innermost) stack frame. This is similar to `step', but function calls that appear within the line of code are executed without stopping
- print [EXP]

For the beginner, it is hard to use

Graphic User Interface to GDB

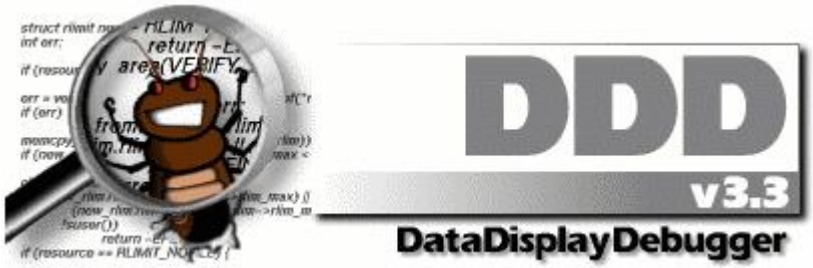
- It makes GDB easier to use
  - DDD
  - gdbgui (browser-based)
  - Insight
  - Code Medic
  - Others …

Graphic User Interface

GDB
(command-line mode)

Data Display Debugger (DDD)

- A graphical front-end for command-line debuggers
  - GDB, DBX, WDB, Ladebug, JDB, XDB, the Perl debugger, the bash debugger, the Python debugger
- GNU project
- http://www.gnu.org/software/ddd/
- Requirement
  - **X-windows**
  - Motif-like library



DDD

DDD: /public/source/programming/ddd-3.2/ddd/cxxtest.C

File Edit View Program Commands Status Source Data Help

(): list->self

Lookup Find« Break Watch Print Disp* Plot Hide Rotate Set Undisp

1: list
(List *) 0x804df80

*()

value = 85
self = 0x804df80
next = 0x804df90

value = 86
self = 0x804df90
next = 0x804df a0

self. next

```
    list->next                  = new List(a_global + start++);
    list->next->next            = new List(a_global + start++);
    list->next->next->next      = list;

    (void) list;                // Display this

    delete list;
    delete list->next;
    delete list;
}

// Test
void lis
{
    list
}

//
void ref
{
    date
    dele
    date
}
```

(List *) 0x804df80

Run
Interrupt
Step Stepi
Next Nexti
Until Finish
Kill
own
edo
lake

DDD Tip of the Day #5

If you made a mistake, try **Edit→Undo**. This will undo the most recent debugger command and redisplay the previous program state.

Close Prev Tip Next Tip

```
(gdb) graph display *(list->next->next->self) dependent on 4
(gdb)
```

list = (List *) 0x804df80

Insight

- A graphic user interface to GDB
- Written by Tcl/Tk, since 1994
- From Red Hat and Cygnus Solutions
- http://sources.redhat.com/insight/
- GPL license

Insight

pi1.c - Source Window

File Run View Control Preferences Help

Find:

pi1.c | main | MIXED

```
          35 {
- 0xa01002d8 <main>:                    addiu   sp,sp,-64
- 0xa01002dc <main+4>:                  sd      ra,56(sp)
- 0xa01002e0 <main+8>:                  sd      s8,48(sp)
- 0xa01002e4 <main+12>:                 move    s8,sp
- 0xa01002e8 <main+16>:                 sw      a0,64(s8)
- 0xa01002ec <main+20>:                 sw      a1,72(s8)
          36        int i=0;
■ 0xa01002f0 <main+24>:                 sw      zero,32(s8)
          37        char *endp;
          38        struct captured_main_args args;
          39
          40        args.argc = argc;
- 0xa01002f4 <main+28>:                 lw      v0,64(s8)
- 0xa01002f8 <main+32>:                 sw      v0,40(s8)
          41        args.argv = argv;
- 0xa01002fc <main+36>:                 lw      v0,72(s8)
- 0xa0100300 <main+40>:                 sw      v0,44(s8)
          42
          43        stor[i++] = 0;
- 0xa0100304 <main+44>:                 lw      v1,32(s8)
- 0xa0100308 <main+48>:                 move    v0,v1
- 0xa010030c <main+52>:                 sll     a0,v0,0x2
- 0xa0100310 <main+56>:                 lui     v0,0xa011
- 0xa0100314 <main+60>:                 addiu   v0,v0,-3952
- 0xa0100318 <main+64>:                 addu    v0,a0,v0
- 0xa010031c <main+68>:                 sw      zero,0(v0)
- 0xa0100320 <main+72>:                 addiu   v1,v1,1
- 0xa0100324 <main+76>:                 sw      v1,32(s8)
          44        if (argc == 0)
- 0xa0100328 <main+80>:                 lw      v0,64(s8)
- 0xa010032c <main+84>:                 bnez    v0,0xa0100344 <main+108>
- 0xa0100330 <main+88>:                 nop
          45                        n = 20;
- 0xa0100334 <main+92>:                 li      v0,20
- 0xa0100338 <main+96>:                 sw      v0,-32488(gp)
- 0xa010033c <main+100>:                j       0xa0100358 <main+128>
- 0xa0100340 <main+104>:                nop
          46        else
          47                        n = atoi(argv[1]);
```

Program is running. 0xa01002f0 36

Insight

pi1.c - Source Window

File Run View Control Preferences Help

Find:

pi1.c | main | SOURCE

```
-   69        while (cnt < n)
    70        {
-   71              for (i = 0; ++i <= (int)n - (int)cnt; )
    72                {
-   73                  mf[i] *= 10L;
-   74                  ms[i] *= 10L;
    75              }
-   76              for (i =(int)(n - cnt + 1); --i >= 2; )
    77              {
-   78                    temp = 2 * i - 1;
-   79                    shift(&mf[i - 1], &mf[i], temp - 2, temp * kf);
-   80                    shift(&ms[i - 1], &ms[i], temp - 2, temp * ks);
    81              }
-   82              nd = 0;
-   83              shift((long *)&nd, &mf[1], 1L, 5L);
-   84              shift((long *)&nd, &ms[1], 1L, 239L);
-   85              xprint(nd);
-   86        }
-   87        printf("\n\nCalculations Completed!\n");
-   88        free(ms);
-   89        free(mf);
-   90        return(0);
```

Program stopped at line 79 | 0x804882b | 79

Stack

```
__libc_start_main
main
shift
```

Local Variables

```
  argc =  (int) 2
⊞ argv =  (char **) 0xbffffab4
  i =  (int) 20
⊞ endp =  (char *) 0xbffffa68 "\210Ãº Ã¿Â¿Ã¤X\001B\002"
⊞ args =  (struct captured_main_args) {...}
```

Registers

Register		
eax	0x0	st0
ecx	0xffffffff	st1
edx	0xa0	st2
ebx	0x50	st3
esp	0xbffffa28	st4
ebp	0xbffffa68	st5
esi	0x40012020	st6
edi	0xbffffab4	st7
eip	0x804882b	fctrl
eflags	0x396	fstat
cs	0x23	ftag
ss	0x2b	fiseg
ds	0x2b	fioff
es	0x2b	foseg
fs	0x0	fooff
gs	0x0	fop

Console Window

```
(gdb) b shift
Breakpoint 3 at 0x8048b17: file pi3.c, line 7.

(gdb) c
Continuing.

Breakpoint 3, shift (l1=0x8049ef4, l2=0x8049ef8, lp=37, lmod=975) at pi3.c:7

(gdb) next
No symbol "lmod" in current context.

(gdb)
```

Build the Cross Debugger

Prepare to Build Cross Debugger

- **Cross compiler, binutils and newlib are ready**
- Download insight source package
  - ftp://sources.redhat.com:/pub/insight/releases/insight-6.6.tar.bz2
- Uncompress source package
  - tar –jxvf insight-6.6.tar.bz2

GDB Source Package (1)

- gdb-6.6/configure (and supporting files)
  - script for configuring GDB and all its supporting libraries
- gdb-6.6/gdb
  - the source specific to GDB itself
- gdb-6.6/bfd
  - source for the Binary File Descriptor library
- gdb-6.6/include
  - GNU include files
- gdb-6.6/libiberty
  - Source for the "liberty" free software library
  - It is a collection of subroutines used by various GNU programs.
  - Current members include:
    - getopt -- get options from command line
    - obstack -- stacks of arbitrarily-sized objects
    - strerror -- error message strings corresponding to errno
    - strtol -- string-to-long conversion
    - strtoul -- string-to-unsigned-long conversion

GDB Source Package (2)

- gdb-6.6/opcodes
  - source for the library of opcode tables and disassemblers
- gdb-6.6/readline
  - source for the GNU command-line interface
- gdb-6.6/sim
  - Sources for various simulators
    - ARM
    - MIPS
    - PowerPC
    - M68hc11
    - SH

Build Cross Debugger (1)

- Build cross binutils (--prefix=/foo)
- **Add /foo/bin to PATH**
- Build cross compiler (--prefix=/foo)
- Build cross newlib (--prefix=/foo) (optional)
- Configure GDB
  - **./configure --prefix=*/foo* --target=arm-elf **
    --enable-sim
- make
- make install

"/foo"只是舉例，你應該設定為你可以寫入且希望安裝的目錄路徑

Build Cross Debugger (2)

- Cygwin users
  - ~~insight 6.6 is ok ???~~
  - Some bugs in insight 6.5
  - ftp://sources.redhat.com:/pub/insight/releases/insight-6.4.tar.bz2
  - Download patch cyg-tcl-sehfix.patch from 課程網頁

Try: insight 6.8-1 ??

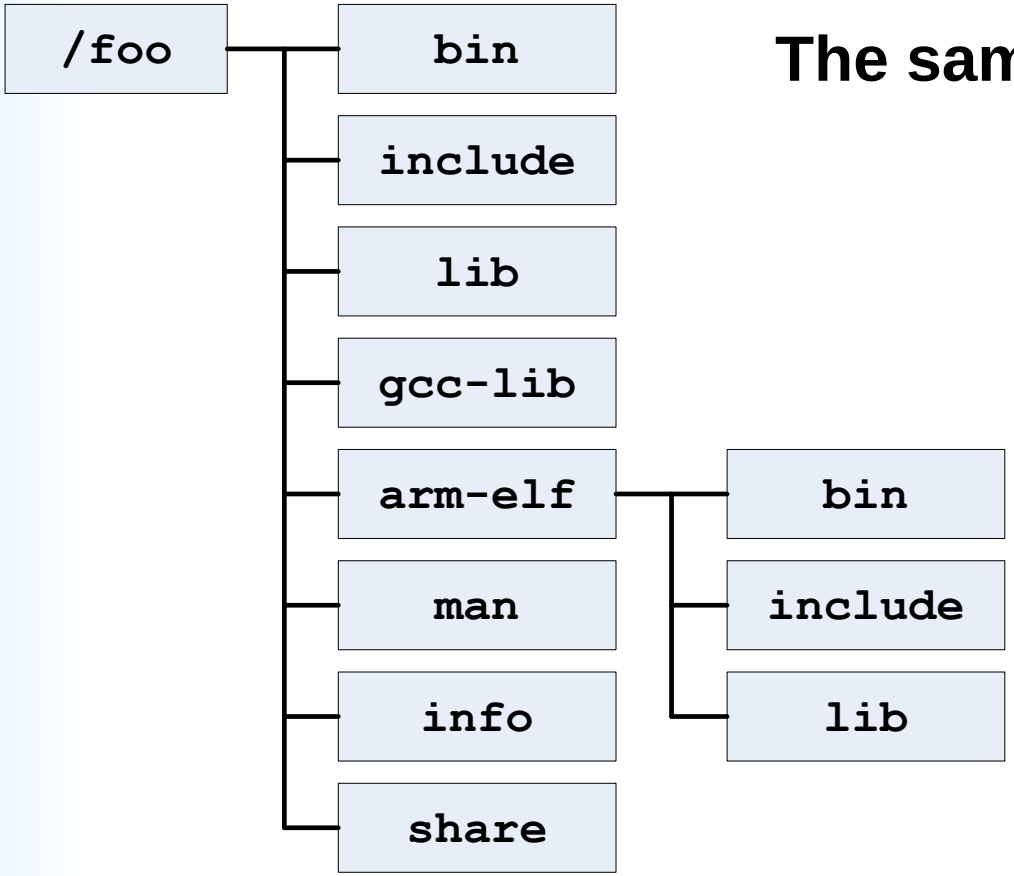
```
% tar -jxvf insight-6.4.tar.bz2
% cd insight-6.4
% patch -p0 < /tmp/cyg-tcl-sehfix.patch
```

(假設cyg-tcl-sehfix.patch存放在/tmp目錄下)

Build Cross Debugger (3)

- **--enable-tui**
  - enable full-screen terminal user interface (TUI)
- **--enable-gdbtk**
  - enable gdbtk graphical user interface (GUI)
- **--enable-profiling**
  - enable profiling of GDB
- **--enable-sim**
  - Link gdb with simulator
- **Check GDB document for others**

Directory Structure of GDB



The same with GCC, binutils

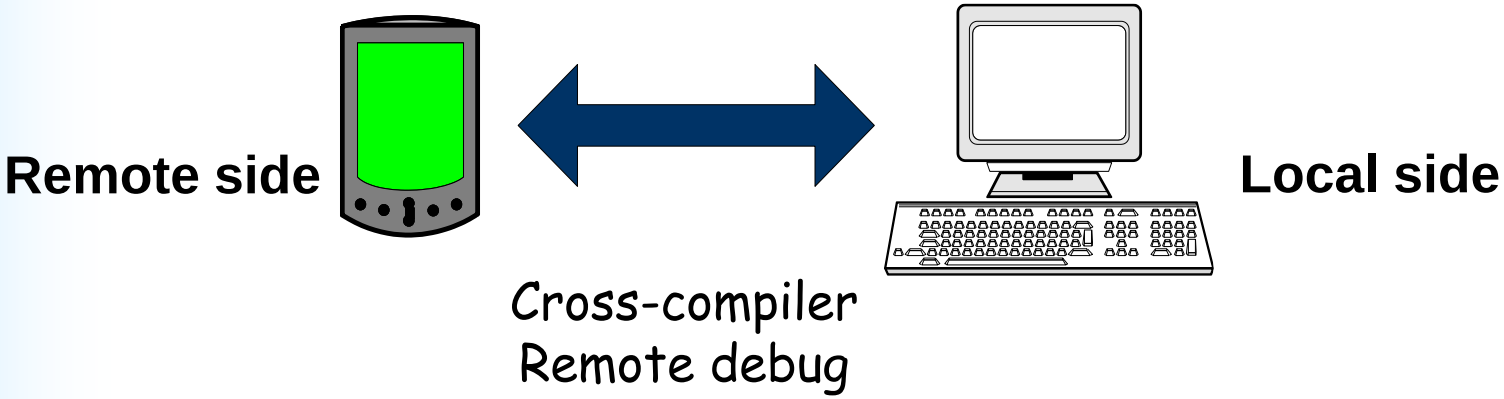
Run Insight with ARM Emulator

- Cygwin (Tcl/Tk)：直接執行
- Linux or BSD：需要**X-window**來執行

Practice on Source-Level Debug (1)

- Use cross compiler to compile test.c
  - "**-g**" option: add debug information which makes possible source-level debug

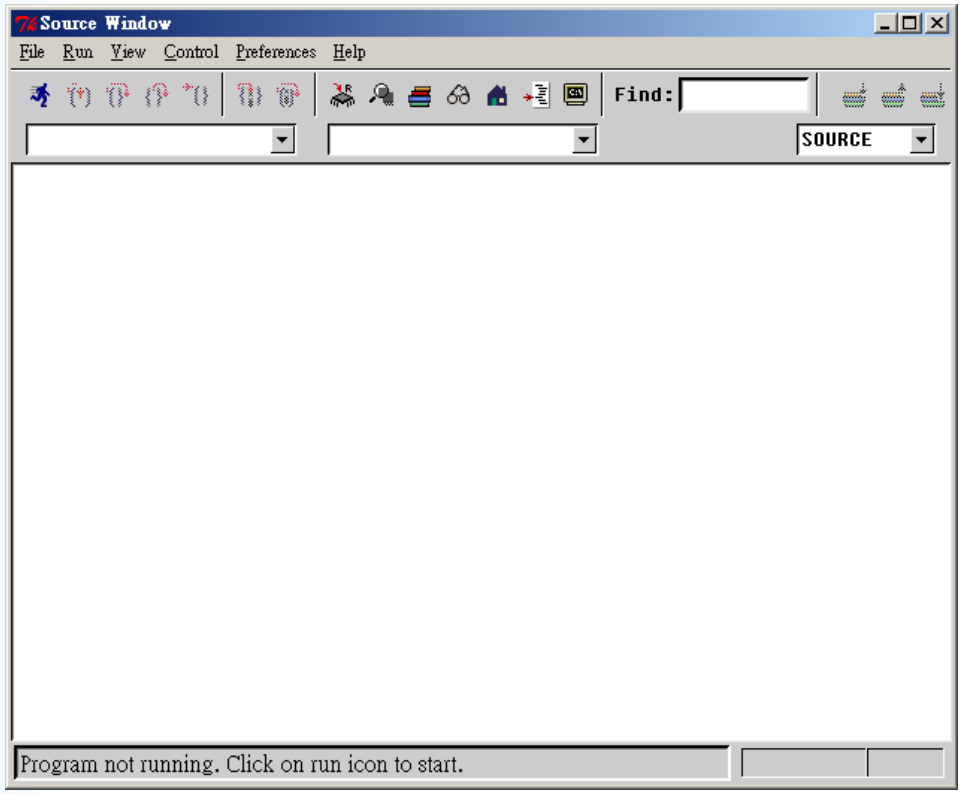
```
% /foo/bin/arm-elf-gcc -g test1.c -o test1.exe
```



Practice on Source-Level Debug (2)

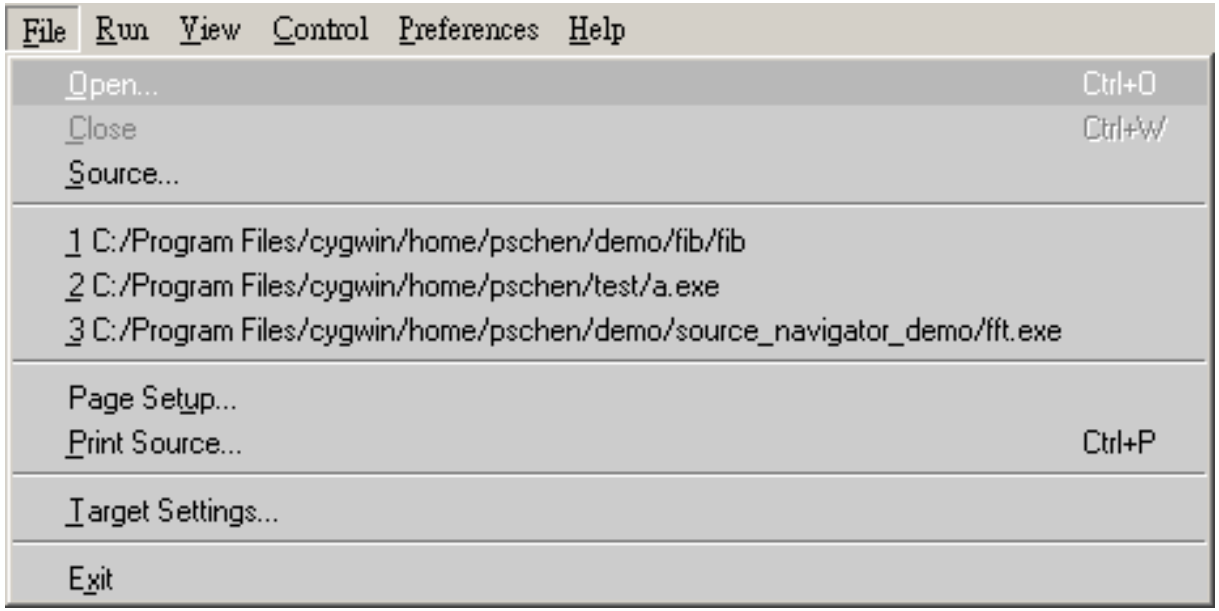
- **Local side**: Run **insight** (GUI of GDB)

```
% /foo/bin/arm-elf-insight
```



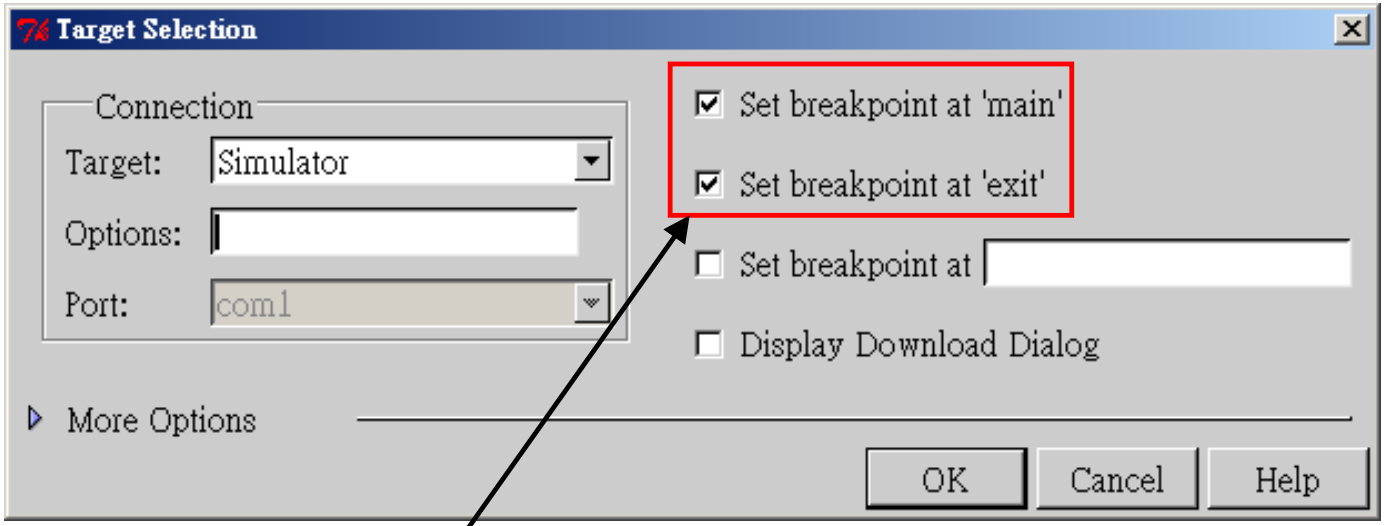
Practice on Source-Level Debug (3)

- **Local side**: Load file: test.exe
  - Select menu: File -> Open
  - Load the program we want to debug



Practice on Source-Level Debug (4)

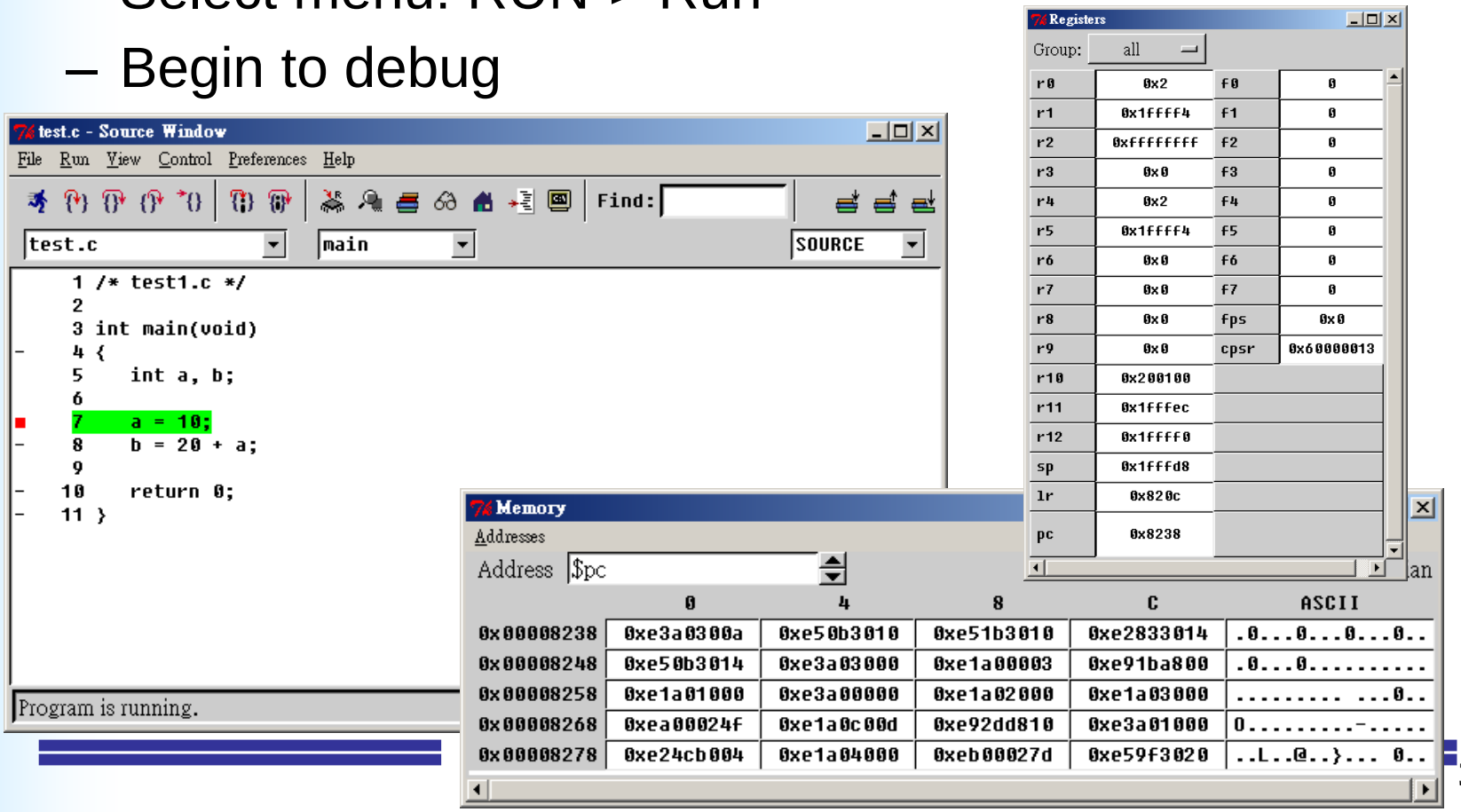
- **Local side**: Set target machine to simulator
  - Select menu: File -> Target Settings
  - Set **Target** to **ARM simulator**



Set breakpoint at “main” and “exit”

Practice on Source-Level Debug (5)

- **Local side**: Begin to do source-level cross debug on ARM simulator
  - Select menu: RUN-> Run
  - Begin to debug



How to Run a ARM program?

ARM assembly program

ARM object code

ARM executable code

Run ARM program

GCC cross-compiler & binutils

GDB & ARM emulator

Program Execution (1)

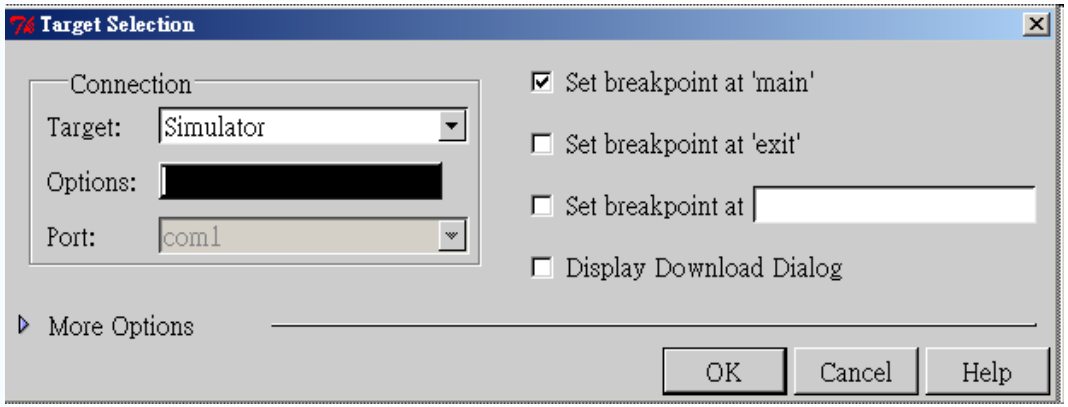
- 編譯你的程式

```
% arm-elf-gcc -g -O0 test.s -o test.exe
```

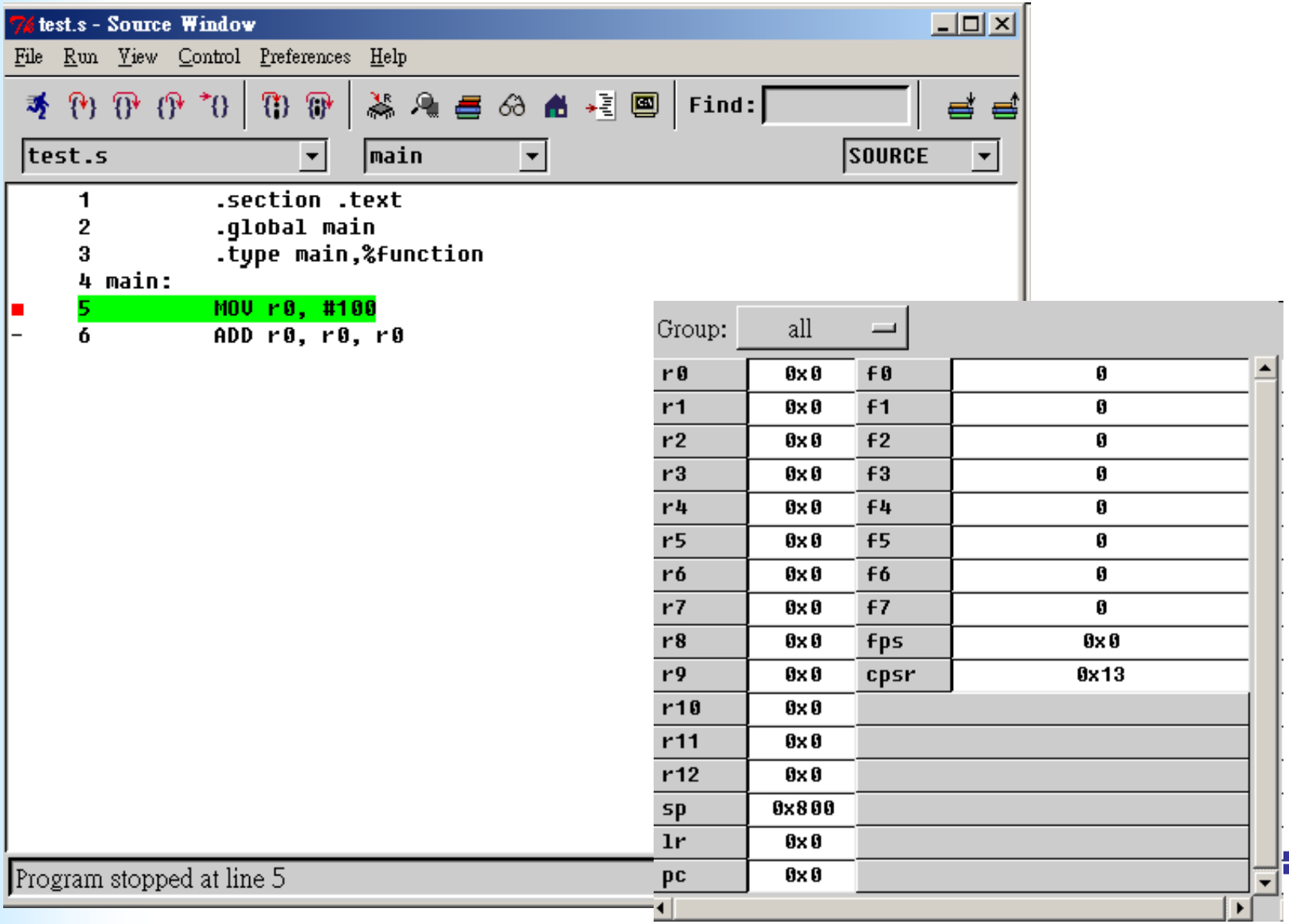
- 執行debugger & emulator

```
% arm-elf-insight
```

  – Select execution model:
    - File -> Target Settings… -> simulator



Program Execution (2)



```
.type name, type description
```

- **`type description`**
  - function: Mark the symbol as being a function name.
  - object: Mark the symbol as being a data object.
  - common: Mark the symbol as being a common data object.

Example (1)

```
/* ========================== */
/*        DATA section        */
/* ========================== */
        .data

/* --- variable a --- */
        .type a, %object
a:
        .word 1
        .word 2

/* --- variable b --- */
        .type b, %object
b:
        .word 3

/* --- variable c --- */
        .type c, %object
c:
        .space 8, 0
```

$$a = \begin{bmatrix} 1 & 2 \end{bmatrix}$$

$$b = \begin{bmatrix} 3 \end{bmatrix}$$

$$c = b \times a$$

$$c = \begin{bmatrix} ? & ? \end{bmatrix}$$

Example (1)

```
/* ========================== */
/*        DATA section        */
/* ========================== */
      .data

/* --- variable a --- */
a:
      .word 1
      .word 2

/* --- variable b --- */
b:
      .word 3

/* --- variable b --- */
c:
      .space 8, 0
```

$$a = \begin{bmatrix} 1 & 2 \end{bmatrix}$$

$$b = \begin{bmatrix} 3 \end{bmatrix}$$

$$c = b \times a$$

$$c = \begin{bmatrix} ? & ? \end{bmatrix}$$

Example (2)

```
/* ========================== */
/*        TEXT section        */
/* ========================== */
        .section .text
        .global main
        .type main,%function
.matrix:
        .word a
        .word b
        .word c
main:
        ldr r0, .matrix
        ldr r1, [r0], #4    /* r1 := mem32[r0] */
                            /* r0 := r0 + 4    */
        ldr r2, [r0]

        ldr r0, .matrix + 4
        ldr r3, [r0]        /* r3 := mem32[r0] */

        ldr r4, .matrix + 8

        mul r5, r3, r1
        mul r6, r3, r2

        str r5, [r4], #4    /* mem32[r4] := r5 */
                              /* r4 := r4 + 4    */
        str r6, [r4]
        nop
```

抓到a的address，放入r0

抓到第一個a的值

抓到第二個a的值

Example (2)

```
/* ========================== */
/*        TEXT section        */
/* ========================== */
        .section .text
        .global main
        .type main,%function
.matrix:
        .word a
        .word b
        .word c
main:
        ldr r0, .matrix
        ldr r1, [r0], #4    /* r1 := mem32[r0] */
                            /* r0 := r0 + 4    */
        ldr r2, [r0]

        ldr r0, .matrix + 4
        ldr r3, [r0]        /* r3 := mem32[r0] */

        ldr r4, .matrix + 8

        mul r5, r3, r1
        mul r6, r3, r2

        str r5, [r4], #4    /* mem32[r4] := r5 */
                              /* r4 := r4 + 4    */
        str r6, [r4]
        nop
```

抓到b的address，放入r0

抓到第一個b的值

抓到c的address，放入r4

Example (2)

```
/* ========================= */
/*        TEXT section       */
/* ========================= */
        .section .text
        .global main
        .type main,%function
.matrix:
        .word a
        .word b
        .word c
main:
        ldr r0, .matrix
        ldr r1, [r0], #4    /* r1 := mem32[r0] */
                            /* r0 := r0 + 4    */
        ldr r2, [r0]

        ldr r0, .matrix + 4
        ldr r3, [r0]        /* r3 := mem32[r0] */

        ldr r4, .matrix + 8

        mul r5, r3, r1
        mul r6, r3, r2

        str r5, [r4], #4    /* mem32[r4] := r5 */
                            /* r4 := r4 + 4    */
        str r6, [r4]
        nop
```

將結果存入第一個c的位置

將結果存入第二個c的位置

Reference

- **ARM instruction set**
  - 請參閱**Ecourse**課程網頁上之**ARM instruction set**相關說明
- **GNU Binutils**
  - http://sources.redhat.com/binutils
- **GNU GDB**
  - http://sources.redhat.com/gdb
- **Insight**
  - http://sources.redhat.com/insight