# ARM Assembly Programming: SWI by Using GAS

**Peng-Sheng Chen**

Fall, 2017

# Software Interrupt: SWI

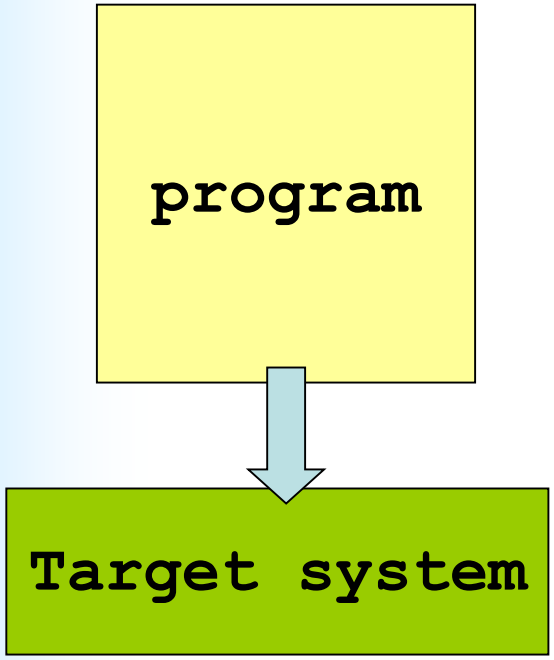
- SWI
  - 產生software interrupt
  - 應用程式可以透過SWI呼叫系統服務函式
  - Ex: I/O、timer、…

```
; This routine returns control from a user program
; back to the monitor program

    SWI     SWI_Exit   ; return to monitor
```

# CASE 1 (1)

## Embedded System





Ex: 汽車安全氣囊的程式

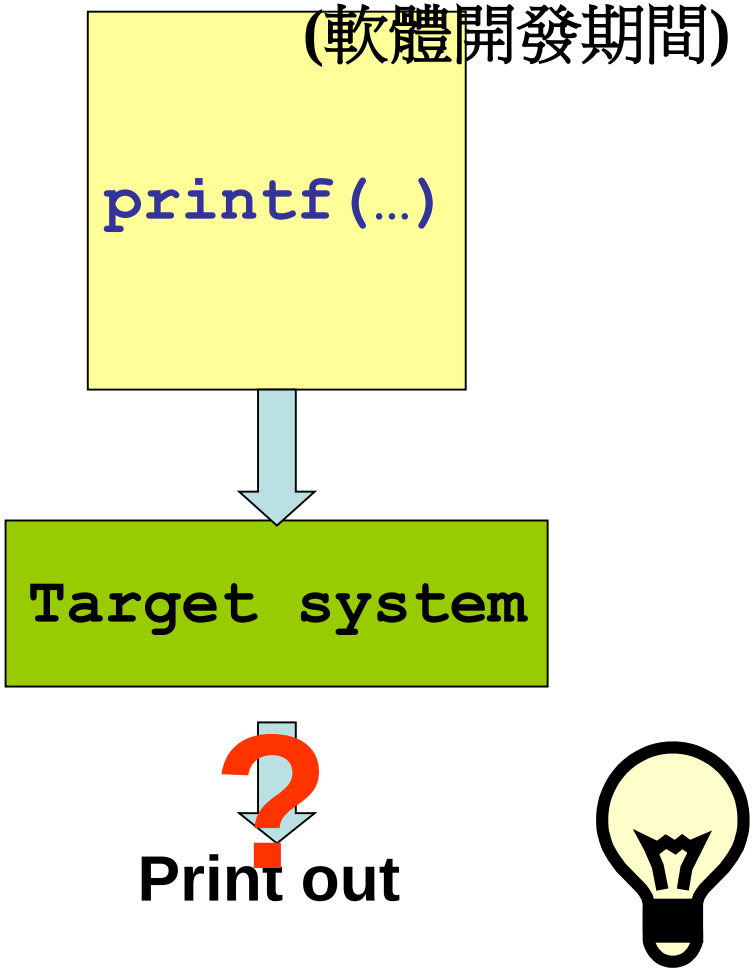
# CASE 1 (1)

## Embedded System

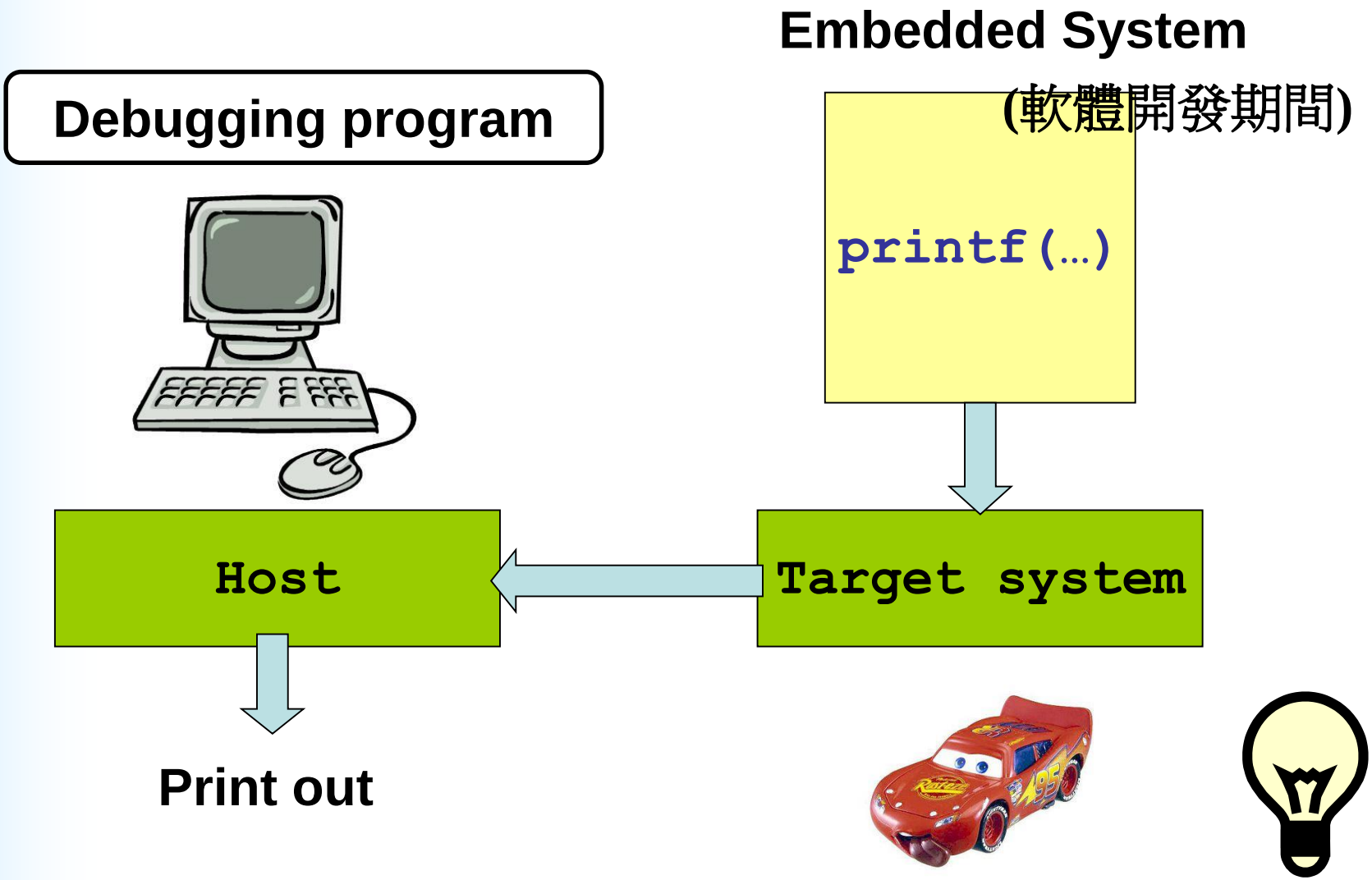

Ex: 汽車安全氣囊的程式

## Embedded System

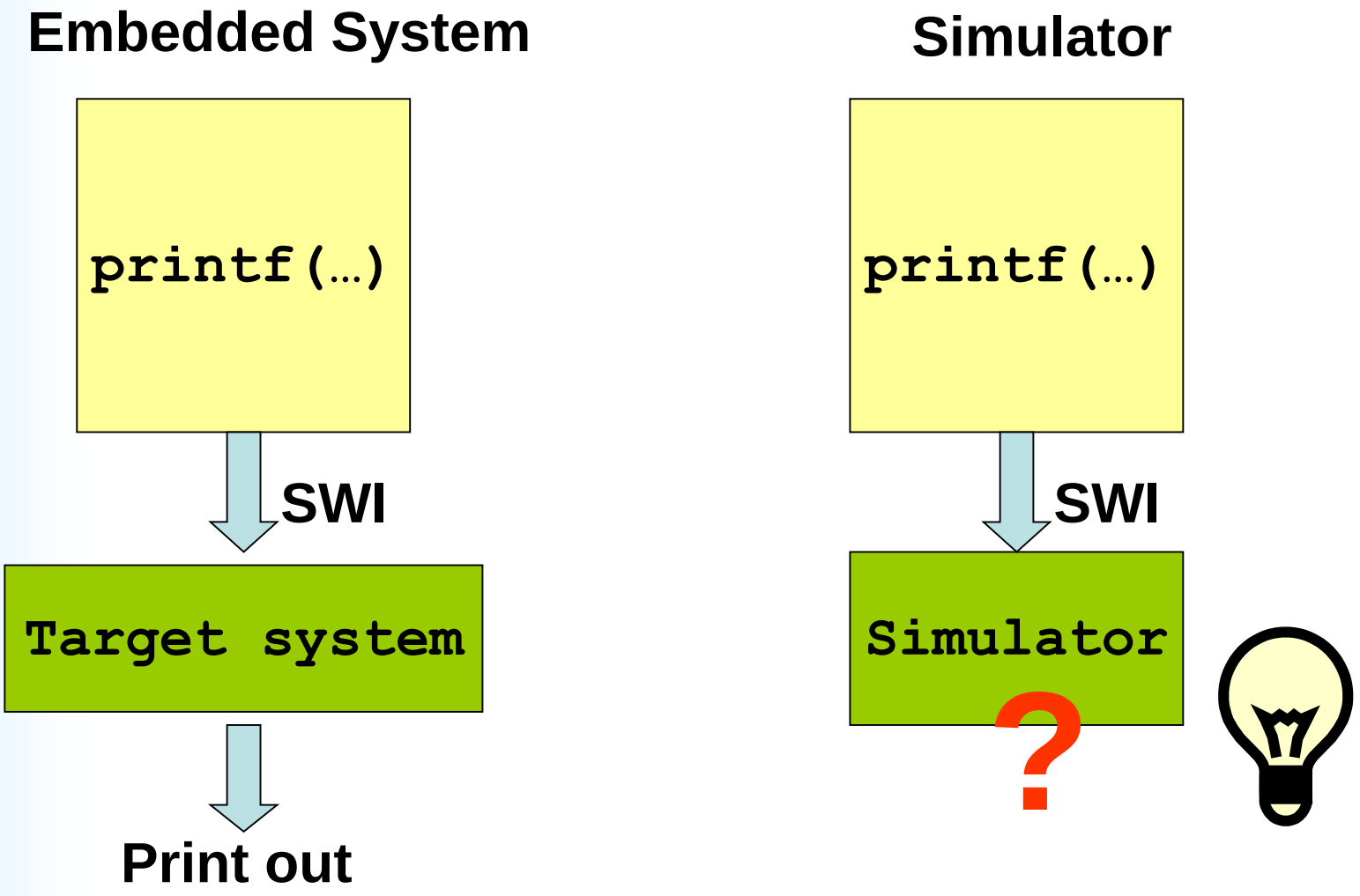
(軟體開發期間)

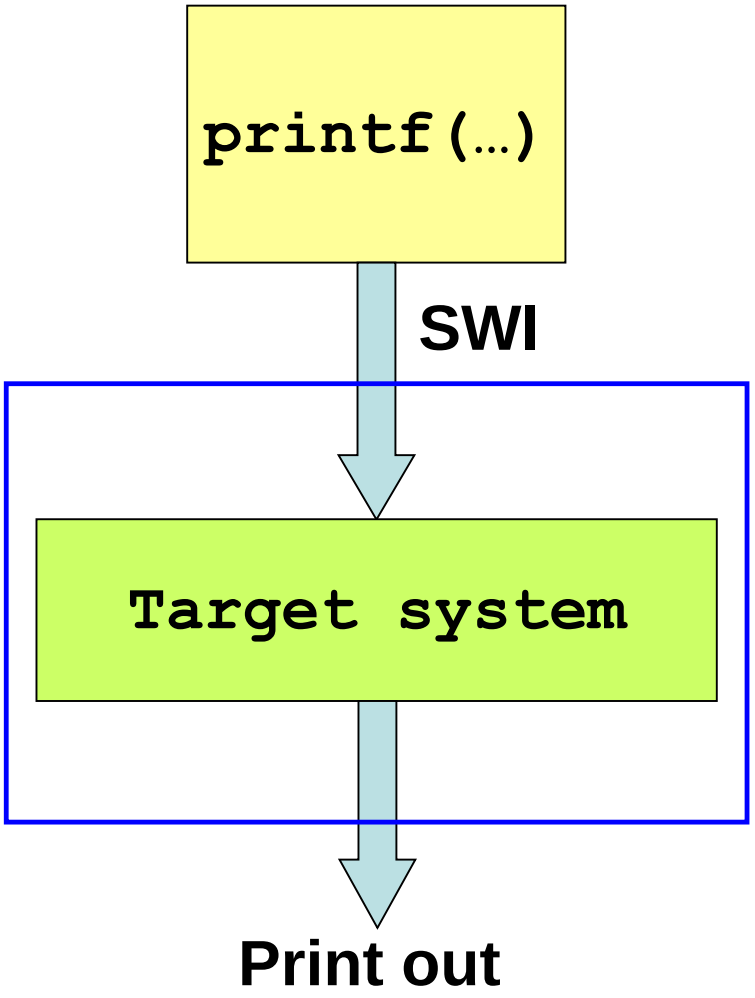


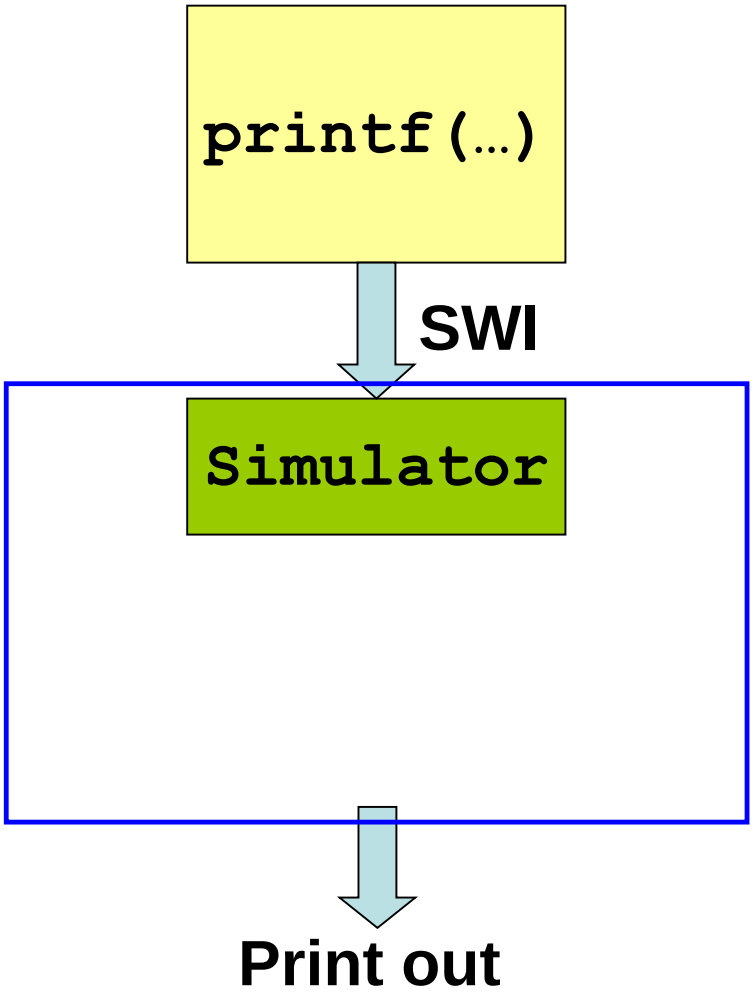
# CASE 1 (2)

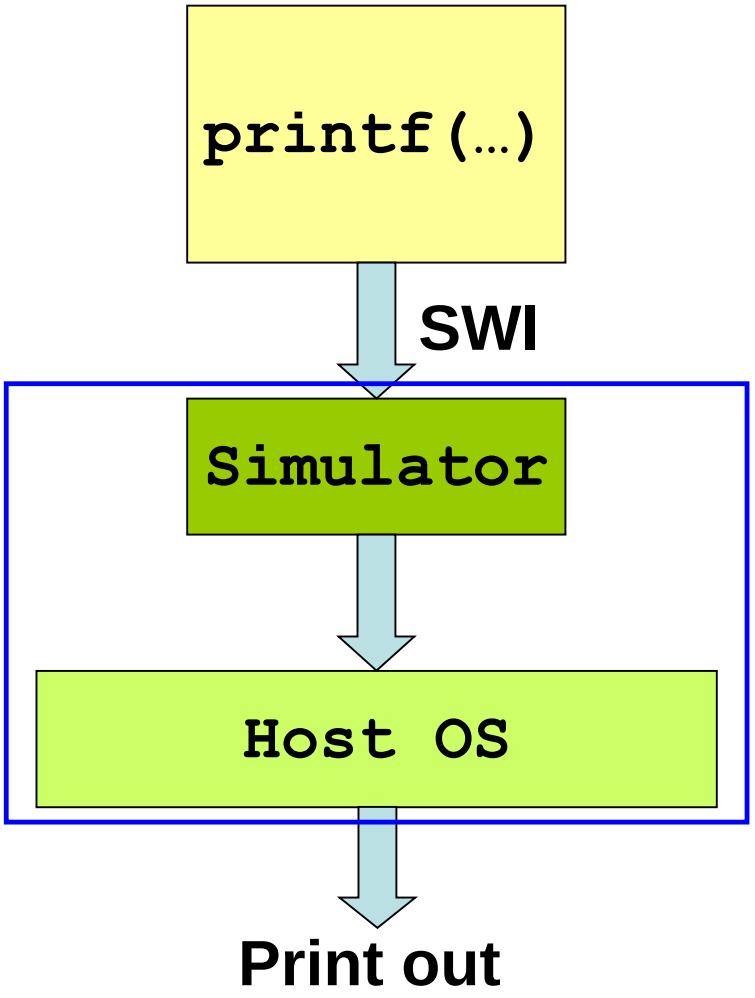

Embedded System
(軟體開發期間)
Debugging program
printf(…)
Host
Target system
Print out

# CASE 2


Embedded System
printf(…)
SWI
Target system
Print out
Simulator
printf(…)
SWI
Simulator
?


printf(…)
SWI
Target system
Print out


printf(…)
SWI
Simulator
Print out


printf(…)
SWI
Simulator
Host OS
Print out

# Semihosting (1)

- **Q: If target has no printf(), how to use printf() in the program?**
- **Semihosting**
  - Host: the ARM debugger is running on
  - Enable a target system which doesn't support various features required by the ANSI C library to use the features of the host instead
  - **Ex: 當target system沒有支援輸出，請host OS幫我們輸出，以方便debug**
  - It is very useful during software development

# Semihosting (2)

- **Simulator沒有支援輸出，請host OS幫我們輸出**

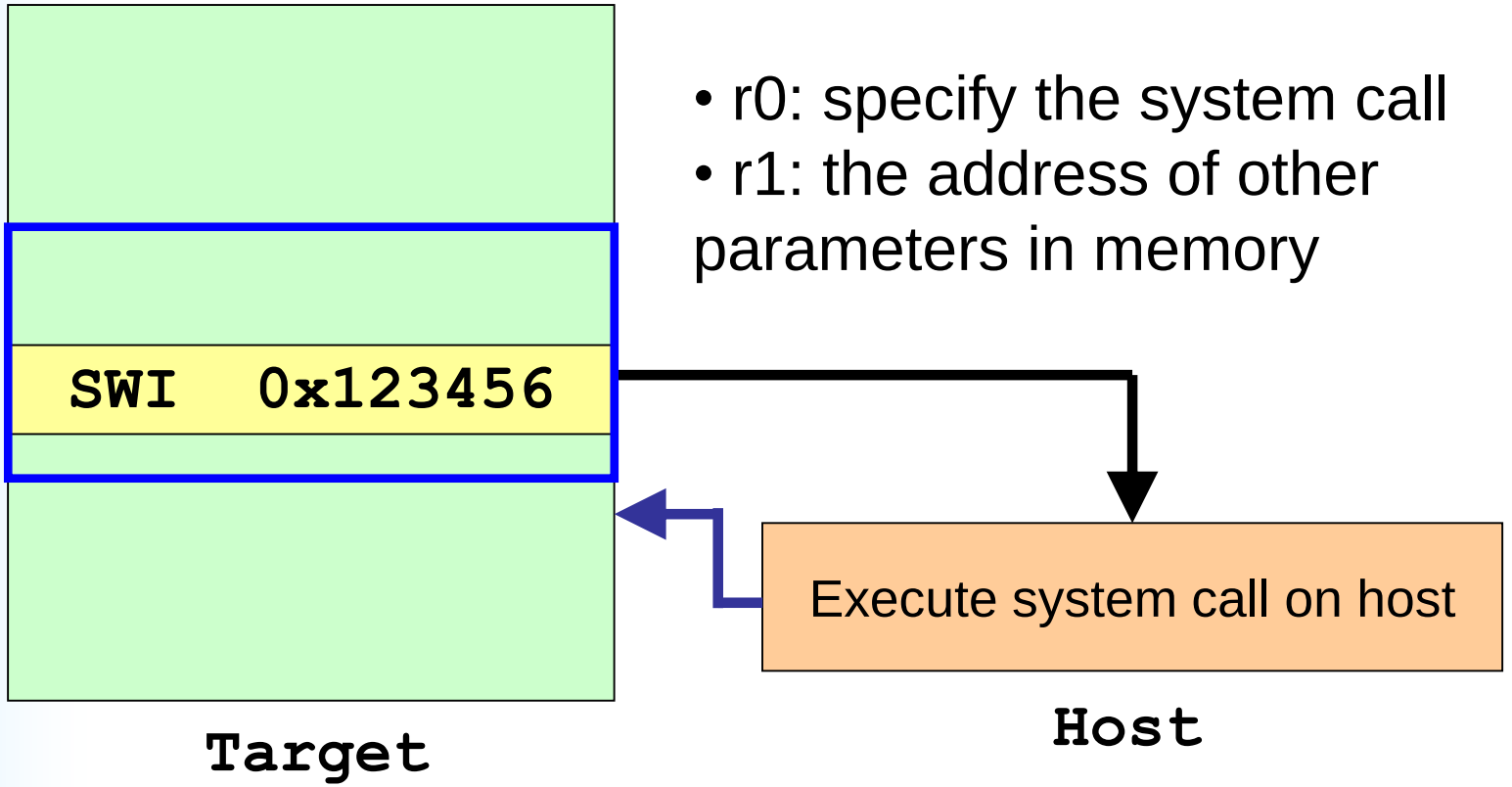
# Software Support for Semihosting

- ARM Toolkit: ARM angel debug monitor
- GNU Toolkit: newlib + GDB

# ARM Angel Debug Monitor

- **It provides the following services to developers**
  - Debug capability, including memory inspection, image download and execution, breakpointing and single step
  - CPU and board startup and basic exception handling
  - A full ANSI C library, using semihosting to provide services from the host which are not available on the target
  - A full source distribution, allowing developers a kickstart in developing standalone applications
- **It's similar to "newlib + gdb" in GNU toolchain**

# Semihosting on GNU Newlib (1)

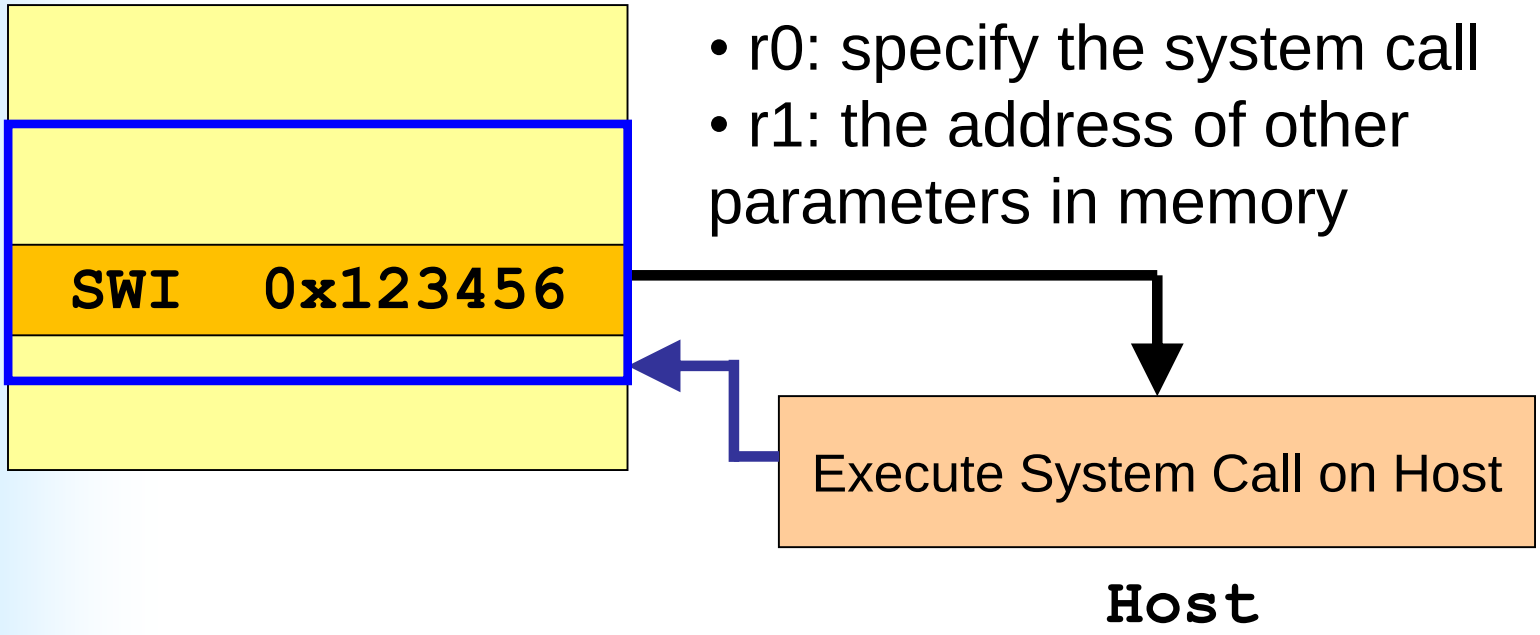



- r0: specify the system call
- r1: the address of other parameters in memory

**do_AngleSWI()** (newlib-x.xx.x/newlib/libc/sys/arm/libcfunc.c)

```
main (void)
{
…
    fclose();
…
}
```

```
_swiclose (void)
{
 /* inline assembly code */
}
```



# Implementation of SystemCall (newlib)

```
int _swiwrite (int  file, char*  ptr, int  len)
{
  int fh = remap_handle (file);
#ifdef ARM_RDI_MONITOR
  int block[3];

  block[0] = fh;
  block[1] = (int) ptr;
  block[2] = len;

  return do_AngelSWI (AngelSWI_Reason_Write, block);
#else
  asm ("mov r0, %1; mov r1, %2;mov r2, %3; swi %a0"
       : /* No outputs */
       : "i"(SWI_Write), "r"(fh), "r"(ptr), "r"(len)
       : "r0","r1","r2");
#endif
}
```

**(newlib-x.xx.x/newlib/libc/sys/arm/syscalls.c)**

# Semihosting on GNU Newlib (2)

**Format**

```
MOV    r0, function
MOV    r1, address of args
SWI    AngelSWI              /* 0x123456 */
MOV    output, r0
```

# Semihosting on GNU Newlib (3)

- **Functions**

```
#define AngelSWI_Reason_Open            0x01
#define AngelSWI_Reason_Close           0x02
#define AngelSWI_Reason_WriteC          0x03
#define AngelSWI_Reason_Write0          0x04
#define AngelSWI_Reason_Write           0x05
#define AngelSWI_Reason_Read            0x06
#define AngelSWI_Reason_ReadC           0x07
#define AngelSWI_Reason_IsTTY           0x09
#define AngelSWI_Reason_Seek            0x0A
#define AngelSWI_Reason_FLen            0x0C
#define AngelSWI_Reason_TmpNam          0x0D
#define AngelSWI_Reason_Remove          0x0E
#define AngelSWI_Reason_Rename          0x0F
#define AngelSWI_Reason_Clock           0x10
#define AngelSWI_Reason_Time            0x11
#define AngelSWI_Reason_System          0x12
#define AngelSWI_Reason_Errno           0x13
#define AngelSWI_Reason_GetCmdLine      0x15
#define AngelSWI_Reason_HeapInfo        0x16
#define AngelSWI_Reason_EnterSVC        0x17
#define AngelSWI_Reason_ReportException 0x18
```

**(newlib-x.xx.x/newlib/libc/sys/arm/swi.h)**

# Example 1: Print a character

```
/* ========================== */
/*        DATA section        */
/* ========================== */
        .data
        .align 4
/* --- variable a  (character) --- */
        .type a, %object
        .size a, 1
a:
        .byte 65
/* ========================== */
/*        TEXT section        */
/* ========================== */
        .section .text
        .global main
        .type main,%function
.char:
        .word a
main:
        mov r0, #0x3
        ldr r1, .char
        swi 0x123456
        mov r3, r0
```

character ‘A’

Function: WriteC

Address of ‘A’

# Example 2: Hello ARM

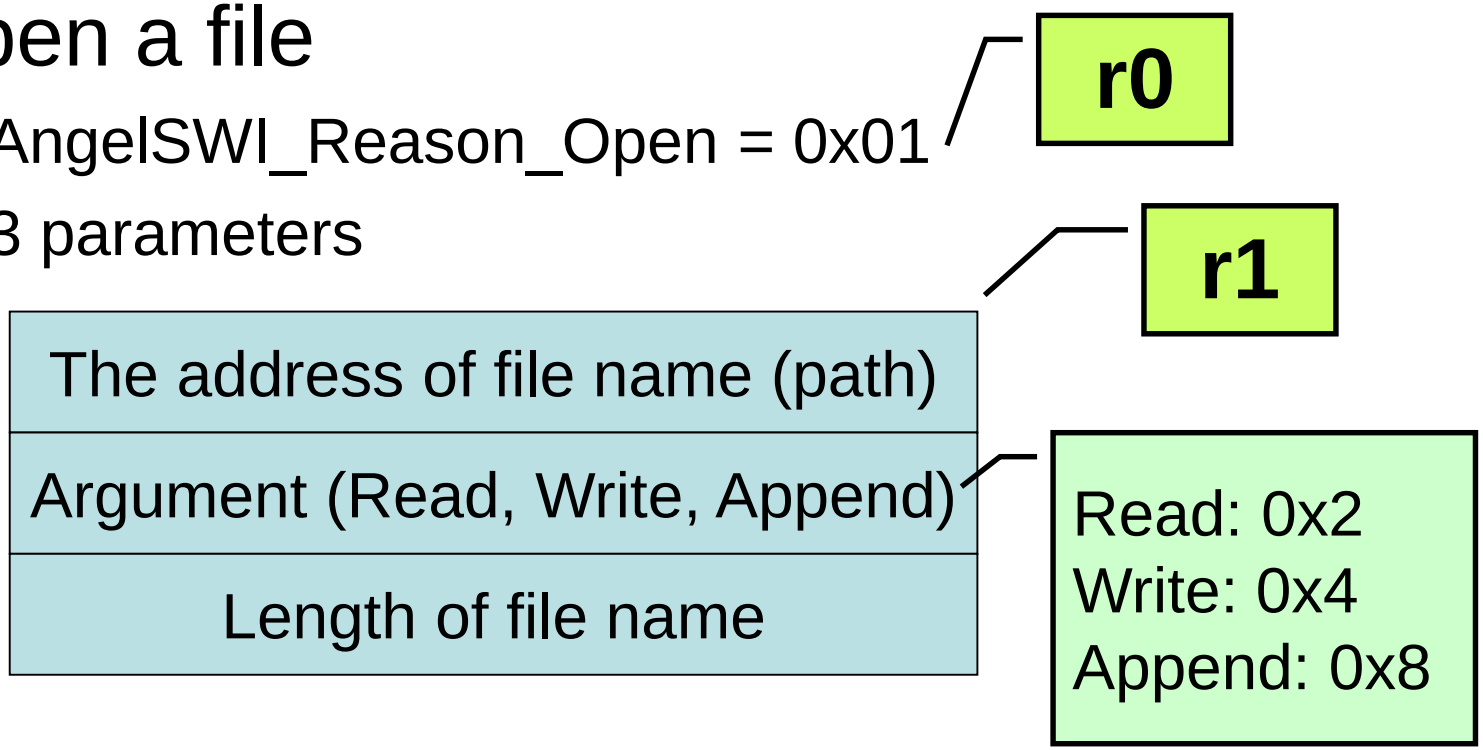
```
/* ========================= */
/*       DATA section        */
/* ========================= */
        .data
        .align 4

/* - a string "Hello ARM" - */
a:
        .ascii "Hello ARM\n"
```

```
/* ========================= */
/*       TEXT section        */
/* ========================= */
        .section .text
        .global main
        .type main,%function
.string:
        .word a
main:
        mov r3, #9
        mov r0, #0x3
        ldr r1, .string
loop:
        cmp r3, #0
        swigt 0x123456
        addgt r1, r1, #1
        subgt r3, r3, #1
        bgt loop
```

# File I/O: Open

- Open a file
  - AngelSWI_Reason_Open = 0x01
  - 3 parameters




  - **Return:** **r0 will have a file descriptor**

# File I/O: Close

- Close a file
  - AngelSWI_Reason_Close = 0x02
  - 1 parameter

| File descriptor |
|---|

  - **Return: r0 (0=> success)**

# File I/O: Write

- Write a file
  - AngelSWI_Reason_Write = 0x05
  - 3 parameters

| File descriptor |
| --- |
| Address of the string |
| Length of the string |

  - **Return: r0 (0=> success)**

# File I/O: Read

- Read a file
  - AngelSWI_Reason_Read = 0x06
  - 3 parameters

| File descriptor |
| --- |
| Address of the read buffer |
| # of bytes to be read |

  - **Return: r0 (0=> success)**

# File I/O: Flen

- The length of a file
  - AngelSWI_Reason_Flen = 0x0C
  - 1 parameters

| File descriptor |
|---|

  - **Return: r0 is the current length of the file**

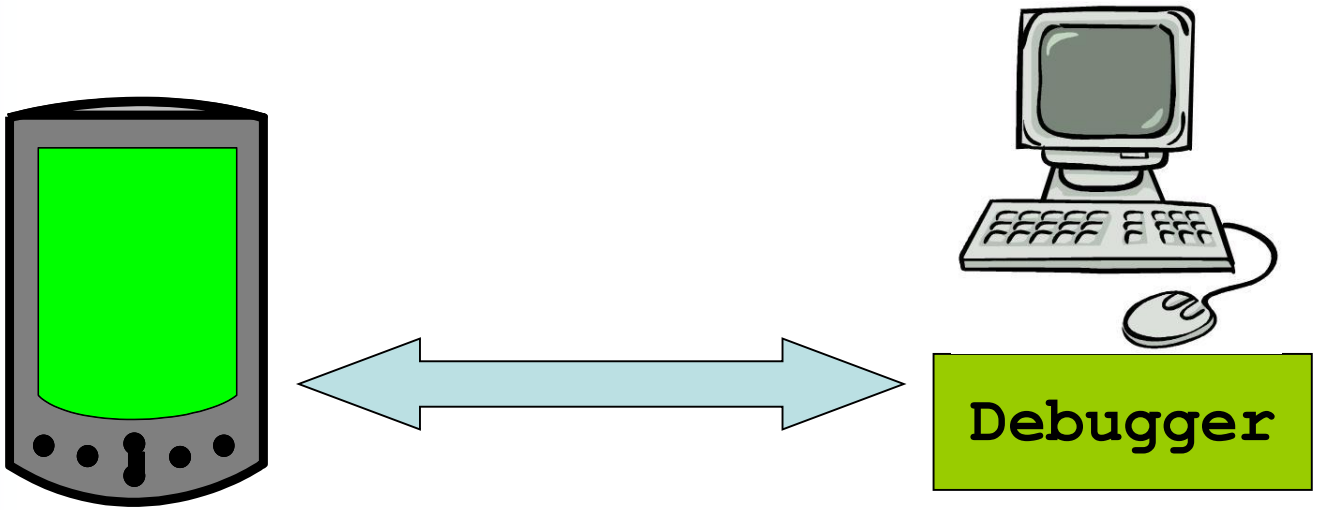
# Backup

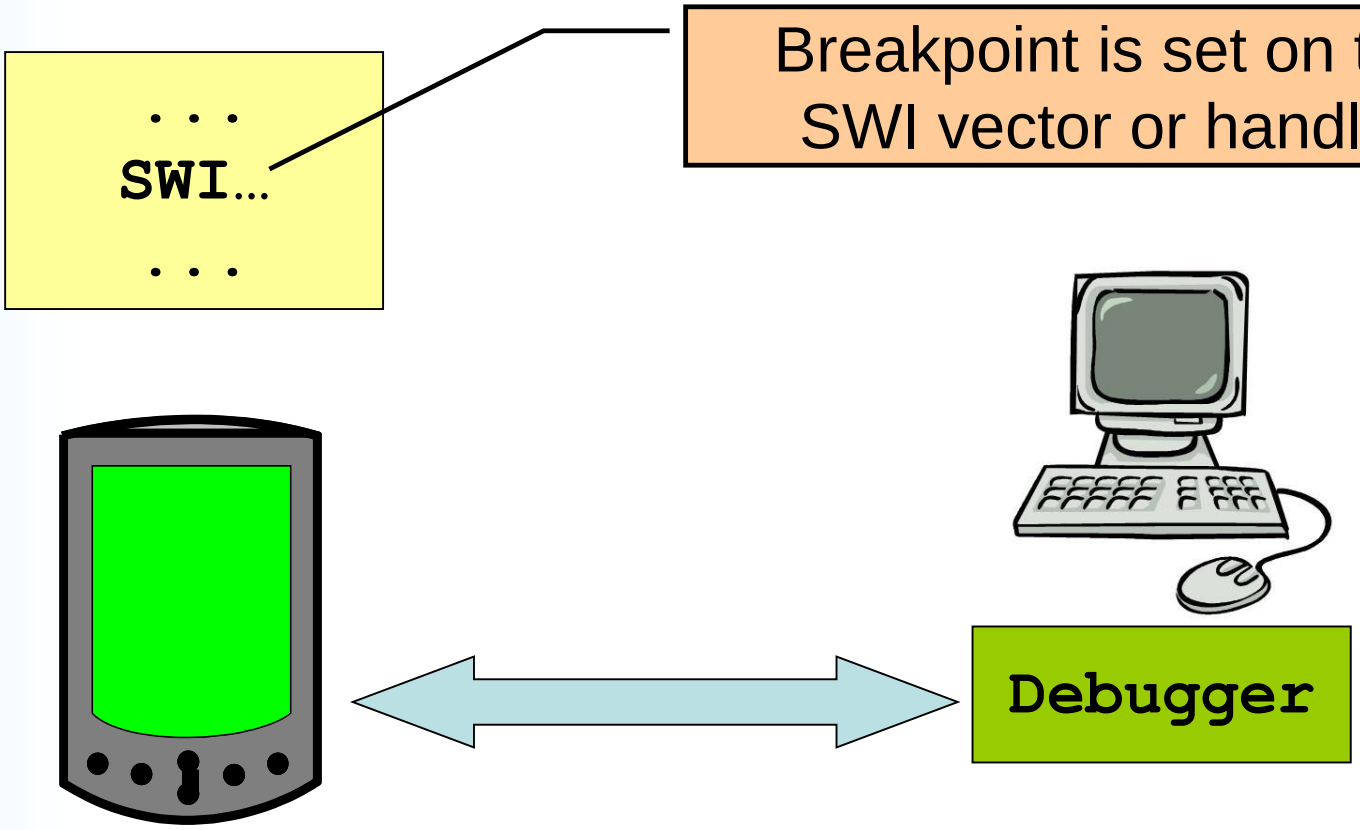
# Semihosting Implementation (1)

- **Standard semihosting**
  - A breakpoint is set on the SWI vector or handler
  - The target system stops when the breakpoint is encountered and control passes to the debugger, which then handles the semihosting operation
    - For example: printf()
    - Debugger displays the message to be printed in a dedicated console window
    - Retrieve the information for display via debugger access to the target memory system
  - When the operation is complete, control is returned to the target and normal operation resumes

1. Remote debug to develop software
2. Target platform does not have the support of standard I/O

```
. . .
SWI…
. . .
```



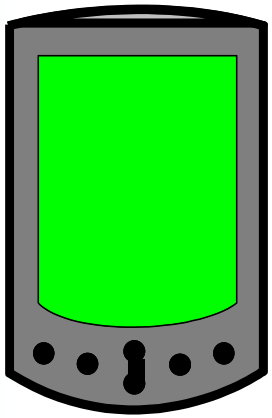



**Standard semihosting**

# When SWI instruction is executed,

1. 執行到Breakpoint
2. 控制權轉移到Debugger
3. **Debugger執行適當的SWI動作**
4. 控制權返回target platform
5. Target platform繼續執行接下來的指令

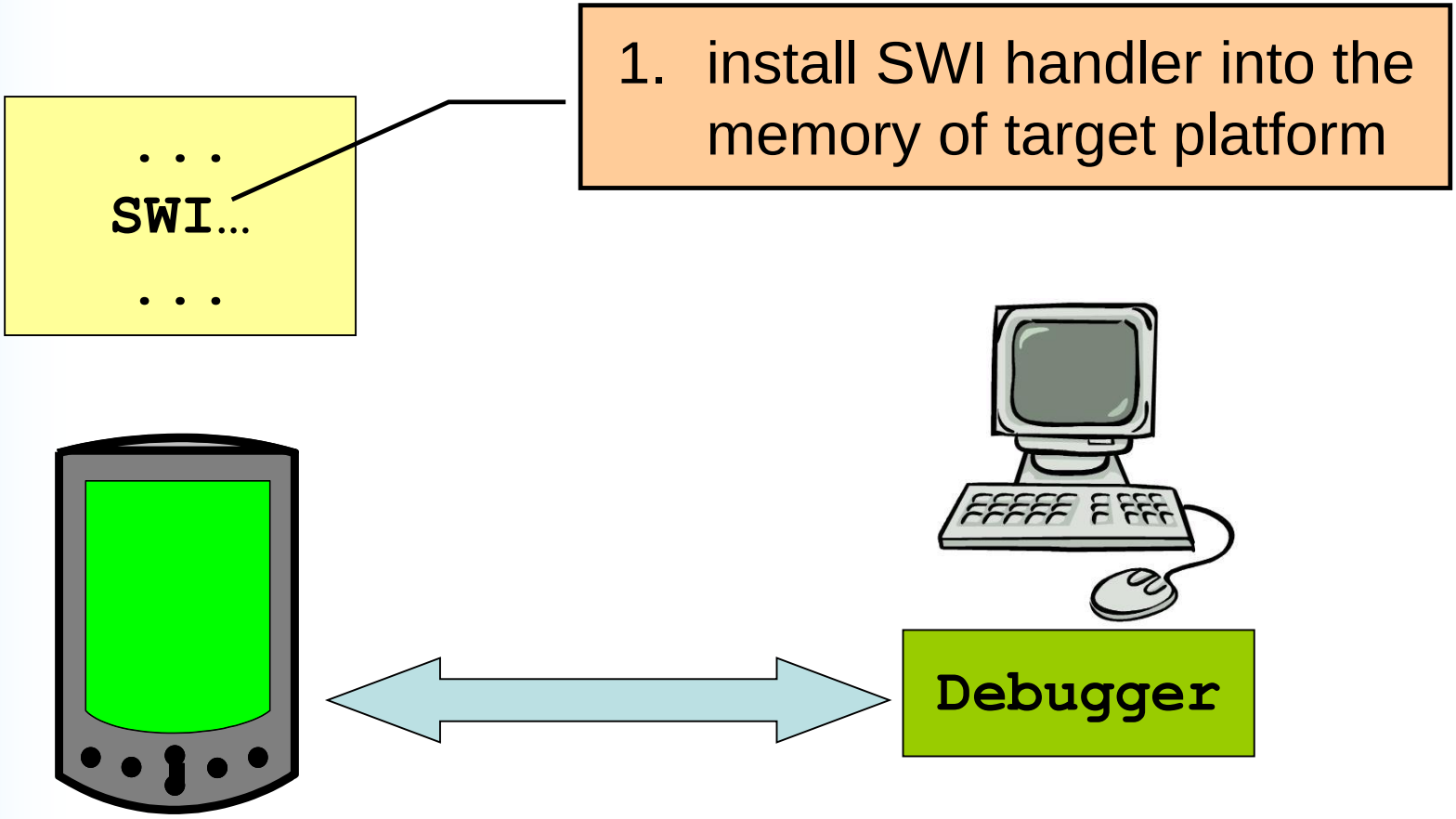
**SWI…**



**Debugger**

# Semihosting Implementation (2)

- **DCC-based semihosting**
  - Do not cause the target processor to stop
  - A SWI handler is installed in the target's memory which intercepts semihosting SWIs
  - Send a request for a semihosting operation which is carried out using the processor's Debug Comms Channel
  - The debugger hardware receives the request and communicates with the SWI handler on the target, requesting that memory is read and written as necessary
  - On completion, execution restarts from the instruction after the semihosting SWI




**DCC-based semihosting**

1. 執行到SWI，需要semihosting operation時，透過debug channel通知debugger
2. **Debugger與target platform上的SWI handler溝通，完成semihosting operation**
3. Target platform繼續執行接下來的指令

**SWI…**