Standard IO

Speaker: Juin-Nan Liu

Adopted from National Taiwan University
SoC Design Laboratory

Goal of This Lab

- Familiarize with ARM I/O architecture
- Know what Semihosting is
- Semihosting exercise

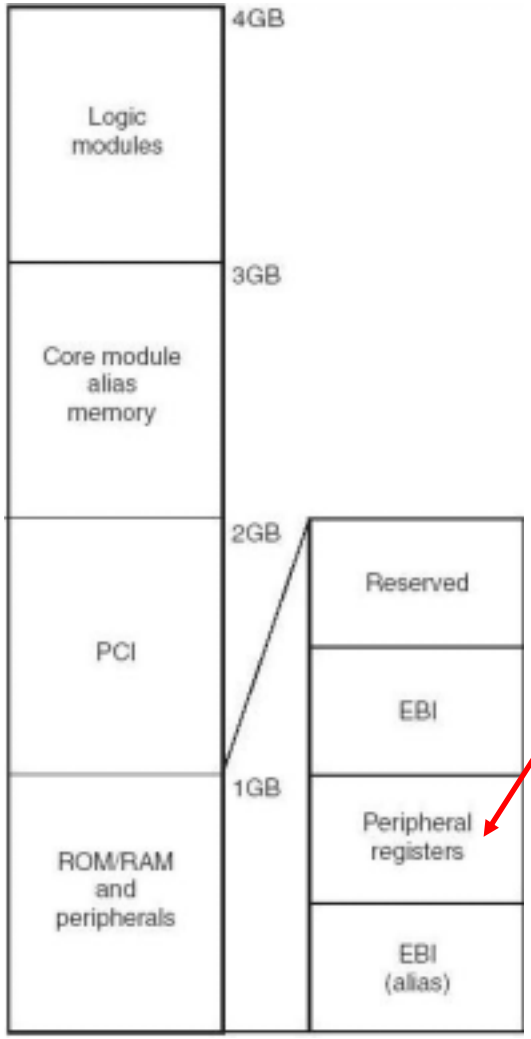
Outline

- ***ARM system input/output (I/O) functions***
- Semihosting [5]
- Lab - Semihosting

Memory-Mapped Peripherals

- The input/output (I/O) functions are implemented in an ARM system using a combination of **memory-mapped** addressable peripheral registers and the **interrupt inputs**.
- A peripheral device contains a number of registers. In a memory-mapped system, each of these registers appears like a memory location at a particular address.

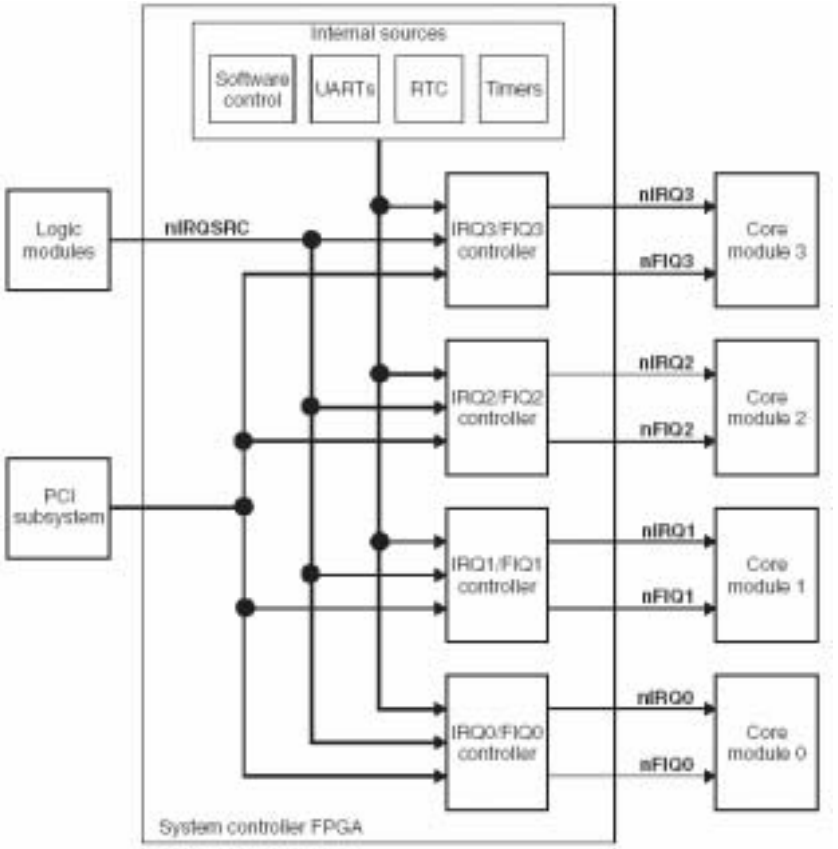
Peripheral Registers

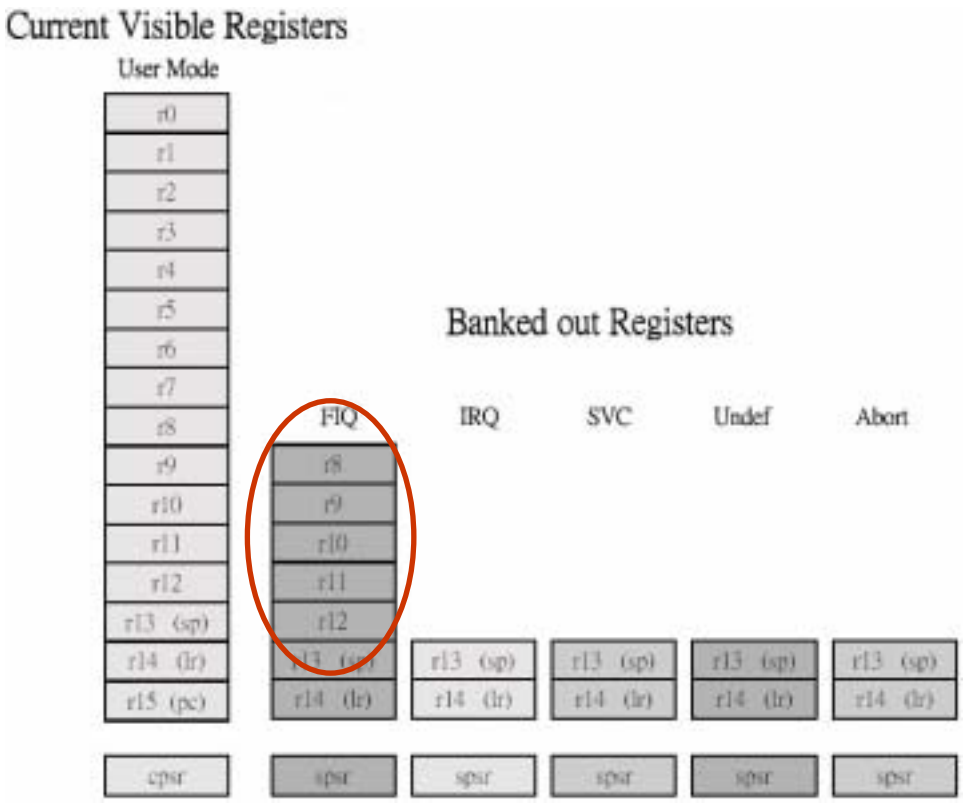


Base address	Size	Description
0x1F000000	16MB	Spare
0x1E000000	16MB	Spare
0x1D000000	16MB	Spare
0x1C000000	16MB	Spare
0x1B000000	16MB	GPIO
0x1A000000	16MB	LED display and boot switch
0x19000000	16MB	Mouse
0x18000000	16MB	Keyboard
0x17000000	16MB	UART1
0x16000000	16MB	UART0
0x15000000	16MB	RTC
0x14000000	16MB	Interrupt controller
0x13000000	16MB	Counter/timers
0x12000000	16MB	EBI configuration registers
0x11000000	16MB	System controller registers
0x10000000	16MB	Core module registers (for core modules) Spare (for logic modules)

[2]

Fast Interrupt Request





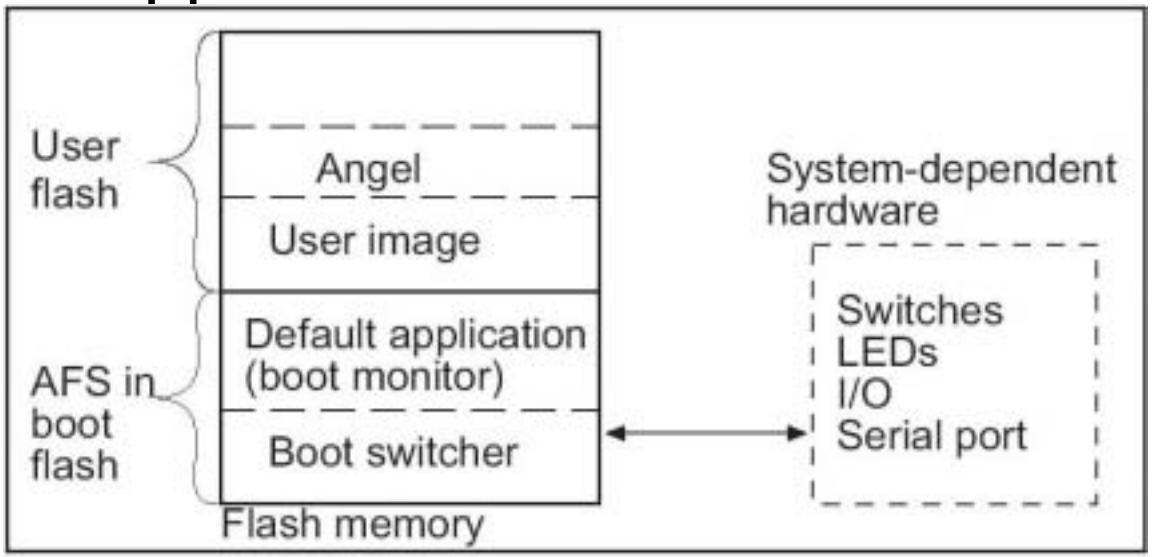
- The ARM fast interrupt (FIQ) architecture includes more ***banked registers*** than the other exception modes in order to minimize the register save and restore overhead associated with handling one of these interrupts. [1]

Input/Output

- In many ARM systems I/O locations are made inaccessible to user code, so the only way the devices can be accessed is through *supervisor calls* (SWIs) or through C library functions written to use those calls.
- The I/O area of memory is normally marked as **uncache-able**, and accesses bypass the cache.
- All the low-level detail of the I/O device registers and the handling of interrupts is the responsibility of the OS.

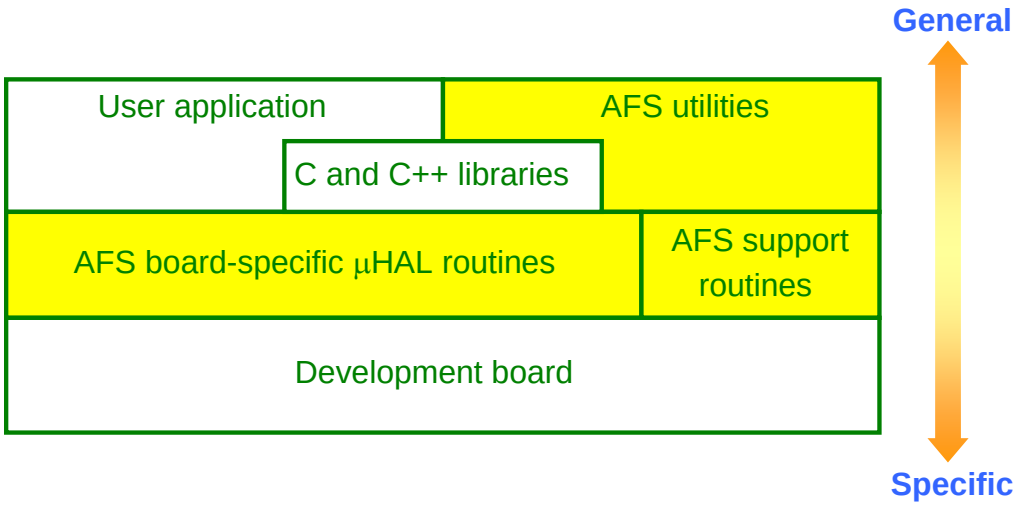
ARM µHAL

- *µHAL* is a *Hardware Abstraction Layer* that is designed to conceal hardware difference between different systems
- ARM µHAL provides a standard layer of board-dependent functions to manage I/O, RAM, boot flash, and application flash.



µHAL Examples

- µHAL provides simple & extended functions that are linkable and code reusable to control the system hardware.



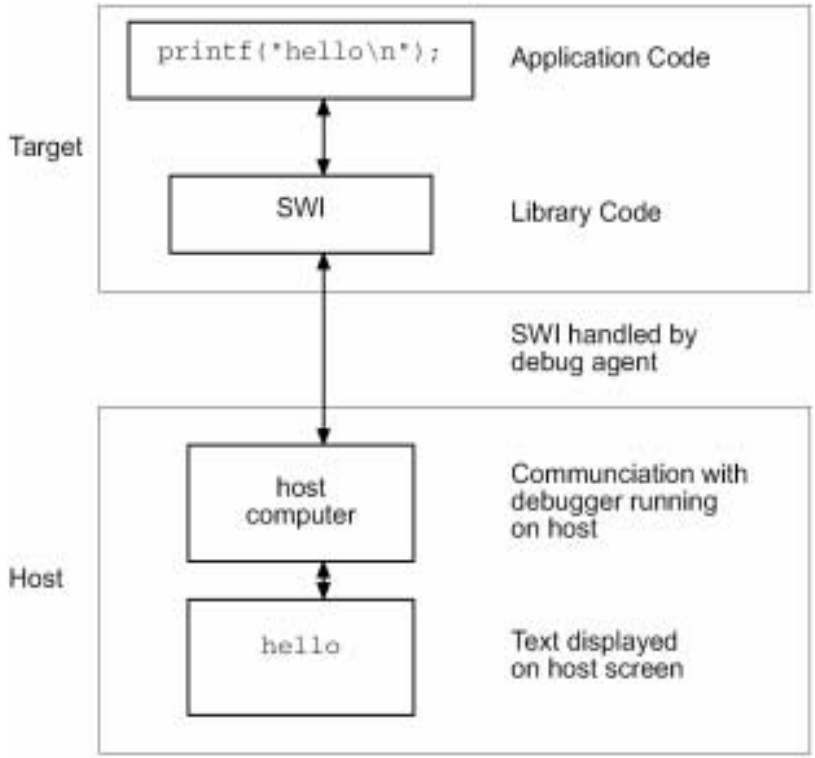
AFS: ARM Firmware Suit

Outline

- ARM system input/output (I/O) functions
- ***Semihosting [5]***
- Lab - Semihosting

Semihosting

- What is Semihosting?
  - A mechanism whereby the target communicates I/O requests made in the application code to the host system, rather than attempting to support the I/O itself.
- Semihosting overview



How Semihosting Work

- The application invokes the semihosting SWI (Software Interrupt)
- The debug agent then handles the SWI exception.
- The debug agent provides the necessary communication to the host system.
- Semihosting operations are requested using a semihosted SWI numbers:
  - 0x123456 in ARM state.
  - 0xAB in Thumb state.

SWI Interface

- A Software Interrupt (SWI) is requested with an SWI number.
  - Semihosting SWI numbers: 0x123456 (ARM), 0xAB (Thumb)
- Different operations in the SWI are identified using value of *r0*.
- Other parameters are passed in a block that is pointed by *r1*.
- The result is returned in *r0*. It could be an immediate value or a pointer.

Semihosting SWIs

- Semihosting operations used by C library functions such as *printf(), scanf().*
- No need to implement semihosting operations for default standard I/O functions.

SWI	Description
SYS_OPEN (0x01) on page 5-12	Open a file on the host
SYS_CLOSE (0x02) on page 5-14	Close a file on the host
SYS_WRITEC (0x03) on page 5-14	Write a character to the console
SYS_WRITE0 (0x04) on page 5-14	Write a null-terminated string to the console
SYS_WRITE (0x05) on page 5-15	Write to a file on the host
SYS_READ (0x06) on page 5-16	Read the contents of a file into a buffer
SYS_READC (0x07) on page 5-17	Read a byte from the console
SYS_ISERROR (0x08) on page 5-17	Determine if a return code is an error

Outline

- ARM system input/output (I/O) functions
- Semihosting
- ***Lab - Semihosting***

Lab 7: Standard I/O

- Goal
  - introduce students to control IO and learn the principle of polling, interrupt, and semihosting through this Lab.
- Principle
  - How to access I/O via the existing library function call.
- Guidance
  - Micro Hardware Abstraction Layer
  - How CPU access input devices
- Steps
  - This program controls the Intergator board LED and print strings to the host using *uHal* API.
- Requirements and Exercises
  - Modify the LED example. When it counts, we press any key to stop counting and then press any key to continue counting numbers.
- Discussion
  - Explain the advantage and disadvantage of polling & interrupt.
  - A system can be divided into hardware, software, and firmware. Which one contains uHAL.

References

[1] http://access.ee.ntu.edu.tw/course/SOC_LAB/index.html

[1] **ARM System-on-Chip Architecture** by S.Furber, Addison Wesley Longman: ISBN 0-201-67519-6.

[2] DUI0098B_AP_UG.pdf.

[3] DUI0102C_FirmSuite_rg.pdf

[4] DUI0102E_AFS1_3_rg.pdf

[5] ADS_DebugTargetGuide_D.pdf