

ELEC2041

Microprocessors and Interfacing

Lectures 28: Exceptions & Interrupts - II

<http://webct.edtec.unsw.edu.au/>

May 2006

Saeid Nooshabadi

saeid@unsw.edu.au

ELEC2041 lec28-exception-II.1

Saeid Nooshabadi

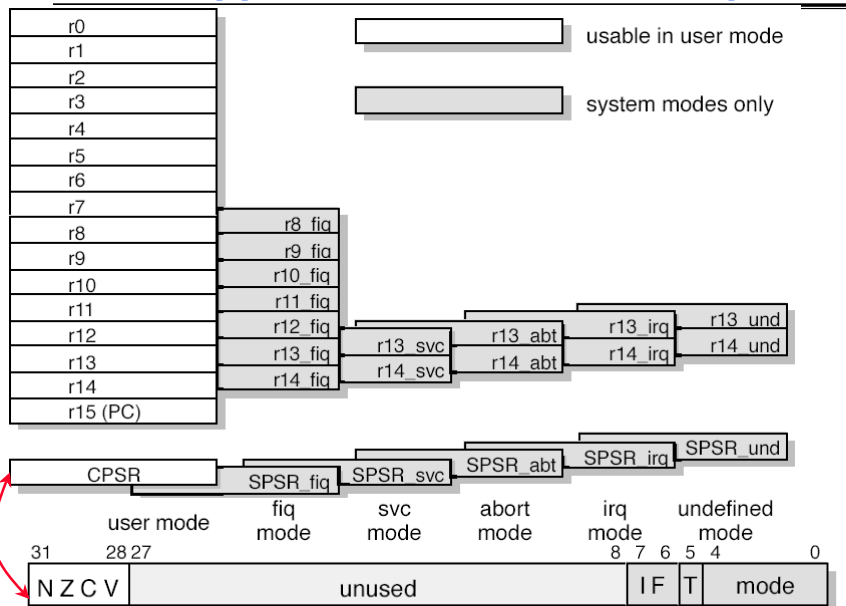
Overview

- Instruction Set Support for Exceptions
- Role of O/S in Handling Exceptions
- Prioritized Exceptions
- Re-entrant Exception Routine

ELEC2041 lec28-exception-II.2

Saeid Nooshabadi

Review: Support for ARM Modes of Operations



Review: Exception Handling and the Vector Table

- When an exception occurs, the core:
 - Copies CPSR into SPSR_<mode>
 - Sets appropriate CPSR bits
 - Interrupt disable flags if appropriate.
 - Maps in appropriate banked registers
 - Stores the “return address” in LR_<mode>
 - Sets PC to vector address

0x00000000	Reset
0x00000004	Undefined Instr
0x00000008	SWI
0x0000000C	Prefetch Abort
0x00000010	Data Abort
0x00000014	Reserved
0x00000018	IRQ
0x0000001C	FIQ

◦ Single instruction stored at these locations should be a branch to a handler

◦ To return, exception handler needs to:

- Restore CPSR from SPSR_<mode>
- Restore PC from LR_<mode> via `movs pc, lr` or `subs pc, lr, #4`

ELEC2041 lec28-exception-II.4

Saeid Nooshabadi

Review: Privileged vs User Mode

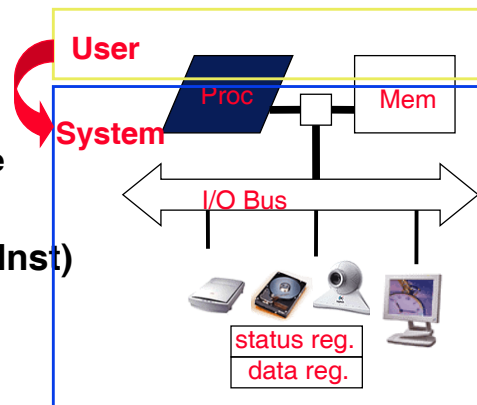
- Privileged Modes vs. User Mode: OS can provide security and fairness
- `swi`: provides a way for a programmer to avoid having to know details of each I/O device.
- To be acceptable, Exception handler must:
 - service all Exceptions (no drops)
 - service by priority
 - make all users believe that no Exception has occurred

Review: OS: I/O Requirements

- The OS must be able to prevent:
 - The user program from communicating with the I/O device directly
- If user programs could perform I/O directly:
 - No protection to the shared I/O resources
- 3 types of communication are required:
 - The OS must be able to give commands to the I/O devices
 - The I/O device notify OS when the I/O device has completed an operation or an error
 - Data transfers between memory and I/O device

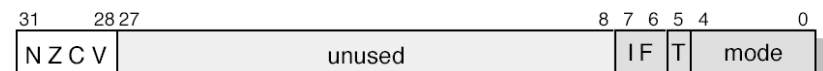
Review: Crossing the System Boundary

- System loads user program into memory and 'gives' it use of the processor
- Switch back
 - `swi`
 - request service
 - I/O
 - exception (und. Inst)
 - Interrupt



Instruction Set Support for OS (#1/2)

- How to turn off interrupts during Exception routine?
- Bits in CPSR determines whether or not interrupts disabled:
 - **Interrupt Request bit (I)** (1 ⇒ disabled, 0 ⇒ enabled)
 - Once an exception occurs (**I**) bit sets to 1 automatically
 - Return from exception restores the original value
 - **Fast Interrupt Request bit (F)** only gets disabled if a fast Interrupt occurs



Instruction Set Support for OS (#2/2)

- How to prevent user program from turning off interrupts (forever)?
- **Interrupt Request bit (I)** and **Fast Interrupt Request bit (F)** bits can only be changed from the privileged modes

Multiple Simultaneous Exceptions/Interrupts

- **Problem:** What if Data Abort exception and an I/O interrupt (printer ready, for example) come in both at the same time?
- **Options:**
 - drop any conflicting interrupts/ exceptions: unrealistic, they may be important
 - simultaneously handle multiple interrupts exceptions: unrealistic, may not be able to synchronize them (such as with multiple I/O interrupts)
 - queue them for later handling: sounds good

Exceptions Prioritization in ARM

Exception Type	Priority (1=High, 6=Low)
Reset	1
Data Abort	2
Fast Interrupt (FIQ)	3
Interrupt (IRQ)	4
Prefetch Abort	5
Software Interrupt (SWI)	6
Undefined Instruction	6

FIQ Priority

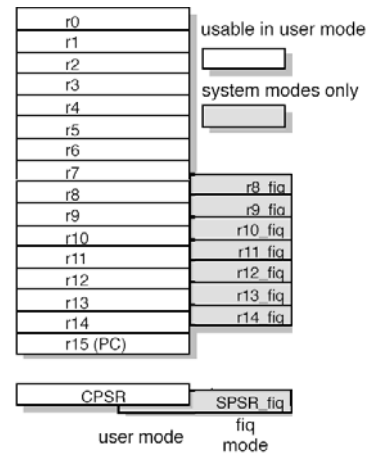
- **Placing the Data Abort exception above the FIQ exception in the priority list ensures that the Data Abort is actually registered before the FIQ is handled.**
- **The Data Abort handler is entered first but control is then passed immediately to the FIQ handler.**
- **Once the FIQ has been handled, control returns to the Data Abort Handler.**
- **This means that the data transfer error does not escape detection as it would if the FIQ were handled first.**

Hardware support for FIQ

- Extra register set `r12_fiq – r8_fiq` in FIQ bank would mean that they don't need to be saved.
- That means faster processing
- FIQ vector address at `0x0000001C` is the last vector, and therefore can start executing immediately without a branch instruction

<code>0x00000000</code>	Reset
<code>0x00000004</code>	Undefined Instr
<code>0x00000008</code>	SWI
<code>0x0000000C</code>	Prefetch Abort
<code>0x00000010</code>	Data Abort
<code>0x00000014</code>	Reserved
<code>0x00000018</code>	IRQ
<code>0x0000001C</code>	FIQ

ELEC2041 lec28-exception-II.13



Saeid Nooshabadi

Preempting Exceptions by Interrupts

- Question: Suppose we're dealing with a computer running a nuclear facility. What if we're handling an Floating Point Divide by 0 Exception and a Nuclear Meltdown Imminent interrupt comes in?
- Answer: We need to categorize and allow some interrupts preempt other Exceptions so we can handle them in order of urgency: emergency vs. luxury.

ELEC2041 lec28-exception-II.14

Saeid Nooshabadi

ARM Support for Preempting Exceptions

- ARM Architecture Support to simplify software:
 - An Exception Process sets IRQ bits (disables it).
 - No further IRQ interrupt is possible
 - However, No exception can disable FIQ interrupt
- If an FIQ interrupt comes while servicing an exception
 - Take FIQ immediately
 - Return to interrupted exception code as soon as no more FIQ Interrupt
 - State information for the preempted exception are saved on banked registers and as well as some internal registers

ELEC2041 lec28-exception-II.15

Saeid Nooshabadi

Reading Material

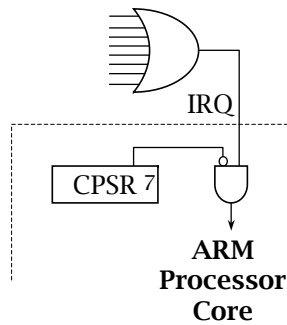
- Experiment 5 Documentation
- Steve Furber: ARM System On-Chip; 2nd Ed, Addison-Wesley, 2000, ISBN: 0-201-67519-6. [Chapter 5](#).
- ARM Architecture Reference Manual 2nd Ed, Addison-Wesley, 2001, ISBN: 0-201-73719-1, [Part A , Exceptions, chapter A2 Section 6](#)

ELEC2041 lec28-exception-II.16

Saeid Nooshabadi

So Many Devices One FIQ/IRQ

- Two interrupt request signals FIQ and IRQ never enough for all of the I/O devices
- Need a mechanism to attach multiple devices to the same IRQ pin.
- Solution: Use Interrupt Controller to attach multiple devices to the same IRQ pin.**

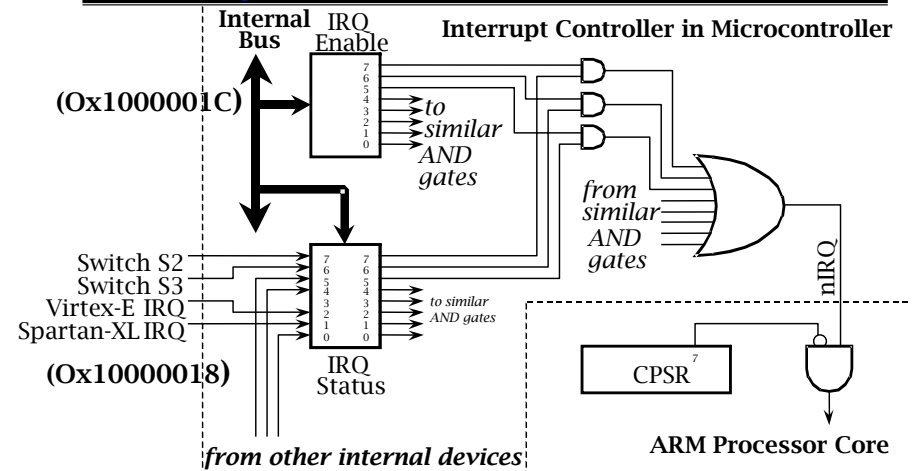


- Interrupt Controller** controls how multiple peripherals can interrupt the ARM processor. Essentially, the interrupt controller acts as a large AND-OR gate
- In the event of an IRQ a combination of hardware and software techniques are required to detect the sources of interrupt and provide a system of priority and queuing.

ELEC2041 lec28-exception-II.17

Saeid Nooshabadi

Interrupt Controller in DSLMU/Komodo



- IRQ Enable : Enables individual interrupts
- IRQ Status: Indicates raising interrupt.
- Prioritization:** in software

ELEC2041 lec28-exception-II.18

Saeid Nooshabadi

DSLMU/Komodo IRQ Status/Enable Regs

Bit	Mode	Function
7	R/W	Push-button switch S2 on the Expansion Board
6	R/W	Push-button switch S3 on the Expansion Board
5	R/W	Serial port transmitter ready
4	R/W	Serial port receiver ready
3	—	(Reserved)
2	R/W	Xilinx Virtex-E interrupt request
1	R/W	Xilinx Spartan-XL interrupt request
0	R/W	Timer Compare interrupt request

IRQ status reg. address = 0x10000018

IRQ Enable reg. address = 0x1000001C

ELEC2041 lec28-exception-II.19

Saeid Nooshabadi

IRQ Interrupt Levels in ARM?

- What are they?



- It depends what the ARM chip is inside of: differ by app: GameBoy, Digital Set top Box, Digital Answering Machine, Digital Satellite Decoder, Pocket PC, Mobile Phone
- ARM architecture and Interrupt controller enables priorities for different I/O events

ELEC2041 lec28-exception-II.20

Saeid Nooshabadi

Improving Data Transfer Performance

- Thus far: OS give commands to I/O, I/O device notify OS when the I/O device completed operation or an error
- What about data transfer to I/O device?
 - Processor busy doing loads/stores between memory and I/O Data Register
- Ideal: specify the block of memory to be transferred, be notified on completion?
 - **Direct Memory Access (DMA)** : a simple computer transfers a block of data to/from memory and I/O, interrupting upon done

Example: FIQ Code for DMA controller

- DMA code from Disk Device to Memory

```
.data
Count:  .word  4096
Start:  .space 4096
.text
Initial: ldr a1, =Count
         ldr a1, [a1] ; No. chars
         ldr a2, =Start ; next char
Wait:    ldr a3, DiskControl
         tst a3,1      ; select Ready
         beq Wait     ; spinwait
         ldrb a4, DiskData ; get byte
         strb a4, [a2], #1 ; transfer,
                           ; Start++
         subs a1, a1, #1 ; Count--
         bne Wait     ; next char
```

- DMA “computer” in parallel with CPU

Re-entrant Interrupt Routine?

- Prioritization and queuing of interrupts attached to IRQ Interrupt controller requires one interrupt preempting another interrupt (reentrant interrupts)
- All exceptions set IRQ disable interrupt bit (I)
 - FIQ can interrupt IRQ and other exceptions
- FIQ sets FIQ disable interrupt bit (F) as well
- How allow one IRQ interrupt to interrupt another IRQ interrupt and and safely save registers?
- Interrupt service Routine needs to save:
 - Return address on `lr_irq` for the pre-empted IRQ handler on IRQ Mode Stack
 - all registers that it is going to use on IRQ Mode Stack
 - Reset the IRQ disable bit
- Same thing for FIQ

Pre-empting Other Exceptions by IRQ?

- All exceptions set IRQ disable interrupt bit (I)
 - FIQ can interrupt IRQ and other exceptions
- How allow IRQ interrupts to interrupt other exceptions (other than IRQ and FIQ exceptions)?
- Interrupt service Routine needs to save:
 - all registers that it is going to use on IRQ Mode Stack
 - Reset the IRQ disable bit

IRQ Controller Support for Queuing and Priority

- **IRQ Controller Support to simplify software:**
 - An Interrupt Process cannot be preempted by Exception **at same** or lower **"level"**
 - When an interrupt is handled, take the highest priority interrupt on the queue
 - If a higher priority interrupt comes in while servicing a lower priority one
 - Preempt the lower priority one and take the higher priority one
 - current priority value is pushed onto a first-in last-out stack and the current priority is updated to the higher priority.
 - Return to interrupted code as soon as no more Interrupts at a higher level
 - On return the current interrupt level is updated with the last stored interrupt level from the stack.

Things to Remember

- **Privileged Mode v. User Mode: OS can provide security and fairness**
- **Exceptions prioritization in the case of multiple concurrent exceptions.**
- **Multiple devices can be connected to single IRQ interrupt via an Interrupt Controller.**
- **Re-entrant Interrupt: One interrupt preempting another interrupt.**
 - **Prioritization and queuing of interrupts in hardware or software is required for this to work.**