
ELEC2041

Microprocessors and Interfacing

Lectures 27: Exceptions & Interrupts - I

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

May 2006

Saeid Nooshabadi

saeid@unsw.edu.au

ELEC2041 lec27-exception-I.1

Saeid Nooshabadi

Overview

- Instruction Set Support for Exceptions
- Privileged vs User modes of operation.
- Handling a Single Interrupt

ELEC2041 lec27-exception-I.2

Saeid Nooshabadi

Review

- I/O gives computers their 5 senses
- I/O speed range is million to one
- Processor speed means must synchronize with I/O devices before use
- Polling works, but expensive
 - processor repeatedly queries devices
- Interrupts works, more complex

ELEC2041 lec27-exception-I.3

Saeid Nooshabadi

Definitions for Clarification

- **Exception**: signal marking that something “out of the ordinary” has happened and needs to be handled. Caused by internal and external sources
- **Interrupt**: Externally asynchronous exception (by I/Os)
- **Software Interrupt (SWI)**: User defined synchronous exception
- **Trap**: Processor’s diversion to a code to handle exception

ELEC2041 lec27-exception-I.4

Saeid Nooshabadi

Example of Exception Sources in ARM

- Externally generated interrupts
- An attempt by the processor to execute an undefined instruction
- Accessing privileged operating system functions via software interrupts (SWI).
 - Print
 - Access Ethernet

I/O Interrupt

- An I/O interrupt is like an undefined instruction exceptions except:
 - An I/O interrupt is “asynchronous”
 - More information needs to be conveyed
- An I/O interrupt is asynchronous with respect to instruction execution:
 - I/O interrupt is not associated with any instruction, but it can happen in the middle of any given instruction
 - I/O interrupt does not prevent any instruction from completion

Architecture Support for Exceptions

- Save the PC for return
 - But where?
- Where to go when Exception occurs?
- How to determine the Cause of exception?
- How to handle exceptions?

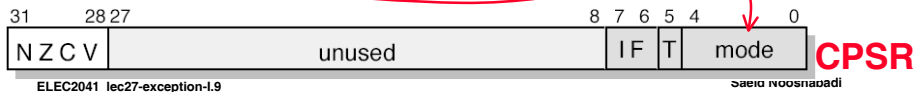
Exception Sources in ARM

- **Reset:** Occurs when the processor reset pin is asserted. (Signalling power-up)
- **Undefined Instruction:** Occurs if the processor does not recognize the currently executing instruction.
- **Software Interrupt (SWI):** This is a user-defined intentional synchronous interrupt instruction.
- **Prefetch Abort:** Occurs when the processor attempts to execute an instruction that was not fetched, because the address was illegal.
- **Data Abort:** Occurs when a data transfer instruction attempts to load or store data at an illegal address.
- **IRQ:** Occurs when the processor external Interrupt ReQuest pin is asserted
- **FIQ:** Occurs when the processor external Fast Interrupt reQuest pin is asserted

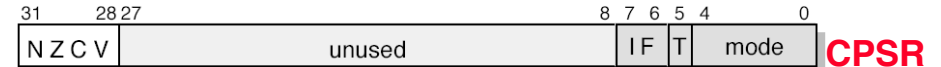
ARM Modes of Operations

Non-Privileged Mode

- 10000 **User** Normal user code
 - 10001 **FIQ** Processing fast interrupts
 - 10010 **IRQ** Processing standard interrupts
 - 10011 **SVC** Processing software interrupts (SWIs)
 - 10111 **Abort** Processing memory faults
 - 11011 **Undef** Handling undefined instruction traps
 - 11111 **System** Running privileged operating system tasks
- Privileged Modes**



CPSR Encoding for Operating Modes & Interrupts



mode	Mode of Operation
T	ARM vs Thumb State (We only use ARM in ELEC2041)
F	FIQ Fast interrupts Disable bit
I	IRQ Normal interrupt Disable bit
NZCV	Condition Flags

Changing mode bits is only possible in privileged modes

ELEC2041 lec27-exception-l.10 Saeid Nooshabadi

User Mode vs Privileged Modes

- User Applications run in User Mode
- Privileged modes, used to
 - service interrupts
 - exceptions
 - access protected resources (via SWI Instruction)
- Privileged modes → User mode **OK**
- User mode → Privileged modes **NOT OK**
 - Only possible through Controlled mechanisms: SWI, Exceptions, Interrupts

ELEC2041 lec27-exception-l.11

Saeid Nooshabadi

Rationale Privileged Modes

- Privileged modes protect system from getting trashed by user.
- Some Instructions can only be executed in privileged mode
- Example: User can't directly read/write information from disk I/Os.
- Why not allow direct access to non-disk I/O devices in user mode?
- **Komodo on DSLMU runs in Privileged mode. Can access everything while running Komodo**

ELEC2041 lec27-exception-l.12

Saeid Nooshabadi

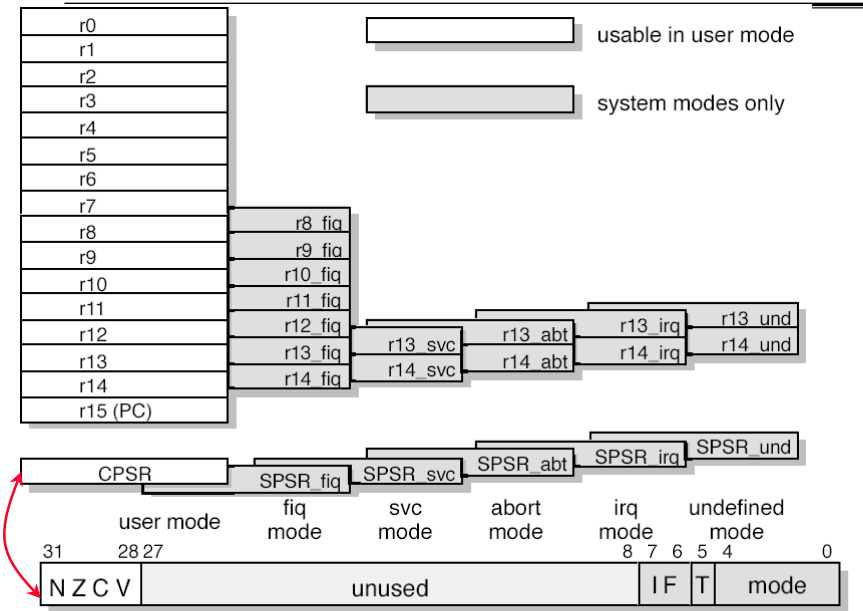
I/O Requires Privileged Modes

- Transmitter/receiver Status and Data words are in privileged data space; thus we must be in privileged mode to read or write them.
- Device drivers run in privileged mode.
- To access the I/O devices the user application has to make a Supervisor call to the OS via SWI instruction
- Komodo on DSLMU Allows access to I/O ports in User mode!
 - Security hole !

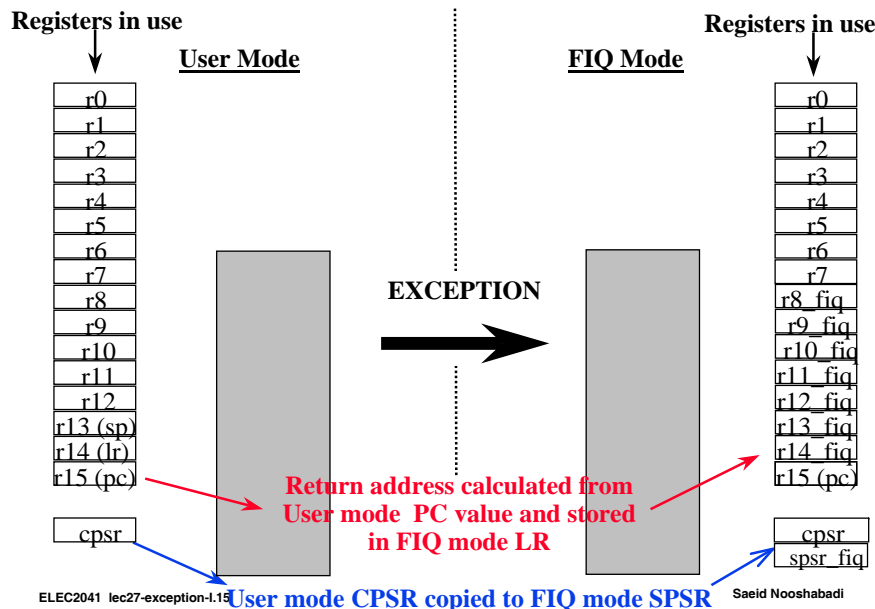
ELEC2041 lec27-exception-1.13

Saeid Nooshabadi

Support for ARM Modes of Operations



Switching between Modes (User to FIQ Mode)



ELEC2041 lec27-exception-1.15

Saeid Nooshabadi

Exception Handling Mechanism (#1/2)

- The processor's response to an exception
 - Copies the Current Program Status Register (CPSR) into the appropriate mode Saved Program Status Register (SPSR)
 - Sets the appropriate CPSR bits
 - Mode bits : set appropriately. maps in the appropriate banked registers for that mode.
 - I bit : to disable interrupts. IRQs are disabled once any other exception occurs
 - F bit: FIQs are also disabled when a FIQ occurs.
 - Stores the address of the return instruction (generally PC - 4) in LR_<mode>.
 - Sets the PC to the appropriate vector address. This forces the branch to the appropriate **exception handler**.

ELEC2041 lec27-exception-1.16

Saeid Nooshabadi

Exception Handling Mechanism (#2/2)

- Returning from an exception handler
 - Restore the CPSR from the SPSR_<mode>.
 - Restore the PC using the return address stored in LR_<mode>.
 - These can be achieved in a single instruction


```
movs pc, lr or
```

```
subs pc, lr, #4
```
 - Adding the **S flag** (update condition codes) to a data processing instruction when in a privileged mode with the PC as the destination register, also transfers the SPSR to CPSR
 - Same thing for Load Multiple instruction (using the ^ qualifier)


```
ldmfd sp! {r0-r12, pc}^
```

ELEC2041 lec27-exception-1.17

Saeid Nooshabadi

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
- To return, exception handler needs to:
 - Restore CPSR from SPSR_<mode>
 - Restore PC from LR_<mode> via

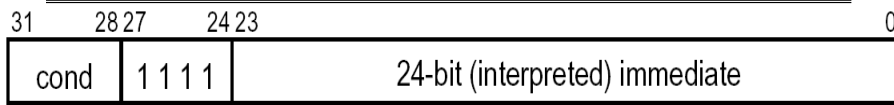
```
movs pc, lr or subs pc, lr, #4
```

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

ELEC2041 lec27-exception-1.18

Saeid Nooshabadi

Software Interrupt (SWI)



- In effect, a SWI is a user-defined instruction,
- A planned Exception from User Application to request privileged O/S services via SWI Supervisor call.
- It causes:
 - A switch to privileged Supervisor Mode.
 - A branch to an exception trap to the SWI exception vector (0x00000008)
 - a SWI exception handler to be called.
- The handler can then examine the comment field of the instruction to decide what operation has been requested.

ELEC2041 lec27-exception-1.19

Saeid Nooshabadi

SWI Invocation

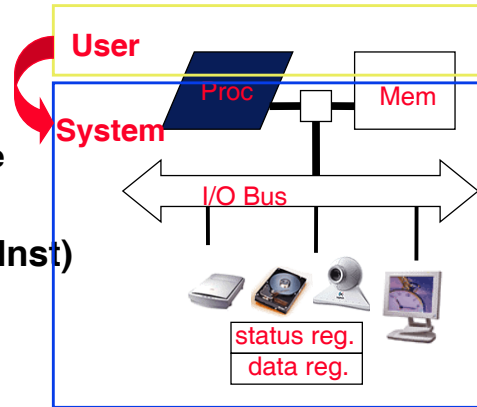
- How does user invoke the OS?
 - swi instruction: invoke the OS code (Go to 0x00000008, change to privileged mode)
 - By software convention, number xxx in swi xxx has system service requested: OS performs request

ELEC2041 lec27-exception-1.20

Saeid Nooshabadi

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



ELEC2041 lec27-exception-L21

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 lec27-exception-L22

Saeid Nooshabadi

SWI Example under GNU Debugging tools

```

7;Print to console a single char in
R0
swi 0x0

;Read a single char from Kboard
into R0
swi 0x4

;Print nul-terminated string
prt_str to the console
ldr    r0,=prt_str
swi    0x2

Swi 0x11 ; Terminate the program

.data
prt_str: .asciz "Hello World:\n"
    
```

ELEC2041 lec27-exception-L23

Saeid Nooshabadi

CPSR and SPSR Transfer Instructions

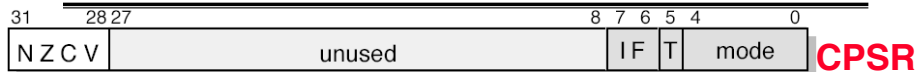


- MRS and MSR transfer content CPSR/SPSR to /from a general purpose register.
 - All of status register, or just the flags, can be transferred.
 - `mrs Rd, <psr> ; Rd ← <psr>`
 - `msr <psr>,Rm ; <psr> ← Rm`
 - `msr <psrf>,Rm ; <psrf> ← Rm`
- Where `<psr>` = CPSR, CPSR_all, SPSR or SPSR_all and `<psrf>` = CPSR_flg or SPSR_flg
- Also an immediate form
 - `msr <psrf>,#Imm32`
 - a 32-bit immediate, of which the 4 most significant bits are written to the flag bits.

ELEC2041 lec27-exception-L24

Saeid Nooshabadi

Modifying CPSR



- **Unused reserved bits, may be used in future, therefore:**
 - they must be preserved when altering PSR
 - the value they return must not be relied upon when testing other bits.
- **Thus read-modify-write strategy must be followed when modifying any PSR:**
 - Transfer PSR to register using MRS
 - Modify relevant bits
 - Transfer updated value back to PSR using MSR
- **Note:**
 - In User Mode, all bits can be read but only the flag bits can be written to.

ELEC2041 lec27-exception-1.25

Saeid Nooshabadi

Questions Raised about Interrupts

- **Which I/O device caused interrupt?**
 - Needs to convey the identity of the device generating the interrupt
- **Can avoid interrupts during the interrupt routine?**
 - What if more important interrupt occurs while servicing this interrupt?
 - Allow interrupt routine to be entered again?
- **Who keeps track of status of all the devices, handle errors, know where to put/supply the I/O data?**

ELEC2041 lec27-exception-1.26

Saeid Nooshabadi

4 Responsibilities leading to OS

- **The I/O system is shared by multiple programs using the processor**
- **Low-level control of I/O devices is complex because requires managing a set of concurrent events and because requirements for correct device control are often very detailed**
- **I/O systems often use interrupts to communicate information about I/O operations**
- **Would like I/O services for all user programs under safe control**

ELEC2041 lec27-exception-1.27

Saeid Nooshabadi

4 Functions OS must provide

- **OS guarantees that user's program accesses only the portions of I/O device to which user has rights (e.g., file access)**
- **OS provides abstractions for accessing devices by supplying routines that handle low-level device operations**
- **OS handles the interrupts generated by I/O devices (and other exceptions generated by a program)**
- **OS tries to provide equitable access to the shared I/O resources, as well as schedule accesses in order to enhance system performance**

ELEC2041 lec27-exception-1.28

Saeid Nooshabadi

Things to Remember

- **Privileged Mode v. 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, interrupt handler must:**
 - **service all interrupts (no drops)**
 - **service by priority**
 - **make all users believe that no interrupt has occurred**