Lecture 11 Exceptions and Interrupts

- ARM processor can work in one of many **operating modes**. So far we have only considered **user mode**, which is the "**normal**" mode of operation.
- The processor can also enter "**privileged**" operating modes which are used to handle **exceptions** and **supervisor calls** (i.e. software interrupts SWI's)
- The Current Processor Status Register **CPSR** has 5 bits [bit4:0] to indicate which mode the processor is in:-

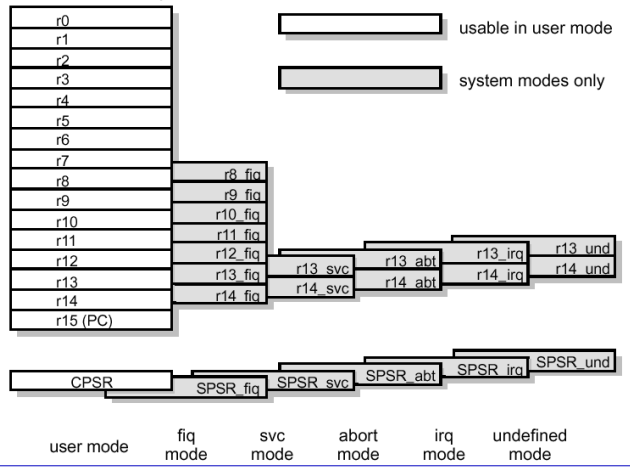
CPSR[4:0]	Mode	Use	Registers
10000	User	Normal user code	user
10001	FIQ	Processing fast interrupts	_fiq
10010	IRQ	Processing standard interrupts	_irq
10011	SVC	Processing software interrupts (SWIs)	_svc
10111	Abort	Processing memory faults	_abt
11011	Undef	Handling undefined instruction traps	_und
11111	System	Running privileged operating system tasks	user

How are exceptions generated?

- By default, the processor is usually in user mode
- It enters one of the exception modes when unexpected events occurs.
- There are three different types of exceptions (some are called interrupts):-
  - 1. As a direct result of executing an instruction, such as:
    - Software Interrupt Instruction (SWI)
    - Undefined or illegal instruction
    - Memory error during fetching an instruction
  - 2. As a side-effect of an instruction, such as:
    - Memory fault during data read/write from memory
    - Arithmetic error (e.g. divide by zero)
  - 3. As a result of external hardware signals, such as:
    - Reset
    - Fast Interrupt (FIQ)
    - Normal Interrupt (IRQ)

Shadow Registers

- As the processor enters an exception mode, some new registers are automatically switched in:-



Shadow Registers (con't)

- For example, an external event (such as movement of the mouse) occurs that generates a Fast Interrupt (on the FIQ pin), the processor enters FIQ operating mode.
- It sees the same r0 - r7 as before, but sees a new set of r8 - r14, and in addition, an extra register called the Saved Processor Status Register (SPSR).
- By swapping to some new registers, it makes it easier for the programmer to preserve the state of the processor. For example, during FIQ mode, r8 - r14 can be used freely. On returning back to user mode, the original values of r8 - r14 will be automatically restored.

What happens when an exception occurs?

- ARM completes current instruction as best it can.
- It departs from current instruction sequence to handle the exception by performing the following steps:-
  - 1. It **changes the operating mode** corresponding to the particular exception.
  - 2. It **saves the current PC** in the r14 corresponding to the new mode. For example, if FIQ occurs, the PC value is stored in r14(FIQ).
  - 3. It **saves the old value of CPSR** in the Saved Processor Status Register of the new mode.
  - 4. It **disables exceptions of lower priority** (to be considered later).
  - 5. It forces the PC to a new value corresponding to the exception. This is effectively a forced jump to the **Exception Handler** or **Interrupt Service Routine**.

Where is the exception handler routine?

- Exceptions can be viewed as "forced" subroutine calls.
  - When and if an exception occurs is not predictable (unless it is a SWI exception).
  - A unique address is pre-defined for each exception handler (IRQ, FIQ, etc), and a branch is made to this address.
  - The address to which the processor is forced to branch to is called the **exception/interrupt vector**.

Exception vector addresses

- Each vector (except FIQ) is 4 bytes long (i.e. one instruction)
- You put a branch instruction at this address:

```
B          exception_handler
```

- FIQ is special in two ways:-
  - 1. You can put the actual FIQ handler (also called Fast Interrupt Service Routine) at 0x0000001C onwards, because FIQ vector occupies the highest address
  - 2. FIQ has many more shadow registers. So you don't have to save as many registers on the stack as other exceptions - faster.

Exception	Mode	Vector address
Reset	SVC	0x00000000
Undefined instruction	UND	0x00000004
Software interrupt (SWI)	SVC	0x00000008
Prefetch abort (instruction fetch memory fault)	Abort	0x0000000C
Data abort (data access memory fault)	Abort	0x00000010
IRQ (normal interrupt)	IRQ	0x00000018
FIQ (fast interrupt)	FIQ	0x0000001C

Exception Return

- Once the exception has been handled (by the exception handler), the user task is resumed.
- The handler program (or Interrupt Service Routine) must restore the user state exactly as it was before the exception occurred:
  - 1. Any modified user registers must be restored from the handler's stack
  - 2. The CPSR must be restored from the appropriate SPSR
  - 3. PC must be changed back to the instruction address in the user instruction stream
- Steps 1 and 3 are done by user, step 2 by the processor
- Restoring registers from the stack would be the same as in the case of subroutines
- Restoring PC value is more complicated. The exact way to do it depends on which exception you are returning from.

Exception Return (con't)

- We assume that the return address was saved in r14 before entering the exception handler.
- To return from a SWI or undefined instruction trap, use:
  - MOVS pc, r14
- To return from an IRQ, FIQ or prefetch abort, use:
  - SUBS pc, r14, #4
- To return from a data abort to retry the data access, use:
  - SUBS pc, r14, #8
- Note the 'S' modifier is NOT used to set the flags, but instead to restore the CPSR, if the destination register is the PC.
- The differences between these three methods of return is due to the pipeline architecture of the ARM processor. The PC value stored in r14 can be one or two instructions ahead due to the instruction prefetch pipeline.

Exception Priorities

- Since exceptions can arise at the same time, a priority order has to be clearly defined. For the ARM processor this is:
  - Reset (highest priority)
  - Data abort (i.e. Memory fault in read/write data)
  - Fast Interrupt Request (FIQ)
  - Normal Interrupt Request (IRQ)
  - Prefetch abort
  - Software Interrupt (SWI), undefined instruction
- Consider the case when a FIQ and an IRQ occurring at the same time. The processor will process the FIQ handler first and "remember" that there is IRQ pending.
- On return from FIQ, the process will immediately go to the IRQ handler.