Exception Handling and the Vector Table

- When an exception occurs,
  - return link → R14_<exception_mode>
  - CPSR → SPSR_<mode>
  - CPSR[4:0] = exception mode number
  - CPSR[5] = 0 /* Execute in ARM state */
  - if <exception_mode> == Reset or FIQ then CPSR[6] = 1 /* Disable fast interrupts */
  - CPSR[7] = 1 /* Disable normal interrupts */
  - PC ← exception vector address
## Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Mode</th>
<th>Priority</th>
<th>IV Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>Supervisor</td>
<td>1</td>
<td>0x0000000000</td>
</tr>
<tr>
<td>Undefined instruction</td>
<td>Undefined</td>
<td>6</td>
<td>0x0000000004</td>
</tr>
<tr>
<td>Software interrupt</td>
<td>Supervisor</td>
<td>6</td>
<td>0x0000000008</td>
</tr>
<tr>
<td>Prefetch Abort</td>
<td>Abort</td>
<td>5</td>
<td>0x0000000010</td>
</tr>
<tr>
<td>Data Abort</td>
<td>Abort</td>
<td>2</td>
<td>0x0000000018</td>
</tr>
<tr>
<td>Interrupt</td>
<td>IRQ</td>
<td>4</td>
<td>0x000000001C</td>
</tr>
<tr>
<td>Fast interrupt</td>
<td>FIQ</td>
<td>3</td>
<td>0x000000001C</td>
</tr>
</tbody>
</table>

Table 1 - Exception types, sorted by Interrupt Vector addresses
Exit from Exception Handler

➢ To return, exception handler needs to:
  ● CPSR from SPSR_<mode>
  ● Moves the LR_<mode>, minus an offset where appropriate, to the PC. (The offset will vary depending on the type of exception.)
  ● Clears the interrupt disable flags, if they were set on entry
  ● Restore CPSR and resume PC can’t be carried out independently.

➢ Two ways to solve the problems above
  ● using a data-processing instruction with the S bit
  ● using the Load Multiple with Restore CPSR instruction
When Reset is de-asserted,

- \( R14_{\text{svc}} = \text{UNPREDICTABLE value} \)
- \( SPSR_{\text{svc}} = \text{UNPREDICTABLE value} \)
- \( \text{CPSR}[4:0] = 0b10011 \quad /* \text{Enter Supervisor mode} */ \)
- \( \text{CPSR}[5] = 0 \quad /* \text{Execute in ARM state} */ \)
- \( \text{CPSR}[6] = 1 \quad /* \text{Disable fast interrupts} */ \)
- \( \text{CPSR}[7] = 1 \quad /* \text{Disable normal interrupts} */ \)
- if high vectors configured then \( \text{PC}=0xFFFF0000 \) else \( \text{PC} = 0x00000000 \)
Undefined Instruction exception

- When an Undefined Instruction exception occurs (can be used for the simulation of the nonexistent coprocessor)
  - R14_und = address of undefined instruction + #4
  - SPSR_und = CPSR
  - CPSR[4:0] = 0b11011/* Enter Undefined mode */
  - CPSR[5] = 0/* Execute in ARM state */
  - CPSR[7] = 1/* Disable normal interrupts */
  - if high vectors configured then PC = 0xFFFF0004 else PC = 0x00000004

- To return from exception handler
  - MOVs PC,R14/* This restores both the PC (from R14_und) and CPSR (from SPSR_und)*/
Software Interrupt exception

- When a SWI is executed
  - R14_svc = address of next instruction after the SWI instruction
  - SPSR_svc = CPSR
  - CPSR[4:0] = 0b10011 /* Enter Supervisor mode */
  - CPSR[5] = 0 /* Execute in ARM state */
  - CPSR[7] = 1 /* Disable normal interrupts */
  - if high vectors configured then PC=0xFFFF0008 else PC=0x00000008

- To return from this exception
  - MOVS PC, R14 /* This restores both the PC (from R14_svc) and CPSR (from SPSR_svc)*/
A Prefetch Abort exception is generated if the processor tries to execute the invalid instruction.

If the instruction is not executed (for example, as a result of a branch being taken while it is in the pipeline), no Prefetch Abort occurs.

When an attempt is made to execute an aborted instruction:
- R14_abt = address of the aborted instruction + 4
- SPSR_abt = CPSR
- CPSR[4:0] = 0b10111 /* Enter Abort mode */
- CPSR[5] = 0 /* Execute in ARM state */
- CPSR[7] = 1 /* Disable normal interrupts */
- if high vectors configured then PC=0xFFFF000C else PC=0x0000000C

To return from exception:
- SUBS PC,R14,#4 /* This restores both the PC (from R14_abt) and CPSR (from SPSR_abt)*/
Data Abort (data fetch memory abort)

- A Data Abort exception occurs before any following instructions or exceptions have altered the state of the CPU.
  - R14_abt = address of the aborted instruction + 8
  - SPSR_abt = CPSR
  - CPSR[4:0] = 0b10111 /* Enter Abort mode */
  - CPSR[5] = 0 /* Execute in ARM state */
  - CPSR[7] = 1 /* Disable normal interrupts */
  - if high vectors configured then PC=0xFFFFF0010 else PC= 0x00000010

- To return from data abort exception
  - SUBS PC,R14,#8
  - If the aborted instruction does not need to be re-executed
    - SUBS PC,R14,#4
Interrupt request (IRQ) exception

- When an IRQ is detected
  - R14_irq = address of next instruction to be executed + 4
  - SPSR_irq = CPSR
  - CPSR[4:0] = 0b10010              /* Enter IRQ mode */
  - CPSR[5] = 0                    /* Execute in ARM state */
  - CPSR[7] = 1                    /* Disable normal interrupts */
  - if high vectors is configured then PC=0xFFFF0018 else PC= 0x00000018

- To return after servicing the interrupt
  - SUBS PC,R14,#4
Fast interrupt request (FIQ) exception

- When an FIQ is detected
  - R14_fiq = address of next instruction to be executed + 4
  - SPSR_fiq = CPSR
  - CPSR[4:0] = 0b10001 /* Enter FIQ mode */
  - CPSR[5] = 0 /* Execute in ARM state */
  - CPSR[6] = 1 /* Disable fast interrupts */
  - CPSR[7] = 1 /* Disable normal interrupts */
  - if high vectors configured then PC=0xFFFF001C else PC=0x0000001C

- To return after servicing the interrupt
  - SUBS PC, R14, #4
## Exceptions Exit

<table>
<thead>
<tr>
<th>Exception</th>
<th>R14_mode (ARM)</th>
<th>R14_mode (THUMB)</th>
<th>Return instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Undefined instruction</td>
<td>PC+4</td>
<td>PC+2</td>
<td>MOVS PC,R14_und</td>
</tr>
<tr>
<td>Software interrupt</td>
<td>PC+4</td>
<td>PC+2</td>
<td>MOVS PC, R14_svc</td>
</tr>
<tr>
<td>Prefetch Abort</td>
<td>PC+4</td>
<td>PC+4</td>
<td>SUBS PC, R14_abt, #4</td>
</tr>
<tr>
<td>Data Abort</td>
<td>PC+8</td>
<td>PC+8</td>
<td>SUBS PC, R14_abt, #8</td>
</tr>
<tr>
<td>Interrupt</td>
<td>PC+4</td>
<td>PC+4</td>
<td>SUBS PC, R14_irq, #4</td>
</tr>
<tr>
<td>Fast interrupt</td>
<td>PC+4</td>
<td>PC+4</td>
<td>SUBS PC, R14_fiq, #4</td>
</tr>
</tbody>
</table>
GNU Function Attribute

- **interrupt**
  - Use this attribute on the ARM, AVR, CRX, M32C, M32R/D, m68k, and Xstormy16 ports to indicate that the specified function is an interrupt handler. The compiler will generate function entry and exit sequences suitable for use in an interrupt handler when this attribute is present. Note, interrupt handlers for the Blackfin, H8/300, H8/300H, H8S, and SH processors can be specified via the interrupt_handler attribute.
  - Note, on the AVR, interrupts will be enabled inside the function.
  - Note, for the ARM, you can specify the kind of interrupt to be handled by adding an optional parameter to the interrupt attribute like this:

```c
void f () __attribute__ ((interrupt ("IRQ")));  
```
  - Permissible values for this parameter are: IRQ, FIQ, SWI, ABORT and UNDEF.
  - On ARMv7-M the interrupt type is ignored, and the attribute means the function may be called with a word aligned stack pointer.