COL216 Computer Architecture

Lab Assignments 12 Lite

ARM CPU with Exception Handling, with Single Mode

1. Objective

The objective of this assignment is to augment ARM processor design with exception handling capability, without introducing a privileged mode.

2. Scope

The scope of this assignment includes only 4 exceptions (Reset, Undefined, SWI and IRQ) out of a total 7. The scope also includes implementation of two instructions (MRS, MSR).

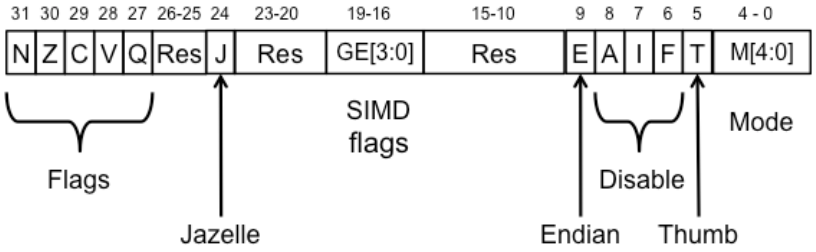
3. Exceptions and Modes

Exceptions defined for ARM and their exception vectors (addresses of handlers) are shown in the table below. The exception modes are also shown, but not to be implemented.

Exception type	Mode	Address
Reset	Supervisor	0x00000000
Undef instr	Undefined	0x00000004
S/W interrupt	Supervisor	0x00000008
Prefetch Abort	Abort	0x0000000C
Data Abort	Abort	0x00000010
Interrupt	IRQ	0x00000018
Fast interrupt	FIQ	0x0000001C

4. Status Registers

Format of status registers, CPSR (current program status register) and SPSR (saved program status register) is shown below.

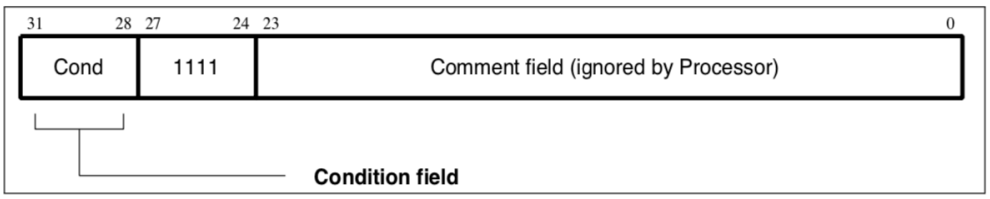


Bits and fields relevant to this assignment are – N, Z, C, V, and I. Bit I is used for disabling IRQ. Other bits will be taken as constant '0'.

CPSR holds the current status and SPSR is used to save CPSR on entering exception handler.

5. Detecting Exceptions

Undefined and *SWI* exceptions are detected by looking at the instruction bits during the *decode* cycle. Detecting an instruction that is not implemented is the cause of *Undefined* exception. SWI instruction (format shown below) causes *SWI* exception. We will use three LSBs of the comment field to specify the particular function to be performed by SWI handler.



IRQ and *Reset* exceptions are detected by looking at external signals. *Reset* signal is checked in every cycle and the current instruction is abandoned, whereas *IRQ* signal is checked in the last cycle of instruction execution (except for BL instruction, in order to prevent overwriting of LR register), thus letting the current instruction finish. *Undefined* and *SWI* are mutually exclusive and do not conflict with each other.

6. Actions Required on Exception

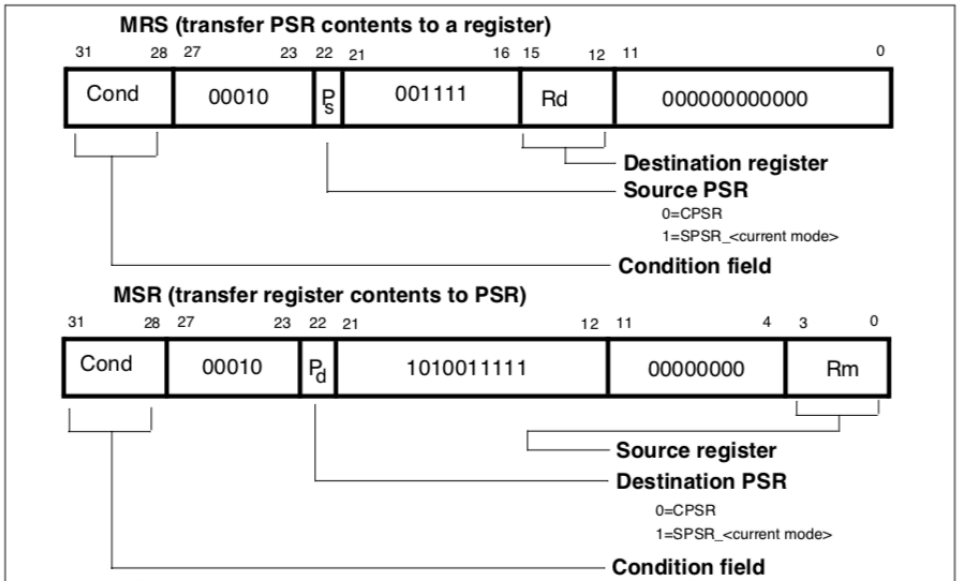
The hardware needs to take the following actions on seeing an exception.

- Save return address in R14 (not required for Reset)
- Save CPSR in SPSR (not required for Reset)
- Transfer exception handler's address to PC
- Set I bit of CPSR to 1 (IRQ is disabled)

Return from exception handler is done by using the instruction MOVS PC, R14. This causes R14_svc to be transferred to PC and SPSR to be transferred to CPSR. Functionality of MOV instruction needs to be augmented to include the second transfer.

7. MRS and MSR instructions

Format of the two instructions to be implemented is shown below. MRS copies status register (CPSR or SPSR) to Rd and MSR copies Rm to status register (CPSR or SPSR). Recall that except for the 5 bits mentioned earlier, the remaining bits of CPSR/SPSR are constant '0'.



There is one more form of MSR instruction that is not to be implemented here.

Summary

See the text shown in green to get a summary of what needs to be done.