COL216 Computer Architecture

Lab Assignments 12

ARM CPU with Exception Handling

1. Objective

The objective of this assignment is to augment ARM processor design with exception handling capability.

2. Scope

There are 7 types of exceptions defined for ARM. The scope of this assignment includes only 4 of these (Reset, Undefined, SWI and IRQ). The scope also includes implementation of two instructions (MRS, MSR) and some elementary protection function.

3. Exceptions and Modes

Exceptions defined for ARM are shown in the table below. The table also shows the corresponding exception vectors (addresses of handlers) and the modes entered on getting exceptions.

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

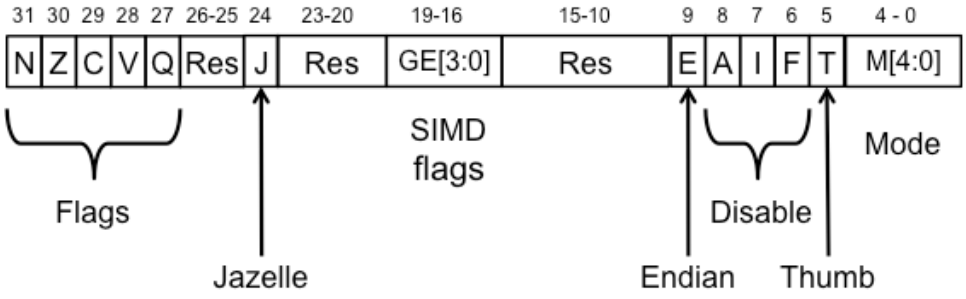
List of all modes and their 5 bit representation is shown in the table below

Processor Mode		Encoding	Description
User	usr	0b10000	Normal program execution mode
FIQ	fiq	0b10001	Supports a high-speed data transfer or channel process
IRQ	irq	0b10010	Used for general-purpose interrupt handling
Supervisor	svc	0b10011	A protected mode for the operating system
Abort	abt	0b10111	Implements virtual memory and/or memory protection
Undefined	und	0b11011	Supports software emulation of hardware coprocessors
System	sys	0b11111	Runs privileged operating system tasks

As mentioned above, only 4 exceptions need to be implemented here. To simplify the design further, only one privileged mode, namely *supervisor mode*, will be used for all 4 exceptions. Apart from this, there will be *user mode*, the non-privileged mode.

4. Status Registers and supervisor mode registers

Format of status registers is shown below.

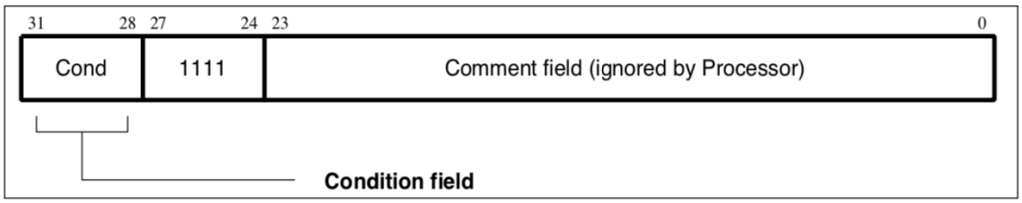


Bits and fields relevant to this assignment are – N, Z, C, V, I and M. Bit I is used for disabling IRQ. Other bits will be taken as constant '0'.

A register called current program status register (CPSR) holds the current status in any mode. There are 3 additional registers accessible only in supervisor mode – R13_svc, R14_svc and SPSR_svc (saved program status register). Function of these registers is as follows. R13_svc is used to maintain a separate stack for supervisor mode, R14_svc is used to save the address where the control has to return after exception, and SPSR_svc is used to save the CPSR value. In supervisor mode, instructions referring to R13 and R14 automatically refer to R13_svc and R14_svc.

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, thus letting the current instruction finish. *Undefined* and *SWI* are mutually exclusive and do not conflict with each other.

6. Actions Required on Exception

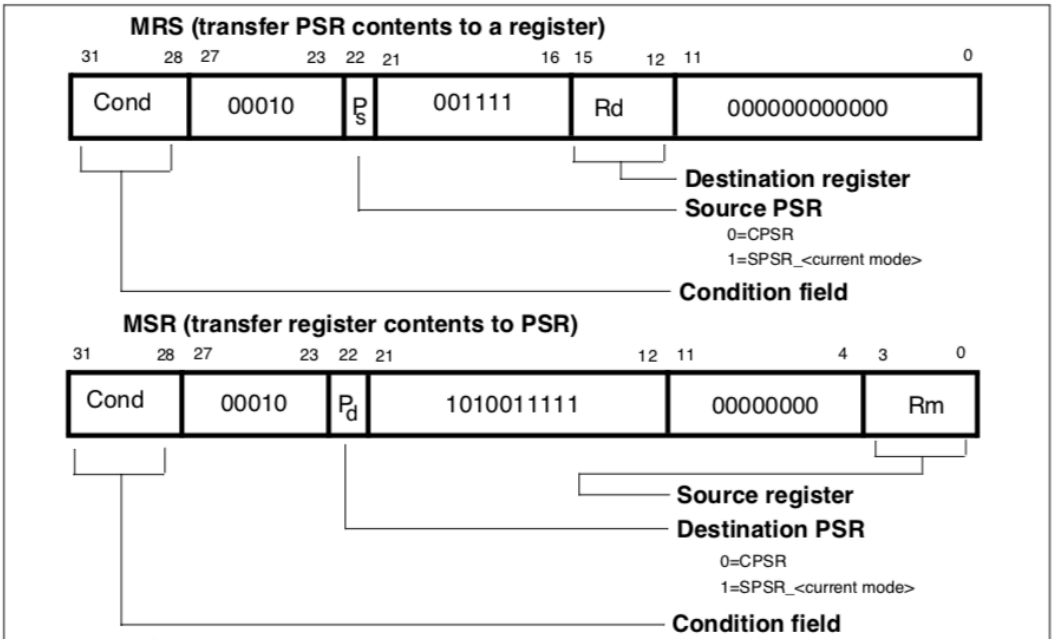
The hardware needs to take the following actions on seeing an exception.

- Save return address in R14_svc (not required for Reset)
- Save CPSR in SPSR_svc (not required for Reset)
- Transfer exception handler's address to PC
- Set M field of CPSR to new mode
- 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_svc 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 10 bits mentioned earlier, the remaining bits of CPSR/SPSR are constant '0'. In user mode, there is no SPSR. Further, in user mode, MSR copies only 4 MSBs of Rm to NZCV bits. This ensures that a user program cannot change the mode field.



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

8. Memory Map and Protection

Use the following memory map (assuming address space of 0x000 to 0x FFF).

0x000 – 0x01F	Exception vectors
0x020 – 0x03F	I/O ports or registers
0x040 – 0x2FF	Supervisor code + data
0x300 – 0x3FF	Supervisor stack
0x400 – 0xDFF	User code + data
0xE00 – 0xFFF	User stack

In user mode prevent access to address range 0x000 to 0x3FF.

Summary

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