CS1022 Tutorial #6
Exceptions

(a) Describe in detail how the ARM7TDMI microprocessor handles the occurrence of an undefined instruction exception with reference to the memory listing shown below.

```
Label        Address   Contents        Disassembly
----------   --------  --------        -----------
             00000000  E59FF018        LDR PC, Reset_Addr
             00000004  E59FF018        LDR PC, Undef_Addr
             00000008  E59FF018        LDR PC, SWI_Addr
             0000000C  E59FF018        LDR PC, PAbt_Addr
             00000010  E59FF018        LDR PC, DAbt_Addr
             00000014  E1A00000        NOP
             00000018  E51FF120        LDR PC, IRQ_Addr
             0000001C  E59FF018        LDR PC, FIQ_Addr
Reset_Addr   00000020  A0001000
Undef_addr   00000024  A0002000
SWI_Addr     00000028  A0003000
PAbt_Addr    0000002C  A0004000
DAbt_Addr    00000030  A0005000
             00000034  00006000
IRQ_Addr     00000038  A0007000
FIQ_Addr     0000003C  A0008000
             ........  ........        ...
Reset_Handler
             A0001000  ????????
             ........  ........        ...
Undef_Handler
             A0002000  ????????
             ........  ........        ...
FIQ_Handler
             A0001000  ????????
```

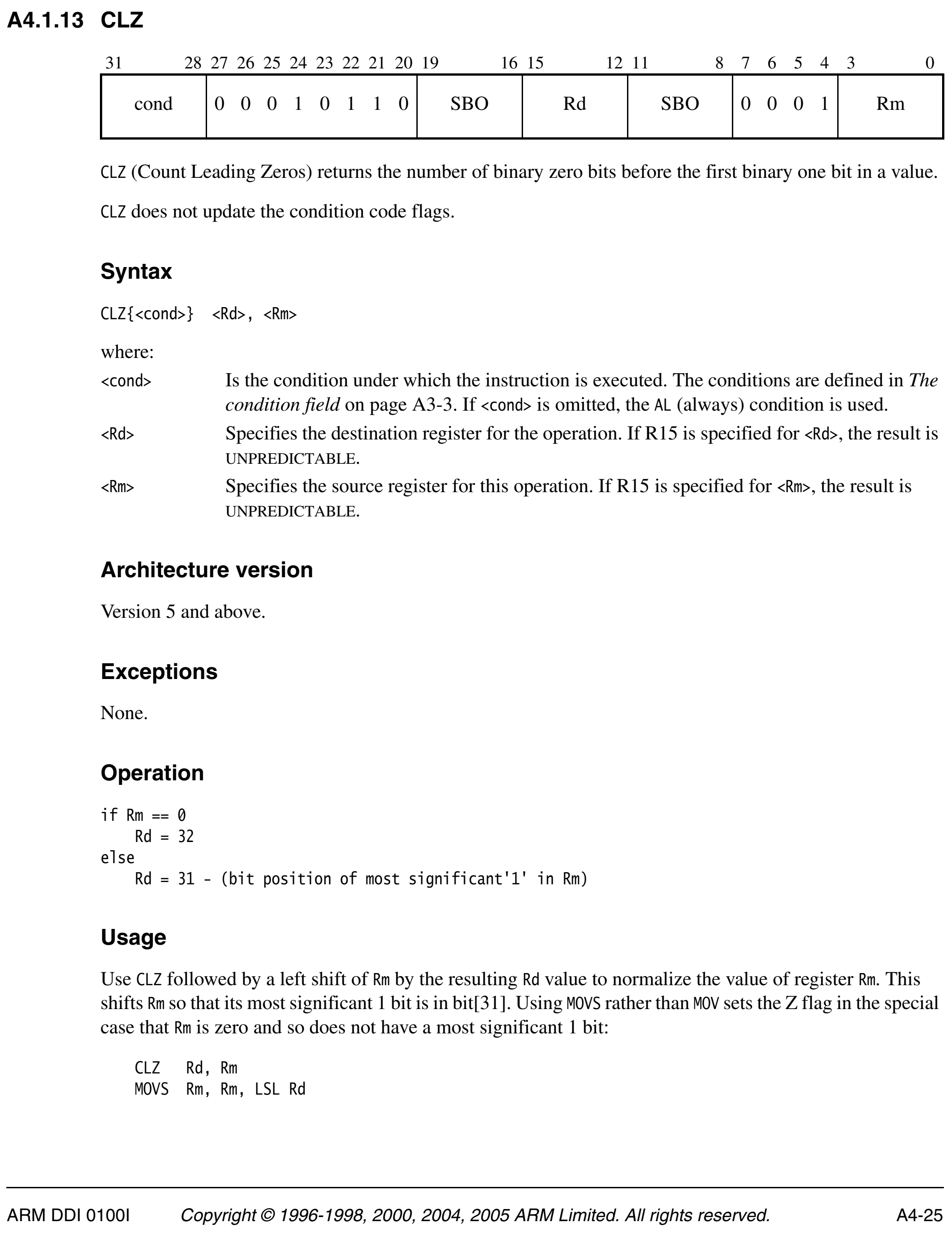
(b) The ARM Architecture instruction set from version 5 onwards contains a `CLZ` instruction that counts the number of leading zeros (number of zero bits to the left of the most significant 1) in a register value. The documentation for this instruction is provided at the end of this handout. (Note: In the documentation for `CLZ`, SBO is an abbreviation for Should Be One.)

Design and write an Undefined Instruction exception handler that will emulate the `CLZ` instruction for ARM Architecture versions prior to version 5. Your exception handler must adhere to the instruction template provided in the documentation. You may ignore the conditional execution of the instruction and the use of R13 and R14 as operands (for now!).

(c) Consider how you would modify the contents of memory listed above in part (a) to cause your new undefined instruction exception handler to be executed and provide a program to do so.

(d) Consider how you would modify your exception handler from part (b) to conditionally execute the CLZ instruction (e.g. if the condition corresponds to `CLZEQ`, you should only execute the instruction if the Zero flag is set.)

(e) Consider how you would modify your exception handler from part (b) to handle the use of R13 and R14 as operands to the emulated CLZ instruction.



A4.1.13 CLZ

31–28	27 26 25 24 23 22 21 20	19–16	15–12	11–8	7 6 5 4	3–0
cond	0 0 0 1 0 1 1 0	SBO	Rd	SBO	0 0 0 1	Rm

CLZ (Count Leading Zeros) returns the number of binary zero bits before the first binary one bit in a value.

CLZ does not update the condition code flags.

Syntax

```
CLZ{<cond>}  <Rd>, <Rm>
```

where:

<cond>	Is the condition under which the instruction is executed. The conditions are defined in *The condition field* on page A3-3. If <cond> is omitted, the AL (always) condition is used.
<Rd>	Specifies the destination register for the operation. If R15 is specified for <Rd>, the result is UNPREDICTABLE.
<Rm>	Specifies the source register for this operation. If R15 is specified for <Rm>, the result is UNPREDICTABLE.

Architecture version

Version 5 and above.

Exceptions

None.

Operation

```
if Rm == 0
    Rd = 32
else
    Rd = 31 - (bit position of most significant'1' in Rm)
```

Usage

Use CLZ followed by a left shift of Rm by the resulting Rd value to normalize the value of register Rm. This shifts Rm so that its most significant 1 bit is in bit[31]. Using MOVS rather than MOV sets the Z flag in the special case that Rm is zero and so does not have a most significant 1 bit:

```
    CLZ   Rd, Rm
    MOVS  Rm, Rm, LSL Rd
```

ARM DDI 0100I *Copyright © 1996-1998, 2000, 2004, 2005 ARM Limited. All rights reserved.* A4-25