EXAMPLE PROGRAM DECIMAL TO BINARY CONVERTER

1. Requirements

Develop and implement some Ada software which, when given a base 10 (decimal) natural number, will convert this integer into a binary number. For example, given the number 41 we would expect the system to output:

```
00000000 00000000 00000000 00101001
```

The standard mechanism, for converting from decimal to binary is to repeatedly divide the decimal number by 2 and, at each division, output the remainder (0 or 1). An example is given below where the binary number 101001 is derived from the decimal number 41.

```
REM                41÷2
1       <--        20÷2
0       <--        10÷2
0       <--         5÷2
1       <--         2÷2
0       <--         1÷2
1       <--         0
```

i.e. 41 (natural) = 101001 (binary).

2. Design

A top down analysis of the problem is given in Table 1.

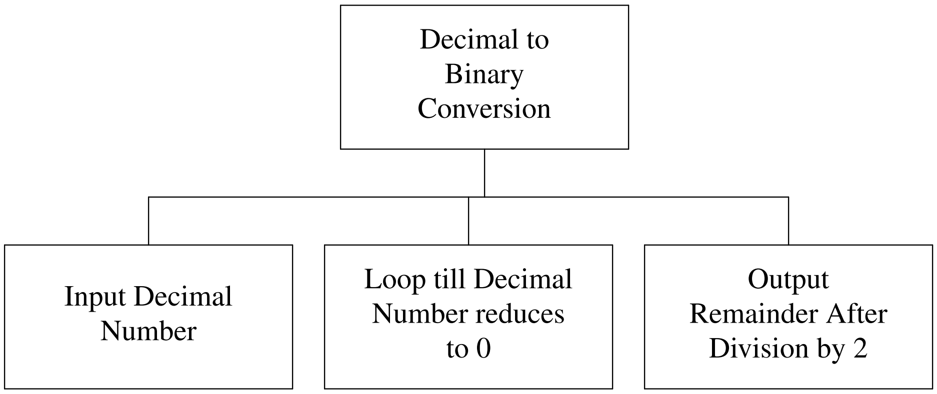

Table 1: *Top down analysis*

The operations given in Table 1 may be implemented using the following procedures:

1) `DECIMAL_2_BINARY` (top level procedure): Input decimal number, call `CONVERT`.

Name	Description	Type	Range
NATURAL_NUMBER	Global input variable	NATURAL	Default

2) `CONVERT` (Level 2 procedure): Convert to binary and output result. Call to `PAD_WITH_ZEROES` when necessary.

Name	Description	Type	Range
DECL_NUM	Formal parameter	NATURAL	Default
DIGIT_COUNTER	Formal parameter	NATURAL	Default

3) `PAD_WITH_ZEROES` (Level 3 procedure): Pad output with appropriate number of zeroes.

Name	Description	Type	Range
DIG_COUNT	Formal parameter	NATURAL	Default
END_VALUE	Local variable	INTEGER	Default

Appropriate Nassi-Shneiderman design is given in Table 2.

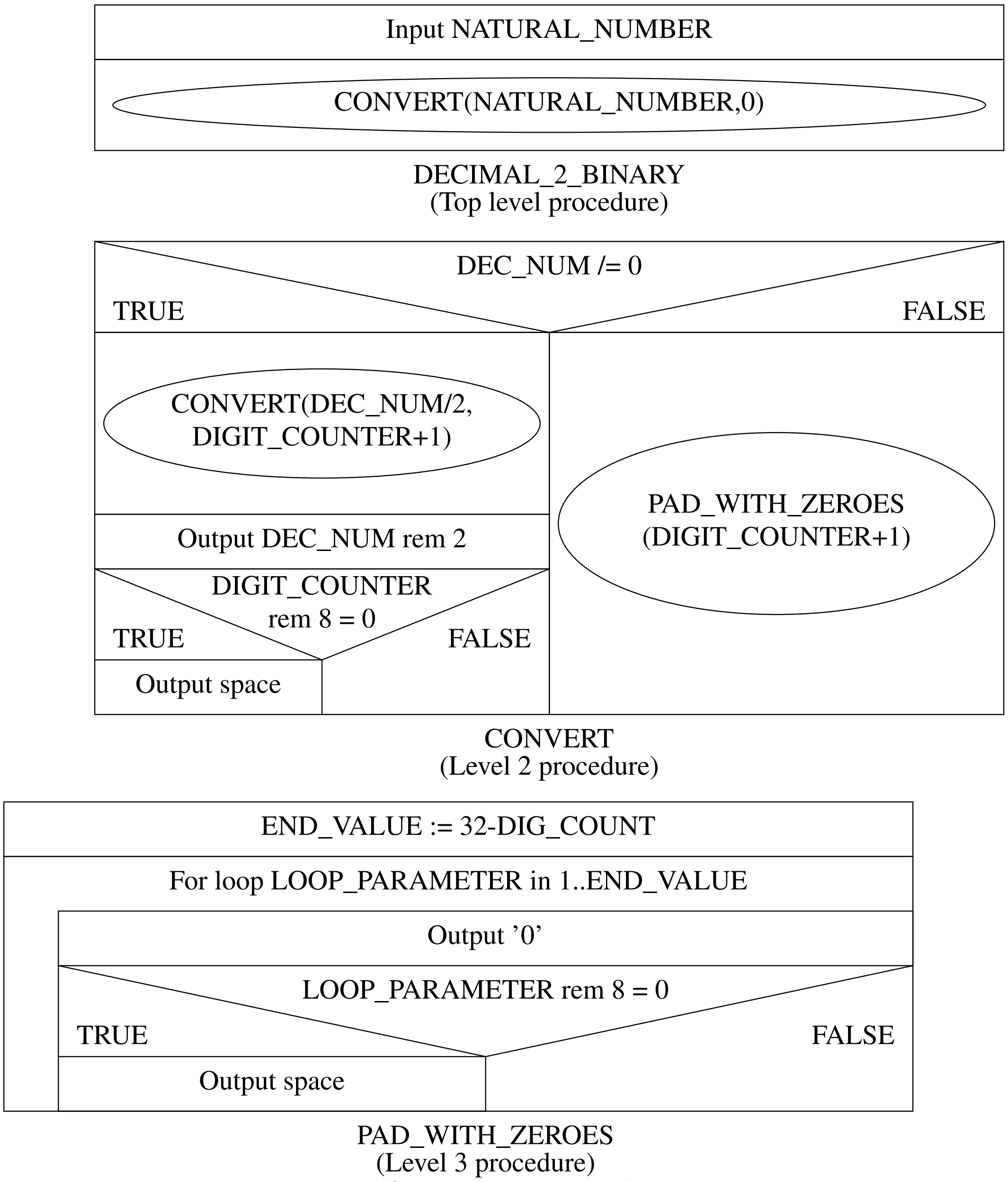

Table 2: *Nassi-Shneiderman chart*

Here we have used a recursive algorithm to ensure that the binary digits are output in the correct order. A flow chart indicating the overall flow of control through the proposed software is given in Table 3.

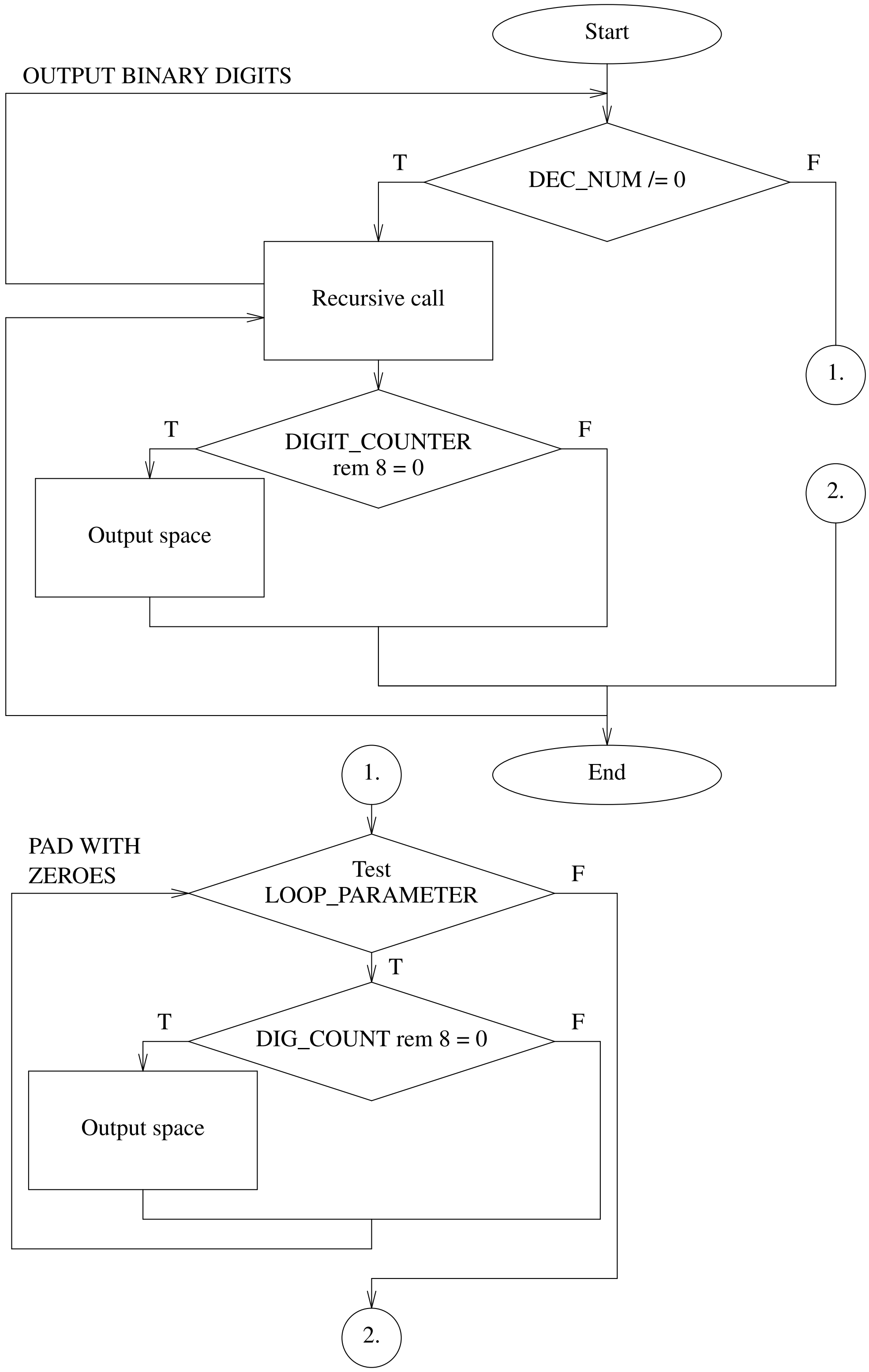

Table 3: *Flow chart*

3. Implementation

The encoding of the above design is given in Table 4.

```
-- DECIMAL 2 BINARY
-- 17 September 1997
-- Frans Coenen
-- Dept Computer Science, University of Liverpool

with TEXT_IO;
use TEXT_IO;

procedure DECIMAL_2_BINARY is
    package INTEGER_INOUT is new INTEGER_IO(INTEGER);
    use INTEGER_INOUT;
    NATURAL_NUMBER: NATURAL;

    -----------------------------------------------------------
    -- CONVERT
    procedure CONVERT(DEC_NUM, DIGIT_COUNTER: NATURAL) is
        -------------------------------------------------------
        -- PAD WITH ZEROES
        procedure PAD_WITH_ZEROES(DIG_COUNT: NATURAL) is
            END_VALUE: INTEGER;
        begin
            END_VALUE := 32-DIG_COUNT;
            for LOOP_PARAMETER in 1..END_VALUE loop
                PUT("0");
                if LOOP_PARAMETER rem 8 = 0 then
                    put(" ");
                end if;
            end loop;
        end PAD_WITH_ZEROES;
        -------------------------------------------------------
    begin
    -- Output digits and spaces between bytes when appropriate
        if DEC_NUM /= 0 then
            CONVERT(DEC_NUM/2, DIGIT_COUNTER+1);
            PUT(DEC_NUM rem 2, WIDTH=>1);
            if (DIGIT_COUNTER) rem 8 = 0 then
                put(" ");
            end if;
        else
            PAD_WITH_ZEROES(DIGIT_COUNTER);
        end if;
    end CONVERT;
    -----------------------------------------------------------

-- TOP LEVEL
begin
    PUT("Input a natural number: ");
    GET(NATURAL_NUMBER);
    NEW_LINE;
    PUT(NATURAL_NUMBER);
    PUT(" = ");
    CONVERT(NATURAL_NUMBER,0);
    NEW_LINE;
end DECIMAL_2_BINARY;
```

Table 4: *Implementation*

4. Testing

4.1. Black box testing

BVA, Limit and Arithmetic testing: Use BVA testing to analyse nature of inputs. Limit testing to test operation of the software at the limits, and arithmetic testing to test the arithmetic expressions. A suggested set of test cases is given in Table 5. The results are presented in Table 6.

TEST CASE NATURAL_NUMBER	EXPECTED RESULT OUTPUT
-1	CONSTRAINT_ERROR
0	00000000 00000000 00000000 00000000
1	00000000 00000000 00000000 00000001
1073741823	00111111 11111111 11111111 11111111
2147483646	01111111 11111111 11111111 11111110
2147483647	01111111 11111111 11111111 11111111
2147483648	DATA_ERROR

Table 5: BVA, Limit and Arithmetic Test Cases

```
kuban-654 $ decimal_2_binary
Input a natural number: -1
Ada-runtime: Exception CONSTRAINT_ERROR raised in
/tmp_mnt/home/staff2/ra/frans/public_html/2CS21/week8/decimal2binary.
ada on line 47.
Rangecheck error: -1 not in range 0..2147483647.

kuban-655 $ decimal_2_binary
Input a natural number: 0
          0 = 00000000 00000000 00000000 00000000

kuban-656 $ decimal_2_binary
Input a natural number: 1
          1 = 00000000 00000000 00000000 00000001

kuban-657 $ decimal_2_binary
Input a natural number: 1073741823
 1073741823 = 00111111 11111111 11111111 11111111

kuban-658 $ decimal_2_binary
Input a natural number: 2147483646
 2147483646 = 01111111 11111111 11111111 11111110

kuban-659 $ decimal_2_binary
Input a natural number: 2147483647
 2147483647 = 01111111 11111111 11111111 11111111

kuban-660 $ decimal_2_binary
Input a natural number: 2147483648
Ada-runtime: Exception DATA_ERROR propagated out of main.
kuban-661 $ decimal_2_binary
Input a natural number: -2147483648
```

Table 6: Results of BVA, Limit and Arithmetic Testing

4.2. White box testing.

Path Testing: We should test all paths identified in the flow chart given in Table 3. This is already achieved by the above test cases.

Loop Testing: The loop is a variable count loop therefore we should test the the loop with 1, 2 and some other number of passes through the loop. Again this will be achieved by the test cases given in Table 5.

4.3. Data Validation Testing

Results of running a number of data validation test cases are given in Table 7.

```
kuban-185 $ decimal_2_binary
Input a natural number: 176.9
        176 = 00000000 00000000 00000000 10110000

kuban-186 $ 9
ksh: 9:  not found

kuban-187 $ decimal_2_binary
Input a natural number: frans
Ada-runtime: Exception DATA_ERROR propagated out of main.

kuban-188 $ rans
ksh: rans:  not found
```

Table 7: *Data validation testing results*