9.1 Number Representation

***Note to tutors**: The important concept to bring across in this question is that a number can be represented (encoded) in many different ways. In the processor, these representations affect the complexity of hardware and the efficiency of computations. In algorithms, confusion in data type casting may result in disastrous failure.*

1. Figure 9.1 shows a 32-bit binary number (the top row shows the bit numbers and the bottom row shows the corresponding binary values). Find the decimal value of the 32-bit number if it is represented as:
   a. Unsigned Integer
   b. Signed-Magnitude
   c. Two's Complement
   d. IEEE 754 Single Precision

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9.1 – 32-bit number

[Suggested Solution]

a. Unsigned integer
$2^{31} + 2^{29} + 2^{28} + 2^{27} + 2^{26} + 2^{25} + 2^{24} + 2^{22}$ = **3,208,642,560**

b. Signed-Magnitude
The MSB indicates that this is a negative number.
$-(2^{29} + 2^{28} + 2^{27} + 2^{26} + 2^{25} + 2^{24} + 2^{22})$ = **-1,061,158,912**

c. Two's Complement
$-2^{31} + 2^{29} + 2^{28} + 2^{27} + 2^{26} + 2^{25} + 2^{24} + 2^{22}$ = **-1,086,324,736**

d. IEEE754 Single Precision
The IEEE 754 Single Precision representation is encoded as follow

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	Exponent (E)								Fractions (F)																						

Sign (S) = 1_2 (Negative)
Exponent (E) = $01111110_2 = 126_{10}$; E – Bias = $126_{10} - 127_{10} = -1_{10}$
1 + Fraction (F) = $(1.100\ldots00)_2 = 1.5_{10}$
Value in decimal = -1.5×2^{-1} = **- 0.75**

9.2 Integer Arithmetic

2. An array consisting of the length of 256 wires is given by L[0], L[1], … , L[255]. Describe a scheme to compute the average length of the 256 wires that will yield a result with the highest precision based on the following specifications:

    - 16-bit registers are used for storing the data and result.
    - Only Single-Precision and Fixed-Point arithmetic is used.
    - Maximum possible length of each wire is 0x3FF and is an integer.

    Illustrate your answer in the form of a mathematical expression and justify your answer.

[Suggested Solution]

- Unsigned arithmetic used to maximise the range since the wire length cannot be negative
- 16-bit => maximum 0xFFFF
- Max length of one wire = 0x400 => Max number of wire length that can be accumulated before the result will potentially overflow = 64
- Math Expression =
  {(L[0]+…+L[63])/64 + (L[64]+…+L[127])/64 +
  (L[128]+…+L[191])/64 + (L[192]+…+L[255])/64}/4
- Always perform Division last to avoid truncating the LSBs too early in the computation. But need to take note of overflow when using addition, subtraction and multiplication.

9.3 Pipelines

3. Consider a processor (not VIP) with 4 pipeline stages: Fetch Instruction (F), Decode (D), Execute (E) and Store (S). Assume that

   - Branch target address is calculated at the execute stage
   - Instruction length for every instruction is one word long
   - Each pipeline stage take 1 cycle to complete

   How many cycles does the code in Figure 9.2 take? Assume delay branching is not enabled.

```
      MOV   AR, #5      ; I1
      MOV   R0, #0x800  ; I2
      MOV   R1, #0x300  ; I3
Loop  SUB   [R0],[R1]   ; I4
      INC   R1          ; I5
      JDAR  Loop        ; I6
      ADD   R3, [R0]    ; I7
      MOV   [R1], R3    ; I8
```

Figure 9.2

[Suggested Solution]

- I1 to I3: 6 cycles (I1 will take 4 cycles to pass through the pipeline)
- I4 to I6. For the first four iterations, each iteration takes (3+2) cycles. This includes 2 cycles discarded in each iteration.
- I4 to I6. Fifth iteration takes 3 cycles since the branch is not taken so no instructions are discarded.
- I7 to I8: 2 cycles.
- Total = 6 + (3+2)*4 + 3 + 2 = 31 cycles.

4. Consider a processor (not VIP) with 4 pipeline stages: Fetch Instruction (F), Decode (D), Execute (E) and Store (S). Assume that

   - Branch target address is calculated at the execute stage
   - Instruction length for every instruction is one word long
   - Each pipeline stage takes 1 cycle to complete
   - No Resource Conflicts
   - Delayed Branching is enabled

   Identify and describe ALL pipeline conflicts the code in Figure 9.3 has when run in the pipeline processor above. Suggest workaround for pipeline conflicts identified.

```
      MOV   R3, #300    ; I1
      MOV   AR, #10     ; I2
Loop  JDAR  Loop        ; I3
      ADD   R1, [R3]    ; I4
      INC   R3          ; I5
      MOV   R1, R3      ; I6
      MOV   R0, R2      ; I7
```

Figure 9.3

[Suggested Solution]

- I2 and I3. Data dependency. AR register value is decremented before I2 completes the AR register store. Resolved by
  - swapping I1 and I2. OR
  - inserting a NOP instruction between I2 and I3
- I5 and I6. Data dependency. At the last iteration of the loop, I6 uses R3 before I5 can store the latest increment to R3. Resolved by
  - swapping I6 and I7. OR
  - inserting a NOP instruction between I5 and I6
- No branch conflicts since delayed branching is used.
- No data dependency between I4 and I5 since I4 is just reading and not writing to R3

(Not necessary to be covered during tutorial)

[Optional, but students are encouraged to attempt these questions]

9.4 Rounding Error

***Note to tutors**: This question is based on an actual example of disasters caused by bad numeric. The inaccurate calculation of time has contributed to fatalities when it was used for intercepting enemy missiles (see http://www.ima.umn.edu/~arnold/disasters/patriot.html). While an error of 0.34 seconds may not seem like much, a missile travelling at 1,676 meters per second would have travelled approximately 500 meters in that time.*

5. You have been tasked to write a program that calculates the actual time based on a counter that is incremented once every 0.10 seconds. For example, if the counter value is 3,600,000, you would expect the actual time to be 100 hours ((3,600,000 x 0.1) / (60 x 60)).

   Suppose you have decided to use a 24-bit fixed point representation as shown in Figure 8.4 to store the value of 0.10 seconds ($2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} + \ldots$).

0	▪	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0

Figure 9.4 – Fixed point representation of 0.1010

   a. Approximate the round-off error (in decimal) of 0.10_{10} due to the fixed-point representation.
   b. What is the effect of this round-off error on the time calculated if the counter value is 3,600,000?

[Suggested Solution]

a. The decimal value of the fixed point representation in Figure 8.4 is 0.0999999046325684_{10}.

   Hence, the round-off error is approximately **0.000000095_{10}** (0.10_{10} - 0.0999999046325684_{10}).

b. The time calculated will be off by approximately **0.34 seconds** (3600000_{10} x 0.000000095_{10}).