CS 2318 (Assembly Language, §253 & §254, Spring 2023) *penalty if not filled--> Name: Samuel Wait

# Assignment 1 Part 2 (posted 02/07/2023, DUE as assigned on Canvas )

Score (by grader):

(Don't have to type but must be legible - **no credits for any parts I have difficulty reading/understanding**.)

5 (8 points) For each (sub-)part, clearly show (may include concise explanations) how you get your answer to not risk incurring penalty.

***Question 5a***: Represent the *decimal signed whole numbers* **472** and **-472** in *sign-magnitude* (SM).

| 472 | -472 |
|---|---|
| 2 \| 472<br>236 R 0<br>118 R 0<br>59 R 0<br>29 R 1<br>14 R 1<br>7 R 0<br>3 R 1<br>1 R 1<br>0 R 1<br>=> $(472)_{10} = (0\,111011000)_2$<br>Sign bit | $(-472)_{10} = (1\,111011000)_2$<br>Sign bit<br>flipped the sign-bit |

***Question 5b***: Represent the *decimal signed whole numbers* **472** and **-472** in *1's-complement* (1C).

| 472 | -472 |
|---|---|
| 1C: $(472)_{10} = (0\,111011000)_2$<br>Positive whole #'s in 1C are the same as in SM | 1C: $(-472)_{10} = (1\,000100111)_2$<br>flipped each bit from the original binary number |

***Question 5c***: Represent the *decimal signed whole numbers* **472** and **-472** in *2's-complement* (2C).

| 472 | -472 |
|---|---|
| 2C: $(472)_{10} = (0\,111011000)_2$<br>Positive whole #'s in 2C are the same as in SM and 1C | 2C: $(-472)_{10} = (1000100000)_2$<br>kept the first 3 bits on the right<br>then flipped the rest |

***Question 5d***: Represent the *decimal signed whole numbers* **472** and **-472** in *excess-511* (*excess*DASH*511*).

| 472 | -472 |
|---|---|
| $472_{10} + 511 = 983_{10}$<br>2 \| 983<br>491 R 1<br>245 R 1<br>122 R 1<br>61 R 0<br>30 R 1<br>15 R 0<br>7 R 1<br>3 R 1<br>1 R 1<br>0 R 1<br>=> $(472)_{10}$ in Excess-511: $(983_{10}) = (0\,111101011\,1)_2$ | $-472_{10} + 511 = 39_{10}$ => 2 \| 39<br>19 R 1<br>9 R 1<br>4 R 1<br>2 R 0<br>1 R 0<br>0 R 1<br>=> $(-472)_{10}$ in excess-511: $(39)_{10} = (1\,100111)_2$ |

6 (3 points) Using *Horner's scheme*, represent in *decimal* (evaluate the *decimal* value of) the *2's complement number* **10011010**.

CAUTION: You will earn NO CREDITS if you don't use *Horner's scheme* to more efficiently evaluate polynomial.

You MUST apply *Horner's scheme* ***directly*** on the polynomial expansion for the number ***as given*** (NOT on the polynomial expansion for the 2's complement of the given number).
(This means that you should NOT, for instance, evaluate **1011** as *-<corresponding positive value of* **1011** *using Horner's scheme>*.)
(It also means that you should NOT, for instance, evaluate **1011** as *<unsigned value of* **1011** *using Horner's scheme>* **- 2**$^4$.)

You MUST apply *Horner's scheme* to cover all contributing bits, i.e., all contributing bits (not just the positive contributing bits, for instance) must participate in the repeated factoring.

You will earn NO CREDITS if you simply show the final result and not *clearly show working* (i.e., intermediate steps).

2C: $(10011010)_2$ => SM: $(1\,1100010)_2$

Horner's Scheme: $(11100010)_2$

$2 \cdot 1 + 1$

$= 2 \cdot 3 + 1$

$= 2 \cdot 7 + 0$

$= 2 \cdot 14 + 0$

$= 2 \cdot 28 + 0$

$= 2 \cdot 56 + 1$

$= 2 \cdot 113 + 0 = \boxed{(226)_{10}}$

7 (8 points) For each (sub-)part, do or answer the following:

- Show how you would add the two 8-bit binary numbers by filling in each of the blank boxes with a `0` or `1`.
  NOTE1: Addition of 2's complement numbers is done just like the addition of unsigned whole numbers, so you only need to show each addition once.
  NOTE2: There are 9 blank boxes for the `Carry` bits and 8 blank boxes for the `Sum` bits.
  CAUTION: Every blank box should be filled with a `0` or `1`; if you leave any boxes blank, you risk incurring penalty.
- Circle one observation from the list below that is applicable and useful to help determine whether overflow has occurred, if the numbers are *unsigned whole numbers* and only 8 bits are available for holding the result.
  CAUTION: There will be penalty for additional observations circled that are not applicable/helpful.
- Circle Y or N to indicate your inference on whether overflow has occurred or not (if the numbers are *unsigned whole numbers* and only 8 bits are available for holding the result) based on the circled observation(s).
- Circle two observations from the list below either of which is applicable and useful to help determine whether overflow has occurred, if the numbers are *2's complement numbers* and only 8 bits are available for holding the result.
  CAUTION: There will be penalty for observations circled that are not applicable/helpful.
- Circle Y or N to indicate your inference on whether overflow has occurred or not (if the numbers are *2's complement numbers* and only 8 bits are available for holding the result) based on the circled observation(s).

List of observations (→ each is in terms of the state(s) of certain bit(s) in 1 or 2 positions -- good for hardware implementation):
(A) Carry-out is 0 when bits at most-significant-bit position are added.
(B) Carry-out is 1 when bits at most-significant-bit position are added.
(C) Addends have opposite signs.
(D) Addends are both positive, sum is positive.
(E) Addends are both positive, sum is negative.
(F) Addends are both negative, sum is positive.
(G) Addends are both negative, sum is negative.
(H) Carry-in and carry-out at most-significant-bit position are the same.
(I) Carry-in and carry-out at most-significant-bit position are not the same.

(a)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Carry: | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| | | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| Sum: | | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

Observation if *unsigned*: ( A / [✓ B] / C / D / E / F / G / H / I ) (← circle only 1)
Overflow if *unsigned*? ( [✓ Y] / N )

Observations if *2's complement*: ( A / B / C / D / E / [✓ F] / G / H / [✓ I] ) (← circle exactly 2)
Overflow if *2's complement*? ( [✓ Y] / N )

(b)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Carry: | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| | | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| Sum: | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

Observation if *unsigned*: ( [✓ A] / B / C / D / E / F / G / H / I ) (← circle only 1)
Overflow if *unsigned*? ( Y / [✓ N] )

Observations if *2's complement*: ( A / B / C / D / [✓ E] / F / G / H / [✓ I] ) (← circle exactly 2)
Overflow if *2's complement*? ( [✓ Y] / N )

(c)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Carry: | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| | | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| Sum: | | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

Observation if *unsigned*: ( A / [✓ B] / C / D / E / F / G / H / I ) (← circle only 1)
Overflow if *unsigned*? ( [✓ Y] / N )

Observations if *2's complement*: ( A / B / C / D / E / F / [✓ G] / [✓ H] / I ) (← circle exactly 2)
Overflow if *2's complement*? ( Y / [✓ N] )

(d)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Carry: | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| Sum: | | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |

Observation if *unsigned*: ( A / [✓ B] / C / D / E / F / G / H / I ) (← circle only 1)
Overflow if *unsigned*? ( [✓ Y] / N )

Observations if *2's complement*: ( A / B / [✓ C] / D / E / F / G / [✓ H] / I ) (← circle exactly 2)
Overflow if *2's complement*? ( Y / [✓ N] )