Note: ASSN 1 Date: / /

Q1

a) Convert unsigned decimal → binary value (w = 8)
— binary value → hexadecimal value.

I) 157_{10} : To binary: $157_{10}/2 = 78$ (r=1)
$78/2 = 39$ (_0)
$39/2 = 19$ (_1)
$19/2 = 9$ (_1)
$9/2 = 4$ (_1)
$4/2 = 2$ (_0)
$2/2 = 1$ (_0)
$1/2 = 0$ (_1)
→ $157_{10} = 1001\,1101_2$
→ $= (9D)_{16}$

II) 248_{10} To binary: $248 - 128 = 120$ ($128 = 2^7$)
$120 - 64 = 56$ ($64 = 2^6$)
$56 - 32 = 24$ ($32 = 2^5$)
$24 - 16 = 8$ ($16 = 2^4$)
$8 - 8 = 0$ ($8 = 2^3$)
→ $248_{10} = 1111\,1000_2$
$= (F8)_{16}$

b) Convert signed decimal → T's C binary (w=8)
→ hexadecimal

I) 123_{10} ⇒ $123 - 64 = 59$ ($64 = 2^6$)
$59 - 32 = 27$ ($32 = 2^5$)
$27 - 16 = 11$ ($16 = 2^4$)
$11 - 8 = 3$ ($8 = 2^3$)
$3 - 2 = 1$ ($2 = 2^1$)
$1 - 1 = 0$ ($1 = 2^0$)
⇒ $123_{10} = 0111\,1011_2$
$(7B)_{16}$

HONG HA

Note: Date: / /

II) -74_{10} => $74 - 64 = 10$ $(64 = 2^6)$
(<0) $10 - 8 = 2$ $(8 = 2^3)$
$2 - 2 = 0$ $(2 = 2^1)$
$74_{10} = 0100\ 1010_2$
$\therefore -74_{10} = 1011\ 0110_2$
$= (B\ 6)_{16}$

e) Interpret binary val as [unsigned / T's C signed

I) $1110\ 1001_2$ → unsigned : $= 2^7 + 2^6 + 2^5 + 2^3 + 2^0$
$= 128 + 64 + 32 + 8 + 1$
$= (233)_{10}$
→ signed : $= -2^7 + 2^6 + 2^5 + 2^3 + 2^0$
$= (-23)_{10}$

II) $1001\ 0110_2$ → unsigned : $= 2^7 + 2^4 + 2^2 + 2^1$
$= 128 + 16 + 4 + 2$
$= (150)_{10}$
→ signed : $= -2^7 + 2^4 + 2^2 + 2^1$
$= (-106)_{10}$

d) $\overset{247}{V}$ Signed val : $247 - 256 = -9$ $(256 = 2^8)$
e) $\overset{-112}{V}$ Unsigned val : $-112 + 256 = 144$ (_____)

Q2

a) Convert to binary & compare true vs actual Sum (w=8)

I) 74_{10} => $74 - 64 = 10$ $(64 = 2^6)$
$10 - 8 = 2$ $(8 = 2^3)$
$2 - 2 = 0$ $(2 = 2^1)$
$\therefore 74_{10} = 0100\ 1010_2$

63_{10} => $63 - 32 = 31$ $(32 = 2^5)$
$31 - 16 = 15$ $(16 = 2^4)$
$15 - 8 = 7$ $(8 = 2^3)$
$7 - 4 = 3$ $(4 = 2^2)$
$3 - 2 = 1$ $(2 = 2^1)$
$1 - 1 = 0$ $(1 = 2^0)$
$\therefore 63_{10} = 0011\ 1111$

74 + 63 = 137

$0100\ 1010$ ← carry in
$+\ 0011\ 1111$
$1000\ 1001$ (True = Actual Sum)

=> No overflow

HONG HA

Note: Date: / /

II) 123_{10} => 123 - 64 = 59 (64 = 2^6)
59 - 32 = 27 (32 = 2^5)
27 - 16 = 11 (16 = 2^4)
11 - 8 = 3 (8 = 2^3)
3 - 2 = 1 (2 = 2^1)
1 - 1 = 0 (1 = 2^0)

123_{10} = 0111 1011 (1 1 1 1 1 1 1 1 => carry in)

157_{10} => 157 - 128 = 29 (128 = 2^7)
29 - 16 = 13 (16 = 2^4)
13 - 8 = 5 (8 = 2^3)
5 - 4 = 1 (4 = 2^2)
1 - 1 = 0 (1 = 2^0)

=> 157_{10} = 1001 1101

$$\begin{array}{r} 123 \\ +\ 157 \\ \hline 280 \end{array} \quad + \begin{array}{r} 0111\ 1011 \\ 1001\ 1101 \\ \hline 1\ 0001\ 1000 \end{array}$$

True Sum
≠
Actual Sum
1000 1000

carry out => Overflow (280 out of range)

⟹ Detecting overflow: Overflow
(=) 1) At bit level: carry out bit = 1
(=) 2) Decimal operands: decimal sum > $2^w - 1$ (max unsigned)

b) Convert to binary & compare True vs Actual Sum (w = 8)

I) 28_{10} => 28 - 16 = 12 (16 = 2^4)
12 - 8 = 4 (8 = 2^3)
4 - 4 = 0 (4 = 2^2)

-74_{10} = 1011 0110$_2$ (see @1)

=> 28_{10} = 0001 1100$_2$ (1 1 1 1 1 1 1 0 => carry in)

$$\begin{array}{r} 28 \\ +\ -74 \\ \hline -46 \end{array} \quad + \begin{array}{r} 0001\ 1100 \\ 1011\ 0110 \\ \hline 1101\ 0010 \end{array}$$

(True = Actual Sum) => No overflow

II) -117_{10} => $-117 + 2^8 = 139$
139 - 128 = 11 (128 = 2^7)
11 - 8 = 3 (8 = 2^3)
3 - 2 = 1 (2 = 2^1)
1 - 1 = 0 (1 = 2^0)
=> -117_{10} = 1000 1011$_2$

126_{10} => 126 - 64 = 62 (64 = 2^6)
62 - 32 = 30 (32 = 2^5)
30 - 16 = 14 (16 = 2^4)
14 - 8 = 6 (8 = 2^3)
6 - 4 = 2 (4 = 2^2)
2 - 2 = 0 (2 = 2^1)
=> 126_{10} = 0111 1110$_2$

HONG HA

Note: Date: / /

```
                 1 1 1 1 1 1 1  <- carry in
  126            0 1 1 1 1 1 1 0
+ -117         + 1 0 0 0 1 0 1 1
-----          -----------------
    9        1 | 0 0 0 0 1 0 0 1   True Sum   => no arithmetic overflow
             ↓ | 0 0 0 0 1 0 0 1   Actual Sum   (carry out discarded)
          carry out
```

III) 74_{10} = 0100 1010$_2$; 63_{10} = 0011 1111$_2$

```
                 1 1 1 1 1 1 1  <- carry in
  74             0 1 0 0 1 0 1 0
+ 63           + 0 0 1 1 1 1 1 1
-----          -----------------
 137             1 0 0 0 1 0 0 1   => Positive Overflow (137 out of range)
```

IV) -119_{10} => $-119 + 2^8 = 137$

137 − 128 = 9 ($128 = 2^7$)

9 − 8 = 1 ($8 = 2^3$)

1 − 1 = 0 ($1 = 2^0$)

=> -119_{10} = 1000 1001$_2$

-105_{10} => $-105 + 2^8 = 151$

151 − 128 = 23 ($128 = 2^7$)

23 − 16 = 7 ($16 = 2^4$)

7 − 4 = 3 ($4 = 2^2$)

3 − 2 = 1 ($2 = 2^1$)

1 − 1 = 0 ($1 = 2^0$)

=> -105_{10} = 1001 0111$_2$

```
                1 1 1 1 1  <- carry in
  -119          1 0 0 0 1 0 0 1
+ -105        + 1 0 0 1 0 1 1 1
-----         -----------------
 -224       1 | 0 0 1 0 0 0 0 0   True Sum   => Negative overflow
carry out ↓ | 0 0 1 0 0 0 0 0   Actual Sum   (-224 out of range)
```

=> Detect overflow:

1) At bit level: let s = actual sum of u & v

$MSB(s) \neq MSB(u) = MSB(v)$ (=) overflow

(negative if $MSB(u) = 1$

positive if $MSB(u) = 0$)

2) Using decimal operands:

if decimal Sum $> 2^{w-1} - 1$ -> positive overflow

$< -2^{w-1}$ -> negative overflow

Q.3 (b) In CSIL computers are Little Endian.

0x7ffcd599845c	0x39 (LSB)	num = 12345 (0x00003039)
d	0x30	↓
e	0x00	↓
f	0x00 (MSB)	

HONG HA