Binary Numbers and Arithmetic

Hexadecimal
$F9 = %1111 1001

		0	0	0	0	+1 CARRY
15	$F	1	1	1	1	
14	$E	1	1	1	0	
13	$D	1	1	0	1	
12	$C	1	1	0	0	
11	$B	1	0	1	1	
10	$A	1	0	1	0	
9	$9	1	0	0	1	
8		1	0	0	0	
7		0	1	1	1	OVERFLOW
6		0	1	1	0	
5		0	1	0	1	
4		0	1	0	0	
3		0	0	1	1	
2		0	0	1	0	+1
1	$1	0	0	0	1	+1
0	$0	0	0	0	0	+1 CARRY
-1		1	1	1	1	
-2		1	1	1	0	+1
-3		1	1	0	1	
-4		1	1	0	0	
-5		1	0	1	1	
-6		1	0	1	0	
-7		1	0	0	1	
-8		1	0	0	0	OVERFLOW

unsigned

signed two's complement

Two's Complement (TC) Properties

① binary number B, $-B = \overline{B}+1$ (2 steps: ① invert all bits, ② add 1)

$-5 = \overline{0101}+1 = 1010+1 = 1011$

② MSB = sign bit
1 = negative, eg %1011 −5
0 = positive, eg %0101 +5

③ sign-extension

$5 = \%0101 = \%0000\ 0101$

$-5 = \%1011 = \%1111\ 1011$ (copy sign bit)

check: $\overline{0000\ 0101}+1 = 1111\ 1010+1 = 1111\ 1011$

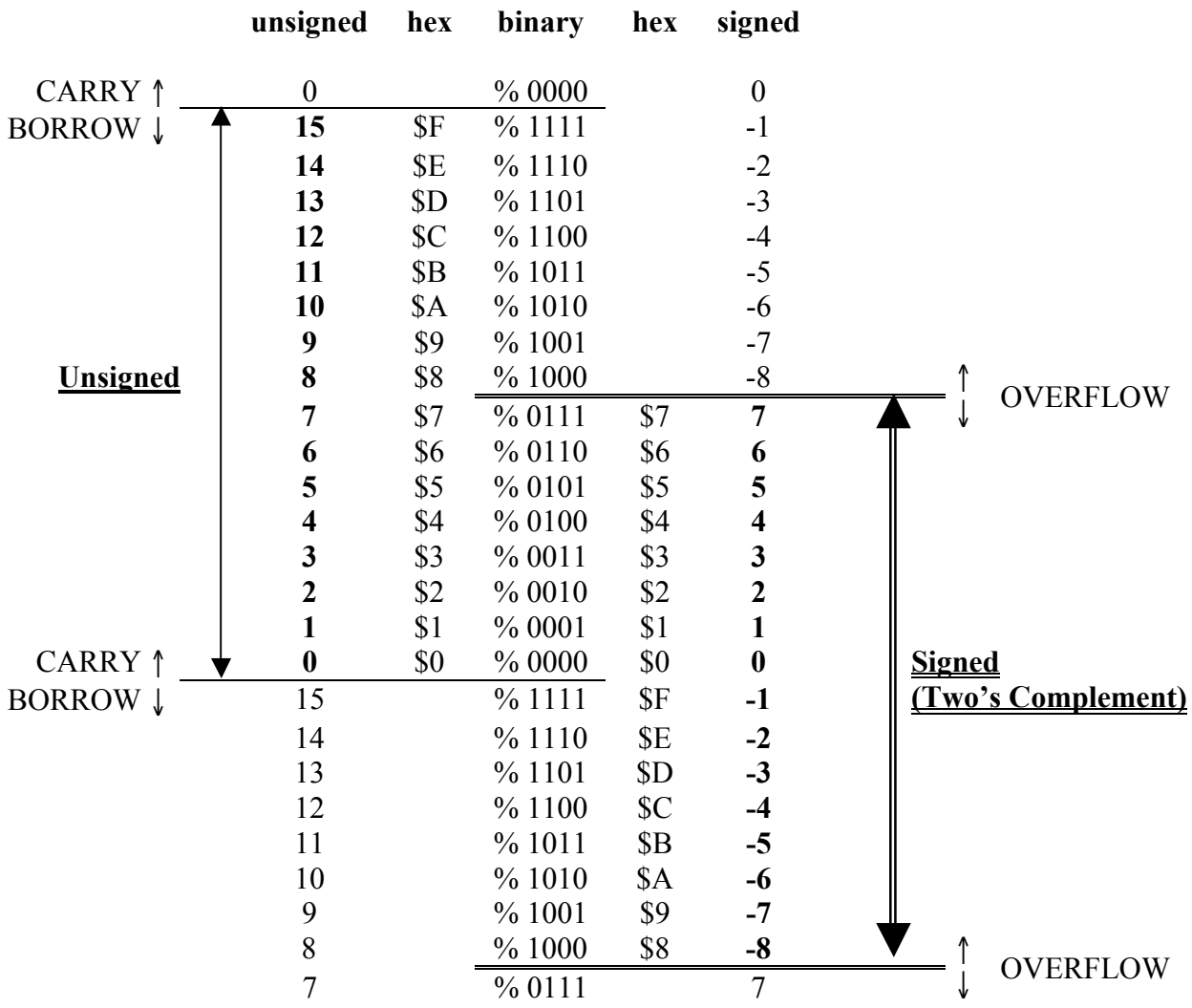

TABLE: Unsigned versus Signed(TC) Number System for 4-bit values.

④ addition / subtraction

signed (TC) ± is identical to unsigned

∴ use exact same digital logic gates

but there is one key difference:

UNSIGNED triggers a { CARRY when adding crosses 1111→0000 (always +ve numbers) ; BORROW when subtracting crosses 0000→1111

C, B

SIGNED triggers an OVERFLOW when adding/subtracting crosses 0111 ↔ 1000

V

Carry Example (always unsigned)

$$\begin{array}{r} \$65 \\ +\$F9 \end{array} \Rightarrow \begin{array}{r} \%\ 0110\ 0101 \\ +\%\ 1111\ 1001 \\ \hline 1\ 0101\ 1110 \end{array}$$

1 = carry; 0101 1110 = answer

detecting carry-out is easy

$$\begin{array}{r} a_7\ a_6 \ldots a_0 \\ b_7\ b_6 \ldots b_0 \\ \hline r_7\ r_6 \ldots r_0 \end{array}$$

exactly 3 cases generate a carry: $a_7 b_7 + a_7 \bar{r}_7 + b_7 \bar{r}_7$

$$C = a_7 b_7 + a_7 \bar{r}_7 + b_7 \bar{r}_7$$

for adds

Borrow Example

(always unsigned)

$25 ⇒ %0010 0101 ← smaller
- $F9 − %1111 1001 ← bigger
0010 1100

small - big < 0 ∴ need to borrow!

borrow is needed

detecting borrow is easy

exactly 3 cases generate borrow: $\bar{a}_7 b_7 + \bar{a}_7 r_7 + b_7 r_7$

$B = \bar{a}_7 b_7 + \bar{a}_7 r_7 + b_7 r_7$ for subtracts

- $\bar{a}_7 b_7$: B > A
- $\bar{a}_7 r_7$: result > A
- $b_7 r_7$: B is big and R is big means A was smaller than B

NOTE: You do not have to remember borrow logic.
You do have to remember carry logic.

Overflow Examples

(always signed)

127 + 1 = 128 ⇒ %0111 1111 + %0000 0001 = %1000 0000
← positive numbers added
← negative number result!
result: −128

−128 + −1 ⇒ %1000 0000 + %1111 1111 = 1 %0111 1111
← negative numbers added
← positive result!
result: 127

carry is ignored / meaningless (only for unsigned numbers)

detecting overflow is easy

exactly 2 overflow cases:

$V = a_7 b_7 \bar{r}_7 + \bar{a}_7 \bar{b}_7 r_7$ for adds

- pos + neg ⇒ never overflows
- pos + pos ⇒ overflows if answer has wrong sign!
- neg + neg ⇒ overflows if answer has wrong sign!