

Overflow

- ▶ When we use fixed size Register for arithmetic operands there is the hazard at each micro-ops of overflow.
- ▶ If Register R stores an n-bit 2's complement number, then:

$$-2^{n-1} \leq R \leq 2^{n-1}-1$$

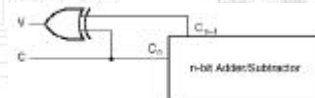
- ▶ Note the asymmetry, so even negation can cause overflow, e.g.

$$R \rightarrow -2^{n-1} \text{ then } R \rightarrow -R \text{ } \Rightarrow \text{ OVERFLOW}$$

Carry and Overflow

- ▶ We can detect overflow for all the previous operations simply by recording the status bits C = Carry and V = Overflow (see section 3.10 I Mano and Kime).

$$K_1: C \rightarrow C_n, V \rightarrow C_n \wedge C_{n-1}$$

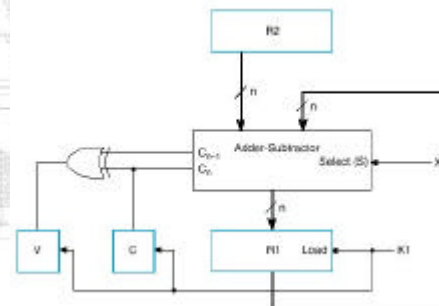


Adder-Subtractor

- ▶ This then is the basis for the adder-subtractor on the next page which uses control input X to select addition and X for subtraction.

$$\begin{aligned} \bar{X}.K_1: R1 &\leftarrow R1 + R2 \\ X.K_1: R1 &\leftarrow R1 + R2 + 1 \\ K_1: C &\rightarrow C_n, V \rightarrow C_n \wedge C_{n-1} \end{aligned}$$

Add & Subtract Micro operation



2BA4 Logic Micro-operations

- The aim here is to provide an effective set of bit-wise functions. A typical basic set follows:

Symbolic micro-op	Description
$R0 \leftarrow \overline{R1}$	Logical bitwise NOT (1's compliment)
$R0 \leftarrow R1 \cap R2$	Logical bitwise AND (clears bits)
$R0 \leftarrow R1 \cup R2$	Logical bitwise OR (sets bits)
$R0 \leftarrow R1 \oplus R2$	Logical bitwise XOR (complements bits)

2BA4 \cap And \cup

- George Bode Prof. Of mathematics in UCC introduced the notation ' \cap ' and ' \cup ' in 1854, and they are used in Register Transfer (RT) notation if it is necessary to distinguish addition from logical OR. For example:

$K_1 + K_2 : R1 \leftarrow R2 + R3, R4 \leftarrow R5 \cap R6$

Logical OR

Logical OR

addition

- The OR micro-ops will always use \cup .

2BA4 Shift Micro-operations

- These provide lateral bitwise shift which are essential for many basic arithmetic algorithms e.g. multiplication division, square root...
- The minimal set is:

$R \leftarrow sr R \circ R_i \leftarrow R_{i+1} \quad i=0, n-2, R_{n-1} \leftarrow 0$
 $R \leftarrow sl R \circ R_i \leftarrow R_{i-1} \quad i=1, n-1, R_0 \leftarrow 0$

2BA4 Logical Shifts

- These are logical shifts and from them you can develop variants which handle the end bits differently, e.g. arithmetic shift, rotates...
- Please see table 9-5 in Mano and Kime.

Transfer Micro-operations

- ▶ Providing choice of path
- ▶ Two approaches are used:
 - ▶ Multiplexer-based transfer for speed
 - ▶ Bus-based transfers for flexibility and economy

Transfer with Multiplexer

- ▶ The if-then else control structure, when applied to identity micro-ops, results in the destination register requiring selective access to two different source registers.

If ($K_1=1$) then $R0 \leftarrow R1$
 else if ($K_2=1$) then $R0 \leftarrow R2$

$K_1: R0 \leftarrow R1, K_2: R0 \leftarrow R2$

Two Sources - One Destination

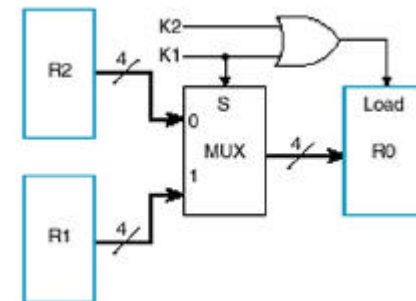
- ▶ To route two sources to one destination we can use a 2:1 MUX with control input S , data input $D0 \cup D1$, and $R0$ must have a load control $LOAD_{R0}$.
- ▶ From the RT description we deduce the following functions for these control inputs.

$$LOAD_{R0} = K_1 + K_1 K_2 = K_1 + K_2$$

$$S = K_1 \quad D1 = R1 \quad D0 = R2$$

Multiplexer

- ▶ From this we derive the circuit schematic, assuming 4-bit Registers.



Multiplexer Detail

- ▶ Note changes in K1 only affect the MUX - R0 path, so transients are short, allowing fast operation.

