Review Problem 7

- Sometimes it can be useful to have a program loop infinitely. We can do that, regardless of location, by the instruction:
- LOOP: B ~~LOOP~~ 0
- Convert this instruction to machine code

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	0	0	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	0	0	0	0

B : 05

0

Conversion example

Compute the sum of the values 0…N-1

```
    ADD X1, X31, X31
    ADD X2, X31, X31
    B ~~TEST~~ +3
TOP:
    ADD X1, X1, X2
    ADDI X2, X2, #1
TEST:
    SUBS X31, X2, X0
    B.LT ~~TOP~~ -3
END:
```

ADD X1, X31, X31	100 0101 1000 (458)	11111 (X31)	000000 (0)	11111 (X31)	00001 (X1)
ADD X2, X31, X31	100 0101 1000 (458)	11111 (X31)	000000 (0)	11111 (X31)	00010 (X2)
B TEST +3	00 0101 (05)	00000000000000000000000011 (+3)			
ADD X1, X1, X2	100 0101 1000 (458)	00010 (X2)	000000 (0)	00001 (X1)	00001 (X1)
ADDI X2, X2, #1	10 0100 0100 (244)	000000000001 (#1)		00010 (X2)	00010 (X2)
SUBS X31, X2, X0	111 0101 1000 (758)	00000 (X0)	000000 (0)	00010 (X2)	11111 (X31)
B.LT TOP -3	0101 0100 (54)	1111111111111111101 (-3)	0 1011 (0B)		

Assembly & Machine Language

Assembly

Simple instructions

Mnemonics for humans

(Almost) 1-to-1 relationship w/machine language

Machine Language

Numeric representations of instructions

Fixed format(s)

Directly controls CPU hardware

Computer Arithmetic

Readings: 3.1-3.3, A.5

Review binary numbers, 2's complement

Develop Arithmetic Logic Units (ALUs) to perform CPU functions.

Introduce shifters, multipliers, etc.

Binary Numbers

Decimal: $469 = 4*10^2 + 6*10^1 + 9*10^0$

Binary: $01101 = 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = (13)_{10}$

Example: $0111010101 = (?)_{10}$

$1+4+16+64+128+256$

$\cancel{20}$ 21

384

85

469_{10}

2's Complement Numbers

Positive numbers & zero have leading 0, negative have leading 1

Negation: Flip all bits and add 1

13_2

Ex: $-(01101)_2 =$ $10010 + 1 = 10011_2$

To interpret numbers, convert to positive version, then convert:

$11010 = -(-11010)$

$= -(00101+1)$

$= -(00110)$

$= -(6)$

$= -6_{10}$

$\underset{+}{0}\underbrace{1100}$ = $+12_{10}$

Sign Extension

Conversion of n-bit to (n+m)-bit 2's complement: replicate the sign bit

$b_3b_2b_1b_0 = b_3b_3b_3b_2b_1b_0 = b_3b_3b_3b_3b_3b_3b_3b_3b_3b_3b_3b_3b_2b_1b_0$

Ex - Convert to 8-bit: 01101 = $(13)_{10}$ 11101 = $(-3)_{10}$

000 01101

111 11101

correct?

= −(− 11111101)

= −(00000010+1)

= −(00000011)

= − 3

Arithmetic Operations

Decimal:

```
 ~~1~~ 1 1 1 0
    5 7 8 9 2
  + 7 8 9 5 6
  -----------
    3 6 8 4 8
```

odometer math

Binary:

```
 ~~0~~ 0 0 0 1 1 1
   1 0 1 0 1 1 1
 + 0 1 0 0 1 0 1
 ---------------
   1 1 1 1 1 0 0
```

Binary:

```
   1 0 1 0 0 1 1 0
 - 0 0 0 1 0 1 1 1
 -----------------
```

flip the bits

```
 ~~1~~ 1 1 0 0 0 0 0 1
       1 0 1 0 0 1 1 0
     + 1 1 1 0 1 0 0 0
     -----------------
       1 0 0 0 1 1 1 1
```

Add one

Overflows

Operations can create a number too large for the number of bits

- n-bit 2's complement can hold $-2^{(n-1)} \ldots 2^{(n-1)}-1$

Can detect overflow in addition when highest bit has carry-in $\neq$ carry-out

- (carry-in) $\oplus$ (carry-out) = 1

$$\begin{array}{r} \\ 5 \\ 3 \\ \hline -8 \end{array} \qquad \begin{array}{ccccc} 0 & 1 & 1 & 1 & \\ & 0 & 1 & 0 & 1 \\ & 0 & 0 & 1 & 1 \\ \hline & 1 & 0 & 0 & 0 \end{array}$$

Overflow

$$\begin{array}{r} \\ -7 \\ -2 \\ \hline 7 \end{array} \qquad \begin{array}{ccccc} 1 & 0 & 0 & 0 & \\ & 1 & 0 & 0 & 1 \\ & 1 & 1 & 1 & 0 \\ \hline & 0 & 1 & 1 & 1 \end{array}$$

Overflow

$$\begin{array}{r} \\ 5 \\ 2 \\ \hline 7 \end{array} \qquad \begin{array}{ccccc} 0 & 0 & 0 & 0 & \\ & 0 & 1 & 0 & 1 \\ & 0 & 0 & 1 & 0 \\ \hline & 0 & 1 & 1 & 1 \end{array}$$

No overflow

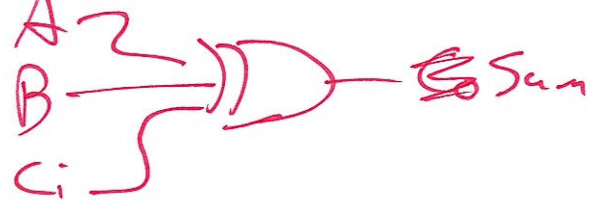
$$\begin{array}{r} \\ -3 \\ -5 \\ \hline -8 \end{array} \qquad \begin{array}{ccccc} 1 & 1 & 1 & 1 & \\ & 1 & 1 & 0 & 1 \\ & 1 & 0 & 1 & 1 \\ \hline & 1 & 0 & 0 & 0 \end{array}$$

No overflow

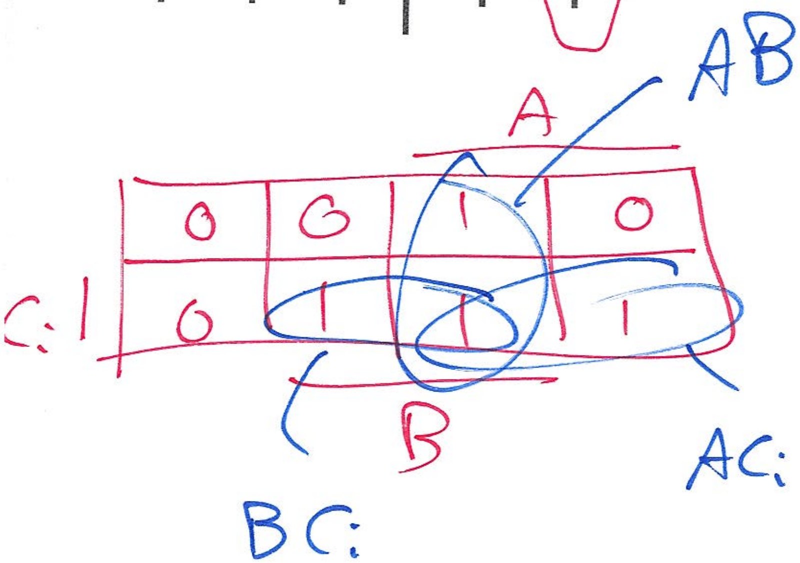
On an overflow, the top bit is flipped

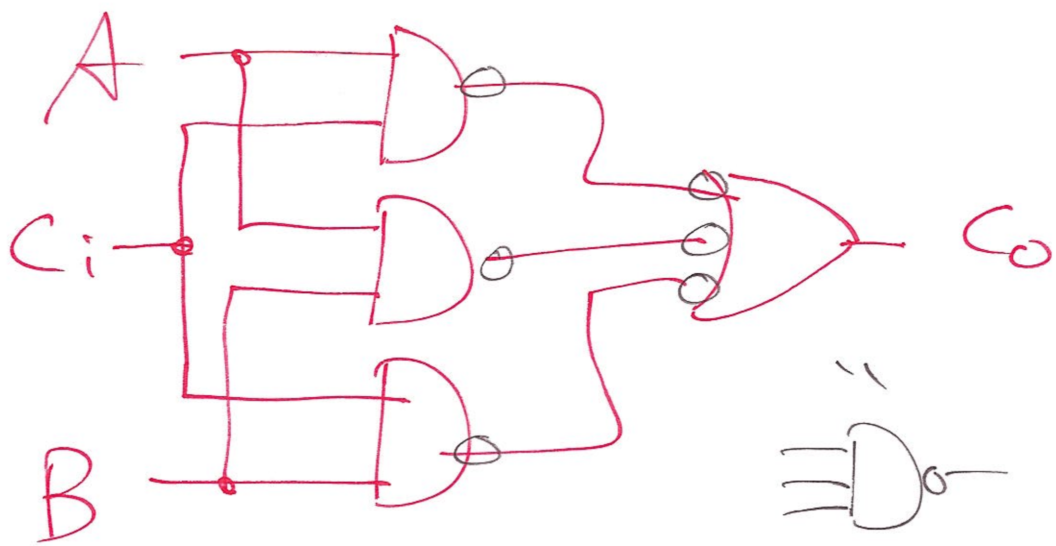
Full Adder

A	B	CI	CO	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



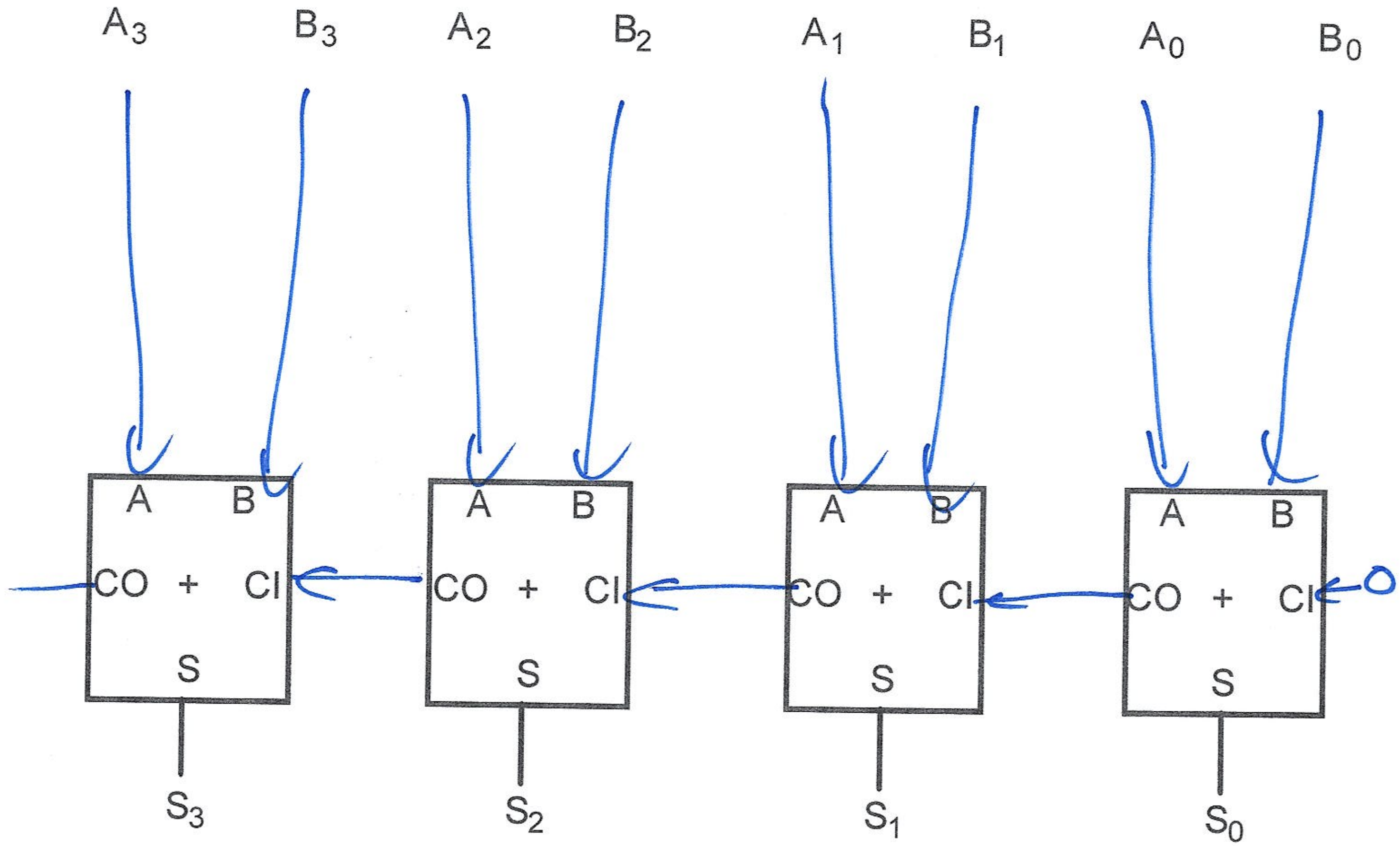
4xN ANDs





Multi-Bit Addition

$$\begin{array}{rcccc} C_4 & C_3 & C_2 & C_1 & \leftarrow 0 \\ & A_3 & A_2 & A_1 & A_0 \\ + & B_3 & B_2 & B_1 & B_0 \\ \hline & S_3 & S_2 & S_1 & S_0 \end{array}$$



Subtract

Adder/Subtractor

0: $A + B = \quad = A + B + 0$

1: $A - B = A + (-B) = A + \bar{B} + 1$